MACHINE LEARNING

ML-Based Next-Generation Vision System for Bare PCB Defect Detection

R. Harshitha, L. Keerthana, N D. Pannaga, H U. Prajna

Department of Electronics and Communication Engineering, GSSSIETW, Mysuru, India

Published November 2026

Abstract

Surface defects on bare printed circuit boards (PCBs), such as scratches, solder-mask misalignment over pads and mask peeling, affect board reliability, yet manual inspection is slow and varies from one inspector to another. This paper presents a machine-learning-based vision system that combines classical image preprocessing with deep-learning object detection to localise selected surface defects on bare PCBs. A dataset of 344 mobile-phone images covering three PCB layouts was annotated in Label Studio with three classes: Scratch, Mask on Pad (MP) and Mask Peel-Off (MF). Every captured image is first processed with OpenCV through BGR-to-HSV conversion, HSV-based segmentation, erosion and dilation, boundary detection, cropping, perspective correction and resizing to a standard resolution. The standardised board image is then passed to a YOLO26n detector trained for 100 epochs on a laptop CPU at an image size of 1280 × 1280 pixels. A Flask-based web application, PCB Vision, ties the workflow together and reports bounding boxes, class labels, confidence scores, severity labels, geometric measurements, Pareto analysis, process-feedback suggestions, annotated images and PDF inspection reports. In a demonstration inspection the system displayed 98 defect regions (65 MP, 21 MF and 12 Scratch) with a recorded end-to-end processing time of 2.1403 s. The work shows how preprocessing, defect localisation and reporting can be integrated into a single prototype; a quantitative evaluation on the held-out test subset is left for future work.

Keywords: bare printed circuit board; automated optical inspection; computer vision; image preprocessing; HSV segmentation; YOLO26n; deep learning; object detection; PCB defect detection

1 Introduction

1.1 Background

Printed circuit boards (PCBs) provide the mechanical support and electrical interconnections on which almost every electronic product depends. They are found in consumer electronics, automotive systems, communication equipment, industrial controllers, medical devices and embedded systems. Because the board sits at the foundation of the product, its quality has a direct bearing on the reliability of everything mounted on it.

During manufacturing, surface defects can arise from variations in fabrication conditions, solder-mask application, material handling and mechanical contact. Some of these defects only change the appearance of the board, while others interfere with pads and conductive regions. Finding them at a suitable stage of production is therefore a basic requirement of PCB quality control.

Inspection is conventionally carried out either visually by trained personnel or with automated optical inspection (AOI) systems [1]. Manual inspection depends on the experience, concentration and eyesight of the inspector. When many boards have to be examined, it becomes slow, and repeated

inspection tends to introduce inconsistencies in how defects are identified and classified. These drawbacks have encouraged the use of computer-vision systems that analyse PCB images and flag regions likely to contain defects.

1.2 Problem statement

Automated inspection of bare PCBs is made difficult by variations in image acquisition, board orientation, lighting and defect appearance. Images taken with an ordinary camera contain background, rotation, perspective distortion and brightness differences, and supplying such images directly to a detection model makes its input inconsistent. In addition, different defects can look alike, and some occupy only a small part of the board, so the input has to be prepared carefully before defects can be localised and told apart.

The problem addressed here is the development of an integrated vision-based inspection system that can:

- acquire PCB images with a mobile-phone camera;
- identify and extract the actual PCB region from the captured image;
- correct orientation and perspective variations;
- standardise the extracted image for subsequent processing;
- detect and classify selected PCB surface defects;
- display defect locations together with relevant inspection information; and
- generate structured inspection summaries and downloadable reports.

1.3 Motivation

The aim is a practical and accessible inspection solution built from image processing and machine learning. Rather than relying entirely on manual examination, a trained object detector indicates the categories and approximate locations of selected defects. The tool is also meant to be easy to operate and to understand, so the system goes beyond model prediction and includes preprocessing, result visualisation, defect-distribution analysis, timing information and inspection documentation. Using a mobile-phone camera and a laptop for processing keeps the prototype independent of specialised industrial inspection equipment.

1.4 Objectives

The objectives of the work are:

1. to develop an automated vision-based system for detecting defects on bare PCBs;
2. to implement OpenCV-based preprocessing for PCB-region extraction and image standardisation;
3. to train and integrate a YOLO26n detector for Scratch, Mask on Pad and Mask Peel-Off defects;
4. to build a Flask-based web application for uploading PCB images and viewing results;
5. to provide defect counts, confidence scores, severity information, bounding-box coordinates and processing-time measurements; and
6. to generate class-wise defect distributions, Pareto analysis, annotated images and downloadable PDF inspection reports.

1.5 Contributions

The contributions of this work are summarised as follows.

1. A preprocessing pipeline that isolates the PCB from the captured background and, through correction and resizing, gives the detector a consistent representation of the board.
2. YOLO26n-based detection of three selected surface-defect categories.
3. A web-based interface that presents detections, statistics and Pareto-based visualisation in an organised form to support inspection review.
4. Automatic generation of annotated images and PDF inspection records.

2 Literature review

This section follows how PCB inspection techniques have developed, the problems each generation addresses and where the present work fits.

2.1 Traditional image-processing-based inspection

Early PCB inspection relied on image-processing operations such as registration, image differencing, thresholding, segmentation and morphological filtering [1]. In reference-based inspection, the captured board is compared with a reference image that represents the expected appearance, and the differences are analysed as possible defects. The main difficulty with such methods is alignment. If the captured image is rotated, shifted or affected by perspective distortion, the difference operation highlights variations caused by positioning rather than by real defects. Zhou [2] proposed a PCB defect detection approach that combines image registration, image differencing, principal component analysis (PCA) and weighted threshold segmentation, which illustrates how mathematical image analysis can be used to find surface defects.

2.2 Machine-learning-based detection

Machine-learning methods offer an alternative to purely rule-based inspection. They learn patterns from image features and use them to separate defect categories. Their performance, however, depends on how well the extracted features represent the variations in defect appearance, and this dependence limits how far hand-crafted features can be pushed.

2.3 Deep-learning-based detection

Deep-learning detectors have become important in automated visual inspection because they learn relevant features directly from annotated images. YOLO-style architectures are particularly suitable, since they perform localisation and classification in a single unified framework. Ghorpade and Patil [3] combine YOLOv8-based detection with graph-based alternate-path analysis to add a connectivity examination of the board. YOLOv5 has been applied to PCB defect detection in [4], and YOLOv8 in [5]. Improved feature extraction and fusion models for PCB surface defects are described in [6], and improved YOLOv8 variants for bare boards and surface defects are reported in [7] and [8]. Deep-learning approaches to PCB defect detection are also studied in [9], further YOLOv8 improvements are proposed in [10], and visual detection and classification of surface defects on PCBs is examined in [11]. Across this body of work, the reported improvements concern feature extraction, feature fusion, attention mechanisms and multi-scale detection, all aimed at small and complex PCB defects.

2.4 Research gap

The reviewed literature points to several practical challenges.

- Registration and alignment variations affect conventional reference-based inspection.
- Small and closely spaced defects are hard to identify.
- Deep-learning approaches need properly annotated datasets.
- Detection performance is sensitive to lighting, orientation and image quality.
- Many approaches concentrate on model predictions and do not provide an integrated inspection and reporting interface.
- Computational requirements can limit training and inference on resource-constrained systems.

2.5 Positioning of the proposed work

The proposed work responds to these requirements with a combined preprocessing and object-detection approach. The captured image is first prepared by extracting the PCB region, correcting it and standardising it, and only then is it supplied to the trained YOLO26n model. Unlike a stand-alone detector, the PCB Vision application also provides inspection visualisation, defect distribution, severity information, processing-time analysis, Pareto charts and PDF reports. The project therefore focuses on integrating the main stages of PCB image inspection into one application.

3 Proposed system architecture

3.1 Overview

The system is organised as a sequence of interconnected stages covering image acquisition, dataset preparation, preprocessing, object detection, post-processing and result analysis, and it operates in two phases (Figure 1). In the model development phase, PCB images are acquired and annotated, the dataset is configured, and the YOLO26n model is trained to produce a trained model. In the PCB inspection phase, a newly uploaded image passes through BGR-to-HSV conversion, HSV-based segmentation, morphological filtering, boundary detection, region identification and cropping, perspective and tilt correction, and standardisation. The standardised image is then processed by YOLO26n, the detections are post-processed and classified, and the results are analysed and compiled into an inspection report.

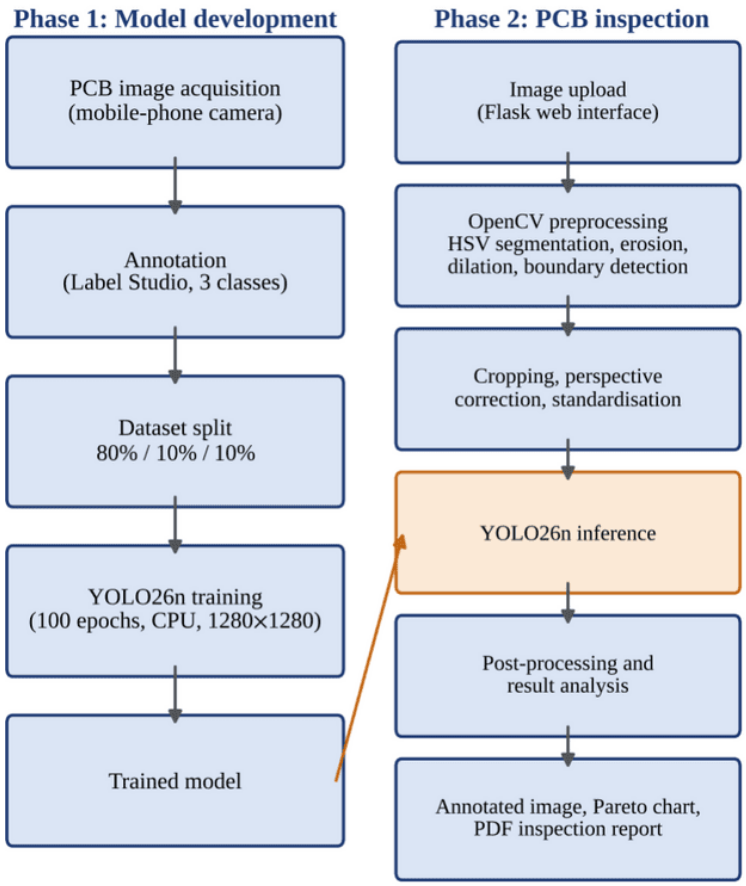


Figure 1. Architecture of the proposed PCB inspection system, showing the model-development phase (left) and the inspection phase (right).

3.2 System description

Images are captured with a mobile-phone camera and organised into a dataset representing three PCB layouts. They are annotated in Label Studio, where each visible defect is marked with a bounding box and a class label. The annotated dataset trains the YOLO26n model for 100 epochs on the laptop CPU at an image size of 1280 × 1280 pixels.

During inspection, the user uploads a PCB image through the PCB Vision web application. OpenCV isolates the board region and corrects variations in orientation and perspective. The resulting standardised image is passed to the trained model, which predicts the locations and categories of possible defects. The application finally processes these predictions and displays them as annotated images, defect counts, class-wise percentages, confidence scores, severity information and processing-time measurements.

3.3 Mathematical representation of the workflow

Table 1. Defect categories considered

Defect category	Description
Scratch	Visible scratch marks on the PCB surface
Mask on Pad (MP)	Solder-mask material present over a pad region
Mask Peel-Off (MF)	Peeling or removal of solder-mask material

The inspection process can be written as a sequence of image transformations and prediction operations. Let $I_0 \in \mathbb{R}^{H\times W\times 3}$ be the original colour image of height H and width W with three colour channels. The preprocessing stage is represented as

$$I_p = \mathrm{P}(I_0) \tag{1}$$

where $P(\cdot)$ is the complete preprocessing function and I_p is the standardised PCB image. Detection by the trained YOLO26n model is

$$D = f_\theta(I_p) \tag{2}$$

where f_θ is the trained detector with learned parameters θ and D is the set of predicted detections. The final inspection output is

$$R = \mathrm{G}(D,\ I_0,\ I_p) \tag{3}$$

where $G(\cdot)$ denotes application-level processing and visualisation. The result R contains the annotated image, defect classes, bounding-box coordinates, confidence scores, statistical summaries and inspection information. These equations describe the architecture at a functional level rather than the exact source-code implementation.

4 Dataset preparation and annotation

4.1 Image acquisition

The quality of the captured images strongly influences both preprocessing and detection. The PCB images used in this work were captured with a mobile-phone camera at a single location and include three different PCB layouts. In total, 344 images were prepared. They contain examples of the three defect categories considered in the project, listed in Table 1. Using several layouts gives some variation in board appearance, but because the images were captured at one location, the dataset does not establish robustness to all industrial lighting and camera conditions.

4.2 Annotation using Label Studio

Before training, each defect must be annotated with its location and class. The images were examined individually in Label Studio and rectangular bounding boxes were drawn around the defect regions, each assigned one of the three class labels (Figure 2). The annotations supply the spatial and categorical information required for supervised object-detection training.

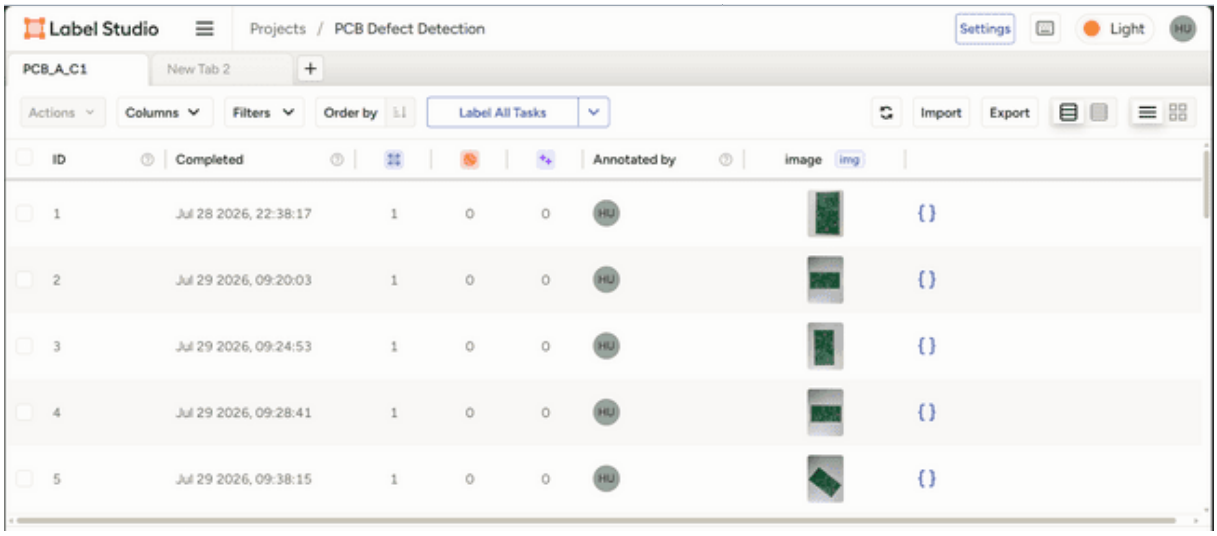


Figure 2. Dataset annotation in Label Studio.

4.3 Bounding-box representation

Each annotated defect is a rectangle that can be written by its top-left and bottom-right corners,

$$B = (x_1,\ y_1,\ x_2,\ y_2) \tag{4}$$

with width and height

$$w = x_2 - x_1,\ h = y_2 - y_1 \tag{5}$$

and centre coordinates

$$x_c = \frac{x_1 + x_2}{2},\ y_c = \frac{y_1 + y_2}{2} \tag{6}$$

For detector training, the box is commonly expressed in normalised form,

$$x'_c = \frac{x_c}{W},\ y'_c = \frac{y_c}{H},\ w' = \frac{w}{W},\ h' = \frac{h}{H} \tag{7}$$

where W and H are the image width and height. The normalised values lie between 0 and 1, which describes the box independently of the original image size.

4.4 Dataset split

The 344 images were divided into training, validation and testing subsets in the ratio 80:10:10. The training subset is used to learn the visual characteristics of the defect classes, the validation subset to monitor training behaviour during model development, and the testing subset is reserved for evaluating the trained model on images not used in training or validation. With N images in total,

$$N = N_{train} + N_{val} + N_{test},\ N_{train} = 0.80N,\ N_{val} = 0.10N,\ N_{test} = 0.10N \tag{8}$$

For N = 344 this corresponds to a total of 275 training, 34 validation and 35 testing images (Table 2).

Table 2. Dataset configuration

Parameter	Value
Total images	344

Parameter	Value
PCB layouts	3
Camera	Mobile phone
Annotation tool	Label Studio
Defect classes	Scratch, MP, MF
Training (80%)	275 images
Validation (10%)	34 images
Testing (10%)	35 images

5 Image preprocessing using OpenCV

The YOLO26n model is not applied to the raw camera image. The image is first prepared with OpenCV, with the aim of removing unnecessary background, isolating the PCB region, correcting orientation and producing a consistent input for the detector. The stages are shown in Figure 3, and example outputs of the stages are shown in Figure 4.

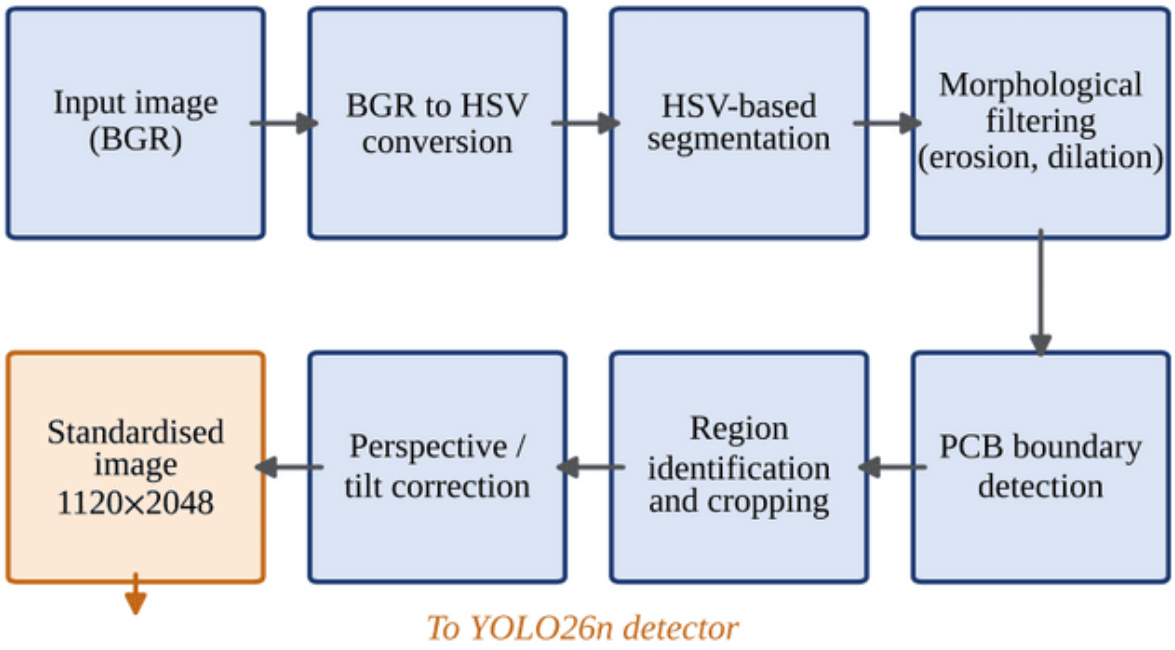


Figure 3. Image preprocessing pipeline.

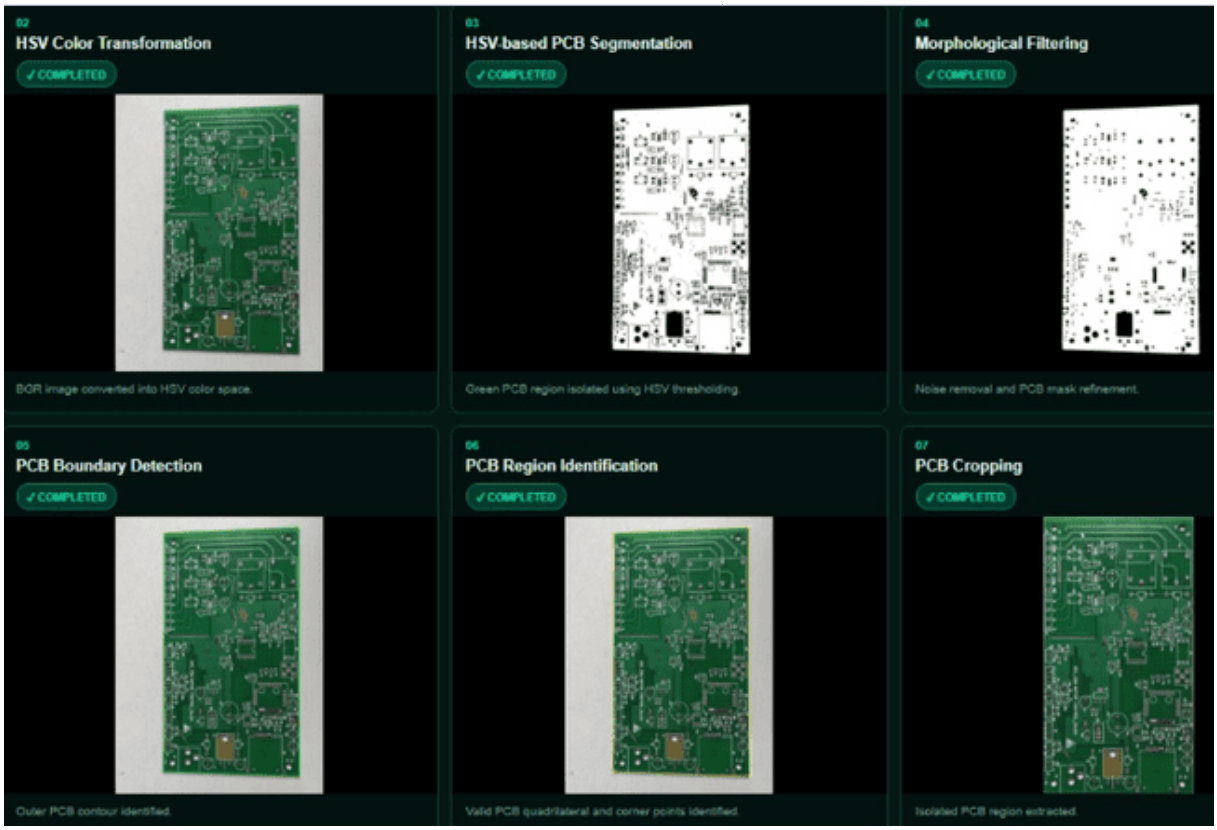


Figure 4. Outputs of the preprocessing stages.

5.1 BGR-to-HSV colour-space conversion

OpenCV loads the captured image in BGR channel order. Although BGR is convenient for storage and display, colour-based object identification is easier in HSV space, where hue (H) represents the colour type, saturation (S) its purity and value (V) its brightness. Separating colour from brightness makes it easier to distinguish the green PCB from its surroundings.

For normalised red, green and blue components $R, G, B \in [0, 1]$, define

$$C_{\max} = \max(R, G, B),\ C_{\min} = \min(R, G, B),\ \Delta = C_{\max} - C_{\min} \tag{9}$$

The hue, saturation and value are then

$$H = \begin{cases} 0^\circ, & \Delta = 0 \\ 60^\circ\left(\dfrac{G-B}{\Delta} \bmod 6\right), & C_{\max} = R \\ 60^\circ\left(\dfrac{B-R}{\Delta} + 2\right), & C_{\max} = G \\ 60^\circ\left(\dfrac{R-G}{\Delta} + 4\right), & C_{\max} = B \end{cases} \tag{10}$$

$$S = \begin{cases} 0, & C_{\max} = 0 \\ \dfrac{\Delta}{C_{\max}}, & C_{\max} \neq 0 \end{cases} \tag{11}$$

$$V = C_{\max} \tag{12}$$

These are the general RGB-to-HSV equations in normalised form. OpenCV applies its own channel conventions; for 8-bit images the hue is stored in the range 0–179 and saturation and value in 0–255.

5.2 HSV-based segmentation

Segmentation divides an image into meaningful regions according to selected characteristics. Here it separates the green PCB from the background by thresholding the HSV image into a binary mask M. For a pixel (x, y),

$$M(x,\ y) = \prod_{c\in\{H,S,V\}} 1\left[c_{\min} \le c(x,\ y) \le c_{\max}\right] \tag{13}$$

where $\mathbf{1}[\cdot]$ equals 1 when the condition holds and 0 otherwise, and $[c_{\min}, c_{\max}]$ is the selected range for each channel. A value of 1 marks a pixel inside the selected PCB colour range and 0 marks a pixel outside it. Segmentation therefore reduces the influence of the surrounding scene and isolates the board surface.

5.3 Morphological filtering

The raw mask usually contains small spurious regions, gaps and irregular boundaries. Morphological operations, which act on the shape of regions in a binary image through a small structuring element, are applied to refine it. In this implementation, erosion and dilation are used. With A the foreground of the binary mask and B the structuring element, erosion is

$$A \ominus B = \{z \mid B_z \subseteq A\} \tag{14}$$

where B_z is B translated by z. Erosion shrinks foreground regions by removing boundary pixels, which eliminates small unwanted regions and smooths irregularities. Dilation is

$$A \oplus B = \{z \mid (\widehat{B})_z \cap A \ne \varnothing\} \tag{15}$$

where $\hat{B}$ is the reflection of B. Dilation enlarges foreground regions, restoring or connecting regions that erosion may have separated. Used together, the two operations improve the mask before boundary detection, and the refined mask is then used to find the board in the original image.

5.4 Boundary detection and region identification

Boundary detection finds the outer extent of the segmented region. If the detected PCB boundary is the contour

$$C = \{(x_1,\ y_1),\ (x_2,\ y_2),\ \ldots,\ (x_n,\ y_n)\} \tag{16}$$

its spatial extent follows from

$$x_{\min} = \min_i x_i,\ x_{\max} = \max_i x_i,\ y_{\min} = \min_i y_i,\ y_{\max} = \max_i y_i \tag{17}$$

and the bounding rectangle of the board has width and height

$$W_b = x_{\max} - x_{\min},\ H_b = y_{\max} - y_{\min} \tag{18}$$

These coordinates provide the spatial information needed for cropping.

5.5 PCB region cropping

Cropping removes the unnecessary background. The cropped image I_c is obtained from the original image I as

$$I_c(x,\ y) = I(x + x_{\min},\ y + y_{\min}),\ 0 \le x < W_b,\ 0 \le y < H_b \tag{19}$$

This gives a focused view of the board and is particularly useful when the mobile-phone image contains a large background area.

5.6 Perspective and tilt correction

During acquisition the board is not always held perpendicular to the camera, which produces rotated or tilted boards, perspective distortion and variations in apparent dimensions. Perspective or tilt correction is therefore applied after the PCB region has been extracted. In general, a perspective transformation is described by a 3 × 3 homography matrix **H** that maps a point (x, y) of the original image plane to (x_s, y_s) in the corrected plane,

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \; x_s = \frac{x'}{w'}, \; y_s = \frac{y'}{w'} \tag{20}$$

In a typical procedure, corresponding points in the original PCB region and in the desired rectangular output are used to determine the matrix. The correction yields a more consistent representation of the board for the later stages.

5.7 Image standardisation

After cropping and correction, the PCB image is resized to a fixed resolution of $W_s \times H_s = 1120 \times 2048$ pixels. If W_c and H_c are the width and height of the corrected image before resizing, the scaling factors are

$$S_x = \frac{W_s}{W_c}, \; S_y = \frac{H_s}{H_c} \tag{21}$$

and a point (x_c, y_c) of the corrected image maps to

$$x_s = S_x x_c, \; y_s = S_y y_c \tag{22}$$

Standardisation gives the detector a consistent input format. The image sizes at the different stages of the pipeline are different and should not be confused (Table 3): the original example image is 3024 × 4032 pixels, the preprocessing output is 1120 × 2048 pixels, and the YOLO26n training image size is 1280 × 1280 pixels.

Table 3. Image sizes at different stages

Processing stage	Image size (pixels)
Original example image	3024 × 4032
Preprocessing output	1120 × 2048
YOLO26n training size	1280 × 1280

6 YOLO26n-based defect detection

6.1 Object detection with YOLO

YOLO (You Only Look Once) is a family of models that identify and localise objects within an image. Unlike image classification, which assigns one label to an entire image, an object detector finds individual objects and associates each with a class label and a bounding box. This suits PCB

inspection, where several defects may be present at different locations on one board. The YOLO26n model performs two tasks in the proposed system: localisation, which gives the approximate position of a defect as a bounding box, and classification, which assigns the region to a defect category. Its output is then processed by the PCB Vision application.

6.2 Model training

The annotated dataset was used to train YOLO26n on the laptop CPU for 100 epochs at an image size of 1280 × 1280 pixels (Table 4). The bounding boxes tell the model where each defect is, and the class labels tell it what the defect is. In general, training seeks the parameters that minimise a loss function,

$$\theta^{*} = \arg\min_{\theta} \mathrm{L}(\theta) \tag{23}$$

where θ denotes the model parameters, $\mathcal{L}(\theta)$ the training loss and θ* the parameters that minimise it. The objective is to reduce the difference between the predicted detections and the annotated ground truth. Equation (23) states this general objective and does not describe a specific loss implementation.

Table 4. Training configuration

Parameter	Setting
Model	YOLO26n
Epochs	100
Image size	1280 × 1280
Hardware	Laptop CPU
Training data	275 images
Annotation format	Bounding boxes

6.3 Detection output

During inspection, the standardised image is passed to the trained model. For each predicted defect region the model provides a bounding box, a predicted class and a confidence score, so that the *i*-th detection is

$$D_i = (B_i,\ c_i,\ s_i),\ c_i \in \{\mathrm{Scratch},\ \mathrm{MP},\ \mathrm{MF}\} \tag{24}$$

where B_i is the box, c_i the class and s_i the confidence score. Several defect regions can be identified in a single PCB image.

6.4 Bounding-box geometry

A bounding box approximates the rectangular region containing a predicted defect. For $B_i = (x_1, y_1, x_2, y_2)$ the width, height and area are

$$w_i = x_2 - x_1,\ h_i = y_2 - y_1,\ A_i = w_i h_i = (x_2 - x_1)(y_2 - y_1) \tag{25}$$

expressed in square pixels when pixel coordinates are used. The application displays these quantities for each detection. It should be noted that A_i is the area of the bounding rectangle and not the exact area of the defect, and that no physical measurement in millimetres is established without a calibration procedure.

6.5 Confidence score

The confidence score expresses the model's confidence in an individual prediction. A score of 0.85, for example, is displayed as 85%:

$$s_{\%} = 100s \tag{26}$$

where $s \in [0, 1]$. A higher score indicates stronger confidence but does not guarantee that the prediction is correct. Confidence should not be confused with precision, recall or accuracy, which are defined under a specified evaluation protocol (Section IX).

6.6 Intersection over Union

Intersection over Union (IoU) is the standard measure of the overlap between a predicted box B_p and a ground-truth box B_g (Figure 5):

$$\text{IoU} = \frac{|B_p \cap B_g|}{|B_p \cup B_g|} \tag{27}$$

where the numerator is the overlapping area and the denominator the combined area of the two boxes. IoU ranges from 0 (no overlap) to 1 (perfect overlap) and indicates how accurately a predicted box matches the actual defect location. It is used here as a methodological definition; no measured IoU values are reported in this paper.

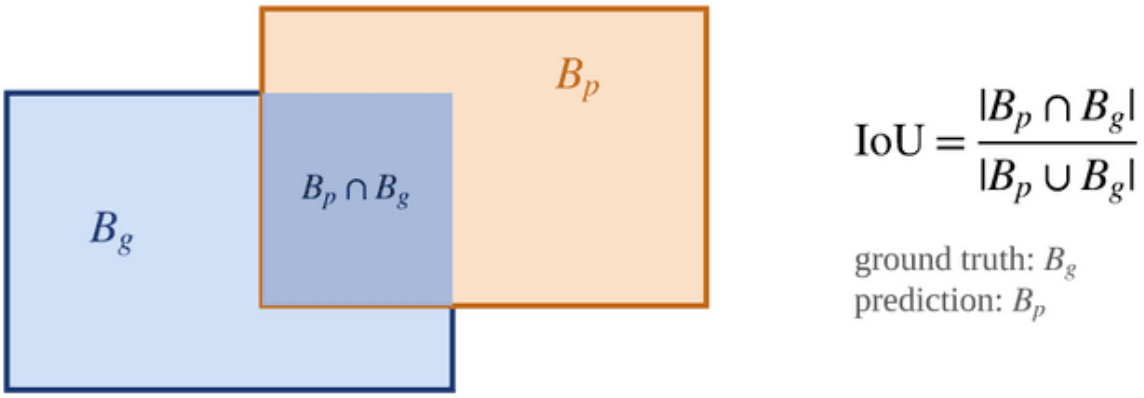


Figure 5. Intersection over Union between predicted and ground-truth boxes.

6.7 Non-maximum suppression

A detector may generate several overlapping boxes around the same object. Non-maximum suppression (NMS) reduces such duplicates. The detection with the highest confidence is retained, the IoU between it and the remaining overlapping boxes is calculated, and boxes whose overlap exceeds a threshold are suppressed; the procedure is repeated until all detections have been processed. A lower-scoring box B_j is suppressed when

$$\text{IoU}\left(B_i, B_j\right) \geq T_{IoU} \tag{28}$$

where T_{IoU} is the suppression threshold and B_i is the retained detection. NMS yields a clearer final detection output.

6.8 Defect classification

The trained model supports three categories (Table 1). A *scratch* is a visible surface mark caused by mechanical contact, handling or abrasion. *Mask on Pad* refers to solder-mask material that appears over a pad region where it is not intended to remain. *Mask Peel-Off* refers to the removal or peeling of solder-mask material from the PCB surface. The classification output allows the application to organise detections by category.

7 Web-based PCB Vision application

7.1 Application development

PCB Vision was developed with the Flask framework to give users a simple interface for uploading PCB images, running the inspection pipeline and examining the results. It integrates the software components listed in Table 5 into one workflow.

Table 5. Software components

Technology	Purpose
Python	Main programming language
OpenCV	Image preprocessing
Label Studio	Dataset annotation
YOLO26n	Defect detection
Flask	Web application framework
HTML, CSS	Page structure and interface design
ReportLab	PDF report generation

7.2 Image upload

The interface allows the user to select a JPG, JPEG or PNG image from the local system and start the inspection with the *Run PCB Inspection* option. A demonstration option executes the workflow on a built-in test image.

7.3 Pipeline and model integration

After upload, Flask coordinates the workflow. The image is passed to the OpenCV preprocessing pipeline, which identifies the PCB region, removes the background, applies correction and standardises the image. The standardised image is supplied to the trained YOLO26n model, which returns defect coordinates, class labels and confidence scores. The application then prepares these predictions for display.

7.4 Detection visualisation and defect information

Predicted defect regions are drawn as bounding boxes, each associated with a class, a confidence score and an approximate location, so that users can find the regions needing further examination without scanning the whole image. A detailed table lists the individual detections with the fields shown in Table 6, so that the user can inspect each detection rather than rely on the total count alone.

Table 6. Fields in the defect information table

Field	Description
ID	Unique serial number
Class	Predicted defect category
Severity	Assigned severity level
Confidence	Model confidence
X1, Y1	First bounding-box corner
X2, Y2	Opposite bounding-box corner
Width, Height	Bounding-box dimensions
Area	Bounding-box area (px^2)

7.5 Severity labels

Each detection is given a severity label (Normal, Minor, Major or Critical) as an additional way to organise the detected regions for review. The labels are an aid to inspection review and are not a physical-damage measurement.

7.6 Pareto analysis

Pareto analysis identifies the defect categories that contribute most to the total number of detections in an inspection. The application shows category counts as bars and cumulative percentages as a line. If N_1, N_2, ..., N_k are the numbers of detections in the k categories, arranged in descending order, the total is

$$N_T = \sum_{i=1}^{k} N_i \tag{29}$$

the percentage contribution of category i is

$$P_i = \frac{N_i}{N_T} \times 100 \tag{30}$$

and the cumulative percentage is

$$CP_i = \sum_{j=1}^{i} P_j \tag{31}$$

The worked values for the demonstrated inspection are given in Section 8.4.

7.7 Production feedback

The application also offers feedback on the production process, based on the detected categories and their distribution. For Mask on Pad it suggests reviewing solder-mask alignment, pad registration, application consistency and PCB positioning during the solder-mask process. For Mask Peel-Off it suggests examining solder-mask adhesion, surface preparation, curing conditions and PCB handling. For Scratch it suggests reviewing handling practices, fixtures, transport surfaces and mechanical contact points. These suggestions point to process areas that may deserve further investigation; they do not establish the actual manufacturing root cause.

7.8 Processing-time analysis

The application records the time taken by the stages of the inspection pipeline: image loading, HSV transformation, segmentation, morphological filtering, boundary detection, region identification, alignment, cropping, perspective correction, standardisation, YOLO inference and post-processing. The total processing time is

$$T_{total} = \sum_{i=1}^{n} T_i \tag{32}$$

where T_i is the time of the i-th stage and n the number of stages.

7.9 Annotated image and PDF report

The application can save the annotated image, in which detected regions are marked with bounding boxes and class labels, and can generate a downloadable PDF inspection report with ReportLab containing the available detection information and an inspection summary. These facilities make it easier to retain, review and share inspection results.

8 Experimental results and discussion

This section reports outputs from the completed prototype. It distinguishes between demonstrated application functionality and experimentally validated detection performance; the results below belong to the first category.

8.1 Experimental environment

The system was developed and run on a laptop and a mobile-phone camera. The camera was used to collect the PCB images, and the laptop was used for dataset preparation, image processing, model training and application execution. YOLO26n was trained for 100 epochs on the CPU at an image size of 1280 × 1280 pixels.

8.2 PCB Vision interface

The home page gives access to the main inspection functions through the navigation options Dashboard, New Inspection, Inspection Pipeline, System and History. The upload page lets the user select a PCB image and start the inspection (Figure 6).

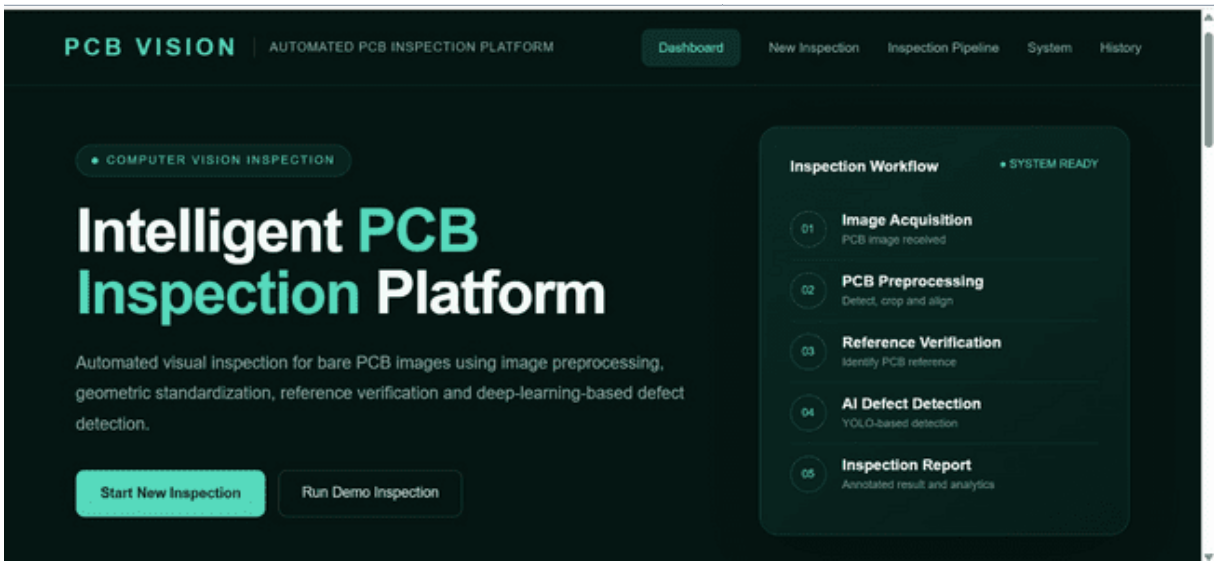


Figure 6. PCB Vision home page and image-upload interface.

8.3 Defect detection output

After preprocessing, the image is passed to YOLO26n, and the predicted defect locations, class labels and confidence scores are displayed as bounding boxes over the PCB image (Figure 7). The output shows that the system can identify and visually represent all three supported defect categories in the reported inspection.

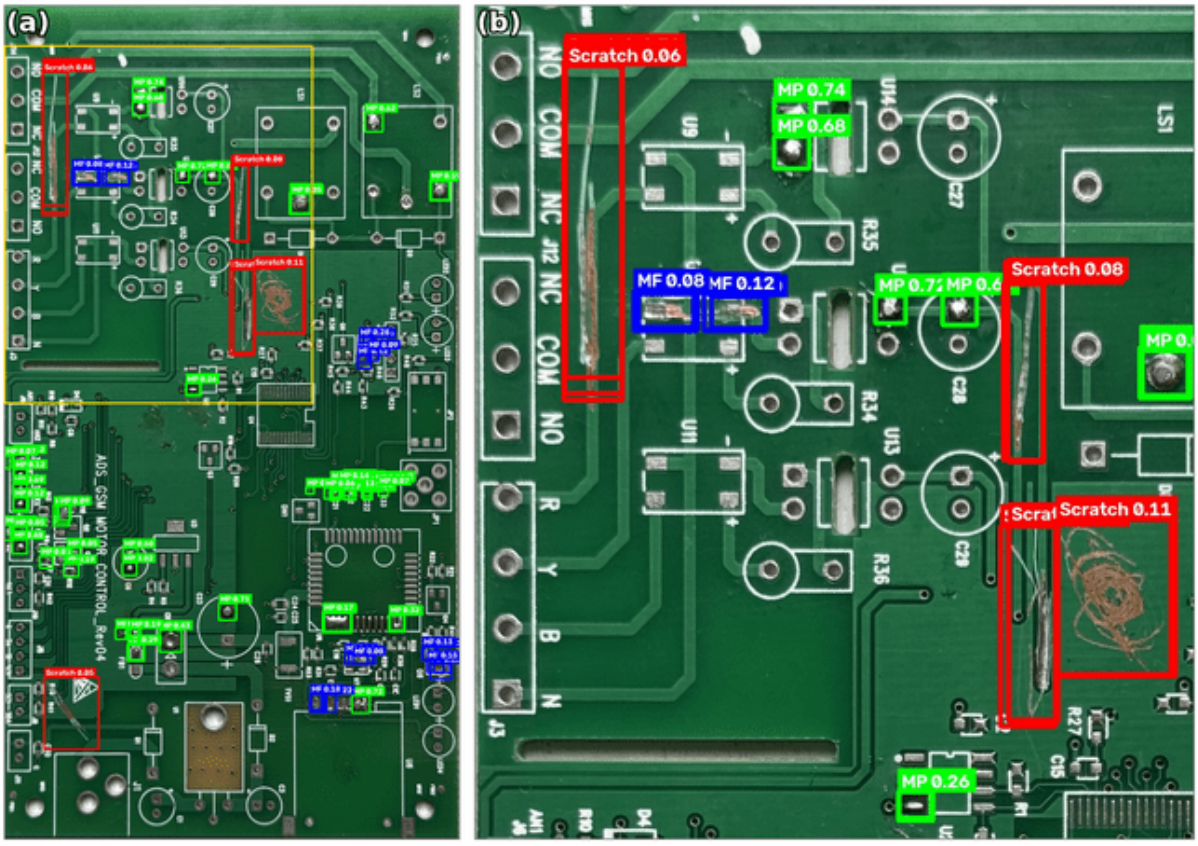


Figure 7. YOLO26n-based PCB defect detection output.

8.4 Class-wise defect distribution

The demonstration inspection contains 98 predicted defect regions (Table 7). Using Eqs. (29)–(31), the percentage contributions are P_{MP} = 65/98 × 100 = 66.33%, P_{MF} = 21/98 × 100 = 21.43% and P_S = 12/98 × 100 = 12.24%. The cumulative percentages are CP_1 = 66.33%, CP_2 = 66.33 + 21.43 = 87.76% and CP_3 = 100%. Mask on Pad is therefore the dominant category in this inspection, and Mask on Pad together with Mask Peel-Off accounts for 87.76% of all detections (Figure 8). This is the distribution of one reported inspection and not a dataset-wide defect-frequency measurement. It can nevertheless help inspection personnel prioritise further examination of the dominant category.

Table 7. Class-wise defect distribution

Defect category	Detections	Share (%)	Cumulative (%)
Mask on Pad	65	66.33	66.33
Mask Peel-Off	21	21.43	87.76
Scratch	12	12.24	100.00
Total	**98**	**100.00**	

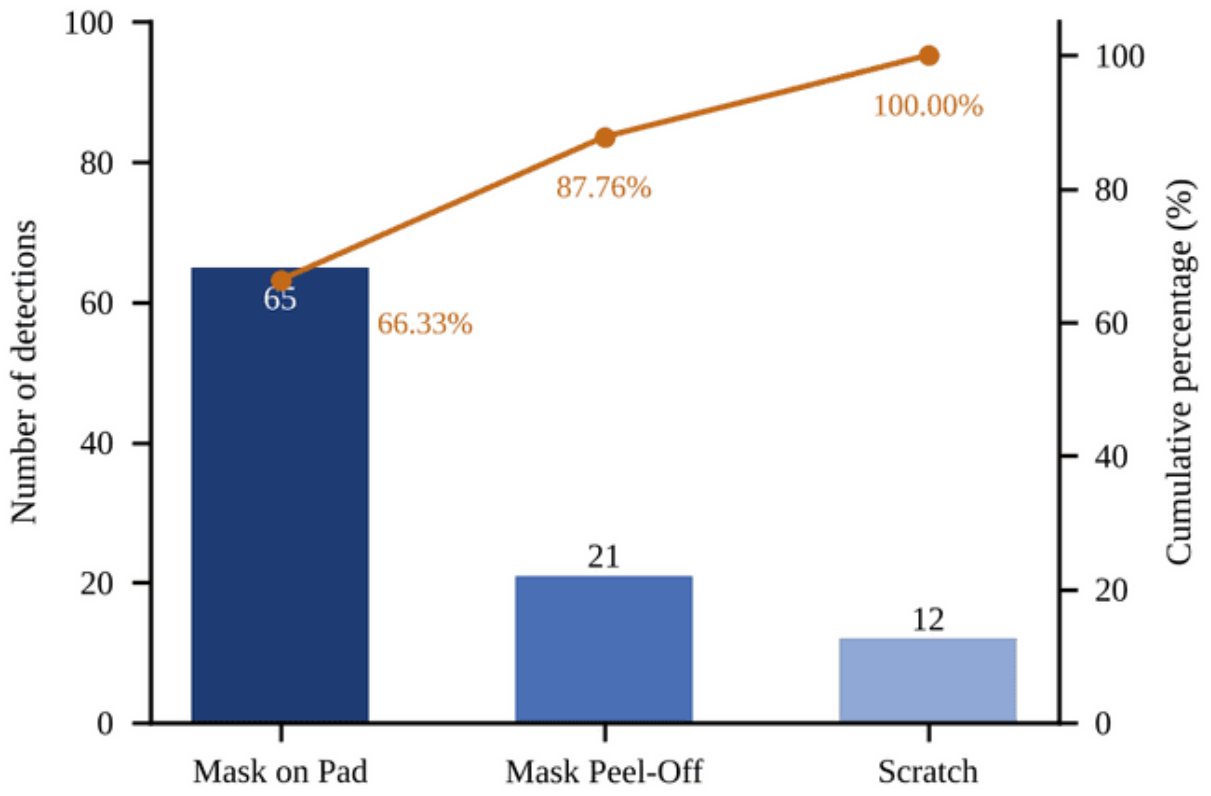


Figure 8. Pareto analysis of the detected PCB defects.

8.5 Processing performance

Table 8 lists the processing times documented for the demonstration inspection, and Figure 9 plots them. YOLO26n inference required 0.6163 s of the 2.1403 s needed for the complete inspection. The remaining 0.7561 s corresponds to the stages and application-level operations that are not itemised in the table. These values come from a single execution and should not be taken as a guaranteed processing time for every image.

Table 8. Processing time of the demonstration inspection

Processing stage	Time (s)
Image loading	0.2223
HSV transformation	0.2159
PCB region identification	0.3297
YOLO26n inference	0.6163
Remaining stages (derived)	0.7561
Complete inspection	**2.1403**

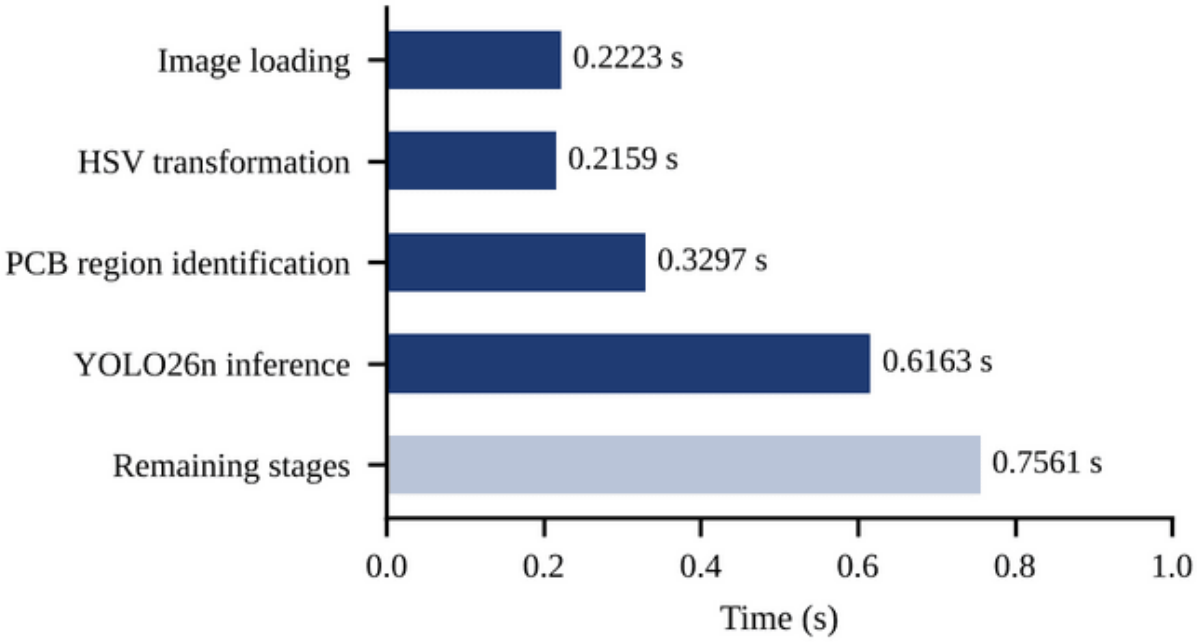


Figure 9. Processing time of the inspection stages for the demonstration image.

8.6 Image geometry, defect details and reports

The application displays the original and standardised image dimensions (Table 3): the original example image is 3024 × 4032 pixels and the standardised image is 1120 × 2048 pixels, which gives a consistent format for the detection stage. The detailed defect table lists the class, confidence, severity, box coordinates, dimensions and area of every predicted region, and the application offers downloads of the annotated image and of the PDF inspection report (Figure 10).

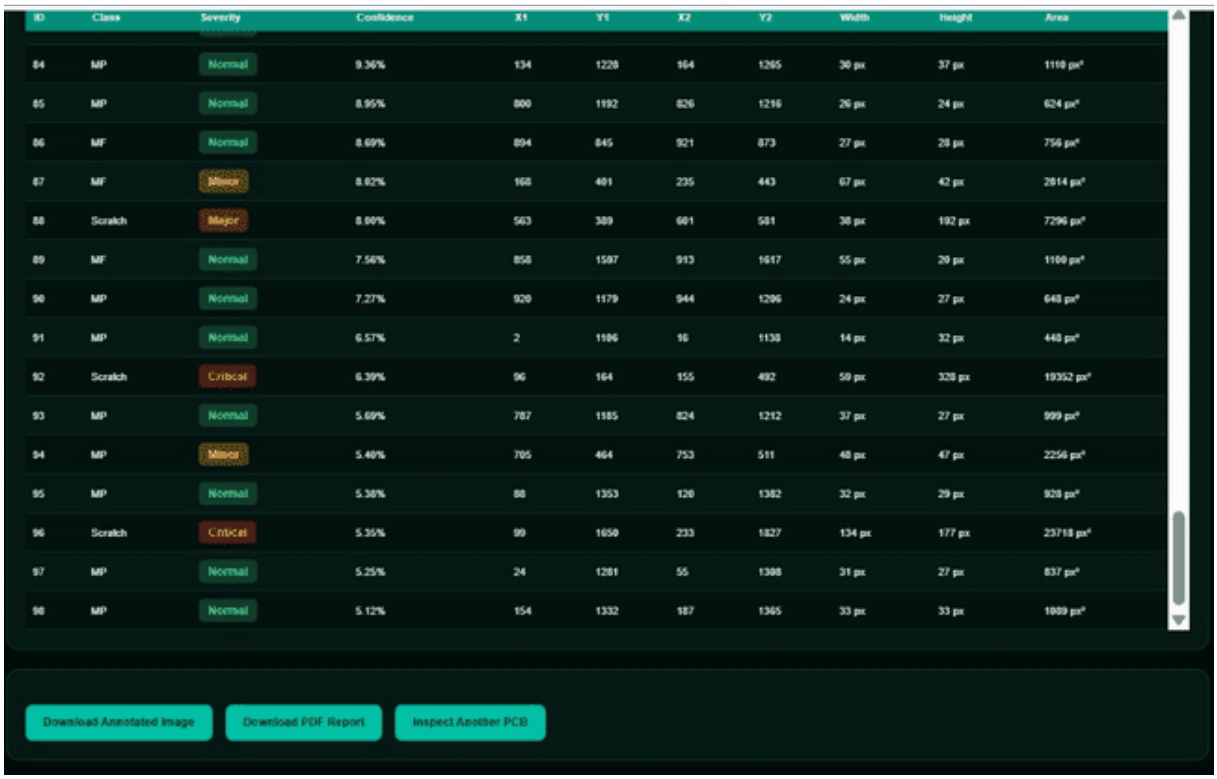


Figure 10. Detailed defect information, annotated image and PDF report options.

8.7 Overall discussion

The observations show that the system integrates image acquisition, preprocessing, deep-learning-based detection and result presentation. The OpenCV pipeline isolates the board, corrects its orientation and standardises its dimensions, and the trained YOLO26n model then identifies regions belonging to the supported defect categories. The web application organises the predictions into a structured interface that adds class-wise statistics, confidence scores, severity labels, geometric measurements, timing information and Pareto analysis to the annotated output. The demonstration inspection produced 98 predicted defect regions in a total processing time of 2.1403 s. These observations demonstrate the working functionality of the prototype; further quantitative testing is needed to establish its detection accuracy and its generalisation across PCB designs and image acquisition conditions.

9 Model evaluation metrics

A complete assessment of an object detector requires standard metrics computed on labelled test images. This section defines the metrics intended for evaluating the model on the 10% testing subset. Test-set values are not reported in this paper, because no test evaluation is documented (validation: Table 9).

Detection results are first counted as follows. A *true positive* (TP) is a predicted defect that matches an actual labelled defect in class and satisfies the chosen IoU criterion. A *false positive* (FP) is a prediction that does not match any ground-truth defect. A *false negative* (FN) is a labelled defect that the model fails to detect. True negatives are generally not used in object detection, because background pixels are not counted as matched object instances.

Precision is the proportion of predicted detections that are correct, and recall is the proportion of labelled defects that are detected:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{33}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{34}$$

A high precision means few false alarms, and a high recall means few missed defects. The F1-score is their harmonic mean:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{35}$$

Average precision (AP) is the area under the precision–recall curve of a single class, evaluated at a specified IoU threshold,

$$AP = \int_0^1 p(r)dr \tag{36}$$

where $p(r)$ is the precision at recall r. The mean average precision (mAP) averages AP over the C classes; for the three classes of this work,

$$mAP = \frac{1}{C}\sum_{c=1}^{C} AP_c = \frac{AP_{Scratch} + AP_{MP} + AP_{MF}}{3} \tag{37}$$

For speed, the average processing time over N test images and the corresponding throughput are

$$T_{avg} = \frac{1}{N}\sum_{i=1}^{N} T_i, \; FPS = \frac{N}{\sum_{i=1}^{N} T_i} \tag{38}$$

provided that the same, clearly defined processing stages are timed for every image. Only a single execution time (2.1403 s) was recorded in this work, which does not support a reliable average FPS. Finally, a three-class confusion matrix, with Scratch, MP and MF along both axes, shows correct classifications on its diagonal and class confusions off the diagonal. For object detection, its construction must also account for the matching of predicted and ground-truth boxes at a defined IoU threshold, and for unmatched predictions and missed objects.

10 Advantages

The developed system offers the following advantages. The trained YOLO26n model identifies the selected surface defects automatically and gives their approximate locations. OpenCV preprocessing isolates the board and prepares the image before detection, and standardisation to 1120 × 2048 pixels gives a consistent output. Three defect categories are supported, and bounding boxes with class labels make the detected regions easy to find. The application provides confidence scores, severity labels, coordinates, dimensions and area information for each detection, class-wise counts and percentages for the inspection as a whole, and the time taken by individual stages. Defect-based feedback points to process areas that merit investigation, and annotated-image downloads and PDF reports keep a record of every inspection.

11 Limitations

Although the system demonstrates an integrated inspection workflow, several limitations remain.

- **Dataset size and diversity.** The model was developed with 344 images from three PCB layouts, and its performance on substantially different designs has not been established. A larger and more diverse dataset would help it handle different designs and defect appearances.
- **Defect scope.** Only Scratch, Mask on Pad and Mask Peel-Off are supported; other PCB defects are outside the present scope.
- **Acquisition conditions.** Images were captured with a mobile-phone camera at a single location. Variations in illumination, camera angle, clarity and reflections can affect preprocessing and detection, and blurred, poorly lit or reflective images and unclear board boundaries can degrade segmentation and the defect identification that follows.
- **Hardware dependency.** The model was trained on a laptop CPU, and training duration and inspection speed will differ on other hardware.
- **No independent test results.** The project does not yet document an independent test-set evaluation with standard object-detection metrics, so overall accuracy, precision, recall and mAP cannot be established from the demonstration results alone.
- **Production feedback.** The recommendations are based on the detected categories and require human verification, since they do not establish the actual manufacturing root cause.

12 Future scope

The PCB Vision system is a foundation for further development in automated inspection. The dataset can be expanded with images from additional layouts, manufacturing batches and acquisition conditions, with more examples of each defect category. The defect scope can be extended to categories such as missing holes, mouse bites, open circuits, short circuits, spurious copper, pinholes, spurs and other solder-mask and surface defects, once suitable training images have been collected and annotated. A fixed camera arrangement with controlled illumination and stable board positioning would reduce unwanted variation and make preprocessing more consistent.

Quantitative evaluation is the most immediate need: an independent labelled test set should be used to report precision, recall, F1-score, AP, mAP, the confusion matrix and false-positive and false-negative analysis. Model-optimisation techniques may reduce inference time for resource-constrained systems, provided speed gains are checked against detection performance. The application can be extended with batch-wise defect statistics, historical comparisons, configurable quality thresholds and more detailed inspection records, and it may be integrated with manufacturing inspection and quality-management platforms. Before industrial deployment, the complete system would need validation with representative production data and actual acceptance criteria.

13 Conclusion

This paper presented an ML-based vision system for detecting selected surface defects on bare PCBs. It combines OpenCV-based preprocessing, YOLO26n object detection and a Flask-based web application into one inspection workflow. The preprocessing pipeline performs BGR-to-HSV conversion, HSV-based segmentation, morphological filtering with erosion and dilation, boundary detection,

region identification, cropping, perspective correction and standardisation, and so prepares each captured image for detection.

A dataset of 344 mobile-phone images from three PCB layouts was annotated in Label Studio and divided 80:10:10 into training, validation and testing subsets. YOLO26n was trained for 100 epochs on a laptop CPU at 1280 × 1280 pixels and integrated into the PCB Vision application to identify Scratch, Mask on Pad and Mask Peel-Off defects. The application provides visual localisation, confidence scores, severity information, geometric measurements, processing-time analysis, class-wise distribution, Pareto analysis, production-feedback suggestions, annotated-image downloads and PDF reports. In the demonstration inspection it displayed 98 defect regions (65 Mask on Pad, 21 Mask Peel-Off and 12 Scratch) in 2.1403 s. The results demonstrate the integration and functionality of the prototype on the tested images. Greater dataset diversity, a quantitative evaluation on the held-out test subset and more controlled image acquisition would help establish its suitability for broader PCB inspection applications.

Table 9. Overall validation-subset results of YOLO26n

Metric	Value (%)
Precision	73.62
Recall	72.69
mAP@50	78.29
mAP@50–95	38.83

Table 9 gives the overall scores of the trained YOLO26n model on the validation subset of 34 images, computed over all three classes; per-class values and a confusion matrix are not available. Because this subset was also used to monitor training, the scores are indicative only and do not replace an independent test-set evaluation.

References

[1] Ling Q, Mat Isa NA. Printed circuit board defect detection methods based on image processing, machine learning and deep learning: a survey. *IEEE Access*. 2023;11:15921–15944.

[2] Zhou L. Research on PCB defect detection method based on principal component analysis. In: *Proceedings of the IEEE 8th Information Technology and Mechatronics Engineering Conference (ITOEC)*; 2025. p. 1070–1074.

[3] Ghorpade DD, Patil SR. A hybrid system for PCB defect detection using YOLOv8 and graph-based alternate path analysis. In: *Proceedings of the 5th Asian Conference on Innovation in Technology (ASIANCON)*; 2025.

[4] Nitish K, Koduru UCR, Gurucharan GJ, Selvam P, Jayabal R. Printed circuit board (PCB) defect detection using YOLOv5. In: *Proceedings of the 3rd International Conference on Emerging Applications of Material Science and Technology (ICEAMST)*; 2025.

[5] Zhu S, Luo S, Liu J. PCB defect detection based on YOLOv8 model. In: *Proceedings of the IEEE 7th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*; 2025. p. 392–396.

[6] Li J, et al. An improved feature extraction and fusion model for PCB surface defect detection. In: *Proceedings of the 6th International Conference on Internet of Things, Artificial Intelligence and Mechanical Automation (IoTAIMA)*; 2025.

[7] Yu J, Gao Y. Defect detection on PCB bare boards based on improved YOLOv8 algorithm. In: *Proceedings of the 3rd International Conference on Cloud Computing, Big Data Application and Software Engineering (CBASE)*; 2024.

[8] Ai Y, Ye T. Surface defect detection algorithm for PCB based on improved YOLOv8. In: *Proceedings of the 8th International Conference on Electrical, Mechanical and Computer Engineering (ICEMCE)*; 2024. p. 1421–1425.

[9] Hu T, Lu Y, Liu C, Zhuang D. Research on PCB defect detection method based on deep learning. In: *Proceedings of the 5th International Conference on Computer Engineering and Intelligent Control (ICCEIC)*; 2024. p. 303–307.

[10] Zhu H, Luo R, Lan H, Ren Q, Chen C. PCB defect detection algorithm of improved YOLOv8. In: *Proceedings of the 8th International Conference on Image, Vision and Computing (ICIVC)*; 2023.

[11] Ji X, Zhao S. Research on visual detection and classification technology for surface defects of printed circuit boards. In: *Proceedings of the 3rd International Conference on Electronic Information Engineering and Computer Science (EIECS)*; 2023. p. 313–316.