PCB DEFECT DETECTION USING YOLOv8

Project Overview

This project implements an automated **Printed Circuit Board (PCB) defect detection system** using the **YOLOv8 object detection model**. The complete pipeline includes dataset preparation, annotation conversion, model training and fine-tuning, inference on PCB images, and a user-friendly **Streamlit frontend** connected to a **FastAPI backend**.

The final system detects PCB defects, draws bounding boxes, generates analytical graphs, and provides **downloadable reports in CSV, TXT, PDF, and ZIP formats**. The objective is to reduce manual inspection effort and improve consistency and accuracy in PCB quality control.

---

1. Detailed Analysis – Graphs and Charts

1.1 Training Curves (Loss and Metrics)

The training process was analyzed using:

- Training loss curve
- Validation loss (where available)
- Metric curves such as **Precision**, **Recall**, and **mAP@0.5**

Interpretation:
The training loss decreased rapidly during the initial epochs, indicating that the model learned basic PCB defect features quickly. As training progressed, the loss curve flattened, showing convergence. Validation mAP@0.5 improved gradually; saturation or slight decline during later epochs suggests mild overfitting, which can be addressed using additional data and augmentation.

1.2 Detection Distribution

- A **histogram** shows the number of detected defects per class.
- A **pie chart** represents the relative proportion of defect classes.

Interpretation:
Class imbalance was observed, with some defect types appearing less frequently. This imbalance affects recall for minority classes and highlights the need for targeted data augmentation or additional labeled samples.

1.3 Per-Image Detection Examples

Sample PCB images with predicted bounding boxes and confidence scores were analyzed.

Interpretation:
Correct detections exhibit tight bounding boxes and confidence scores above 0.5. Missed or inaccurate detections typically occur for very small defects or visually ambiguous regions, indicating scope for higher-resolution training and preprocessing.

1.4 Confusion Matrix

A confusion matrix was used to evaluate class-wise prediction behavior.

Interpretation:
Certain defect classes such as *Short* and *Spur* show partial confusion due to visual similarity. This insight helps guide further dataset refinement and augmentation strategies.

---

2. Comprehensive Explanation of Model Building

2.1 Dataset Preparation

- The dataset was inspected to remove corrupted files.
- A total of **693 valid annotated PCB images** were used.
- XML annotations were converted to **YOLO TXT format** containing normalized bounding box coordinates.
- The dataset was split into **training and validation sets** (approximately 80/20).

Reasoning:
Clean annotations and a proper split are essential to ensure correct supervised learning and reliable evaluation.

2.2 Model and Training Setup

- **Model Used:** YOLOv8 (Ultralytics)
- **Approach:** Transfer learning using pretrained weights
- **Image Size:** 640 × 640

- **Optimizer:** AdamW
- **Learning Rate:** Tuned for stable convergence
- **Augmentation:** Mosaic loading, flips, and rotations

Reasoning:
YOLOv8 offers a good balance between accuracy and speed. Transfer learning reduces training time and improves performance on limited datasets.

2.3 Inference and User Interface

Inference is performed on uploaded PCB images. The system outputs:

- Annotated images with bounding boxes
- Tabular defect details
- Graphical analysis
- Downloadable reports

A Streamlit UI enables non-technical users to interact with the system easily.

---

3. Evaluation Metrics

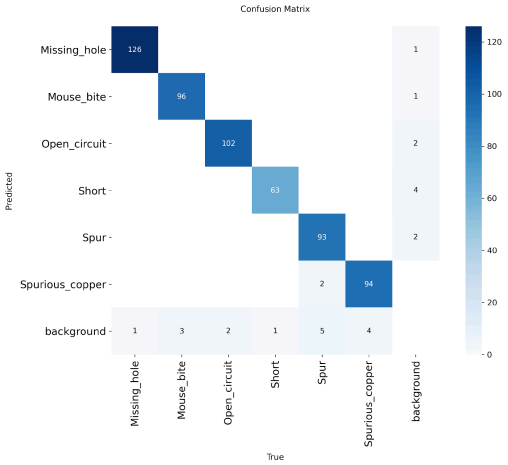
The following metrics were used:

- **Precision:** Correct predictions out of total predictions
- **Recall:** Detected defects out of actual defects
- **mAP@0.5:** Primary detection accuracy metric
- **mAP@0.5–0.95:** Stricter averaged metric

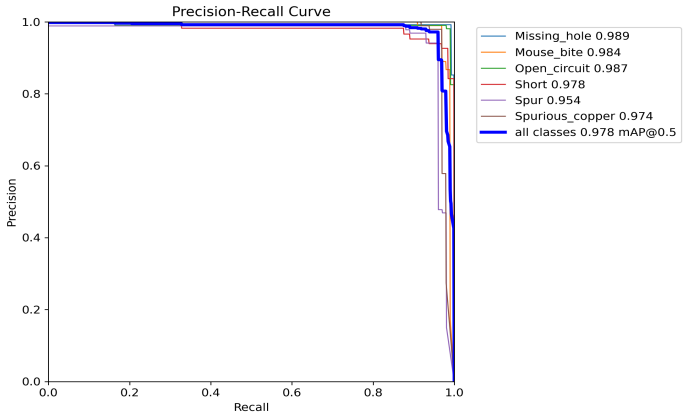
These metrics are automatically computed during YOLOv8 validation.

3.1 Display the figures:

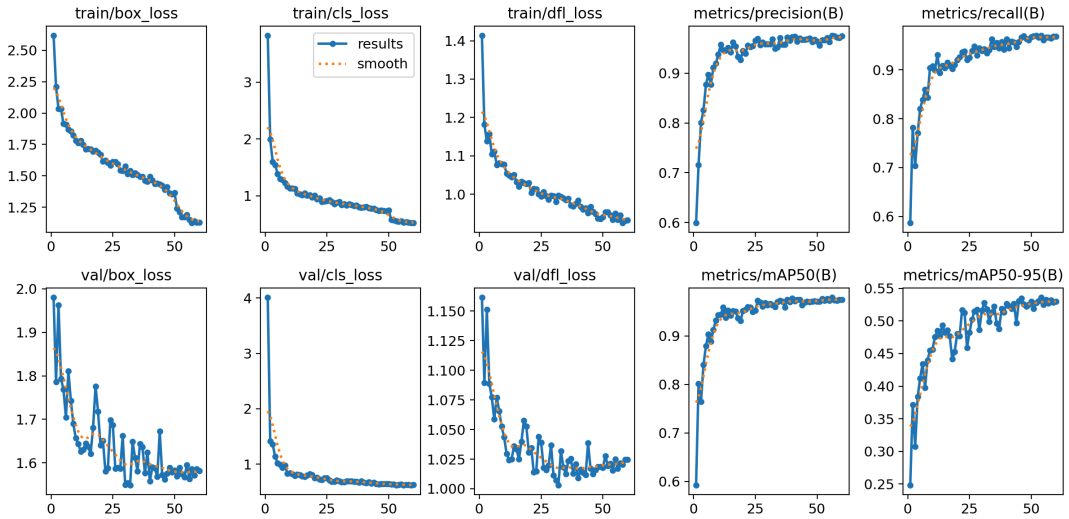
Confusion matrix.



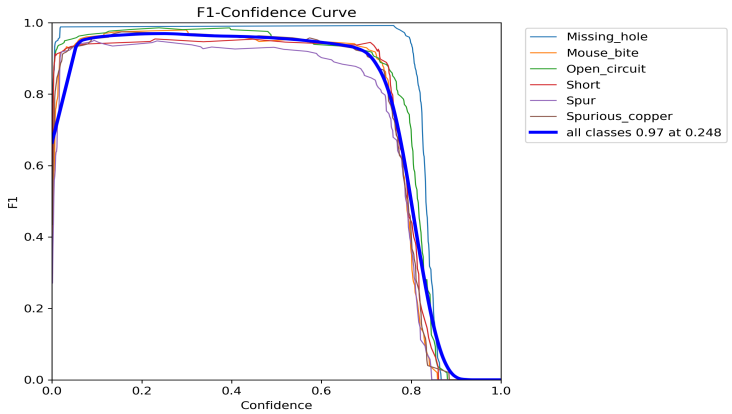
Precision-Recall



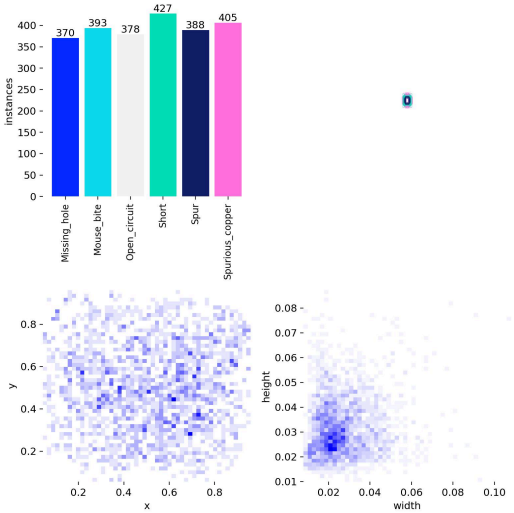
Training/validation loss and metrics.



F1 score confidence score.



Histogram of detected defects by class



---

4. Frontend and Backend Implementation

4.1 Backend Implementation (FastAPI)

The backend handles all computation and file management tasks.

Responsibilities:

- Receive uploaded images
- Run YOLOv8 inference
- Extract defect class, confidence, and bounding box coordinates
- Save annotated images
- Generate CSV, TXT, PDF, and ZIP reports
- Serve downloadable files via API endpoints

Job-based processing is implemented using unique job IDs to avoid file conflicts and enable scalability.

4.2 Report Generation

The backend generates:

- **CSV reports:** combined and per-image
- **TXT reports:** combined and per-image
- **PDF reports:**
    - One page per image
    - Original vs annotated image (side-by-side)
    - Defect table
    - Confidence distribution graph
    - Defect count graph
- **ZIP archive:** containing all reports and annotated images (for multiple images)

4.3 Frontend Implementation (Streamlit)

The frontend provides a clean and professional interface.

Key features:

- Multiple image upload support
- Single-row file display with remove option
- Annotated image grid view
- Defect analysis graphs
- Detailed defect table
- Structured download section

4.4 Download Management

The UI provides:

- Horizontal pop-down sections for:
    - CSV
    - TXT
    - PDF
- Each section includes:
    - Combined report
    - Separate reports per image
- ZIP download option
- Reset option to clear session data

---

5. Challenges Faced and Solutions

Challenge	Solution
Small dataset and class imbalance	Data augmentation and re-sampling
Tiny defect localization	Higher resolution training and preprocessing
Overfitting	Learning-rate tuning and regularization

Challenge	Solution
Streamlit reruns during downloads	Session state management

---

6. Recent Fine-Tuning Updates

- Increased training epochs
- Unfroze final backbone layers
- Batch XML-to-YOLO conversion
- Tested multiple augmentation strategies

Incremental improvements were observed; additional data is required for further gains.

---

7. Potential Improvements (95–99% Accuracy Target)

- Collect more labelled data for rare defects
- Hard-negative mining
- Model ensembling
- Image subtraction and morphological preprocessing
- Hyperparameter optimization

---

8. Applications and Industry Impact

Applications:

- Automated PCB inspection
- Manufacturing quality assurance
- Pre-shipment defect verification

Industry Impact:

- Reduced inspection time
- Lower human error
- Improved consistency and traceability
- Cost reduction due to fewer defective products

---

9. Deliverables

- Trained YOLOv8 model (best.pt)
- Annotated output images
- Streamlit frontend application
- FastAPI backend
- Downloadable reports (CSV, TXT, PDF, ZIP)

---

10. Conclusion

This project successfully delivers a **complete end-to-end PCB defect detection system** using YOLOv8. The integration of deep learning, backend automation, and an intuitive frontend provides an effective solution for automated PCB inspection. While current results are satisfactory, further dataset expansion and fine-tuning can significantly improve performance for industrial deployment.