MLPR PROJECT END SEM PRESENTATION

PCB DEFECT DETECTION USING ML

Prepared by group CTRL Z CREW (**Group 13**)

Members

Manish Bisht

Saksham Bali

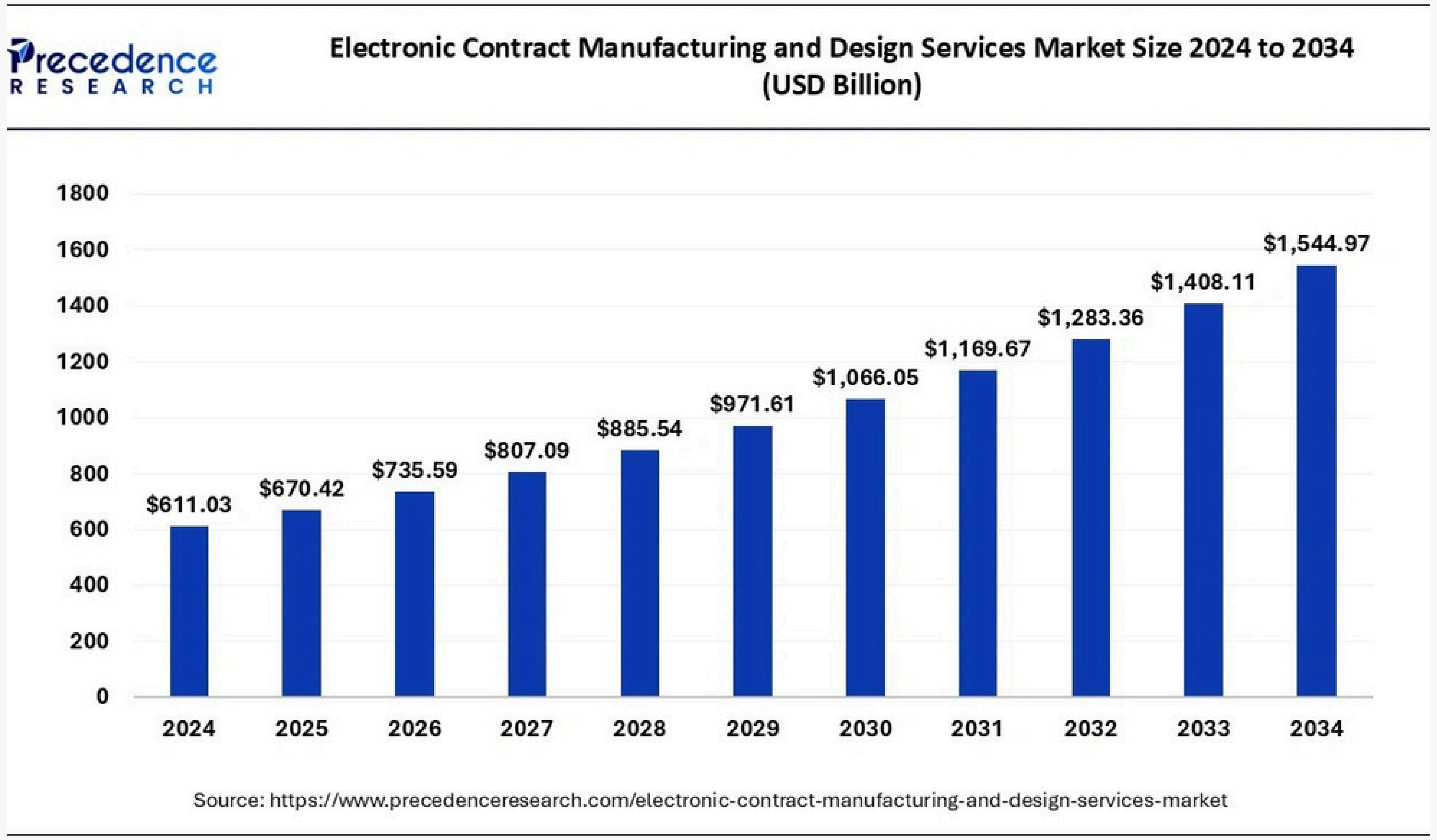
Sanskar Sengar

IMPORTANCE OF PCBs

With the increasing demand of PCB's due to the rise in demand for high performance computing components like GPUs, CPUs in AI and Software Industry, smartphones, computers, and gaming consoles in consumer industry and other industries like medical industry, Automotive industry, Aerospace and Defence Industry, etc., there is a proportional increase in the need to reduce the defects in PCB manufacturing and to detect any defects to prevent any accidents. Moreover, ensuring product quality mandates meticulous defect

inspection, a task exacerbated by the heightened precision of contemporary circuit boards, intensifying the challenge of defect detection. [1]

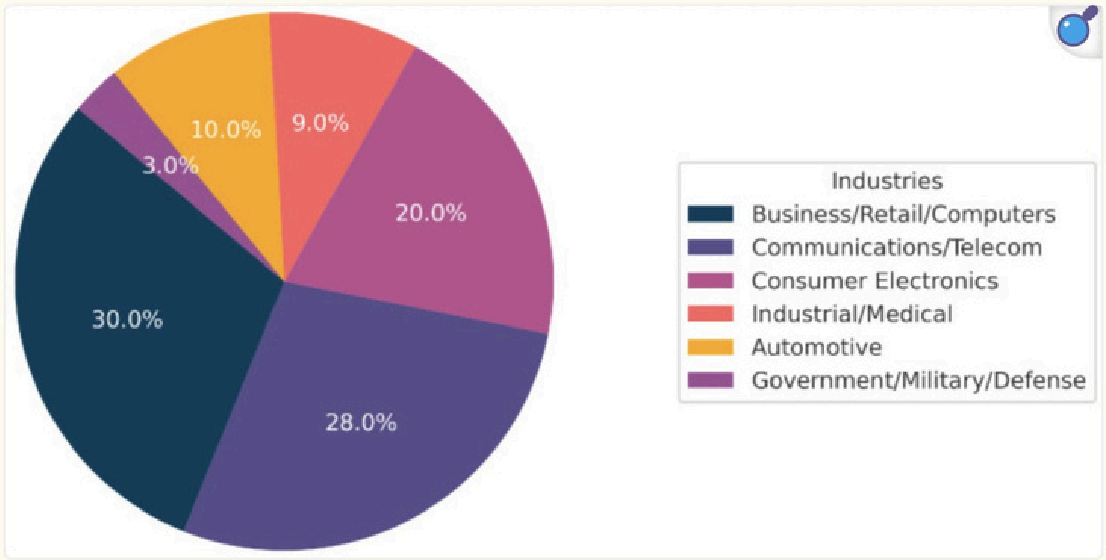
X. Chen, Y. Wu, X. He and W. Ming, "A Comprehensive Review of Deep Learning-Based PCB Defect Detection," in IEEE Access, vol. 11, pp. 139017-139038, 2023, doi: 10.1109/ACCESS.2023.3339561.



The global electronic contract manufacturing and design services market size was accounted for USD 611.03 billion in 2024 and is expected to exceed around USD 1,544.97 billion by 2034, growing at a CAGR of 9.72% from 2025 to 2034.

[2] Precedence Research. Electronic contract manufacturing and design services market. https://www.precedenceresearch.com/electronic-contract-manufacturing-and-design-services-market.

Figure 1.



Open in a new tab

Proportions of PCB Applications in Various Fields.

DEFECTS IN PCBs

PCB can be categorized defects into two primary groups:

1) Functional defects that directly affect circuit operation and can cause complete failure.

2) Cosmetic defects that primarily impact appearance but may eventually lead to performance issues through abnormal heat dissipation or current distribution.

NEED TO DECREASE THESE DEFECTS

Decreasing these defects is crucial for reducing financial loss like if the defects are detected in bare PCBs (those without electronic components) or at the Solder Paste Inspection stage then reworking the PCB only costs one-tenth as compared to after the re-flow oven stage, reducing defects ensures better performance and longevity of end products and it is especially important that PCB has minimal to no defects in critical applications like automotive and medical devices, minimizing defects becomes crucial for safety and reliability.

ml

(Manual

Labour)

PROBLEM STATEMENT

PCB defects can compromise performance, increase costs, and reduce product reliability. Traditional inspections are labor-intensive and expensive, while deep learning solutions demand high computational resources. This project aims to develop a classical ML-based defect detection model using feature extraction and oversampling techniques to improve accuracy, scalability, and efficiency for manufacturers.

ML

(Machine

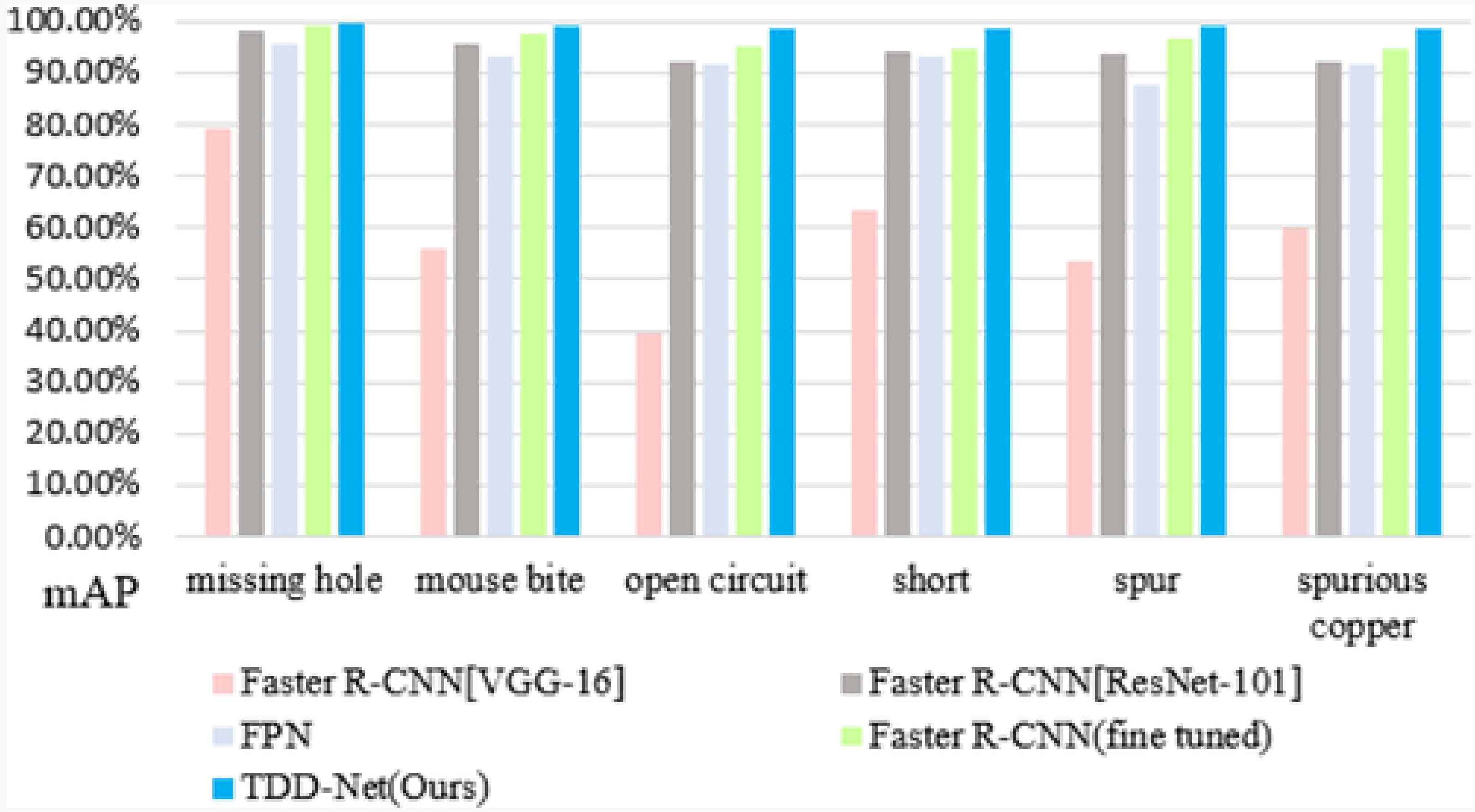
Learning Models)

Literature Review

BarePCB state of art

Key Features of TDD-Net:

- Lightweight Architecture: Designed to be computationally efficient, making it suitable for real-time applications and deployment on devices with limited resources.
- High Accuracy: Achieves a mean Average Precision (mAP) of 98.90% on standard PCB defect datasets, outperforming several existing methods.



Limitations :

1. Simplified Assumptions: The model may rely on assumptions that oversimplify real-world complexities, such as static demand patterns or perfect market conditions.
2. Scalability Concerns: The approach might face challenges when scaling up to larger systems with numerous variables and constraints.
3. Integration with Real-Time Data: Limited consideration for real-time data integration, which is crucial for dynamic decision-making in electricity markets.
4. No time based criteria used for assessment

Model	Backbone	Anchors	Feature	Head	mAP@0.5
Faster R-CNN [[8]]	VGG-16	2k	the last layer	2fc	58.57%
Faster R-CNN [[8]]	ResNet-101	2k	C5	2fc	94.27%
FPN [[14]]	ResNet-101	2k	{Pk}	2fc	92.23%
Faster R-CNN(fine-tu	ResNet-101	2k	C5	2fc	96.44%
TDD-Net(Ours)	ResNet-101	2k	{Pk}	2fc	98.90%

Citation:-

TDD-net: a tiny defect detection network for printed circuit boards Runwei Ding, Linhui Dai, Guangpeng Li, Hong Liu

https://doi.org/10.1049/trit.2019.0019

SoldefAI: State of Art

Model Used:-

- Mask R-CNN – used by the authors for PCB defect segmentation.
- YOLO – one-stage detector mentioned as a faster alternative
- SSD – another one-stage detector cited for comparison

Gap Analysis:-

- No inference-time metrics (FPS/latency) are reported – real-time speed is only discussed qualitatively (YOLO/SSD are noted to be faster).
- Moderate accuracy: the Mask R-CNN model achieves only ~62.1% mAP on the PCB defect task.

Table 9. Evaluation of AP metric—dataset_1.

Metric	IoU	Area	maxDets	Value [%]
Detection				
mAP	@IoU = 0.50:0.05:0.95	All	100	62.11
mAP	@IoU = 0.50	All	100	71.05
mAP	@IoU = 0.75	All	100	71.05
mAP	@IoU = 0.50:0.05:0.95	Small (area < 32^2)	100	NaN
mAP	@IoU = 0.50:0.05:0.95	Medium (32^2 < area < 96^2)	100	NaN
mAP	@IoU = 0.50:0.05:0.95	Large (area > 96^2)	100	62.113
Segmentation				
mAP	@IoU = 0.50:0.05:0.95	All	100	68.57
mAP	@IoU = 0.50	All	100	71.05
mAP	@IoU = 0.75	All	100	71.05
mAP	@IoU = 0.50:0.05:0.95	Small (area < 32^2)	100	NaN
mAP	@IoU = 0.50:0.05:0.95	Medium (32^2 < area < 96^2)	100	NaN
mAP	@IoU = 0.50:0.05:0.95	Large (area > 96^2)	100	68.57

Insipiration Point:-

Their work effectively emphasizes the need for task-specific datasets, such as capturing PCB defects under varied angles and lighting. Their use of Mask R-CNN for segmentation, along with per-class mAP evaluation, offers a detailed performance breakdown — useful inspiration for robustness testing and structured defect analysis.

Citation:-

Fontana, G., Calabrese, M., Agnusdei, L., Papadia, G., & Del Prete, A. (2024). SolDef_AI: An Open Source PCB Dataset for Mask R-CNN Defect Detection in Soldering Processes of Electronic Components. Journal of Manufacturing and Materials Processing, 8(3), 117. https://doi.org/10.3390/jmmp8030117

Evaluation Criteria

- **Mean Average Precision (mAP):** The mean of APs over all classes telling on average, denoting the mean value resulting from the integration of accuracy rates across distinct thresholds, spanning a recall range from 0 to 1.
- **FPS** serves as a metric for assessing inference speed, representing the quantity of images that can be processed per second on specific hardware. FPS holds significance as a key metric for gauging model performance and its applicability in real-time scenarios.
- The **F1 score** provides a balanced mean between precision and recall, effectively harmonizing the model's accuracy and recall
- Precision refers to the ratio of correctly predicted positive samples to the total predicted positive samples by the model

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

Citation : X. Chen, Y. Wu, X. He and W. Ming, "A Comprehensive Review of Deep Learning-Based PCB Defect Detection," in IEEE Access, vol. 11, pp. 139017-139038, 2023, doi: 10.1109/ACCESS.2023.3339561.

GAPS IN CURRENT METHODOLOGIES

Manual Labour

- To date about 45% industries detect the defects using traditional manual inspection methods which are highly labour-intensive, time-consuming, and prone to human error due to lower operating costs.

Existing models in industry

- Technologies like Solder Paste Inspection (SPI) and Automated Optical Inspection (AOI), still face challenges in achieving high accuracy, scalability, and adaptability across diverse PCB designs, as a result industries still need to use manual labor to verify these. But are still used due to time efficieny. However, since AOI is limited to surface-visible defects it becomes hard to catch hidden or functional faults, for which methods like X-ray imaging inspect internal solder joints and multilayer boards to reveal voids or misaligned connections and thermal imaging for detecting thermal anomalies (like in short circuit) were introduced in the industry. But each had its own gaps. The X-ray imaging is not efficient in detecting fine cracks or open solder joints and the problem with thermal imaging is that its resolution is relatively low, making it poorly suited to very small defects. Although all of these methods provided better inspection quality, most of them are limited by financial and time constraints. But the AOI can detect almost any type of defects in a faster and completely indestructible approach

1) Fonseca, L.A.L.O., Iano, Y., Oliveira, G.G.d. et al. Automatic printed circuit board inspection: a comprehensible survey. Discov Artif Intell 4, 10 (2024). https://doi.org/10.1007/s44163-023-00081-5

Challenges in Current Methodologies

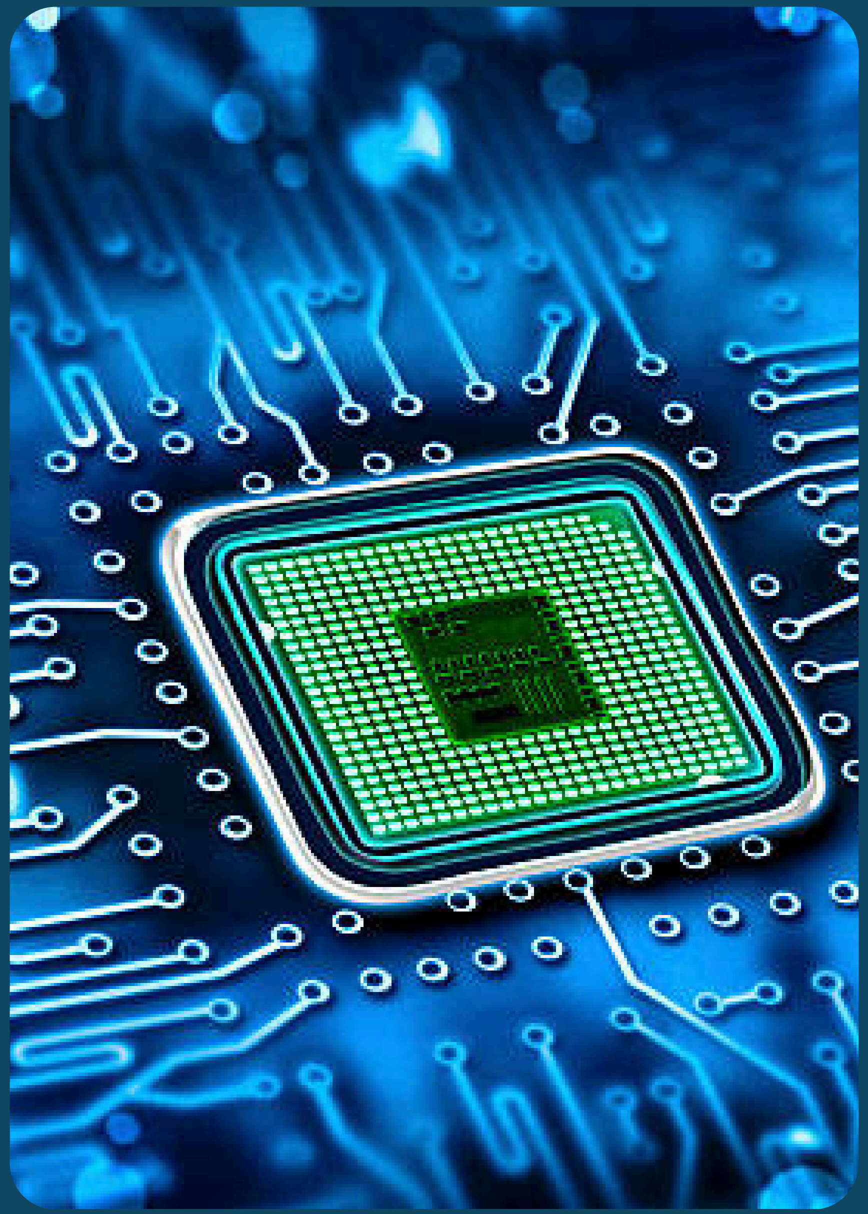
1) Data Imbalance: Minor defects occupy a small area on PCBs, leading to an imbalance between defective and non-defective samples during training leading to poor model performance for rare defect categories.

2) Feature Loss in Deep Networks: While deep learning models excel at feature extraction, the sometimes lose critical information about small defects due to down-sampling layers.

3) Computational Efficiency: Many state-of-the-art models like YOLOv3 or Faster R-CNN achieve high accuracy but are computationally expensive.

4) Not Scalable Across Different Designs : PCB as a technology are continuously evolving and the models start failing as soon as there is a change in its design and hence models need to retrained on new data frequently requiring the need of models that can be retrained efficiently at a scalable level without losing accuracy

5) Current models do not factor in time and most of them use no criteria relating to time in their models.



Data Set Study

Dataset consists images in .JPG format along with JSON file containing their data in it.

1. Bare PCB Inspection

- There are around 8534 training images of these.

For bare PCBs (before soldering), the following defects are examined:

- Missing Hole – A required hole is not drilled or plated.
- Mouse Bite – Small notches or irregularities along the PCB edges.
- Open Circuit – A break in the circuit traces, causing connectivity failure.
- Short Circuit – Unintended connections between traces leading to malfunctions.
- Spur – Unwanted protrusions in the copper traces.
- Spurious Copper – Unintended copper residue, which can cause electrical issues.

https://www.kaggle.com/datasets/akhatova/pcb-defects

2. Solder Paste PCB Inspection

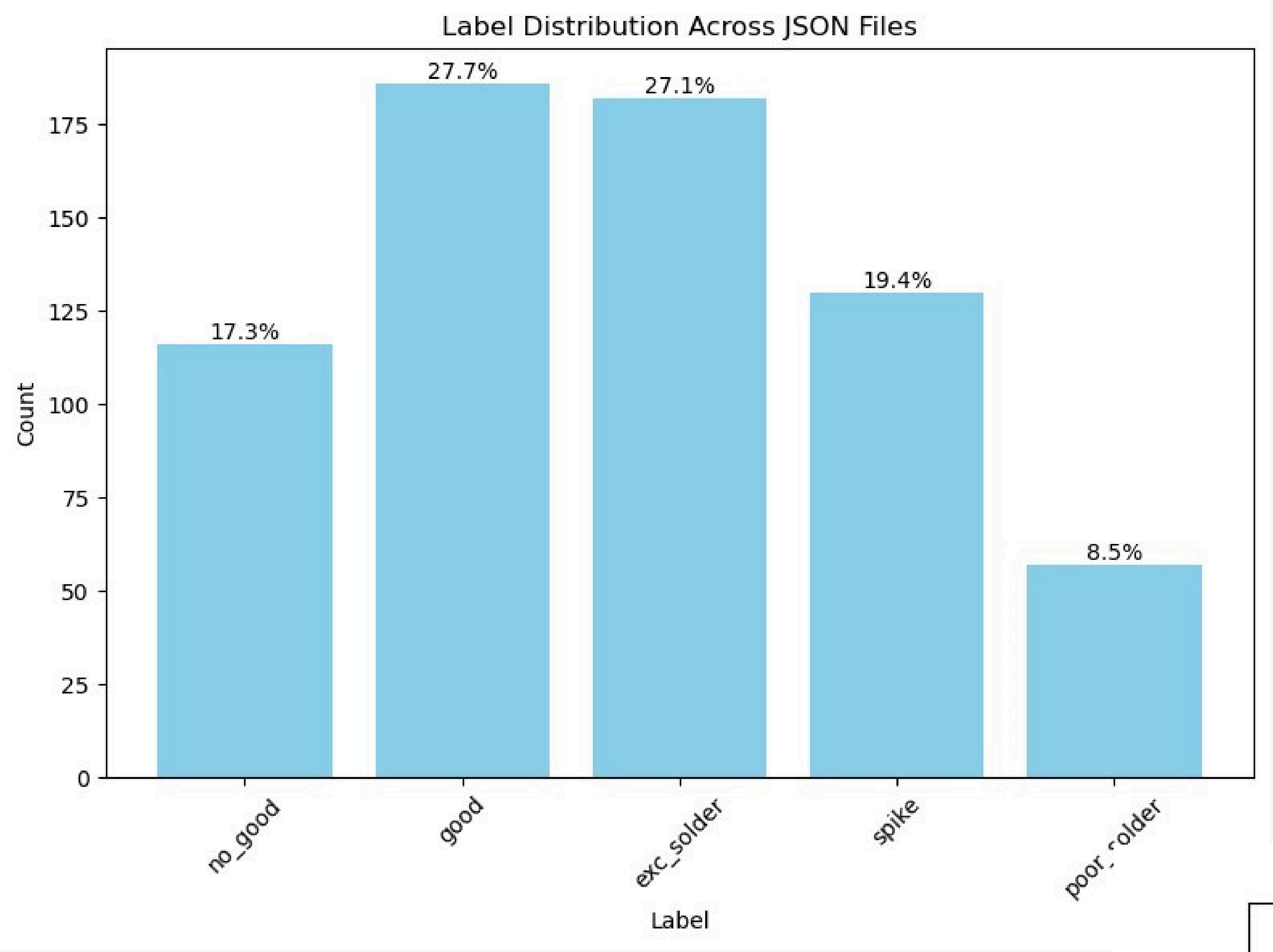
- This datset has around 2000 images in it.

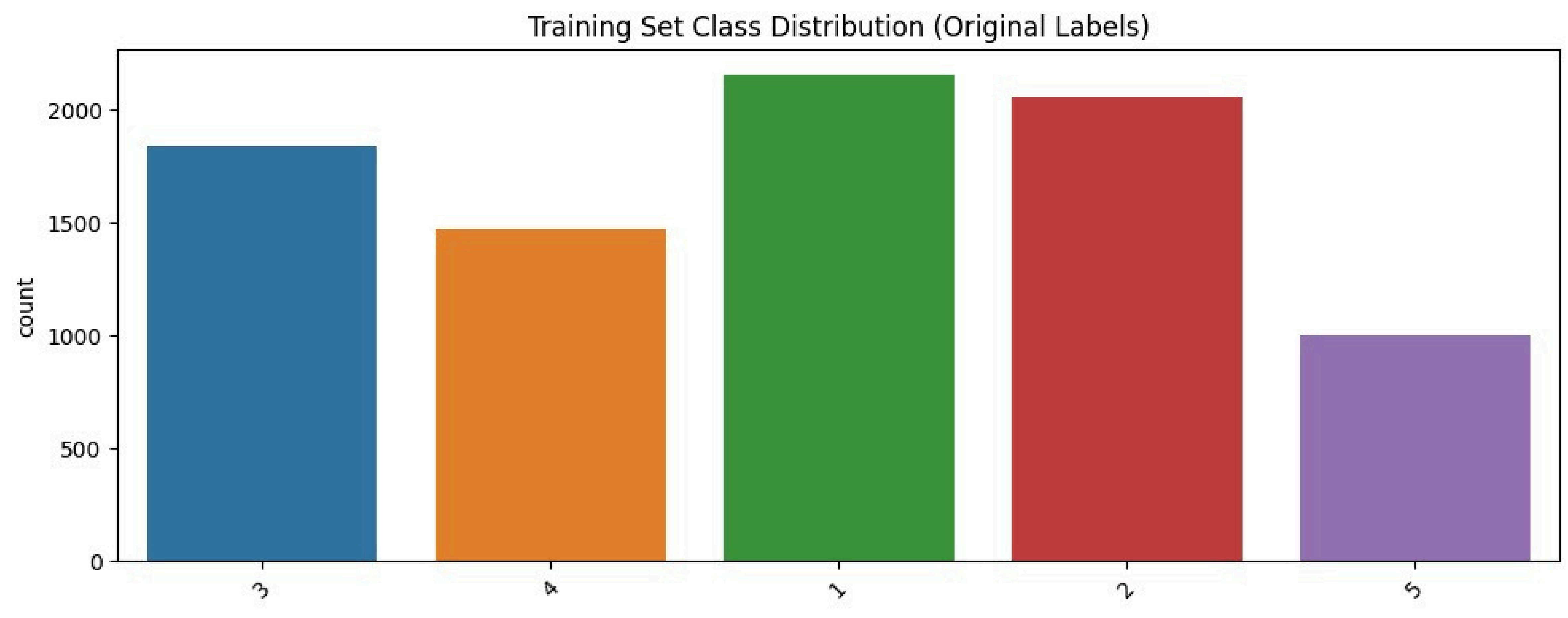
During the inspection of solder paste application on PCBs, the following categories are identified:

- Good Category – Proper solder paste application ensuring reliable connections.
- Excessive Solder – Too much solder, which can lead to bridging or weak joints.
- Not Good – General classification for defects in solder application.
- Poor Solder – Insufficient or uneven solder, leading to weak connections.
- Spike – Sharp solder formations that can cause short circuits.

https://www.kaggle.com/datasets/mauriziocalabrese/soldef-ai-pcb-dataset-for-defect-detection/data -

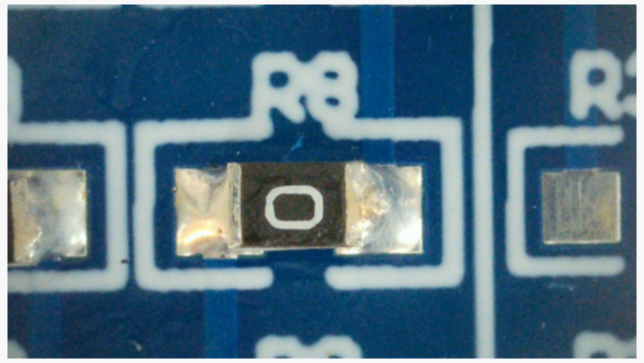
Data Set Study



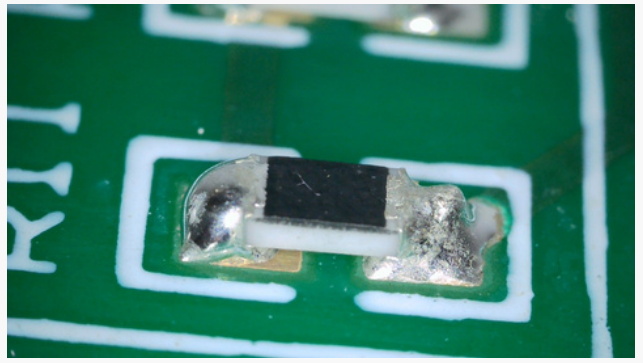


Dataset

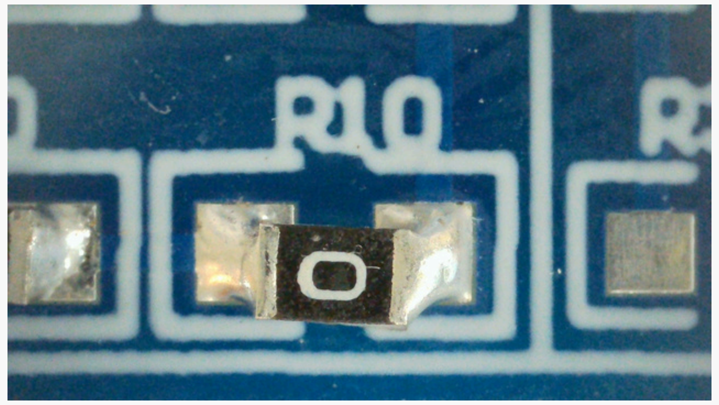
Solder Paste Dataset



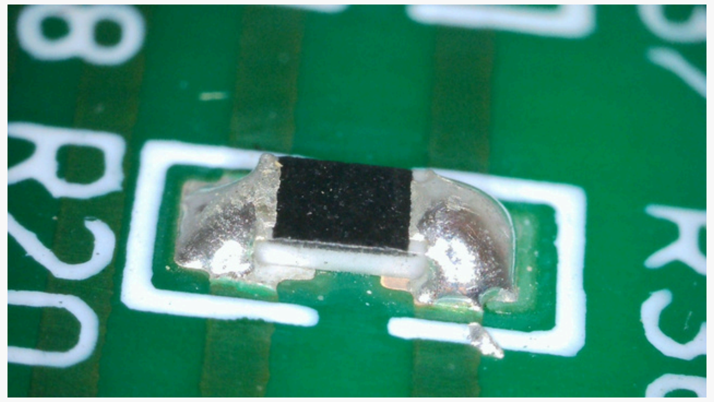
Good



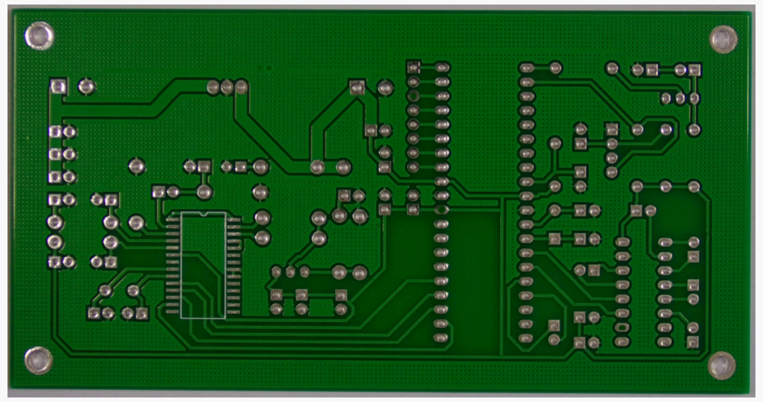
Spike



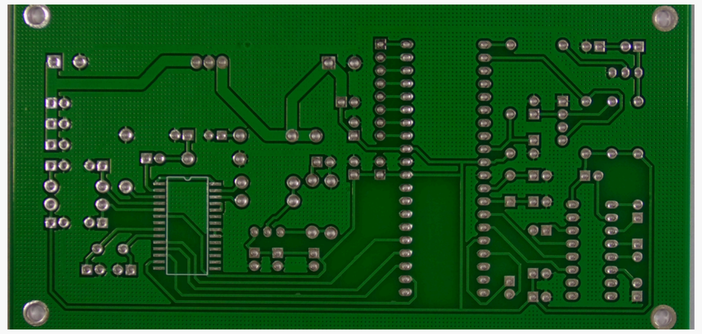
Not Good



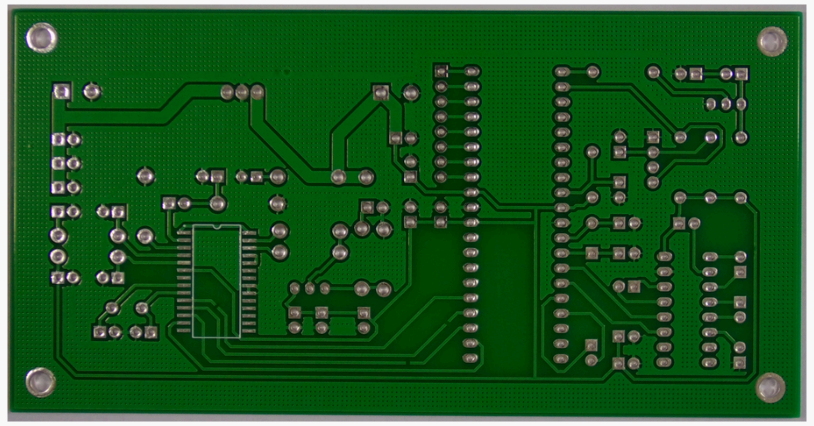
Bare PCB



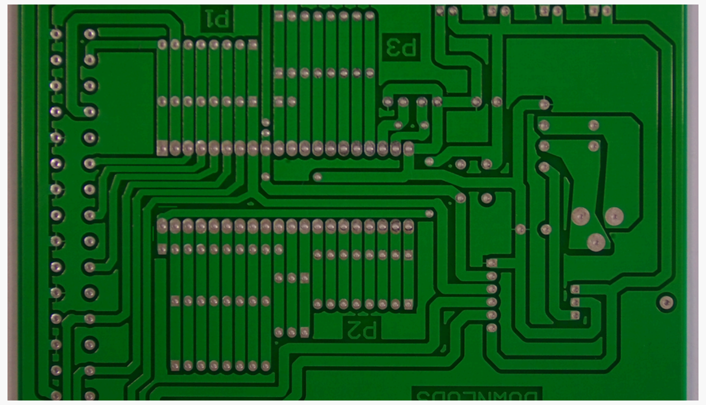
Missing Hole



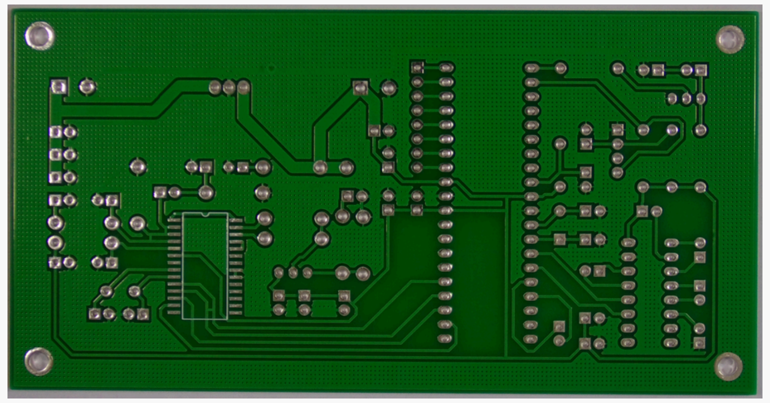
spur



open_circuit



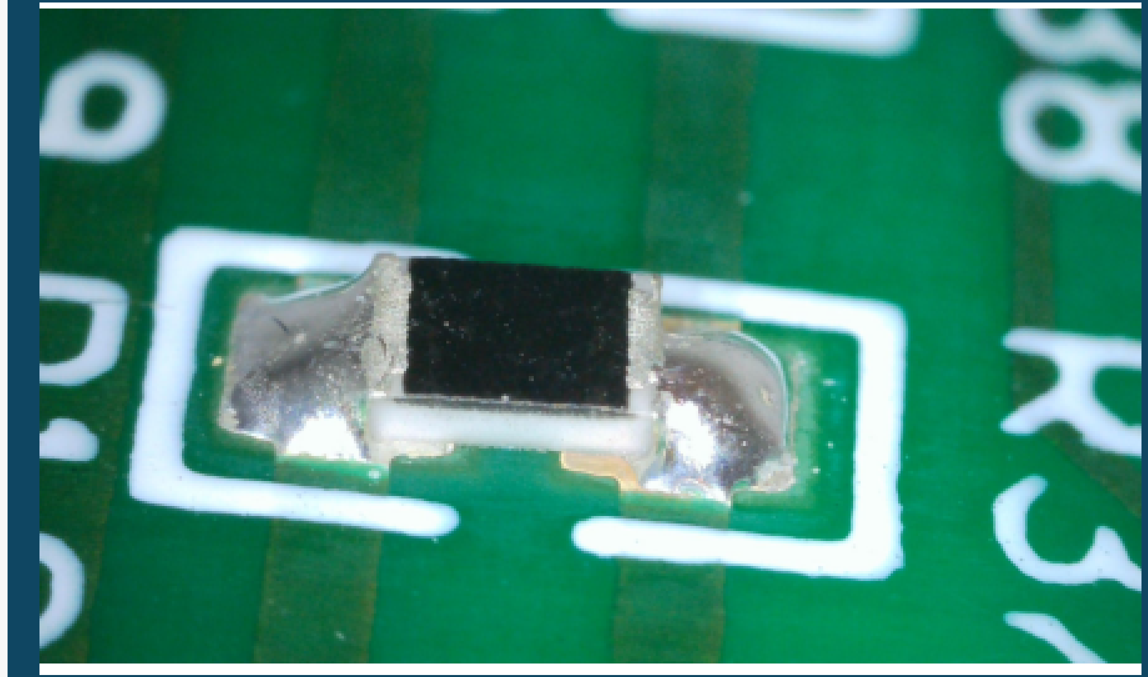
spurious_copper



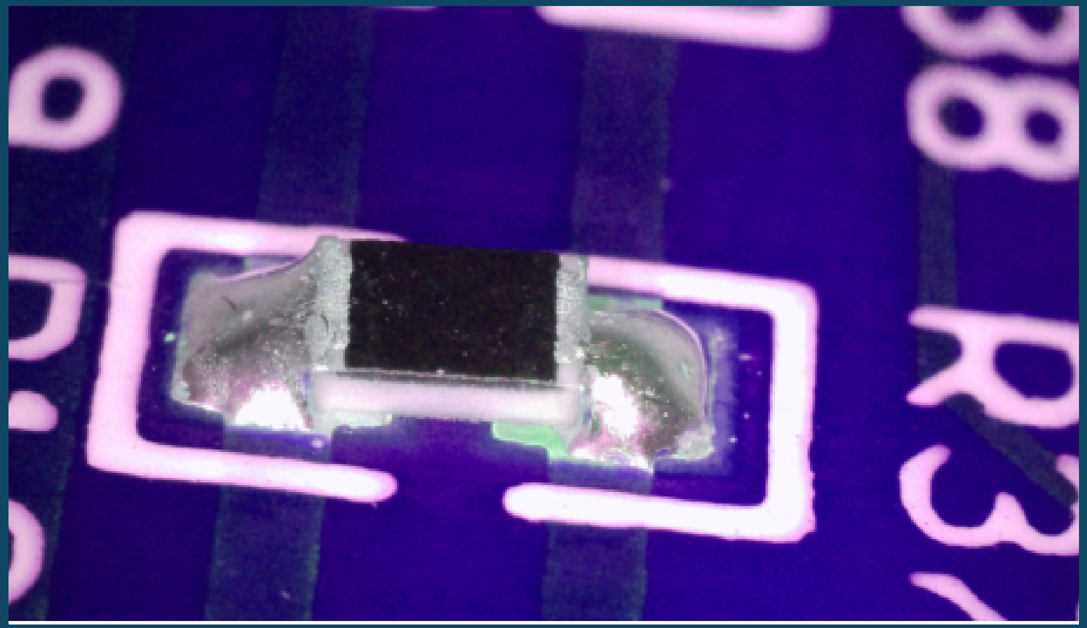
short

Pre-processing

Original



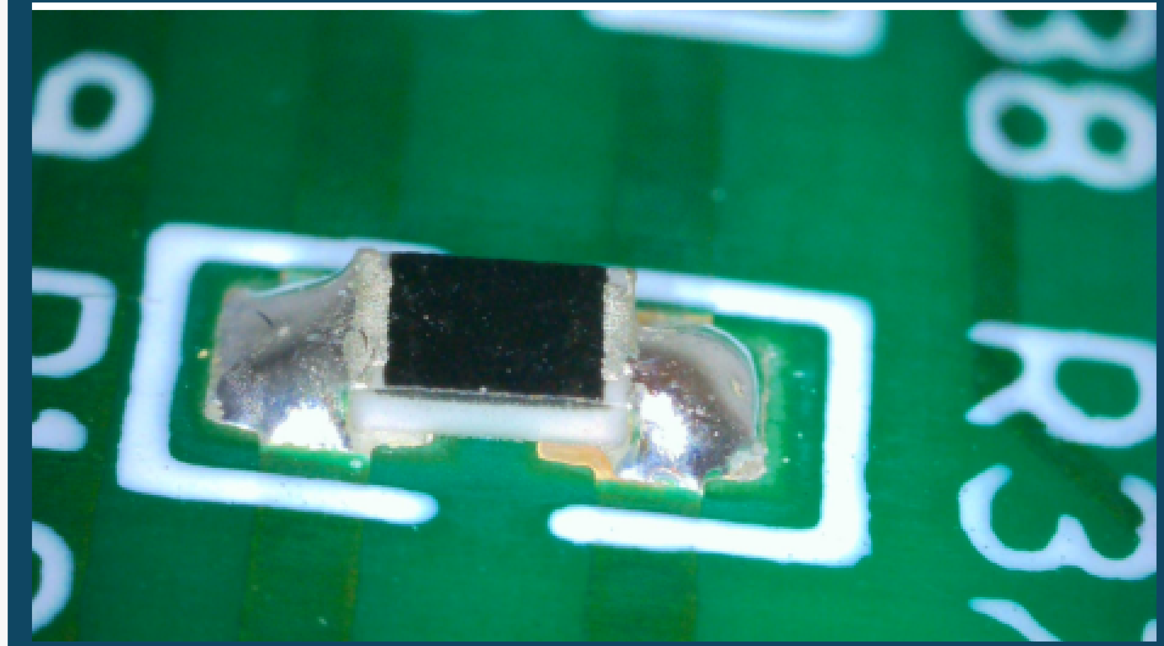
Hue



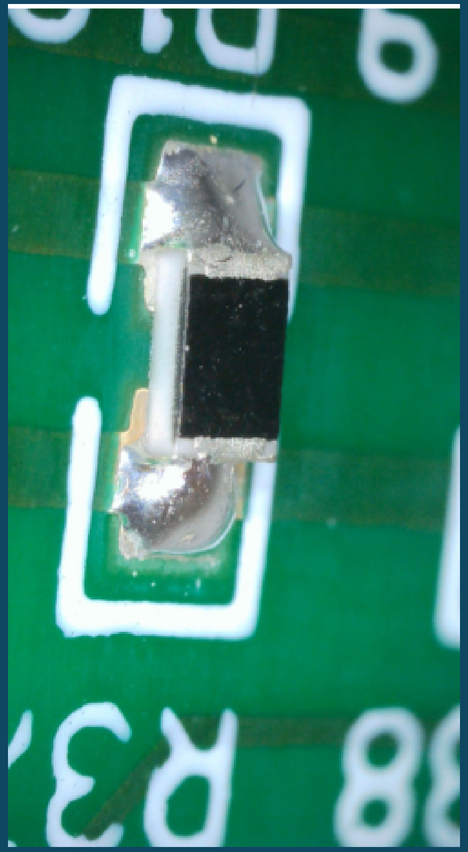
1) Since our dataset consists of images, we do not need to preprocess the data. 2) We are expanding our dataset by applying modifications such as rotating the images at different angles, adjusting the hue, and increasing saturation.
3) In future we may segment a single image into multiple images for better accuracy on minor defects.

Fonseca, L.A.L.O., Iano, Y., Oliveira, G.G.d. et al. Automatic printed circuit board inspection: a comprehensible survey. Discov Artif Intell 4, 10 (2024). Your paragraph text

Saturated



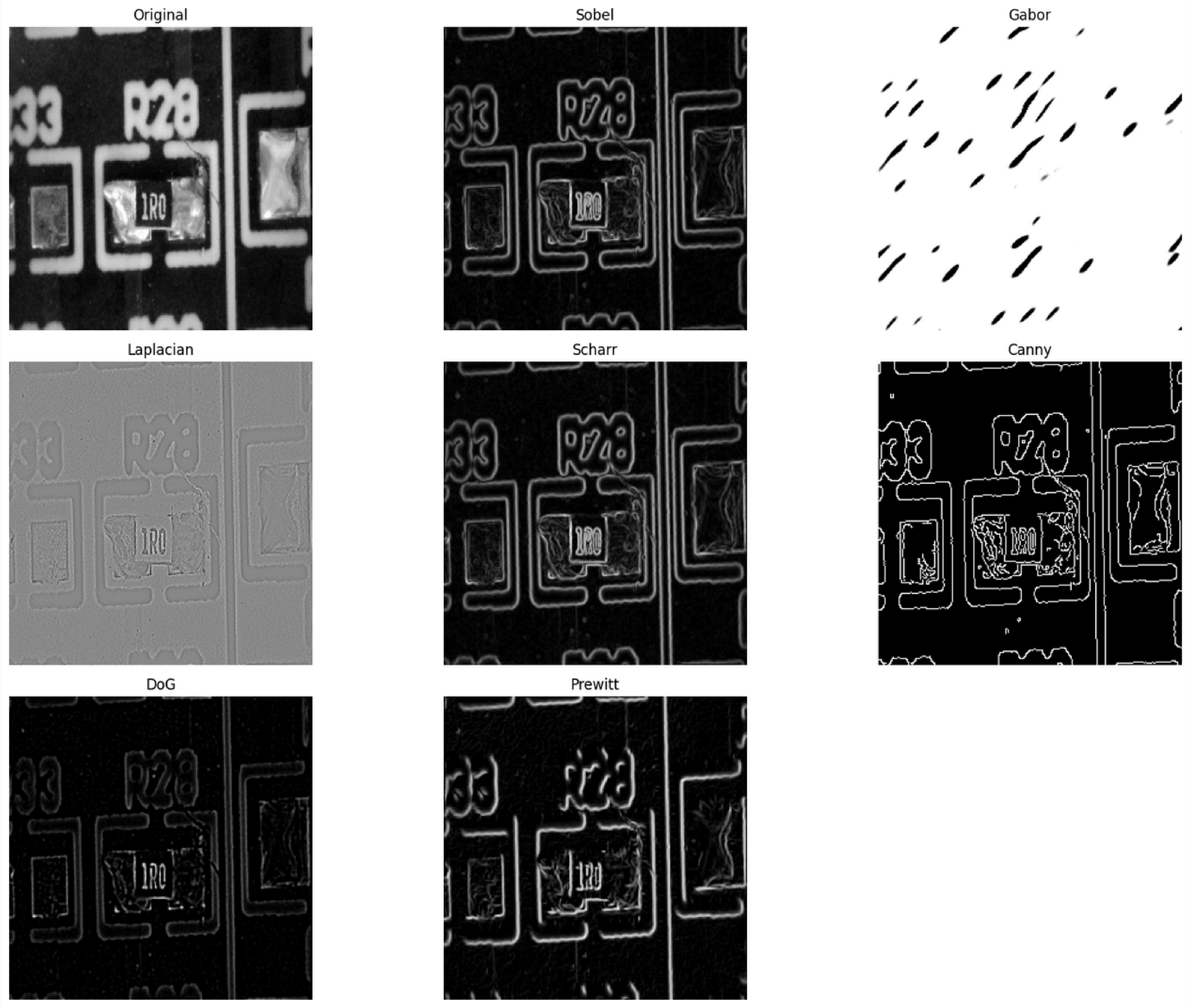
Rotated



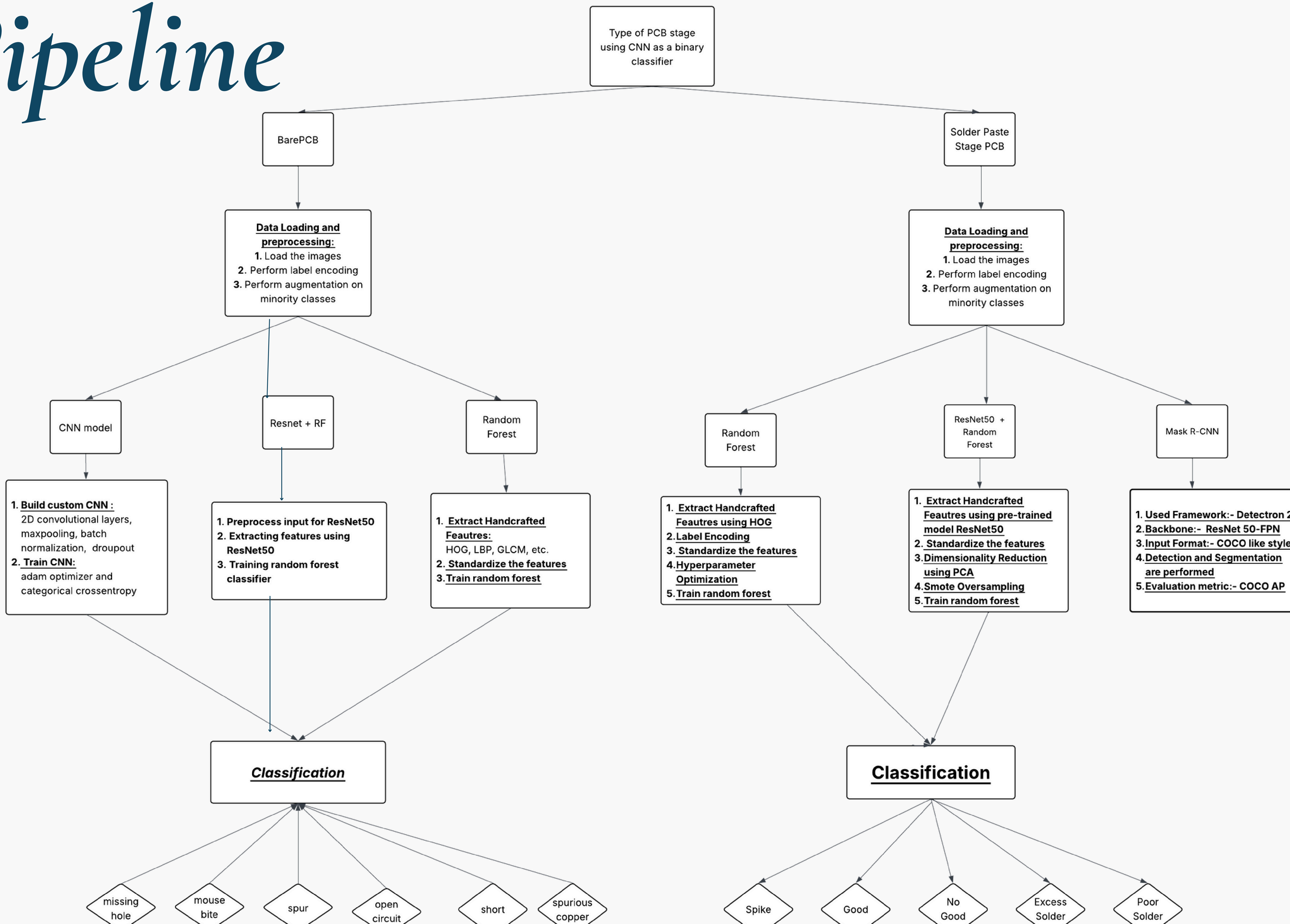
Dataset Preprocessing

We have used several methods to extract features which include:

- Histogram of Oriented Gradients
- Local Binary Pattern (LBP)
- GLCM (Gray Level Co-occurrence Matrix)
- Hu Moments
- Sobel Gradient Features
- Color Histogram
- Contour-Based Shape Features



Model Pipeline



CNN as a binary classifier

Data Loading

- Recursive scan of two folders (bare_pcb_folder, solder_pcb_folder)
- Images resized to 224×224 px, normalized to [0,1]
- Labels: 0 = bare PCB, 1 = solder PCB

Dataset Preparation

- Combined into X (images) and y (labels) arrays
- Stratified train/test split (80 % train, 20 % test, random_state=42)

Model Architecture

- Conv Block 1: Conv2D (32 filters, 3×3) → ReLU → MaxPool (2×2)
- Conv Block 2: Conv2D (64 filters, 3×3) → ReLU → MaxPool (2×2)
- Classifier: Flatten → Dense (64, ReLU) → Dense (1, Sigmoid)

Training & Evaluation

- Compiled with Adam optimizer, binary cross-entropy loss, accuracy metric
- Trained for 1 epoch (batch size 32) with validation on test set

BarePCB Models

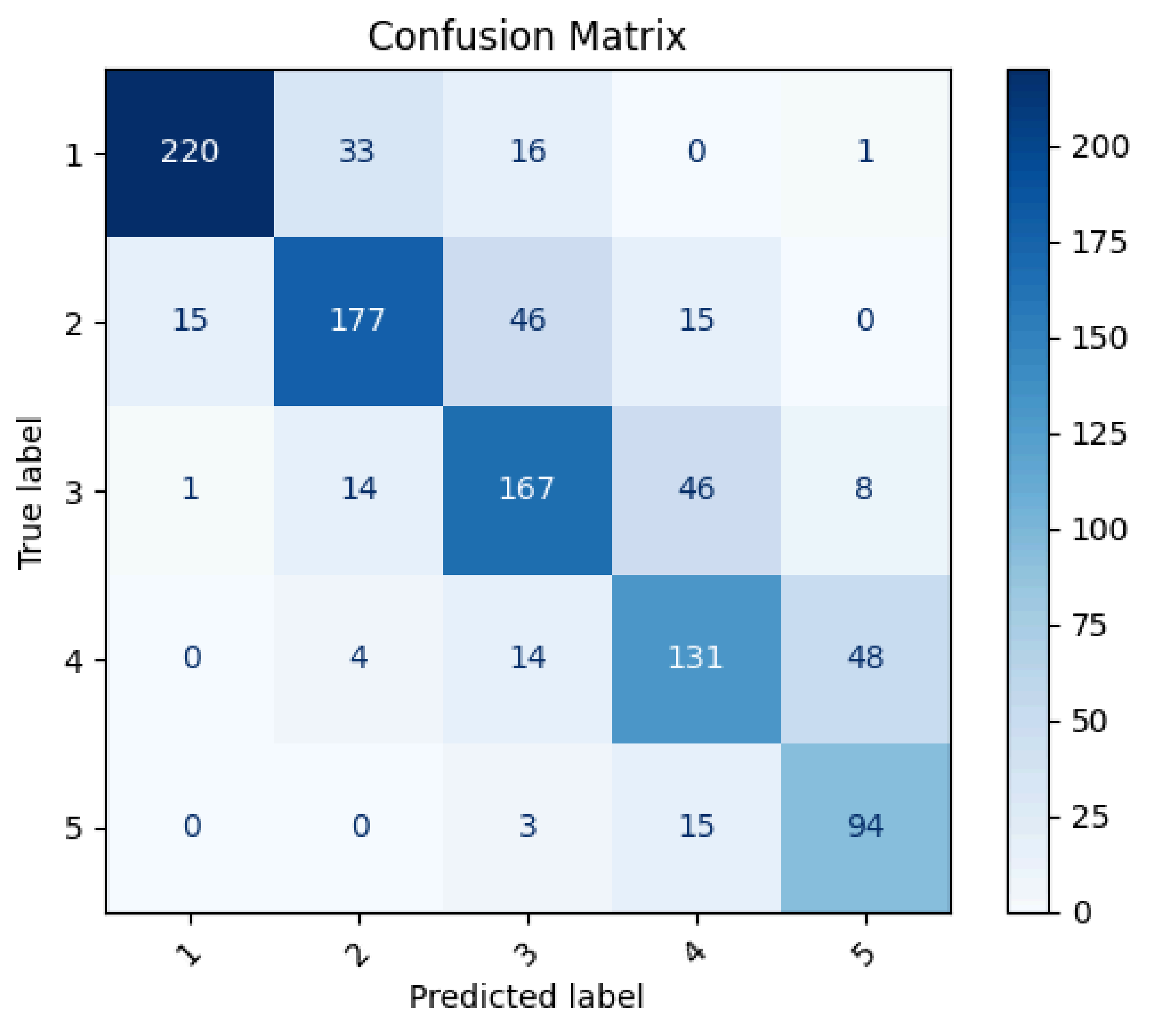
1) Random Forest Classifier

Architecture:

- Feature Extraction:
    - Each image (128×128) is transformed into a fixed-length numeric feature vector using a combination of kernels like HOG features, sobel, Hu moments, etc.
- Data Augmentation & Balancing:
    - Applied albumentations transforms to balance underrepresented classes.
    - All 5 defect classes have 2159 samples each.
- Random Forest Classifier
    - n_estimators: 300
    - max_depth: 30
    - Class weighting: Balanced
    - StandardScaler() for input normalization

Evaluation:

- Test Accuracy: 73.88%
- mAP: 0.8035
- AUC (OvR): 0.9491
- Inference Time: 2.79 s
- FPS: 3828



Class	Precision	Recall	F1-score
1	0.93	0.81	0.87
2	0.78	0.7	0.74
3	0.68	0.71	0.69
4	0.63	0.66	0.65
5	0.62	0.84	0.71

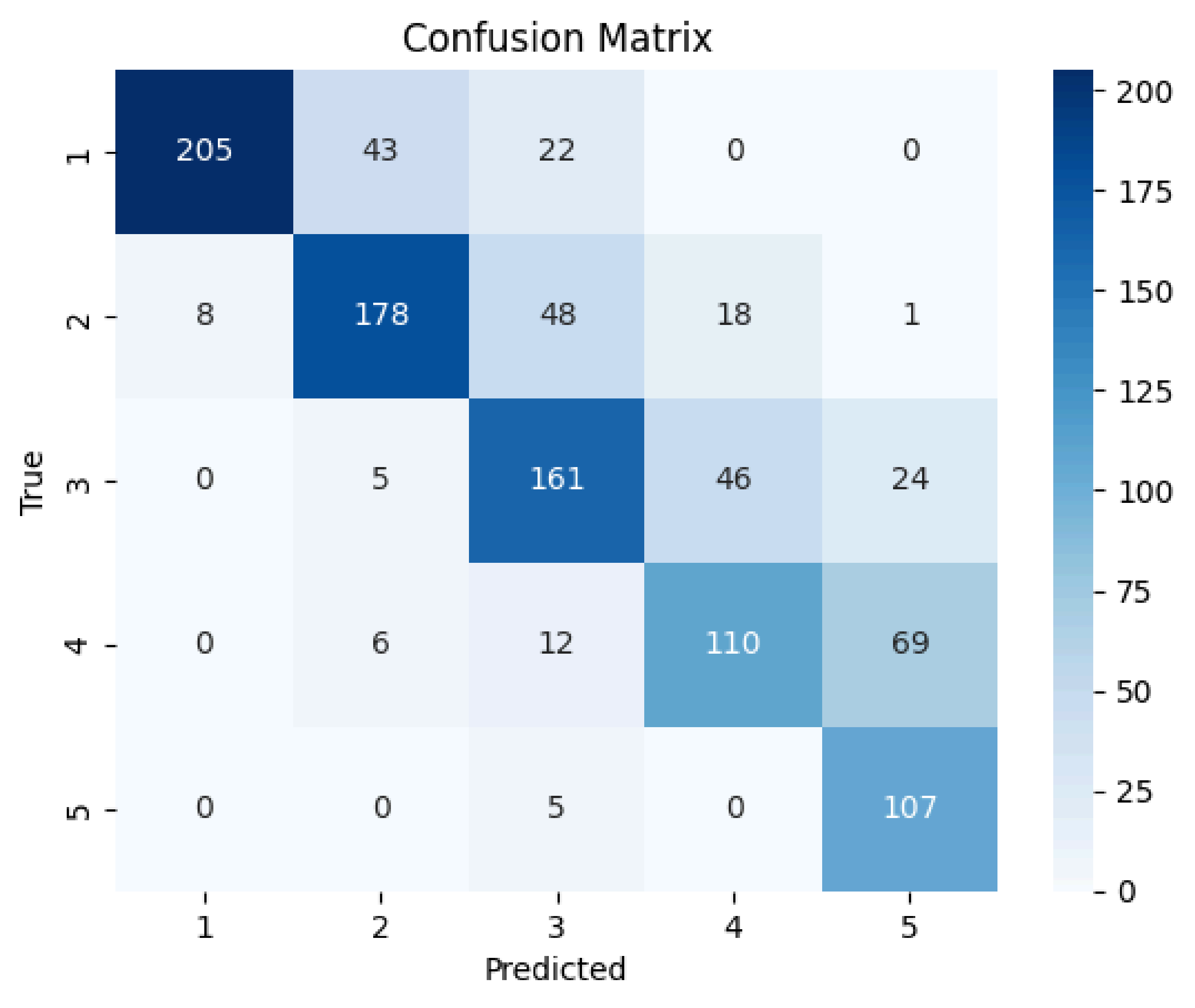
2) CNN

Architecture:

- CNN layers:
  - 3 Conv2D layers with ReLU + BatchNorm + MaxPool
  - Dense(256) → Dropout(0.4) → Output (softmax)
- Loss Function: Custom Focal Loss
- Minority class augmentation
- Optimizer: Adam (lr=1e-5)
- Early Stopping: Patience=5

Evaluation:

- Test Accuracy: 71%
- Macro F1-Score: 0.70
- mAP: 0.7592
- AUC (OvR): 0.935
- Inference Time: 5.66s (FPS: 188.85)



Per-class Precision:

1: 0.9624

2: 0.7672

3: 0.6492

4: 0.6322

5: 0.5323

Per-class F1-score:

1: 0.8489

2: 0.7340

3: 0.6653

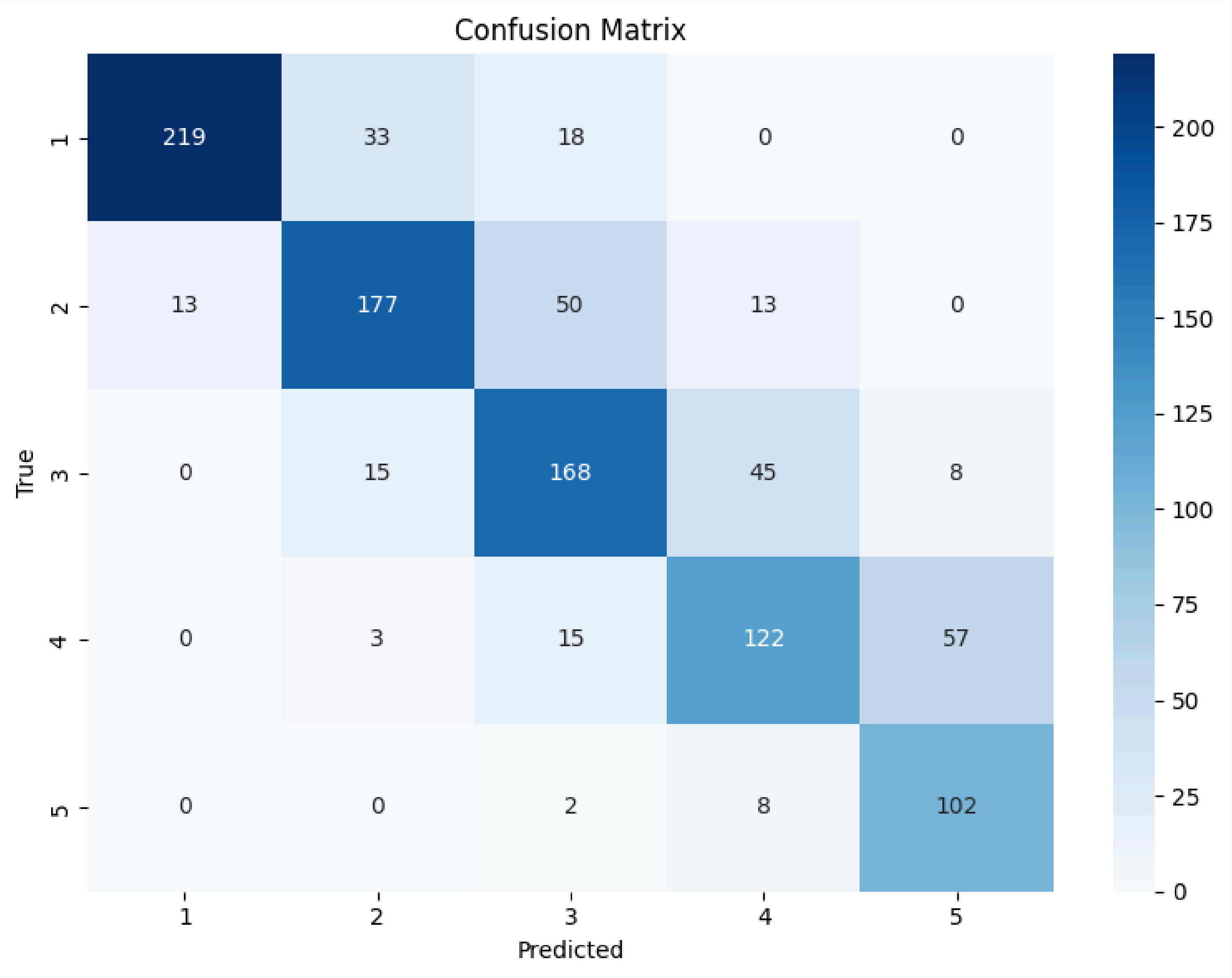
4: 0.5930

5: 0.6837

3) Resnet + Random Forest Classifier

Architecture:

- Feature Extraction using ResNet50
  - All images are resized to 224×224×3 to match ResNet50's input.
  - The last convolutional output is passed through GlobalAveragePooling2D() resulting in a 2048-dimensional vector for each image.
- Random Forest classifier for classification
  - n_estimators = 100 trees
  - random_state = 42 for reproducibility



Evaluation:

- Accuracy: 74%
- Mean Average Precision (mAP): 0.7748
- Overall AUC Score: 0.9423
- Inference Speed:
  - Total Time: 0.06s
  - FPS: ~18,134

Per-class Precision:

1: 0.944

2: 0.776

3: 0.664

4: 0.649

5: 0.611

Per-class F1-score:

1: 0.873

2: 0.736

3: 0.687

4: 0.634

5: 0.731

Solder Paste Models

Random Forest Classifier

Pipeline:-

1) Data pre-processing and Augmentation:

Each image is loaded using OpenCV and resized to a fixed size.

Albumentations library is used to perform:

i) Horizontal Flip (p=0.5)

ii) Random Brightness/Contrast (p=0.2)

iii) Rotation (±15 degrees, p=0.3)

2) Feature Extraction (HOG)

Grayscale conversion of the image as HOG does not require colour

Extracted Using:

i) 9 orientations

ii) 8*8 Pixels

iii) 2*2 block normalization

3) Label Encoding:

Labels extracted from JSON files are encoded into integers using LabelEncoder, suitable for scikit-learn classifiers

4) Hyperparameter Optimization:

Performed using GridSearchCV with 3-fold cross-validation on:

n_estimators: [100, 200] ; max_depth: [10, 20, None]

min_samples_split: [2, 5] ; min_samples_leaf: [1, 2]

max_features: ['sqrt', 'log2']

```
Classification Report (Random Forest + HOG + Augmentation)
              precision    recall  f1-score   support

  exc_solder       0.71      0.52      0.60        33
        good       0.52      0.76      0.62        66
     no_good       0.58      0.55      0.57        47
 poor_solder       1.00      0.15      0.27        13
       spike       1.00      0.47      0.64        15

    accuracy                           0.59       174
   macro avg       0.76      0.49      0.54       174
weighted avg       0.65      0.59      0.57       174


Average Precision (per class):
exc_solder: 0.7974
good: 0.7992
no_good: 0.7463
poor_solder: 0.3169
spike: 0.8026

Mean Average Precision (mAP): 0.6925
```

ResNet50 + Random Forest

Pipeline Description:-

- **Feature Extraction:** Each image was resized to 224x224 and passed through a pre-trained ResNet50 (without the top layer) to extract global average pooled features.
- **Standardization:** Features were scaled using StandardScaler.
- **Dimensionality Reduction:** PCA with 200 components was applied to reduce overfitting and computation.
- **SMOTE Oversampling:** The training set was augmented with synthetic samples to balance the class distribution.
- **Classifier:** A Random Forest with 300 trees and balanced_subsample class weighting was used.

Mean Average Precision

- **mAP (macro-averaged)**: **0.8683**

Inference Performance

- **Total Inference Time**: 0.0959 seconds for 162 images
- **Average Inference Time per Image**: 0.000592 seconds (~0.6 ms)

The pipeline is extremely fast, offering **near real-time classification performance**.

```
Classification Report:
              precision    recall  f1-score   support

  exc_solder       0.74      0.97      0.84        33
        good       0.62      0.41      0.49        32
     no_good       0.68      0.72      0.70        32
 poor_solder       0.97      0.97      0.97        33
       spike       0.97      0.94      0.95        32

    accuracy                           0.80       162
   macro avg       0.80      0.80      0.79       162
weighted avg       0.80      0.80      0.79       162

Confusion Matrix:
 [[32  0  0  0  1]
 [ 7 13 11  1  0]
 [ 1  8 23  0  0]
 [ 1  0  0 32  0]
 [ 2  0  0  0 30]]

Mean Average Precision (mAP): 0.8683

Total Inference Time: 0.0959 seconds
Average Inference Time per Image: 0.000592 seconds
```

MASK R-CNN

1. Dataset Preparation

Input Source: Labelled JSON and image files.

Two Datasets:

Dataset 1: ["good", "no_good"] (for binary classification)

Dataset 2: ["good", "exc_solder", "poor_solder", "spike"]

Steps:

i) Filter valid annotations based on class labels.

ii) Split into 80% training and 20% validation.

iii) Organize into /images/train, /images/val, /labels/train, /labels/val.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.893
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.945
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.945
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.893
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.946
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.967
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.967
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.967
```

2. Dataset Registration with Detectron2

Parsed annotations using shapes from LabelMe JSON format.

Converted to Detectron2 format (polygon + bbox).

Registered with DatasetCatalog and MetadataCatalog.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.691
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.932
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.786
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.500
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.692
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.666
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.780
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.782
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.500
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.783
```

3. Model Training & Evaluation

Model: Mask R-CNN (R-50 FPN) from Model Zoo.

Configuration:

BASE_LR = 0.00025, MAX_ITER = 1500, BATCH_SIZE = 2

Training: Performed using DefaultTrainer.

Evaluation: Used COCOEvaluator to compute mAP on validation set.

Why our Model is better?

1) Higher Accuracy (mAP)

Our models consistently achieve higher mean Average Precision across all defect types, even without relying on complex segmentation-based architectures like Mask R-CNN. This shows superior feature extraction and classification capability.

2) Faster Inference

Unlike their setup, we explicitly optimize and measure inference time, making our models ideal for real-time edge deployment. Like in ResNet50 model we have inference time of approx 0.6 ms.

3) Better Generalization

Our models also achieve high per-class performance, addressing class imbalance and hard-to-detect defects more effectively.

mAP in Random Forest Classifier:- 69.25%

mAP in ResNet50+Random Forest Classifier:- 86.83%

Detection (Bounding Box) AP

Metric	Custom Model	SolDef_AI Paper	Improvement
AP@0.5:0.95	69.34%	42.70%	**+26.64%**
AP@0.5	93.41%	56.90%	**+36.51%**
AP@0.75	78.90%	56.90%	**+22.00%**

Segmentation AP

Metric	Custom Model	SolDef_AI Paper	Improvement
AP@0.5:0.95	69.10%	48.30%	**+20.80%**
AP@0.5	93.15%	56.90%	**+36.25%**
AP@0.75	78.59%	56.90%	**+21.69%**

Thank you