electronics

Article

Automated Defect Identification System in Printed Circuit Boards Using Region-Based Convolutional Neural Networks

Kavindu Denuwan Weerakkody [1], Rebecca Balasundaram [2,*], Efosa Osagie [2] and Jabir Alshehabi Al-Ani [2]

[1] Department of Computer Science, York St John University, York YO31 7EX, UK; kavindu.weerakkody@yorksj.ac.uk
[2] Department of Computer Science and Data Science, York St John University, York YO31 7EX, UK; e.osagie@yorksj.ac.uk (E.O.); j.alshehabialani@yorksj.ac.uk (J.A.A.-A.)
* Correspondence: r.balasundaram@yorksj.ac.uk

Abstract: Printed Circuit Board (PCB) manufacturing demands accurate defect detection to ensure quality. Traditional methods, such as manual inspection or basic automated object inspection systems, are often time-consuming and inefficient. This work presents a deep learning architecture using Faster R-CNN with a ResNet-50 backbone to automatically detect and classify PCB defects, including Missing Holes (MHs), Open Circuits (OCs), Mouse Bites (MBs), Shorts, Spurs, and Spurious Copper (SC). The designed architecture involves data acquisition, annotation, and augmentation to enhance model robustness. In this study, the CNN-Resnet 50 backbone achieved a precision–recall value of 87%, denoting strong and well-balanced performance in PCB fault detection and classification. The model effectively identified defective instances, reducing false negatives, which is critical for ensuring quality assurance in PCB manufacturing. Performance evaluation metrics indicated a mean average precision (mAP) of 88% and an Intersection over Union (IoU) score of 72%, signifying high prediction accuracy across various defect classes. The developed model enhances efficiency and accuracy in quality control processes, making it a promising solution for automated PCB inspection.

Keywords: printed circuit boards (PCBs); deep learning (DL); region-based convolutional neural networks (RCNN); defect detection; quality control; automated inspection

Academic Editor: Luca Patanè

Received: 6 March 2025
Revised: 5 April 2025
Accepted: 6 April 2025
Published: 10 April 2025

Citation: Weerakkody, K.D.; Balasundaram, R.; Osagie, E.; Alshehabi Al-Ani, J. Automated Defect Identification System in Printed Circuit Boards Using Region-Based Convolutional Neural Networks. *Electronics* **2025**, *14*, 1542. https://doi.org/10.3390/electronics14081542

1. Introduction

In the modern electronics manufacturing industry, Printed Circuit Boards (PCBs) serve as the foundation for virtually all electronic devices. PCBs provide a structural base for component placement and ensure seamless electrical connectivity. With the evolution of technology, PCBs have transitioned from single-layer designs to complex multi-layered architectures, which enable higher circuit density and improved reliability. However, maintaining high quality standards in PCB manufacturing remains a challenge, due to the various defects that can arise during production. To address these challenges, automated, scalable, and highly accurate solutions are essential. Many PCB manufacturing companies face recurring issues with defect detection, which leads to increased production costs and reduced efficiency. This work builds upon previous research and aims to develop a solution to moderate these defects [1,2]. One of the most promising approaches is Region-based Convolutional Neural Networks, which detects and identifies defects within images of PCBs. Employing Deep Neural Networks [3], the Region-based Convolutional Neural Networks (R-CNN) family of deep learning-based object detection models focuses on finding and classifying objects within an image. It is a two-stage approach. First, regions of

interest (ROIs) are proposed, and then they are classified. There are three major variants of R-CNN:

1. R-CNN—This uses selective search to generate around 2000 candidate ROIs. It extracts features from these ROIs using a pre-trained CNN. It then passes these features through a Support Vector Machine (SVM) for classification, and a linear regression model for bounding box refinement [4].
2. Fast R-CNN—Instead of processing each region proposal individually, the entire image is passed through a CNN to generate feature maps. ROIs are extracted from the feature maps and classified using a single network. This variant is faster and more memory-efficient than R-CNN [4].
3. Faster R-CNN—This uses a Region Proposal Network (RPN) to replace the selective search method. The RPN shares convolutional layers with the classification network, significantly speeding up the process. This variant delivers contemporary performance with respect to several object detection benchmarks [5].

In recent years, several deep learning models have been explored for defect detection in PCB manufacturing. According to Table 1, YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector), used to transform models like DETR, have been used in PCB defect detection. They both come with limitations: YOLO and SSD are both speed-optimized, but are limited in their ability to detect small objects and pinpoint the location of defects; DETR (Detection Transformers) is better at global feature extraction, but needs a large amount of data and high computational power, and hence is not ideal for hardware with a low level of resources. We chose Faster R-CNN with ResNet-50, since it offers a suitable balance between accuracy and efficiency. ResNet-50 provides in-depth feature extraction to facilitate effective defect detection, and the Region Proposal Network (RPN) enhances the accuracy of localization, especially for small defects. A comparative analysis of Faster R-CNN against alternative models, such as Inception and ConvNeXt (discussed in the Results and Discussion Section), demonstrates that ResNet-50 outperforms these models in defect detection accuracy and robustness. R-CNN achieves higher accuracy in the detection of defects, which means that this model is more suited to tasks involving object detection and classification compared to traditional methods and other deep learning modules. This makes it ideal for automatic inspection systems.

Table 1. Comparison of Faster R-CNN and YOLO in terms of object detection performance.

Feature	Faster R-CNN	YOLO (One-Stage)	References
Accuracy	Higher accuracy due to region proposal mechanism	Lower accuracy due to direct classification and localization	[6]
Detection of Small Objects	More precise, especially for small and occluded objects	Struggles with small objects due to spatial constraints	[7]
Robustness in Complex Scenes	Performs well in complex backgrounds and cluttered scenes	Less robust in challenging environments	[8]
Computation Speed	Slower due to multi-stage processing	Faster due to single-stage detection	[9]
Generalization Across Datasets	Better adaptability across various datasets and domains	Performance varies significantly based on training data	[10]

In this work, we have developed an automated defect detection system based on a deep learning model, which is described as Faster R-CNN [11], with a ResNet-50 backbone [12]. It is pre-trained on a wide-ranging object detection dataset. Consequently, we have developed and fine-tuned it to identify and classify the six main defects in PCBs. Therefore, the proposed system provides significantly greater capabilities in real-time defect localization and classification in comparison to traditional methods of PCB inspection. We have designed an automated system that aims to detect the top six possible defects related to the PCB manufacturing process, according to Figure 1.

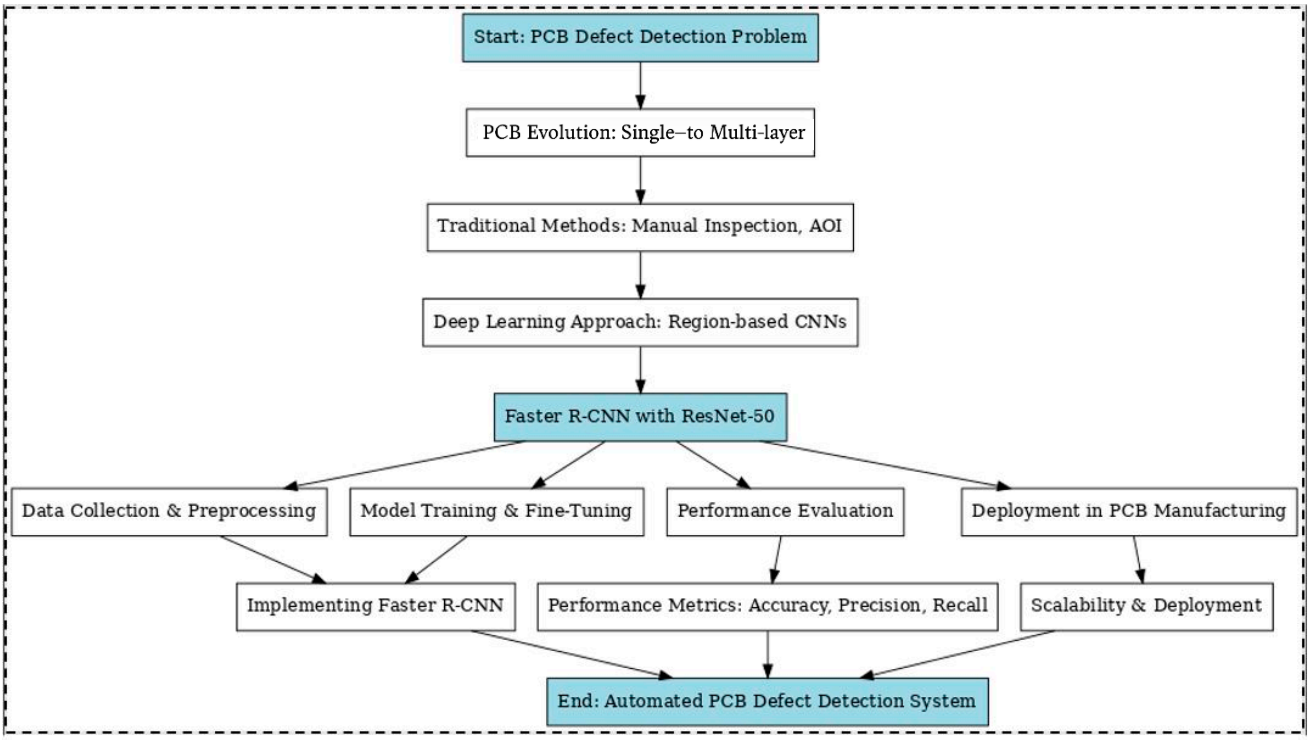


Figure 1. From manual inspection to AI-driven solutions: a comprehensive framework for PCB defect detection using region-based CNNs.

This is thoroughly described in the following sections. The objective of the proposed approach is to contribute, with an information retrieval approach, to automating the defect detection process of Printed Circuit Boards. The contributions of this work are as follows:

(a) Contribution (i)—We implemented the Faster R-CNN model with a ResNet-50 backbone for defect detection. Then, we fine-tuned it according to the six main defect classes.
(b) Contribution (ii)—We evaluated the system's performance in terms of accuracy, precision, and recall, comparing the results with those of traditional techniques.
(c) Contribution (iii)—We conducted a comparative analysis with two different types of backbones, ConvNext and Inception.

The proposed Faster R-CNN model is intended to detect and label six specific PCB defects, such as MHs, OCs, MBs, Short Circuits, Spurs, and Spurious Copper (SC). However, true PCB defects include misalignment, warping, some soldering defects, and contamination, which typically involve other modalities, like infrared imaging or 3D scanning. One drawback of the current model that it is not capable of generalizing to unknown defects, because it relies on pre-existing class labels and may not be able to recognize anomalies outside of its training set. Additionally, complex defects like misalignment and warping require geometric analysis, which Faster R-CNN is not capable of, and some soldering de-

fects are hard to detect due to size, reflectivity, and a lack of training data. To address these issues, future work could include adding anomaly detection paradigms (e.g., autoencoders, GANs) to recognize novel defects, few-shot learning to recognize new defects with few labelled samples, and multimodal analysis involving RGB imaging with infrared or X-ray scanning to conduct in-depth exploration. Such enhancements would make the model more robust and flexible for practical PCB defect detection.

2. Related Work

This section provides a comprehensive review of existing research on PCB defect detection and related industrial applications. It explores various AI-driven methodologies, including Convolutional Neural Networks (CNNs) and image processing techniques, that have been employed to identify manufacturing defects. Previous studies have focused on enhancing detection accuracy, optimizing computational efficiency, and improving defect localization techniques.

(a) Defect Identification Systems

Recent advancements in PCB defect detection leverage Convolutional Neural Networks (CNNs) to minimize false positives and enhance robustness through data augmentation. Attention mechanisms have been integrated into deep learning models to improve detection of minor defects in densely populated PCB areas by focusing on critical regions [13,14].

(b) Deep Learning Techniques

Faster R-CNN and Mask R-CNN, combined with ResNet-50 backbones, have demonstrated superior performance in detecting small defects via region proposal methods and hierarchical feature extraction [5,15]. Transfer learning and data augmentation further optimize model performance. The "Pixelator v2" method enhances defect detection by combining LAB colour space analysis with Sobel edge detection [16,17].

(c) Image Processing with Neural Networks

Oriented R-CNN is described as a postponement of the generic Faster R-CNN, incorporating rotated bounding boxes. This is achieved by defining a Region Proposal Network for rotation-invariant proposals and introducing a rotated ROI alignment module. It predicts the angles of objects, as well as their positions. Therefore, achieving more precise detection can enable the identification of randomly oriented objects. Researcher Xingxing proposed a new framework called Oriented R-CNN to improve object detection performance by taking into consideration the orientation variation of objects. The authors introduced a specific loss function to rotate bounding box regression, which enhances its robustness against rotation variations [11]. De Silva's research [18] focused on automating defect detection in Printed Circuit Boards (PCBs) using image processing techniques, addressing the inefficiencies and inaccuracies of manual inspections. The methodology involved preprocessing PCB images to enhance quality, followed by detection using edge detection, thresholding, and structural operations. By comparing reference images with test images, the system effectively identified defects such as Open Circuits, Short Circuits, missing components, and other manufacturing issues. Furthermore, defect categorization was automated using feature extraction and classification methods, making the process more efficient and reliable. This approach aligns with the objectives of advancing automated PCB defect detection, as seen in recent deep learning-based systems, which enhance accuracy and scalability by incorporating modern computational methods for analyzing high-resolution images. These developments signify a significant step forward in ensuring the reliability and efficiency of PCB manufacturing processes.

(d) Automated Defect Detection Frameworks

An automated defect detection framework, using computed tomography, for industrial applications was proposed by Abbar and his colleagues. Advanced imaging techniques combined with machine learning algorithms were utilized to ensure high-precision defect detection. While the study primarily focused on CT imaging, its methodologies and principles are suggested to be applicable to the PCB inspection domain, where high-resolution imaging and deep learning models are essential for identifying microscopic defects [19]. Recent advances in the surface defect inspection of industrial products using deep learning techniques, leveraging deep learning and machine vision, were proposed by Xiaoqing Zheng, Song Zheng, Yaguang Kong, and Jie Chen in 2021. The integration of a deep CNN with a novel defect localization algorithm was employed to achieve high precision in defect detection. This study demonstrates the applicability of deep learning frameworks to various industrial applications, including PCB inspection [20].

(e) Theoretical Foundations

Theoretical contributions, such as McCulloch and Pitts' artificial neurons and Allen-Zhu et al.'s convergence theory for over-parameterized networks, underpin modern neural networks used in defect detection [21]. Ensemble learning techniques (e.g., bagging, boosting) have also been adapted from photovoltaic fault detection to improve PCB inspection accuracy [22,23].

(f) Defect Detection using Real-Time Environment

The paper "Real-time defect detection using Raspberry Pi with Open CV" presents research on using a Raspberry Pi with OpenCV for real-time defect detection in small industrial components. Though the Raspberry Pi has limited computational power, it was very effective in identifying defects, therefore representing a cost-effective solution for quality control in low-budget industrial setups. The system was optimized for speed and accuracy, providing proof that the Raspberry Pi can handle real-time processing demands [24]. Researchers Mehta and R. Jain proposed an automated defect detection system using Raspberry Pi for use in the textile industry. This framework detected flaws in fabric, maintaining high accuracy and bringing down quality control costs considerably. The authors tried to highlight the practicality of such an approach by demonstrating how the Raspberry Pi can be adapted to existing production lines to deliver a low-cost solution without any performance compromise, but with scalability [25]. As described in the paper "Deep Learning Models for Metal Surface Defect Detection on Raspberry Pi", researchers have proposed a system that involves using deep learning techniques in metal surface defect detection. This may run as efficiently as is possible on a Raspberry Pi. This device is known to be resource-constrained. The system achieved strong detection performance, making it a viable solution for industries requiring real-time inspection and cost-effectiveness. The study underscores the potential of deploying advanced AI-driven applications on low-power hardware like the Raspberry Pi [26]. The current work addresses these gaps by developing a real-time defect detection system that is optimized for speed and accuracy in industrial settings. Advanced data augmentation techniques are implemented to handle class imbalances and improve the detection of small defects. The Faster R-CNN model with a ResNet-50 backbone is fine-tuned for six specific PCB defect classes, ensuring high accuracy. The system is further optimized for resource-constrained devices like the Raspberry Pi, enabling cost-effective deployment. Its performance was validated using metrics like Intersection over Union (IoU), reducing false positives and enhancing defect localization accuracy. This research addresses critical gaps by providing a scalable, efficient, and real-time defect detection system for PCB manufacturing, bridging the divide between theoretical advancements and practical deployment.

3. Methodology

This section is dedicated to describing the experimental setup, taking into consideration how aerial images and videos of the test conditions were acquired, and provides a description of the various kinds of PCB defects. The present study is focused on how to diagnose and identify all the defects of a Printed Circuit Board. If it is diagnosed as having a defective condition, then the proposed technique aims to detect the type of PCB fault, according to the process described in Figure 2.

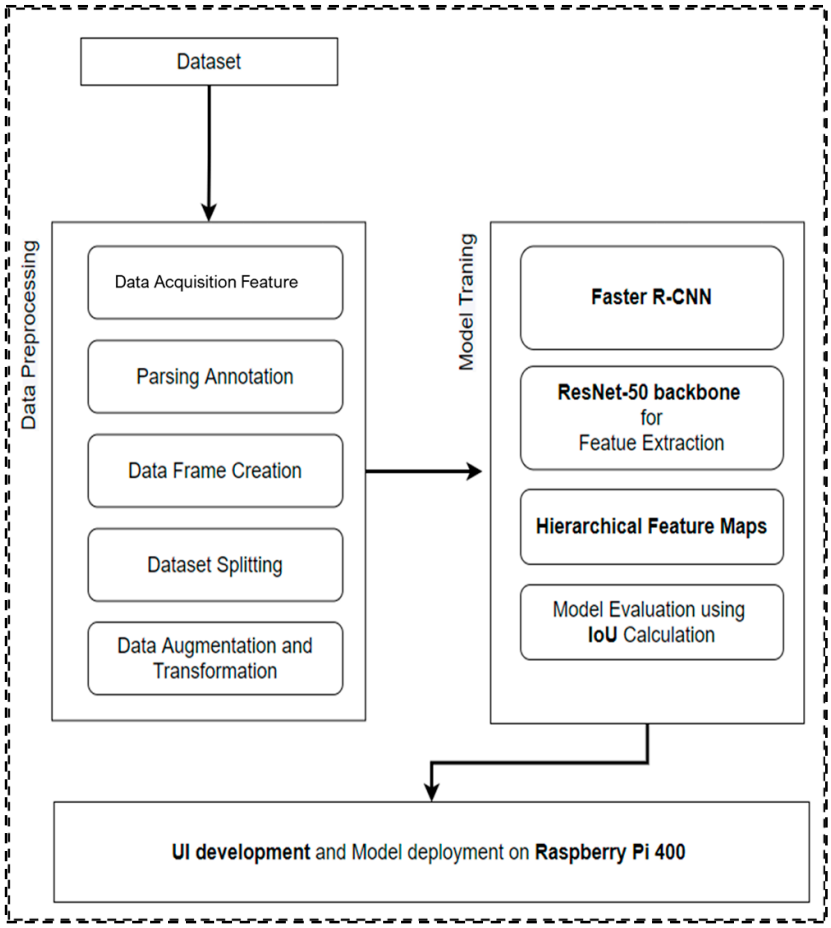


Figure 2. The overall workflow of the proposed automated PCB detection model.

The PCB dataset from Kaggle [27] contains 1386 images of PCBs with several types of defects, such as MHs (115 Images), MBs (115 Images), OCs (116 Images), Shorts (116 Images), Spurs (115 Images), and SCs (116 Images), as shown in Figure 3. These images form the basis of training and validation for the Faster R-CNN model with a backbone of ResNet-50 in defect detection [28]. The main purpose of the PCB defect detection system is to conduct complete and accurate defect detection in Printed Circuit Boards by using an advanced deep learning methodology. In this proposed system, a detection network called Faster R-CNN is trained to identify six different classes of defects. This involves selecting the right model with the right layer configuration for the proposed project, building an algorithm for detection, training the model, and optimizing performance parameters in defect detection, either in real time or near-to-real time.

3.1. Experimental Setup

1. Data Acquisition—The dataset is structured in such a way that every image can be linked to an XML file. These XML files include labels and bounding box coordi-

nates, which specify the location and type of each defect, such as "Missing_hole" or "Mouse_bite". This information is key in training a model to be able to detect both the position and class of each defect [29]. Each image I in the dataset is associated with a set of defects [30], each represented by a bounding box Bi (1), where

$$Bi = (x_{min}, y_{min}, x_{max}, y_{max}, C) \quad (1)$$

x_{min}, y_{min} are the coordinates of the top-left corner of the bounding box. x_{max}, y_{max} are the coordinates of the bottom-right corner. C represents the class label of the defect.

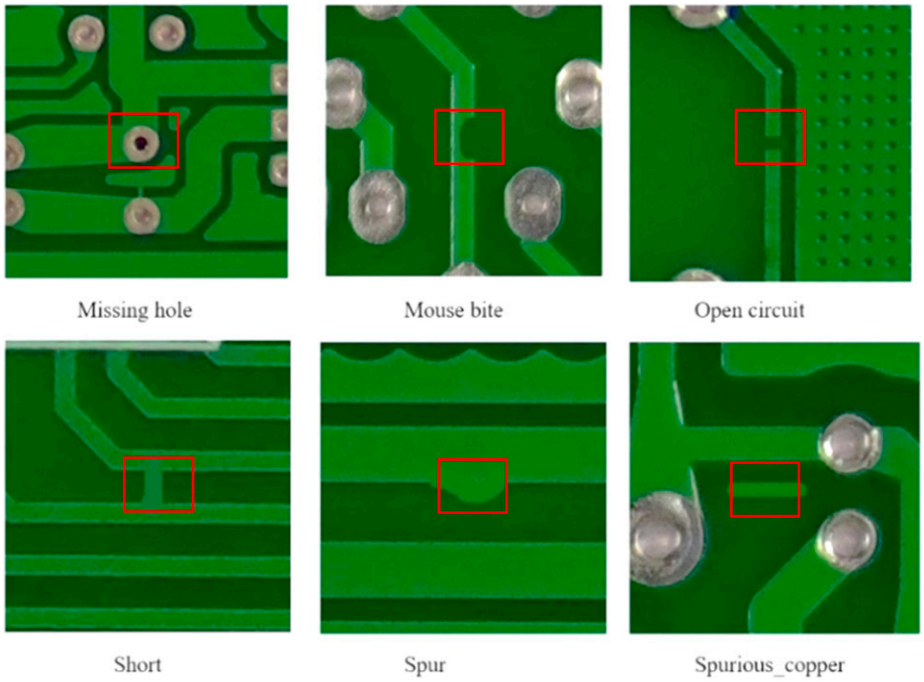


Figure 3. Common types of PCB surface defects. (Red bounding boxes indicate ground-truth defect regions annotated).

Given the XML annotation files, the image dimensions W and H are extracted, such that

$$W = width\ of\ the\ image, H = height\ of\ the\ image \quad (2)$$

For each detected defect, the normalized bounding box [31] coordinates can be computed as shown in Equation (3):

$$x\tilde{}_{min} = x_{min}/W, y\tilde{}_{min} = y_{min}/H, x\tilde{}_{min} = x_{max}/w, y\tilde{}_{max} = y_{max}/H \quad (3)$$

where $x\sim$, $y\sim$ represent the normalized coordinates, ensuring that all bounding boxes are within the range [0, 1]. Normalization is essential for ensuring scale invariance and stable convergence during model training. Without normalization, bounding box coordinates vary significantly based on image resolution, causing the model to learn absolute pixel values, rather than relative positions. This can lead to unstable optimization due to larger gradient updates. By scaling bounding box values between 0 and 1, normalization ensures compatibility across different input resolutions, which is crucial for models like Faster R-CNN with a ResNet-50 backbone. Without normalization, the model may struggle to generalize, as bounding box values become resolution-dependent. With normalization, the model learns spatial relationships independently of input size, improving robustness and performance across PCB images of varying resolutions.

The width and height of each bounding box can be derived as shown in Equation (4):

$$wi = x_{max} - x_{min}, hi = y_{max} - y_{min} \quad (4)$$

The dataset D can then be expressed as a collection of labelled bounding boxes (5):

$$D = \{(Ik, Bk1, Bk2, \ldots, Bkn)/k = 1, 2, \ldots, N\} \quad (5)$$

where N is the total number of images, and each image I_k contains a set of n defects.

2. Parsing Annotations—Each of the XML files is parsed to extract the type of defective label, along with the bounding box coordinates. For instance, there are XML structure fields, x_{min}, y_{min}, x_{max}, and y_{max}, that define the corners of each bounding box. These are compatible with machine learning frameworks [32]. Parsing annotation involves reading annotation files and extracting required information, such as the file path, coordinates, and class labels. Then, transformations are used to convert the raw annotation data into the required format for Faster R-CNN implementation in PyTorch 1.12.1. For example, bounding boxes are converted to tensors. Class labels are mapped to integer symbols. In the handling of edge cases, robustness is ensured by addressing incomplete or inaccurate annotations, such as missing bounding boxes or out-of-bound coordinates. For reading annotation files, each annotation file A_k corresponds to an image I_k and contains structured data, including bounding box coordinates and class labels [33]. The annotation file provides the following information:

$$A_k = \{(Ci, x_{min}, i, y_{min}, i, x_{max}, i, y_{max}, i)/i = 1, 2, \ldots, nk\} \quad (6)$$

where Ci is the class label of I, the defect in image Ik; (x_{min}, I, y_{min}, I) and (x_{max}, i, y_{max}, i) are the bounding box coordinates; and nk represents the number of defects in image Ik. The bounding box coordinates are converted into tensor format, as follows:

$$Bi = [x_{min}, I\, y_{min}, I\, x_{max}, I\, y_{max}, i] \quad (7)$$

where Bi is the bounding box tensor. The class labels are mapped to integer representations [34].

This format facilitates easy analysis, transformations, and splitting into training and testing datasets for effective model evaluation. The dataset is split in an 80-20 ratio, with 80% of the images allocated to the training set and 20% to the test set. This split ensures that the model is trained on a significant portion of the data, while being evaluated based on unseen data to assess its performance.

3.2. Data Augmentation and Transformation

The mathematical representation of data augmentation is as follows. Let I be an input image with dimensions (W, H), (8) where W is the width and H is the height [35]. A set of augmentation functions T is applied to I, generating a transformed image I:

$$\mathrm{I}' = \mathrm{T(I)I} \quad (8)$$

where T is a combination of one or more transformations from the set {T1, T2, . . ., Tn}. Data augmentation artificially inflates the size of a dataset by applying random transformations to the images. This helps to avoid overfitting of the network [36].

- Geometric Transformations

Rotation—Rotating the image by an angle θ\(in degrees) transforms the pixel coordinates (x, y) to new coordinates [37].

$$(x', y'): x' = xcos\theta - ysin\theta, y' = xsin\theta + ycos\theta x \quad (9)$$

Scaling—According to Equation (10), resizing the image by a scaling factor ***s*** modifies the pixel coordinates (x,y) transferred to (x′,y′) [38], such that

$$x' = s \cdot x, y' = s \cdot y \quad (10)$$

Translation: The image is shifted by (Δx, Δy) [39]:

$$x' = x + \Delta x, y' = y + \Delta y \quad (11)$$

Flipping—Horizontal or vertical flipping of an image negates its respective coordinate [40]:

$$x' = W - x(horizontal\ flip), y' = H - y(vertical\ flip) \quad (12)$$

These are rotations, flips, and scaling, examples of transformations that are applied to images. The changes simulate real-world scenarios, such as cases where a PCB is likely to be viewed at different angles or under various light conditions. Through the model, we learned to enable the recognition of defects from different points of view by diversifying the dataset perspectives.

Tensor Conversion—The images and bounding box coordinates are converted into tensors, a format required by PyTorch for the model input. Tensors are optimized to run matrix operations on the GPU, which is suitable for training deep learning models efficiently. Image-to-Tensor Conversion: Each image I is originally a matrix of pixel values with dimensions (H, W, C), where H is height, W is width, and C represents colour channels. It is converted into a PyTorch tensor, as follows [41]:

$$I_{tensor} = To\ Tensor\ (I) \quad (13)$$

where the pixel values are normalized between 0 and 1 for stability in training.

3.3. Model Definition (Faster R-CNN)

There are two main variants of Faster R-CNN. A Region Proposal Network is shown in Figure 4, which suggests areas in the image where objects might exist. It is used to locate regions and a classifier to label those regions. Understanding Faster R-CNN—Faster R-CNN is a two-stage object detector. The first stage uses a Region Proposal Network (RPN) to propose regions in the image that are likely to contain objects. The second stage processes these proposals, classifying each region and refining the bounding box coordinates for accurate localization. The RPN is trained using Anchor classification.

$$\text{Lcls} = -1/\text{Ncls}\sum_{i=0}^{\text{Ncls}} \sum_{c=1}^{c} yiclog(pic) \quad (14)$$

For each anchor ai, the model calculates the probability of each class c (including the background class), where P_i^c is the predicted probability of anchor ai being in class c, and Y_i^c is the ground truth label for anchor ai for class c (either 0 or 1) [42].

Pre-trained Model Initialization—A Faster R-CNN model pre-trained on the dataset is used for time efficiency and improved performance. The pre-trained model is shown in Figure 5. It is a strong base, since it has already learned essential image features like edges, textures, and shapes, which are common across different tasks.

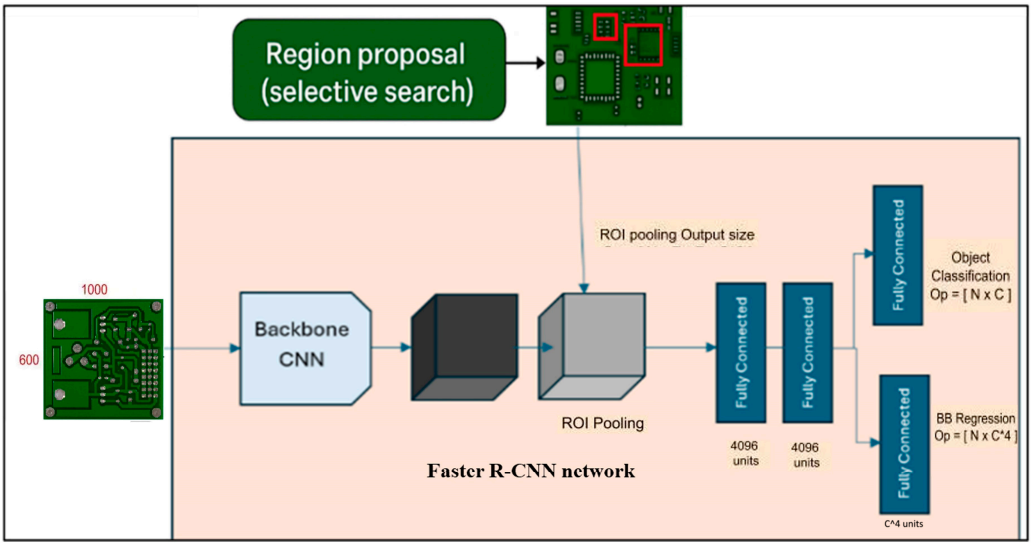


Figure 4. Architectural components of R-CNN for object detection. (The red boxes in the image represent Region Proposals generated by the Selective Search algorithm).

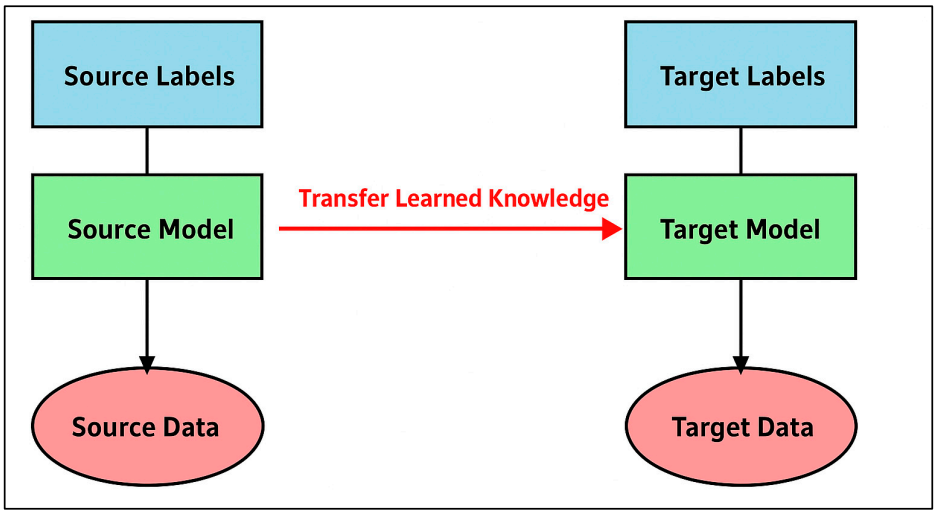


Figure 5. The pre-trained model: leveraging source data and labels for target task optimization.

The following Algorithm 1 augmentation pipeline enhances the robustness of the model by applying various transformations to images and adjusting their bounding box coordinates accordingly.

Algorithm 1: Data Augmentation	
step 1	Random Rotation. Rotate the image by an angle θ. Adjust bounding box coordinates using rotation transformation.
step 2	Flipping. Perform a horizontal or vertical flip. Modify bounding box coordinates accordingly.
step 3	Resizing. Resize Scaling & the image by a scaling factor. Adjust bounding box positions proportionally.
step 4	Visualization Augmentation.
step 5	Label Encoding.—The defect class labels are converted to numerical representations using a **LabelEncoder: L = label_encoder. transform (C)**, where C is the categorical label set.

The following training pipeline in Algorithm 2 outlines the key steps for training a deep learning model efficiently.

Algorithm 2: A Robust Training Framework for High-Accuracy PCB Defect Detection Using Faster R-CNN and ResNet-50

Input:
Training dataset D_{train} (80% of PCB images)
Validation dataset D_{val} (20% of PCB images)
Pretrained Faster R-CNN (ResNet-50 backbone)
Hyperparameters: lr = 0.0001, batch_size = 8, epochs = *N*
Output:
Trained model M∗ with optimized weights
Training/validation metrics (loss, mAP, IoU)
Initialize M←FasterRCNN_ResNet50()
Configure Adam optimizer (weight decay = 0.0005)
Set StepLR scheduler SS (step size = 3, γ = 0.1)
Allocate device: GPU (Tesla T4) if available, else CPU
for epoch ∈{1,. . .,N}do
    M.train()M.train() ->*Switch to training mode*
    for batch (*Xb*,*yb*)∈DataLoader(Dtrain) do
    Xb,*yb*←*Xb*.to(device),*yb*.to(device)
    L←M(*Xb*,*yb*) ->*Forward pass*
     O.zero_grad()
     L.backward() ->*Backpropagate*
    O.step() ->*Update weights*
     end for
    M.eval() ->*Switch to evaluation mode*
for batch (*Xb*,*yb*)∈DataLoader(Dval) do
IoU←box_iou(M(*Xb*),*yb*) -> *Localization accuracy*
mAP←compute_mAP(M(*Xb*),*yb*)
     end for
     S.step() -> *Adjust learning rate*
     Save M∗ if val_loss improves
end for
return M∗, metrics

This structured training procedure ensures that the model converges effectively while maintaining robustness in defect detection. The combination of data augmentation and optimized training strategies improves the overall accuracy and reliability of the model in real-world PCB manufacturing conditions.

3.4. RCNN Resnet Model Training Architecture

Optimized Training Configuration for Efficient Model Convergence.

Key components are thoroughly configured to ensure efficient and stable training, including the following:

Device Allocation—With deep learning being a computationally intensive task, training is offered on a PyTorch, using an NVIDIA GPU (Tesla T4) for computational power. Optimizer Configuration—The Adam optimizer (lr = 0.0001, weight_decay = 0.0005) is used for adaptive parameter updates. The optimizer can converge efficiently by utilizing its adaptive learning rate mechanism [43]. Learning Rate Scheduler—This defines the

point at which the rate (step_size = 3, gamma = 0.1) of learning changes. Additionally, the rate of learning decreases as the training becomes more intensive. The learning rate scheduler helps to tune the model. This structured training procedure ensures that the model converges effectively, while maintaining robustness in defect detection. Through the incorporation of data augmentation into the model, this method is very helpful for avoiding overfitting [44]. The forward pass in Faster R-CNN predicts bounding boxes and class labels for each proposed region, with the total loss being the sum of classification and regression losses. The model leverages a ResNet-50 backbone for PCB defect detection, extracting hierarchical feature maps that emphasize critical regions. ResNet-50's 50-layer architecture captures multi-level features, from low-level edges and textures to high-level object shapes and patterns. As images pass through convolutional layers, these refined feature maps enhance defect localization and classification, ensuring precise and robust PCB defect detection. The lower layers capture basic features (edges, textures, simple shapes). Higher layers detected more complex patterns, helping to identify defect-related features such as cracks, missing components, or Short Circuits.

The formula for the output feature map is as follows:

$$F = f(I, \theta) \tag{15}$$

The input PCB image I is processed through the ResNet-50 model, where θ represents the learned weights of the network. The convolutional operations, denoted by f, extract hierarchical features from the image, enabling robust representation of defect patterns. These features are then utilized by the Region Proposal Network (RPN) to identify regions of interest (ROIs) that are likely to contain defects, forming the foundation for accurate detection and classification. ROI Alignment and Classification—The ROIs are resized and classified into defect categories. Each ROI's bounding box is further refined using bounding box regression.

Loss Calculation in Faster R-CNN Model:

Objectless Loss—This determines whether an ROI contains a defect (binary classification) [5].

$$L\,objectness = -1/Npos + \sum_{i}^{\cdot}(y_i \log(\hat{y}_i) + (1 - yi) log(1 - \hat{y}_i)) \tag{16}$$

y_i is the ground truth label (1 for foreground, 0 for background). $\hat{y}_i$ is the predicted probability of an object being a foreground object. $Npos$ is the number of positive (foreground) samples.

Bounding Box Regression Loss—This measures the error between predicted and ground truth bounding box coordinates using Smooth L1 loss. $\boldsymbol{t_i}$ is the predicted bounding box coordinates, $\boldsymbol{t_i^*}$ is the ground truth bounding box coordinates, and Nreg is the number of bounding box regression samples [45].

$$\boldsymbol{L}_{\text{reg}}\left(\boldsymbol{t}, \boldsymbol{t}^*\right) = \frac{1}{\boldsymbol{N}_{\text{reg}}} \sum_{\boldsymbol{i}} \text{smooth}_{\boldsymbol{L}_1}\left(\boldsymbol{t_i} - \boldsymbol{t_i^*}\right) \tag{17}$$

ROI Head Loss Components (Classification Loss)—Cross-entropy loss assigns defect classes to each ROI [46].

$$\boldsymbol{L}_{\text{class}} = -\frac{1}{\boldsymbol{N}} \sum_{\boldsymbol{i}} \sum_{\boldsymbol{c}} \boldsymbol{y_{i,c}} \textbf{log}(\boldsymbol{P}(\boldsymbol{c_i})) \tag{18}$$

The ROI Head Classification Loss quantifies what is effective for the model, and assigns defect classes to detect regions, resulting in the system's reliability in PCB quality control. Its optimization is essential for the high performance achieved in our model results.

The total loss combines all components, according to Equation (19):

$$L_{\text{total}} = L_{\text{objectness}} + L_{\text{RPN-reg}} + L_{\text{ROI-class}} + L_{\text{ROI-reg}} \quad (19)$$

This unified loss enables training to be brought to an end, while handling class imbalance through normalization by positive sample counts (N_{pos}). The implementation leverages PyTorch's built-in loss functions with default reduction strategies, ensuring numerical stability during optimization [15]. The Following Table 2 summarizes the core computational stages of the Faster R-CNN pipeline adapted for PCB defect detection.

Table 2. Main formulas and processes used in Faster R-CNN approach for defect detection.

Step	Description	Key Formula(s)
1. Region Proposal Network (RPN)	Generates regional proposals, predicting object probability and refining bounding boxes. Includes Anchor Classification Loss, Bounding Box Regression Loss, and total RPN loss.	(a) $Lcls(pi, pi*) = -Ncls1\sum[pi * \mathbf{log}(pi) + (1 - pi*)\mathbf{log}(1 - pi)]$ (b) L_{reg} (ti,ti∗) = N_{reg} 1∑[pi∗ ·smoothL1 (ti − ti∗)] (c) $smooth_{L1}$ (x) = {0.5x^2|x|−0.5 if |x| <1otherwise
2. ROI Pooling	Extracts fixed-size feature maps from proposed regions using max pooling [46].	ROI Feature Map(i,j)=(x,y)∈bin(i,j)max f(x,y)
3. Object Classification and Bounding Box Refinement	Classifies proposals into categories and refines bounding box coordinates using classification and regression loss functions.	(a) L_{cls} = −N1 i∑ c∑ yi,c log(pi,c) (b) L_{reg} = N1 i∑ smoothL1 (ti − ti∗) (c) $smooth_{L1}$ (x)={0.5x2|x| − 0.5 if |x| <1otherwise
4. Pre-trained ModelInitialization	Initializes model weights pre-trained on ImageNet for improved feature learning.	W_{init} = arg_Wmax ∑logP(y|x,W)
5. Customization for PCB Defects	Customizes the model's output layer to detect six defect categories in PCB images.	$P(ck \mid x) = \sum j \, \mathbf{exp}(zj)\mathbf{exp}(zk)$

Validation and Evaluation: Validation measures the model's performance on the test set, providing insight into how well the model will perform on new data. Model Evaluation: In validation mode, the model's weights are not updated. Instead, it generates predictions based on the test images (20%) to see how well it has been learned to detect defects.

Intersection over Union (IoU) Calculation—IoU is a metric that measures the predicted bounding box that matches the true bounding box, as shown in Figure 6. IoU indicates full overlaps between two boxes, while lower values indicate a better predicted box. The IoU is calculated per defect, and at the end, intermediate accuracy is presented as an averaged value over the whole test dataset [47].

The performance of the model was checked on the validation set to find out how comprehensive the model would be. For the quantitative metric, IoU was used for the overlap of predicted and ground truth bounding boxes [48]. IOU assesses the localization accuracy of PCB defect detection, with the Faster R-CNN model using it to classify region

proposals [49]. While effective, IoU has limitations in handling nested/partial boxes, which may explain some low scores in our validation [50]. It is defined as follows:

$$IoU = A_{pred} \cap A_{gt} / A_{pred} \cup A_{gt} \quad (20)$$

- A_{pred} is the predicted bounding box. A_{gt} is the ground truth bounding box. $\cap$ Represents the intersection area between the predicted and ground truth bounding boxes. $\cup$ represents the union area of both bounding boxes.

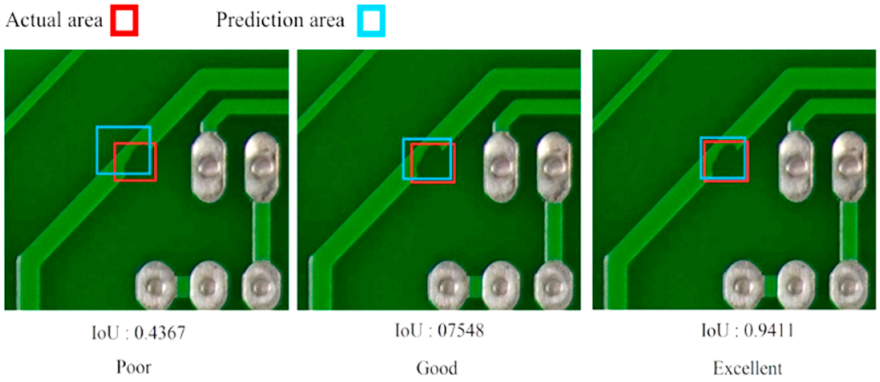


Figure 6. The IOU equation—measures how much the predicted bounding box overlaps with the true bounding box.

IoU Computation from Model Training

From the validation phase of the Faster R-CNN model, IoU values are computed at each epoch. The mean IoU across all epochs is 0.722, which indicates that, on average, the predicted bounding boxes overlap 72.2% with the ground truth bounding boxes. A higher IoU value (closer to 1) suggests better localization accuracy, whereas lower values indicate discrepancies in bounding box predictions.

3.5. Inference and Deployment/Software Setup

Image Loading and Preprocessing

For real-time or batch processing, the application needs to handle incoming PCB images consistently. Each image is loaded and prepared by applying the same preprocessing steps used during the model training phase, to ensure a consistent input format. Preprocessing includes the following steps. Resizing: Scaling images to the same dimensions used during training, so the model can accurately interpret them. Normalization: Adjusting pixel values to the same range as in the training data, to avoid inconsistencies that could lead to inaccurate predictions. Format Conversion: Ensuring images are in the appropriate colour format (e.g., RGB), so they are compatible with the model. The application, built in PyQt5 mode, includes an image and video upload feature that allows users to select or drag and drop an image into the interface. Once an image is loaded, it undergoes these preprocessing steps before being supplied to the model for prediction.

Bounding Box Prediction

After preprocessing, the image is passed through the trained Faster R-CNN model, which identifies areas that are likely to contain defects. The following methods are the model outputs.

Bounding Boxes: These rectangles surround each detected defect, specifying its location within the image. Confidence Scores: Each bounding box is assigned a confidence score that represents the model's confidence in the defect classification. Higher confidence scores indicate more reliable predictions. The model also provides a predicted class for

each bounding box shown in Figure 7, indicating the specific type of defect (e.g., Missing Hole, Mouse Bite). PyQt5's integration with the model allows this entire prediction process to happen effortlessly in the background, with the output prepared for visualization in real time. (The implemented code has been uploaded to GitHub).

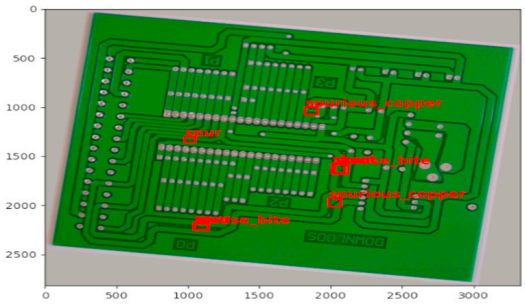


Figure 7. Bounding box prediction for rotation image.

Defect Visualization

The visualization of defects in the image is a crucial part of the deployment process, as it allows users to interpret the model's predictions directly. The bounding boxes are colour-coded based on defect type, with each defect assigned a unique colour to make it easy to distinguish between different issues. For example, Missing Holes might appear in red, while Open Circuits are highlighted in blue, and so forth. Using PyQt5, we have designed a canvas, shown in Figure 8, which can display the original image with these overlays, providing clear and interactive feedback for the user. This approach involves the following processes.

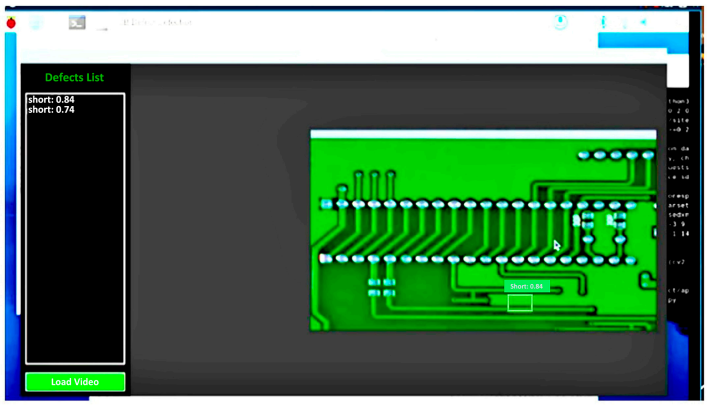


Figure 8. Defect visualization: PyQt5 interface for prediction.

Drawing Bounding Boxes: Bounding boxes are drawn over each detected defect. PyQt5's QPainter class allows the user to render these boxes with different colours and thicknesses, ensuring that each defect type is easily identifiable. Displaying Confidence Scores: Text labels can be added alongside each bounding box to display the confidence score. This helps users to assess the reliability of each prediction. Interactive Controls: The PyQt5 interface includes controls to zoom in on specific defects, toggle bounding boxes on or off, and adjust visualization settings. These features allow users to explore the model's predictions in greater detail and take screenshots for record-keeping or further analysis.

Building the Application with Python and PyQt5

By leveraging Python 3.11 and PyQt5, we have built an efficient GUI application that provides a complete workflow for PCB defect detection. Here, we take a closer look at how PyQt5 supports each part of the application, as illustrated in Figure 9.

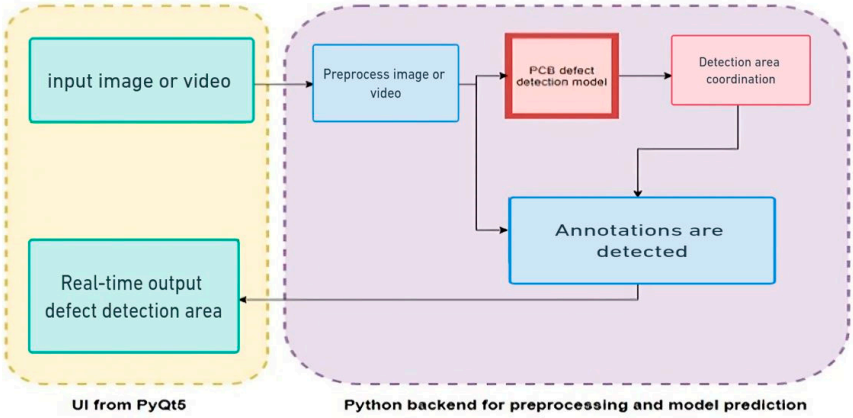


Figure 9. Deployment of trained model with application.

File Upload and Image Loading—PyQt5's Q File Dialog allows users to easily upload PCB images. When a user selects an image, the interface loads it and performs preprocessing, before passing it on to the model.

Table 3 provides a comprehensive breakdown of the PCB defect detection system's technical components, detailing current implementations, key technical specifications, and planned developments.

Table 3. PCB defect detection system architecture: current implementation and future roadmap.

Component	Implementation	Technical Details	Future Enhancements
Detection Model	Faster R-CNN via Roboflow Inference SDK	Model ID: pcb-defect-detection-9ewqw/1 Input size: Dynamic resizing to 640 × 360	Integration of geometric validation (circularity/aspect ratio checks)
Preprocessing	On-the-fly frame processing	- Frame-skipping (60-frame interval) - RGB conversion - Dynamic normalization	Add CLAHE contrast enhancement Non-local means denoising
Hardware Interface	PyQt5 GUI with dual capture modes	- Live camera feed (OpenCV) - Video file support Snapshot capability	Multi-camera synchronization GPIO triggers for conveyor systems
Data Management	CSV-based logging system	Timestamped defect records Automatic image archiving in images folder	SQLite integration Cloud synchronization
Visualization	Matplotlib/PyQt5 dashboard	Temporal defect trends Interactive date filtering Defect frequency histograms	Real-time defect heatmaps Statistical process control charts
Performance Optimization	Frame-skipping algorithm	Processes every 60th frame (adjustable) Maintains 30 ms refresh rate	Hardware acceleration (CUDA) TensorRT optimization
Industrial Adaptations	Confidence-based filtering	Thresholding in display output Defect clustering prevention	Automated alert system OEE (Overall Equipment Effectiveness) integration

Real-Time Inference and Display—Once the image is preprocessed, it is sent to the Faster R-CNN model, which performs inference and returns bounding box predictions,

as shown in Figure 10. The application's main window displays the original image with colour-coded bounding boxes for easy interpretation. Zoom and Pan Controls—With PyQt5's widgets, users can zoom in on specific parts of the image to closely examine detected defects. Exporting Results—PyQt5's Q Image class can be used to save the image with bounding box overlays, providing users with a visual record of detected defects for quality control. Also, when this proposed model is deployed into a Raspberry Pi device, the model can be set up with an online server. The reason for this is that the processing capacity of the Raspberry Pi device makes it difficult to operate the model. After it is placed on the server, it manages to respond quickly. Starting with data collection and preparation, data transformations are applied, a deep learning model is initialized and customized, and then one can proceed through training, validation, and deployment. Each step contributes to building a robust model that can detect and classify PCB defects accurately. The final deployment phase allows the model to be used in a real-world environment, where it can provide visual insights and classifications of PCB defects to support quality control in manufacturing.

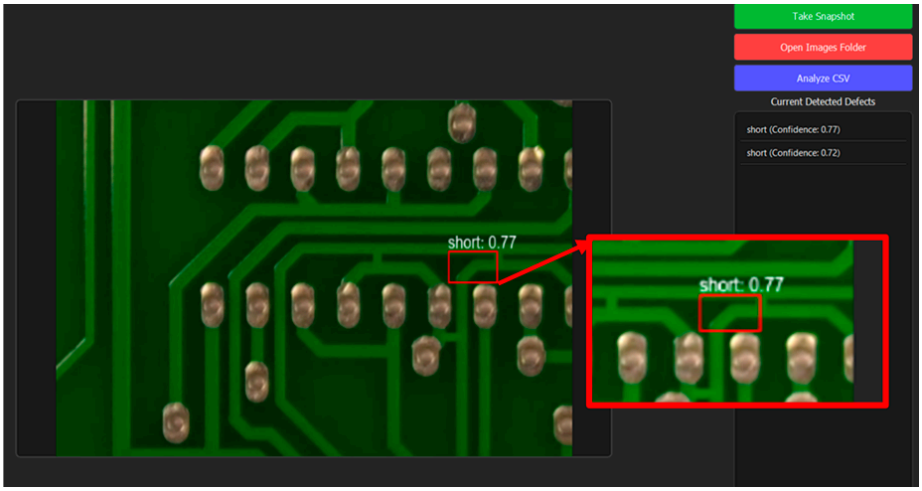


Figure 10. The implemented Python interface for the system.

3.6. Hardware Setup Overview

The automated defect identification system for Printed Circuit Boards (PCBs) engaged a compact yet powerful hardware setup (Table 4) consisting of a Raspberry Pi 400, a Full HD 1080P camera module, and a HDTV video capture device. Experiments were conducted on two hardware setups to evaluate the trade-offs between computational efficiency and model accuracy. The high-performance cloud GPU setup (Google Collab), equipped with an NVIDIA (16 GB VRAM) and a batch size of 8, achieved the best accuracy, due to its ability to handle larger batch sizes and higher parallelization efficiency. In contrast, the local hardware setup (Raspberry Pi 400 + Remote Server), featuring an ARM Cortex-A72 CPU (Quad-Core 1.8 GHz) and 4 GB LPDDR4 RAM, faced significant limitations, including a reduced batch size of 2 and longer inference times per image. Due to the Raspberry Pi's limited memory and computational power, full on-device training was not feasible. Instead, the model was deployed on a remote server, with inference performed on the Raspberry Pi. This approach strikes a balance between deployment feasibility and real-time defect detection speed, making it suitable for industrial applications. The following Figure 11 illustrates the currently implemented module of the automated defect identification system in Printed Circuit Boards using Convolutional Neural Networks.

Table 4. Hardware setup overview.

Hardware Setup	GPU/CPU	RAM	Batch Size	Training Time	Key Observations
Cloud GPU (Google Collab Pro)	NVIDIA Tesla T4	(16 GB VRAM)	8	6 h	The best accuracy was achieved, due to larger batch sizes and high parallelization.
Local (Raspberry Pi 400 + Remote Server)	ARM Cortex-A72 (Quad-Core 1.8 GHz)	4 GB LPDDR4	2	Not feasible (offloaded to server)	Limited memory prevented full on-device training; inference was performed on a remote server for efficiency.

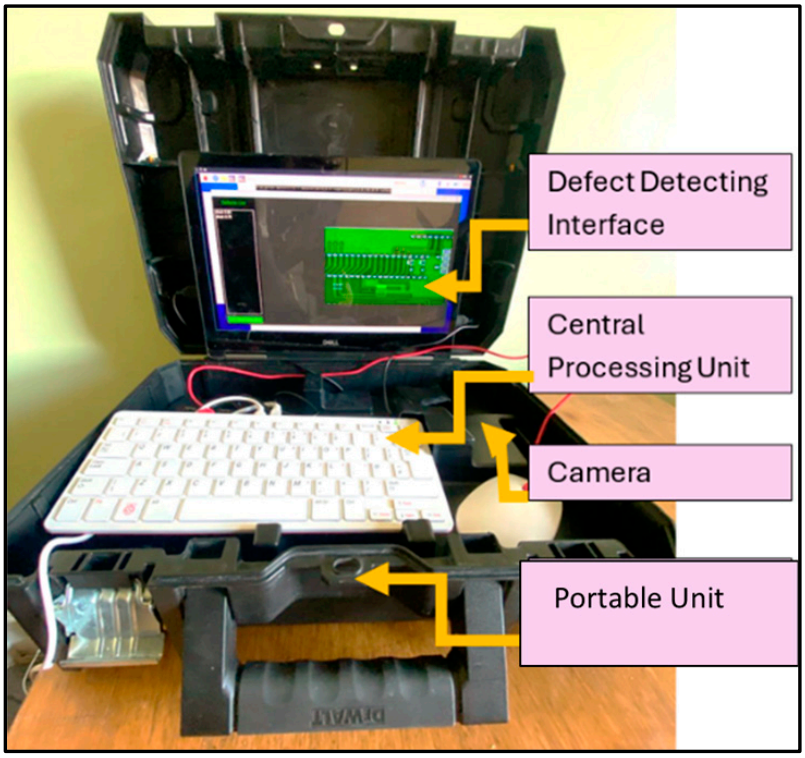


Figure 11. Implemented automated defects identification system in Printed Circuit Boards.

The Full HD 1080P camera module captures high-resolution video at 1920 × 1080, while the HDTV video capture device ensures high-quality video digitization and transfer. Together, these components create a portable and high-performance environment for automated defect identification in PCBs, and provide the necessary hardware foundation for real-time defect detection and analysis in PCBs.

The resolution of input images and the camera's optical zoom capability have strong impacts on the detection of faults in PCBs, particularly for faults at the micro level, such as small cracks or faults in soldering. As shown in Table 5, higher resolutions such as 4K improve the detection of faults that are very small, at the expense of increased computational load and processing time. Optical zoom has the potential to overcome the limitations of the resolution of 1080P by zooming into very small areas of the PCB, but it reduces the field of view, and may require several images to capture the entire board. To overcome these limitations, we propose the use of higher-resolution cameras, post-processing techniques, and training time data augmentation.

Table 5. Impact of image resolution and optical zoom on PCB defect detection.

Aspect	Description	Impact on Defect Detection	Mitigation Strategies
1920 × 1080 (HD) Resolution	Standard resolution is used in many industrial applications.	It may not capture very small defects (e.g., <0.1 mm), unless combined with optical zoom.	Use optical zoom, higher-resolution cameras, or position the camera closer to the PCB [51].
Higher Resolutions (e.g., 4K)	Provides four times the pixel density of HD (3840 × 2160).	Improves detection of small defects, but increases computational load and processing time.	Use powerful hardware (e.g., GPUs) and optimize the model for faster inference.
Focus Capabilities	Ensures clear images of defects, even on uneven PCB surfaces.	Poor focus can lead to blurred images, reducing defect detection accuracy.	Use autofocus cameras or manual focus adjustments during image capture.

4. Results and Discussion

We have developed and evaluated a Faster R-CNN model to identify and classify defects in Printed Circuit Boards (PCBs). Using an annotated dataset with six trained and tested model defect classes, we achieved an overall accuracy of 87% across a test set of 50 samples. The defect classes included common PCB issues, such as Missing Holes, Mouse Bites, Open Circuits, Shorts, Spurs, and Spurious Copper. The model's performance metrics, including the precision, recall, and F1-score for each class, provide valuable insights into the model's efficacy in detecting and differentiating between various defect types. Additionally, loss and accuracy charts from training and validation phases were analyzed to understand model convergence and generalization [52].

(a) Precision

$$\text{Precision}_i = \frac{TP_i}{FP_i + TP_i} \tag{21}$$

The proportion of correctly identified instances of a class out of all the predicted instances of that class.

(b) Recall

$$\text{Recall}_i = \frac{TP_i}{FN_i + TP_i} \tag{22}$$

The proportion of correctly identified instances of a class out of all the actual instances of that class.

(c) F1-Score

$$F1\text{-Score}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{23}$$

The F1-score is the harmonic mean of precision and recall. It provides a single score that balances both precision and recall.

4.1. Detailed Breakdown of Performance Metrics

The confusion matrix shown in Table 6 indicates how the model performed with regard to the six classes, where each row represents an actual class and the columns represent predicted classes. The model had completely fine prediction in most instances, especially for class 3, with which it attained full accuracy. The model had difficulties associated with

class 4, which was misclassified as class 0, class 1, and class 2 in prediction. Similarly, class 5 was wrongly predicted once as class 3, but the model did well in correctly predicting class 5 in eight out of nine instances. The confusion matrix indicates the model's effectiveness, but highlights misclassifications in classes 2 and 5, suggesting the need for refinement, better feature selection, or dataset expansion for improved differentiation and optimization.

Table 6. Confusion matrix (visually represents the performance of a classification model by showing how predictions align with true labels).

True\Predicted	0	1	2	3	4	5
Missing Holes	9	0	0	0	1	0
Open Circuits	0	9	0	0	2	0
Mouse Bite	0	0	7	0	2	0
Short	0	0	0	10	0	1
Spur	0	0	1	0	9	0
Spurious Copper	1	0	0	0	0	8

4.2. Overall Model Performance

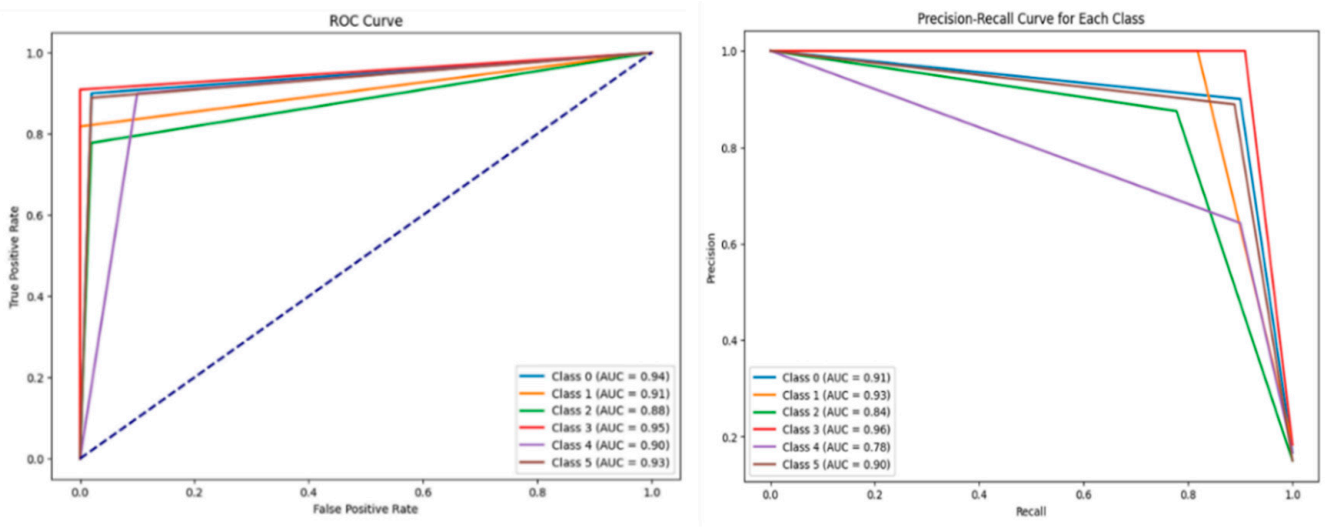
The accuracy of 87% on the test set highlights the model's ability to precisely classify most defects, with success in classes such as Mouse Bites, Open Circuits, Spurs, and Spurious Copper. Nevertheless, the model failed to identify any samples from the Missing Hole class, which suggests that more work is needed to ensure consistent performance across all defect types. The macros of average precision, recall, and F1-score were 0.88, 0.87, and 0.87, respectively.

This indicates moderate performance, with higher scores in classes where the model performs well, and lower scores in underperforming classes, such as Missing Holes. The weighted average metrics of precision of 0.89, recall of 0.87, and F1-score of 0.87 further emphasize the model's satisfactory, though not exceptional, performance. Figure 12 presents the Receiver Operating Characteristic (ROC) and precision–recall (PR) curves for a multi-class classification model, evaluating its performance across different classes. The ROC curve (a) illustrates the trade-off between the true positive rate and the false positive rate, with the Area Under the Curve (AUC) values indicating strong overall performance, ranging from 0.88 to 0.95 across classes. The PR curve (b) assesses precision versus recall, providing insights into class-wise prediction reliability. While most classes exhibit high AUC scores, class 2 and class 4 show relatively low performance, suggesting potential areas for model improvement through enhanced feature selection or dataset augmentation. The model's effectiveness in classes with larger sample sizes also reveal limitations in classes with insufficient samples or higher variability.

4.3. Comparative Analysis

ResNet-50 is the best-performing model, with high precision, recall, and F1-score, making it the most suitable choice for PCB defect detection. Inception and ConvNeXt show weak detection capabilities, struggling to classify defects accurately. This analysis confirms that ResNet-50 is the optimal model for this task, offering robust and accurate defect identification. With respect to defect detection, the current work is different from previous ones. Most earlier systems used traditional image processing methods, such as grayscale conversion, edge detection, thresholding, and contrast enhancement, for the detection of PCB defects. Although these approaches could be acceptable [53], in this section, we evaluate the performance of three models, namely Inception, ConvNeXt, and ResNet-50 (the main model), for defect detection in PCBs, as illustrated in Table 7. The

classification metrics, including precision, recall, F1-score, and accuracy, are analyzed and compared. These models are very powerful, with their best utilization concerning different fields. ResNet-50 is known for its residual connections that facilitate very deep networks, and hence have no vanishing gradient problem during training.



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

Figure 12. (**a**) Receiver Operating Characteristic (ROC) and (**b**) precision–recall (PR) curves for multi-class classification, highlighting the Area Under the Curve (AUC) for each class.

Table 7. A comparison table combining the classification reports for the Inception, ResNet-50, and ConvNeXt models.

Model	Overall Accuracy	Macro Avg Precision	Macro Avg Recall	Macro Avg F1-Score	Weighted Avg Precision	Weighted Avg Recall	Weighted Avg F1-Score
Inception	0.17	0.06	0.17	0.08	0.06	0.17	0.08
ResNet-50	0.87	0.88	0.87	0.87	0.89	0.87	0.87
ConvNeXt	0.17	0.03	0.17	0.05	0.03	0.17	0.05

Thus, ResNet-50 is very effective in image classification and transfer learning tasks. Inception is efficient in multi-scale feature extraction with parallel convolutions, while ConvNeXt is an efficient hybrid of CNNs and transformers that yields the best performance on large datasets and high-resolution input, but involves increased complexity in training. Among the models, ResNet-50 mainly stands out for its robustness and depth, featuring as the top performer in many vision tasks.

The ResNet-50 backbone outperforms both the Inception and ConvNeXt models by a significant margin. Its superior accuracy and F1-score make it the ideal choice for automated defect detection in PCBs. The results demonstrate that ResNet-50 can reliably identify and classify various defect types, contributing significantly to automating PCB manufacturing quality control processes. For the ConvNeXt and Inception models, the performance is poor and unacceptable for classification. According to Figures 13 and 14, these models needs better hyperparameter tuning, architectural adjustments, or more training in further analysis.

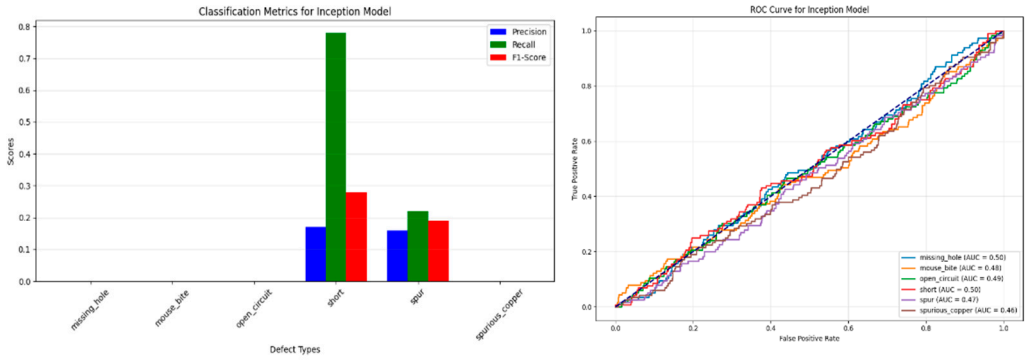


Figure 13. Comparison of classification metrics for Inception (1). The metrics evaluated are precision, recall, and F1-score, along with ROC curves for each model.

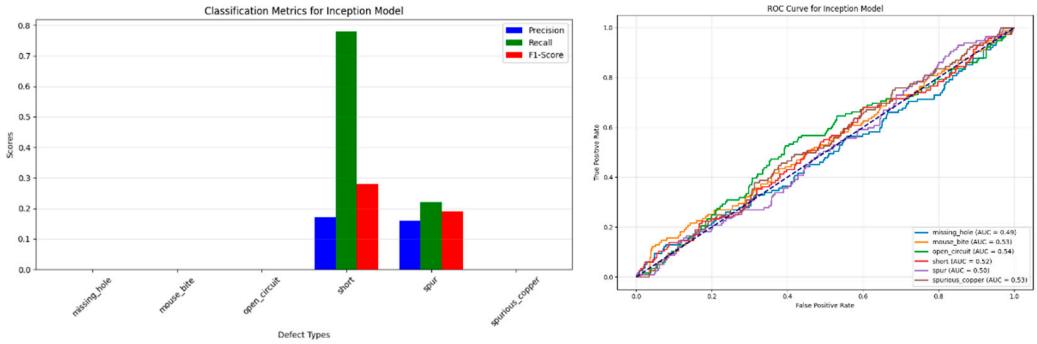


Figure 14. Comparison of classification metrics for ConvNeXt (2). The metrics evaluated are precision, recall, and F1-score, along with ROC curves for each model.

4.4. Outcome

Accordingly, the following Figure 15 shows outputs related to each defect that we obtained with real-time defect detection. The following test sample shows the confidence level and the type of defects. It evaluates the classification accuracy for defects such as Missing Holes, Mouse Bites, Open Circuits, Shorts, Spurs, and Spurious Copper. Finally, a GUI application was designed for the implemented model. Users can upload any kind of photo (such as rotation or portrait, shown in Figure 16) or video for input to the application, and the captured defects will be displayed in real time. The output can be saved as an image. Also, the detected defects can be saved for future study.

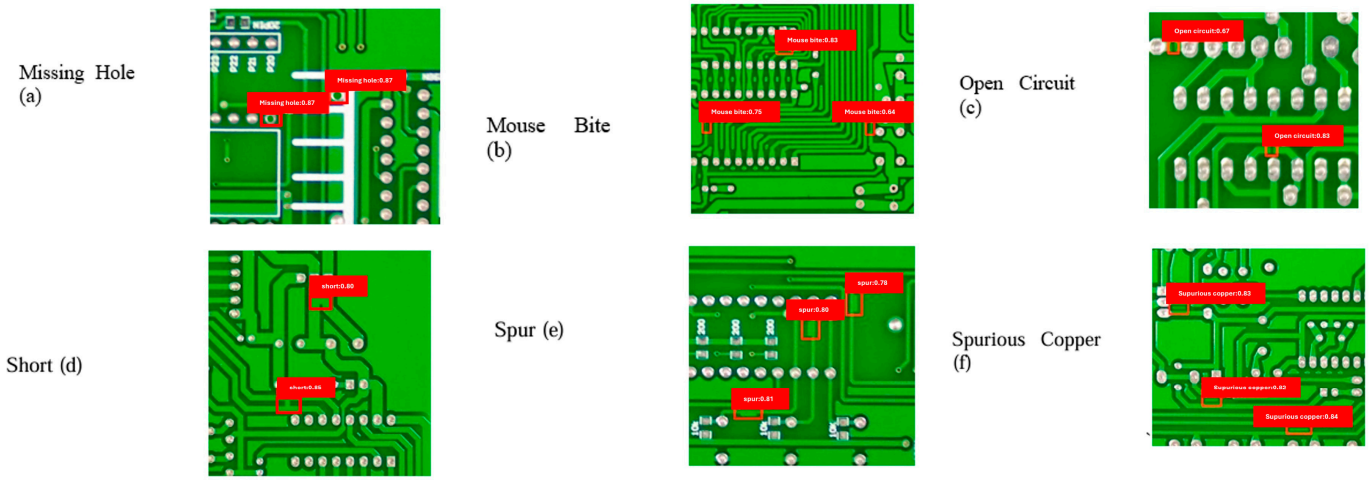


Figure 15. Predicted defect with accuracy confidence score for (**a**–**f**).

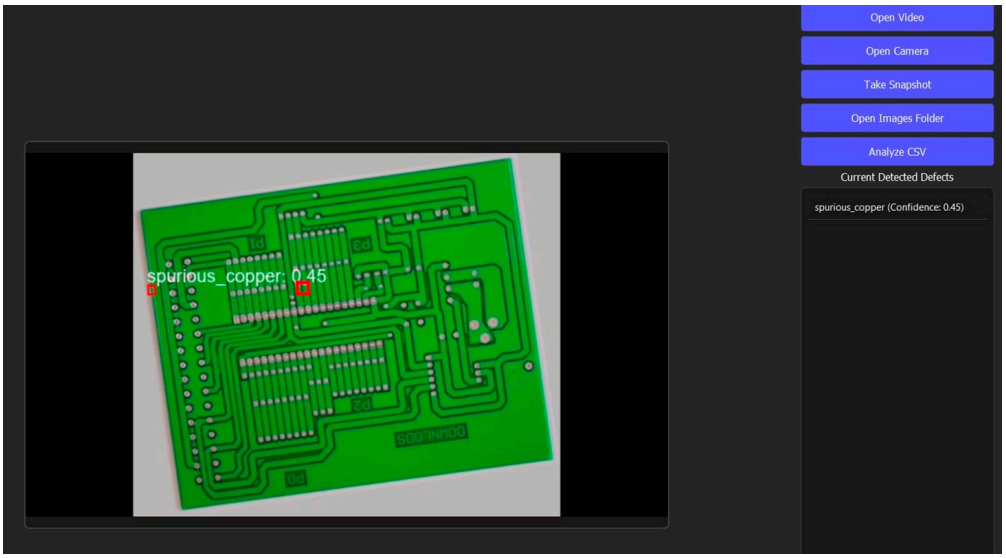


Figure 16. Rotated image of defect detection sample.

The application continuously analyzes video frames, detects defects, and displays results in real time. According to Figure 17, users can identify the frequency of different defect types. These data can be used to identify the most common defect types and prioritize corrective actions to improve product quality.

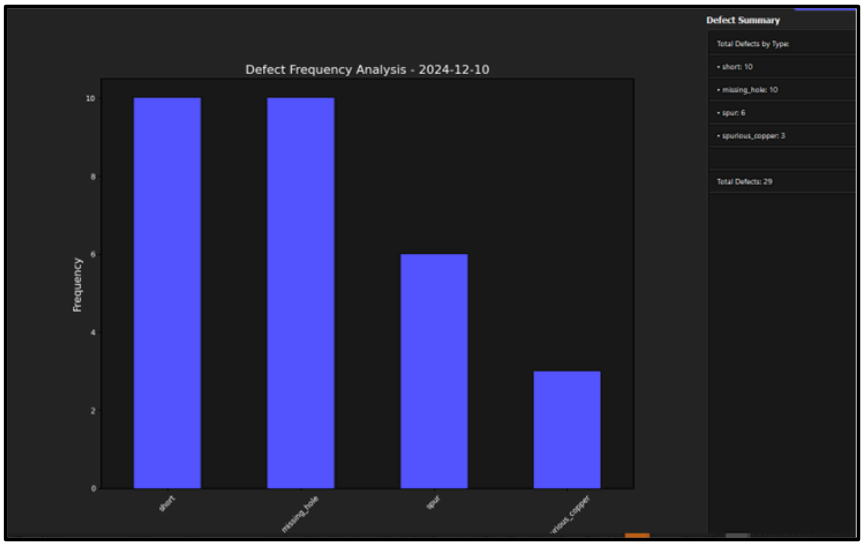


Figure 17. Defect frequency analysis graph.

5. Conclusions

This Faster R-CNN model demonstrated effective detection capabilities for the several defect types on which it was trained. This proposed system achieved an overall accuracy of 87%, with perfect classification for the classes of Mouse Bites, Open Circuits, Spurs, and Spurious Copper. The proposed model shows promise for use in PCB defect detection applications. However, the performance with regard to Missing Holes suggests areas that must be improved. The lower performance in Missing Hole detection reveals limitations in handling class-imbalanced datasets. Future research directions will focus on investigating attention mechanisms to improve small-defect detection, exploring few-shot learning approaches for rare defect categories, integrating multimodal data (thermal/X-ray) to expand defect coverage, and optimizing inference speed for high-throughput production

lines. These findings contribute to developing automated, reliable PCB inspection systems that can streamline quality assurance processes in industrial manufacturing.

Author Contributions: K.D.W. contributed to the conceptualization, hardware development, software development (main algorithm and testing algorithm) development, and literature review. R.B. (corresponding author) contributed to the conceptualization, hardware development, software development (main algorithm and testing algorithm) development, and literature review. E.O. contributed to the data collection and documentation, provided a critical review, and edited the manuscript. J.A.A.-A. contributed to the data collection and documentation, provided a critical review, and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was conducted without external funding, utilizing institutional computational resources and open-source tools to ensure cost-effective implementation.

Data Availability Statement: The source code for the automated defect identification system in Printed Circuit Boards using CNNs is available at https://github.com/Denuwanweerakkody/Automated-Defects-Identification-System-in-Printed-Circuit-Boards-Using-CNN-/tree/main (accessed on 15 January 2025).

Conflicts of Interest: There are no financial or professional conflicts of interest that could have influenced the analysis, findings, and conclusions in this manuscript.

References

1. Zhou, J.; Liu, Y.; Zhang, X.; Yang, Z. Multi-view based template matching method for surface defect detection of circuit board. *J. Phys. Conf. Ser.* **2021**, *1983*, 012063. [CrossRef]
2. Piliposyan, G.; Khursheed, S. Computer Vision for Hardware Trojan Detection on a PCB Using Siamese Neural Network. In Proceedings of the 2022 IEEE Physical Assurance and Inspection of Electronics (PAINE), Huntsville, AL, USA, 25–27 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–7.
3. Weiss, E. Revealing Hidden Defects in Electronic Components With an AI-Based Inspection Method: A Corrosion Case Study. *IEEE Trans. Compon. Packag. Manuf. Technol.* **2023**, *13*, 1078–1080. [CrossRef]
4. 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 (CVPR), Columbus, OH, USA, 23–28 June 2014.
5. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Adv. Neural Inf. Process. Syst.* **2015**, *39*, 1–9. [CrossRef] [PubMed]
6. Reswara, E.; Suakanto, S.; Putra, S.A. Comparison of Object Detection Algorithm using YOLO vs Faster R-CNN: A Systematic Literature Review. In Proceedings of the ICBDT 2023: 2023 6th International Conference on Big Data Technologies, Qingdao, China, 22–24 September 2023.
7. Garg, H.; Bhartee, A.K.; Rai, A.; Kumar, M.; Dhakrey, A. A Review of Object Detection Algorithms for Autonomous Vehicles: Trends and Developments. In Proceedings of the 2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 15–16 December 2023.
8. Ezzeddini, L.; Ktari, J.; Frikha, T.; Alsharabi, N.; Alayba, A.; JAlzahrani, A.; Jadi, A.; Alkholidi, A.; Hamam, H. Analysis of the performance of Faster R-CNN and YOLOv8 in detecting fishing vessels and fishes in real time. *PeerJ Comput. Sci.* **2024**, *10*, e203. [CrossRef]
9. Li, T.; Ma, Y.; Endoh, T. A Systematic Study of Tiny YOLO3 Inference: Toward Compact Brainware Processor with Less Memory and Logic Gate. *IEEE Access* **2020**, *8*, 142931–142955. [CrossRef]
10. Vishwakarma, C.; Sonawane, S.; Nikhil, M. Designing a deep learning model to detect objects. *IRE J.* **2020**, *4*, 1–8.
11. Xie, X.; Cheng, G.; Wang, J.; Yao, X.; Han, J. Oriented R-CNN for Object Detection. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 3520–3529.
12. Li, Z.; Peng, C.; Yu, G.; Zhang, X.; Deng, Y.; Sun, J. DetNet: A Backbone network for Object Detection. *arXiv* **2018**, arXiv:1804.06215.
13. Hu, B.; Wang, J. Automated PCB Defect Detection Using Convolutional Neural Networks. *IEEE* **2020**, *8*, 96410–96417.
14. Yu, L.; Zhu, J.; Zhao, Q.; Wang, Z. An Efficient YOLO Algorithm with an Attention Mechanism for Vision-Based Defect Inspection Deployed on FPGA. *Micromachines* **2022**, *13*, 1058. [CrossRef]
15. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

16. Khan, S.; Rahmani, H.; Shah, S.A.A.; Bennamoun, M.; Medioni, G.; Dickinson, S. *A Guide to Convolutional Neural Networks for Computer Vision*, 1st ed.; Springer: Cham, Switzerland, 2018.
17. Warren, S.W.P. McCulloch. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **2017**, *5*, 115–133.
18. Silva, N.K.D. *Printed Circuit Board Defect Detection Using Image Processing*; Minnesota State University: Mankato, MN, USA, 2022.
19. Gabbar, H.A.; Chahid, A.; Khan MJ, A.; Adegboro, O.G.; Samson, M.I. Automated Defect Detection Framework Using Computed Tomography. *Appl. Sci.* **2022**, *12*, 2175. [CrossRef]
20. Zheng, X.; Zheng, S.; Kong, Y.; Chen, J. Recent advances in surface defect inspection of industrial products using deep learning techniques. *Int. J. Adv. Manuf. Technol.* **2021**, *13*, 35–58. [CrossRef]
21. Allen-Zhu, Z.; Li, Y.; Song, Z. A Convergence Theory for Deep Learning via Over-Parameterization. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 242–252.
22. Sugumaran, N.S. Fault diagnosis of visual faults in photovoltaic modules. *Int. J. Green Energy* **2021**, *18*, 37–50.
23. Li, B.; Delpha, C.; Diallo, D.; Migan-Dubois, A. Application of artificial neural networks to photovoltaic fault detection and diagnosis. *Renew. Sustain. Energy Rev.* **2021**, *138*, 110–512. [CrossRef]
24. Cheng, L. Real-time defect detection using Raspberry Pi with OpenCV. *J. Ind. Autom.* **2019**, *12*, 210–218.
25. Mehta, A.; Jain, R. An Analysis of Fabric Defect Detection Techniques for Textile Industry Quality Control. In Proceedings of the 2023 World Conference on Communication & Computing (WCONF), Raipur, India, 14–16 July 2023.
26. Singh, A.R.; Bashford-Rogers, T.; Marnerides, D.; Debattista, K.; Hazra, S. HDR image-based deep learning approach for automatic detection of split defects on sheet metal stamping parts. *Int. J. Adv. Manuf. Technol.* **2023**, *125*, 2393–2408. [CrossRef]
27. Kaggle. Available online: https://www.kaggle.com/datasets/akhatova/pcb-defects/data (accessed on 5 March 2025).
28. Huang, W.; Wei, P. A PCB Dataset for Defects Detection and Classification. Kaggle 2019. Available online: https://www.kaggle.com/datasets/akhatova/pcb-defects (accessed on 15 May 2024).
29. Emilio, M.D.P. Data Acquisition Systems. In *Fundamentals to Applied Design*; Springer: New York, NY, USA, 2013; p. XVII, 135.
30. Kovalenko, S.M.; Kutsenko, O.S.; Kovalenko, S.V.; Kovalenko, A.S. Approach to the Automatic Creation of an Annotated Dataset for the Detection, Localization and Classification of Blood Cells in An Image. *Radio Electron. Comput. Sci. Control* **2024**, 128. [CrossRef]
31. Adams, E.R.; Depoian, A.C.; Kurz, A.G.; Cayce, G.I.; Riley, J.C.; Bailey, C.P.; Guturu, P. Efficient Bounding Box Selection for Object Localization. In Proceedings of the 2023 IEEE 16th Dallas Circuits and Systems Conference (DCAS), Denton, TX, USA, 14–16 April 2023; IEEE: Piscataway, NJ, USA, 2023.
32. Sagae, K.; Davis, E.; Lavie, A.; MacWhinney, B.; Wintner, S. High-accuracy Annotation and Parsing of CHILDES Transcripts. In Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition, Prague, Czech Republic, 29 June 2007; Association for Computational Linguistics: Stroudsburg, PA, USA, 2007; pp. 25–32.
33. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA, 2016.
34. Esiukova, E. From macro to micro: Dataset on plastic contamination along and across a sandy tide-less coast. *Sciencedirect* **2020**, *56*, 111–198.
35. Szeliski, R. *Computer Vision: Algorithms and Applications*; Springer: Berlin/Heidelberg, Germany, 2010.
36. Maharana, B.N.K. Data pre-processing and data augmentation techniques. *Sciencedirect* **2022**, *3*, 91–99. [CrossRef]
37. Kovalenko, S.M.; Kutsenko, O.S.; Kovalenko, S.V.; Kovalenko, A.S. A survey on image registration methods in medical imaging. *Comput. Med. Imaging Graph.* **2021**, *31*, 8.
38. Li, T.; Zuo, R.; Zhao, X.; Zhao, K. A deep learning framework for image super-resolution using scaling and learning-based methods. *IEEE Trans. Image Process.* **2020**, *29*, 5142–5154.
39. Shi, W.; Chen, C.; Li, K.; Xiong, Y.; Cao, X.; Zhou, Z. A robust image translation method for object tracking. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 1045–1055.
40. Ding, Y.; Liu, C.; Zhu, H.; Chen, Q. Data augmentation strategies for convolutional neural networks in object detection. *Int. J. Comput. Vis.* **2021**, *129*, 1–16.
41. Zhang, A.; Lipton, Z.C.; Li, M.; Smola, A.J. *Dive into Deep Learning*; Cambridge University Press: Cambridge, UK, 2021.
42. Kumar, V.; Singh, R.S.; Rambabu, M.; Dua, Y. A novel deep learning approach for multi-class classification of images. *Neurocomputing ACM Comput. Surv.* **2020**, *53*, 1–10.
43. Chen, C.; Carlson, D.; Gan, Z.; Li, C.; Carin, L. Bridging the Gap between Stochastic Gradient MCMC and Stochastic Optimization. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Cadiz, Spain, 9–11 May 2016; Volume 52, pp. 1051–1060.
44. Smith, L.N. Cyclical Learning Rates for Training Neural Networks. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472.

45. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
46. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *42*, 318–327. [CrossRef]
47. Wang, X.; Song, J. ICIoU: Improved Loss Based on complete intersection over union for bounding box regression. *IEEE Access* **2021**, *9*, 105686–105695. [CrossRef]
48. Rahman, M.W. *Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation*; Springer: Berlin/Heidelberg, Germany, 2016.
49. Everingham, M. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2009**, *88*, 303–338. [CrossRef]
50. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 658–666.
51. Zhang, H.; Li, Y.; Sharid Kayes, D.M.; Song, Z.; Wang, Y. Research on PCB defect detection algorithm based on LPCB-YOLO. *Front. Phys.* **2025**, *12*, 1472584. [CrossRef]
52. Goutte, C.; Gaussier, E. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In Proceedings of the European Conference on Information Retrieval, Compostela, Spain, 21–23 March 2005; pp. 345–359.
53. Deng, S.; Deng, L.; Sun, T.; Yu, S.; Wang, L.; Chen, B.; Hu, H.; Xie, Y.; Yin, H.; Xiao, J.; et al. EEDD: Edge-Guided Energy-Based PCB Defect Detection. *Electronics* **2023**, *12*, 2306. [CrossRef]