**Project Report: PCB Defect Detection**

This report presents the development and outcomes of an automated Printed Circuit Board (PCB) defect detection system built using the YOLOv8 architecture from the Ultralytics framework. The primary aim of this project was to design a fast, accurate, and reliable object detection pipeline capable of identifying and localizing the most common defects found in PCBs. Such automation significantly improves the speed and consistency of quality control processes compared to traditional manual inspection.

**Milestone 1: Project Methodology – Dataset Preparation and Preprocessing**

**Model Architecture (Algorithms Used)**

The system is powered by the lightweight **YOLOv8n (nano)** model, selected for its excellent trade-off between inference speed and detection accuracy. YOLOv8 operates as a fully end-to-end object detection framework, predicting bounding boxes and defect classes simultaneously in a single forward pass. This enables real-time processing, which is essential for industrial applications.

**Data Preprocessing Workflow**

1. **Dataset Setup:**  
   The DeepPCB dataset served as the foundation of the training pipeline. The dataset was systematically organized into training, validation, and testing subsets, ensuring proper model evaluation and generalization.
2. **Configuration File Creation:**  
   A dedicated configuration file (pcb\_data.yaml) was prepared to map dataset paths and define all six defect categories, including missing hole, short, spur, and others. This ensured that YOLOv8 could correctly interpret and process the dataset labels.
3. **Labeling and Normalization:**  
   Each PCB image was paired with a corresponding text annotation file that stored normalized bounding box values in YOLO format.  
    <class\_id> <x\_center> <y\_center> <width> <height>  
   This uniform structure enabled accurate training and consistent scaling across images of varying dimensions.

**Feature Engineering Steps**

Because YOLOv8 is a deep learning–based model, many traditional feature engineering tasks are handled implicitly:

* **Data Augmentation:**  
  To improve model generalization, Ultralytic’s built-in augmentation pipeline was used. Techniques such as horizontal flipping, random scaling, and **Mosaic augmentation** (combining four images into one) enhanced the model’s ability to recognize defects under varied conditions.

**Hyperparameter Tuning**

Several training hyperparameters were adjusted to ensure proper model convergence:

* **Weight Initialization:**  
  Training began using the pretrained yolov8n.pt weights, allowing the model to leverage prior learned features and speed up optimization.
* **Optimization Strategy:**  
  The learning process utilized either the SGD or Adam optimizer along with YOLO’s multi-part loss function, which handles classification loss, objectness loss, and bounding box regression simultaneously.
* **Epoch and Batch Size Selection:**  
  The number of training epochs and batch size were chosen to balance learning stability and computational efficiency, ensuring that the model did not underfit or overfit the dataset

**Model Training and Evaluation**

**The YOLOv8n (nano) architecture—selected for its strong balance between speed and accuracy—was developed and trained using the Ultralytics framework.**

* **Model Implementation:**The training pipeline (train.py) loaded the pretrained yolov8n.pt weights, which were then fine-tuned specifically on the PCB defect dataset to improve detection performance**.**
* **Training Configuration:**The model was trained using a suitable batch size and for an adequate number of epochs to allow stable convergence. Optimization was performed using either the default SGD or Adam optimizer, combined with YOLO’s multi-part loss function that accounts for classification, objectness, and bounding box regression.
* **Data Augmentation:  
  Built-in Ultralytics augmentations**—such as random flipping, scaling transformations, and mosaic composition—were applied automatically during training to improve the model’s ability to generalize, especially when detecting small or subtly varying PCB defects**.**

| **Metric** | **Meaning / Purpose** |
| --- | --- |
| **Precision** | Measures the proportion of predicted defect boxes that are correct. |
| **Recall** | Indicates the fraction of actual (ground-truth) defects successfully detected by the model. |
| **mAP50** | Mean Average Precision at IoU = 0.50; primary metric for evaluating detection accuracy. |
| **mAP50–95** | Average mAP across IoU thresholds from 0.50 to 0.95; a stricter measure of overall localization and classification performance. |

**Performance Metrics :**

**Evaluation Methods :**

| **Computation Method** | **Description** |
| --- | --- |
| **YOLOv8 Validation Pipeline** | Ultralytics automatically computes all metrics after each training epoch. |
| **results\_dict & Logs** | Final metric values, graphs, and plots are generated using the internal results dictionary and saved run logs. |

**Challenges Faced & Solutions Implemented**

**-1 Small Dataset & Class Imbalance**

* **Problem:** Some defect classes had very few samples, causing low recall for rare defects.
* **Solution:** Applied strong augmentation (rotation, scaling, mosaic), oversampled rare classes, and used techniques like focal loss to better handle imbalanced data.

**-2 Small Defects & Bounding Box Accuracy**

* **Problem:** Very tiny defects were difficult for the model to detect and localize accurately.
* **Solution:** Used higher image resolution, multi-scale training, and preprocessing methods such as image subtraction and contour detection to better highlight small defect regions.

**-3 Tooling & Implementation Issues**

* **Problem:** Occasional errors, model loading issues, or environment problems slowed progress.
* **Solution:** Ensured correct dependency versions, reconfigured the training environment, and used Ultralytics’ built-in debugging logs to fix errors quickly.

**Inference and User Interface (UI)**

**Inference Pipeline:**

* The trained model was tested on validation images to generate annotated outputs stored under runs/detect/predict.
* Each prediction includes bounding boxes, class labels, and confidence scores to visually confirm defect detection quality.

**Streamlit-Based UI:**

* A custom Streamlit interface was developed to let users upload one or multiple PCB images.
* Images are shown in a clean grid layout, making it easy to review multiple outputs at once.
* Users can download results in multiple formats, including CSV, TXT, PDF, and ZIP, containing logs and annotated images.
* An intuitive download button allows exporting the complete batch report or individual image outputs.

**Backend functions:**

* **High-Performance Engine:** Built on the **FastAPI** framework to serve a custom **YOLOv8** model, acting as the system's dedicated real-time inference engine.
* **Efficient Data Handling:** Processes raw images via **HTTP POST** requests, utilizing **Pillow** for manipulation and **Base64** encoding for seamless network transmission.
* **Structured Output:** Delivers a comprehensive **JSON payload** containing defect counts, specific bounding box coordinates, and the annotated visual overlay for immediate display.

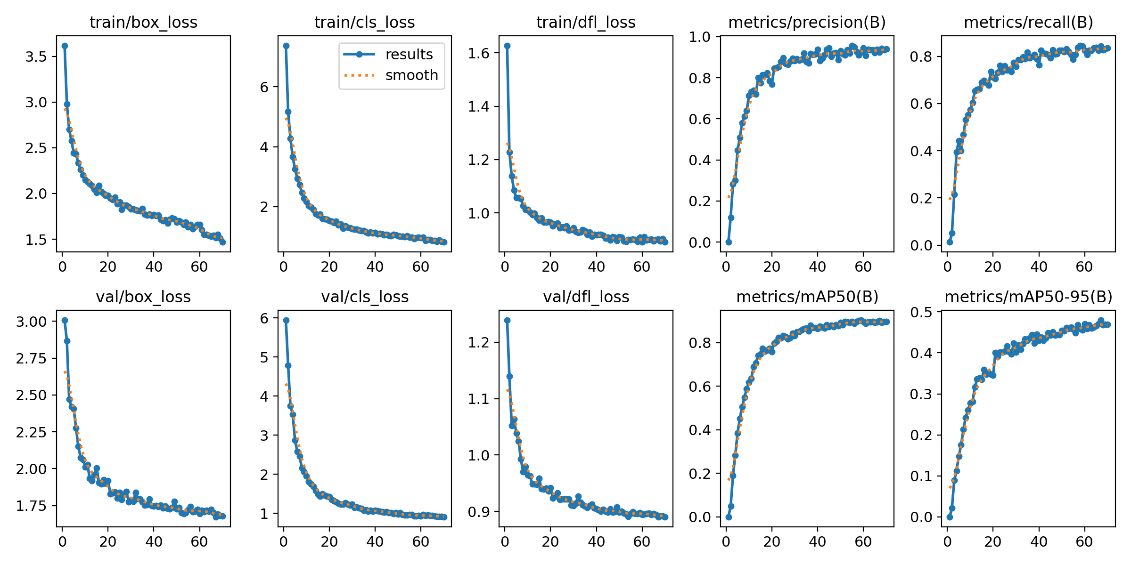
**Purpose:**

* The UI makes the system accessible for non-technical users, including mentors, testers, and operators.
* It provides a smooth, user-friendly experience with responsive design, clean formatting, and clear output visualization.
* This ensures quick testing, easy defect verification, and convenient sharing of model inference results.

A screenshot of a computer

AI-generated content may be incorrect.**Confusion Matrix** (Normalized):

**Accuracy Charts** : Display training and validation accuracy/loss curves to show how well

A graph of different colored lines

AI-generated content may be incorrect.A graph of a graph

AI-generated content may be incorrect.**Frequency of Each PCB Defect Type :**

A bar chart with numbers and symbols

AI-generated content may be incorrect.

**Sample Images of batch 2:**

A collage of green circuit boards

AI-generated content may be incorrect.

**Potential Applications & Industry Impact**

**Potential Applications**

* Automated PCB inspection in large-scale manufacturing for continuous in-line quality monitoring.
* Quality assurance checks for small or medium PCB production batches.
* Pre-shipment verification to minimize customer returns and field failures.

**Industry Impact**

* **Reduced manual effort and errors:** Automated inspection is faster, more accurate, and eliminates inconsistencies in human checking.
* **Lower operational costs:** Fewer defective boards pass through the pipeline, reducing rework, scrap, and warranty claims.
* **High scalability:** The system can be integrated directly into conveyor belts or existing production monitoring setups.
* **Improved traceability:** Annotated images and downloadable inspection reports support defect analysis and process optimization.

**Conclusion :**

The project successfully developed an automated PCB defect detection system using YOLOv8. It accurately identifies and localizes multiple defect types, supports real-time inspection through a Streamlit interface, and generates annotated outputs and reports. The system reduces manual effort, improves accuracy, and can be scaled for industrial applications, demonstrating both practical utility and efficiency in quality control.