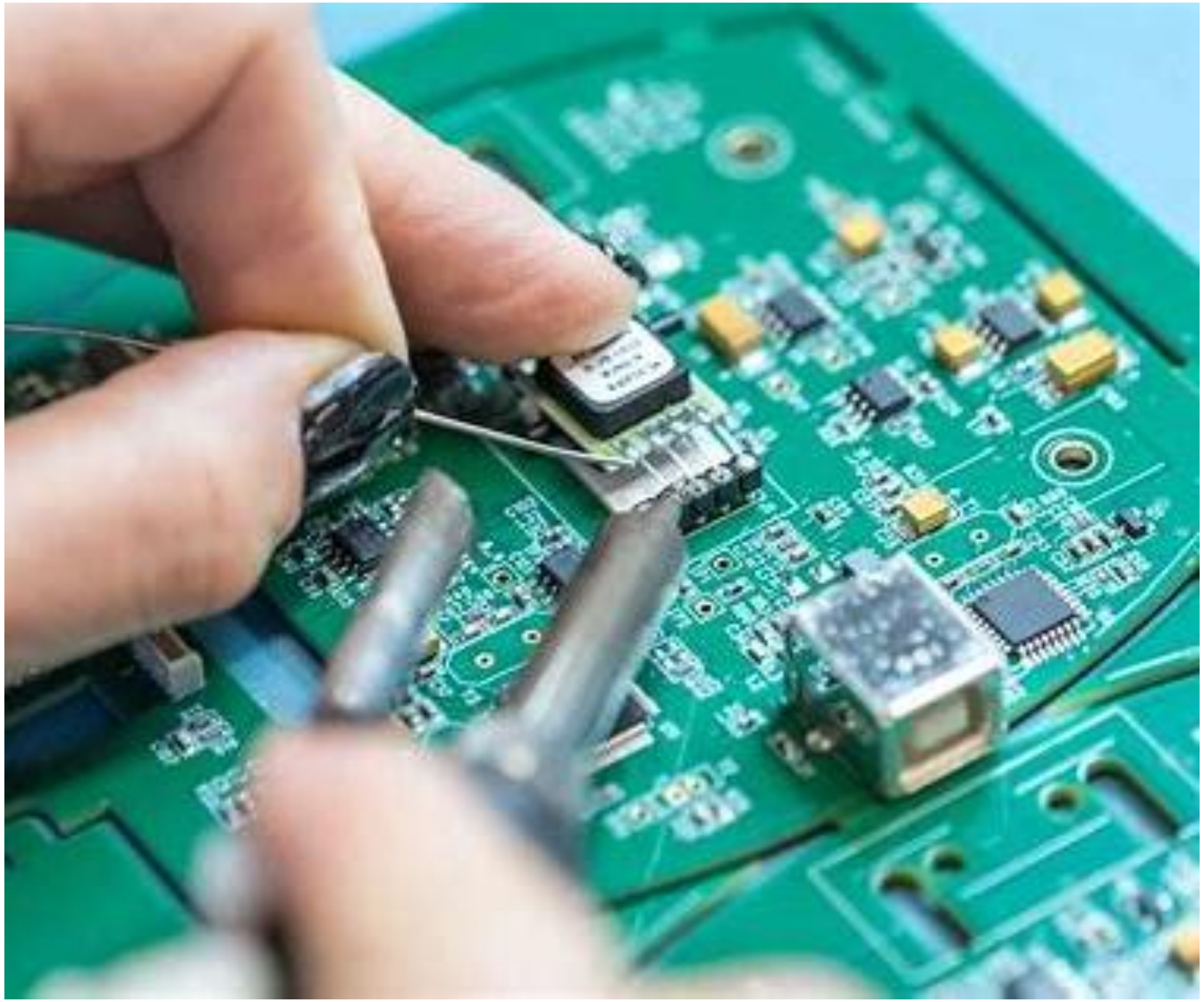
CROSS ANALYSIS OF TRANSFER LEARNING FOR PCB DEFECT DETECTION

Brandon Markham, John Gellerup and Josh Muniga

INTRODUCTION

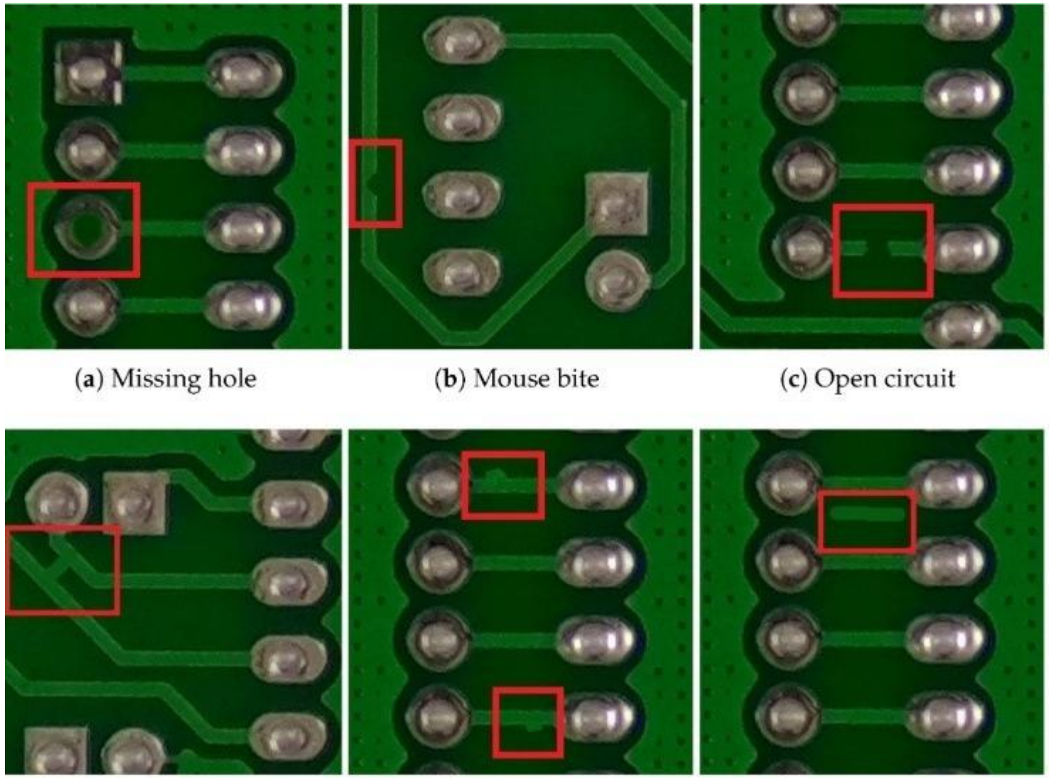
- PCBs are critical to nearly all modern electronic systems
- Manual inspection of PCB defects is slow, labor-intensive, and prone to error
- This project applies transfer learning to automate defect detection and classification, increasing efficiency and accuracy



PROJECT OBJECTIVES

- Evaluate 3 pretrained models: YOLOv8, SSD MobileNet, Faster R-CNN
- Use small, augmented dataset
- Compare models based on:
- Accuracy (Precision/Recall/F1)
- Speed
- Feasibility for real-world deployment

BACKGROUND



(a) Missing hole (b) Mouse bite (c) Open circuit

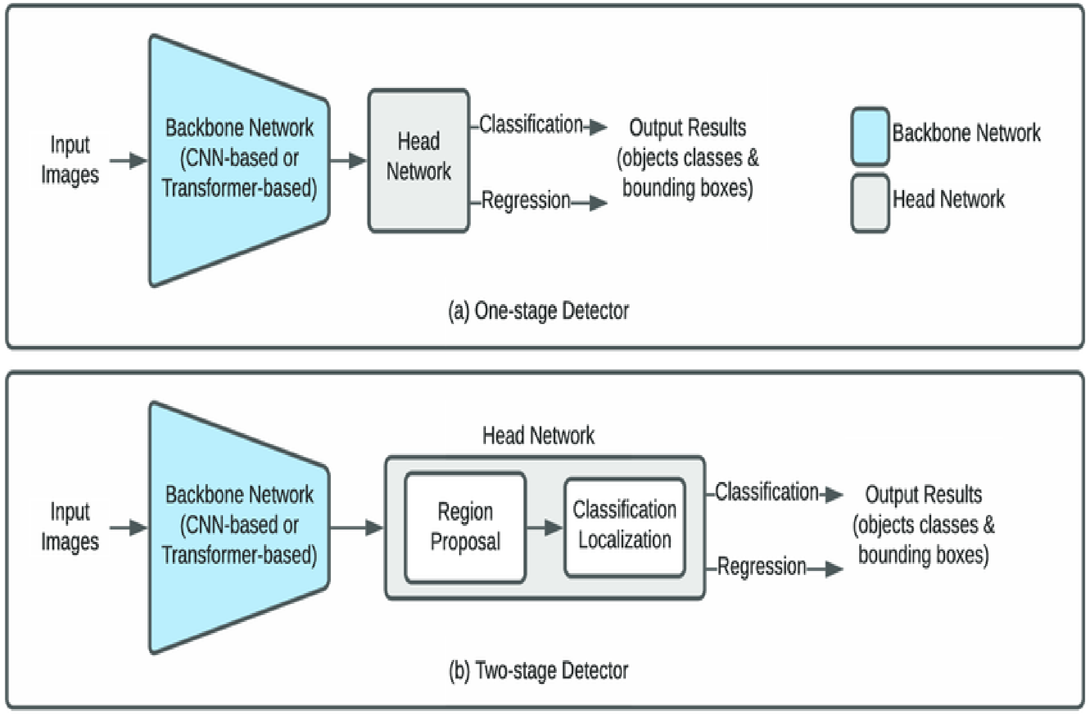
(d) Short (e) Spur (f) Spurious copper

- **Industry Challenge:** Small PCB defects

(shorts, spurs) can break entire systems

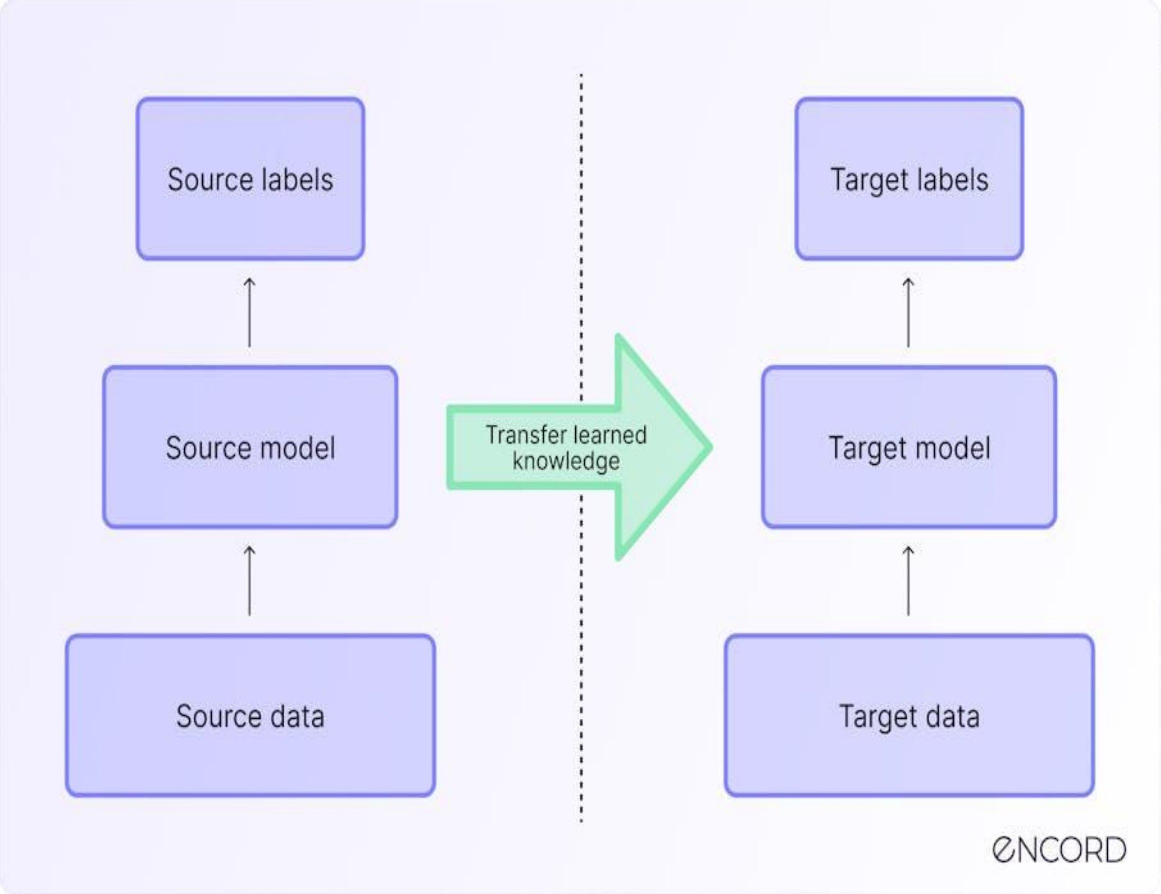
- **Current Method:** Human inspection – slow, error-prone, unscalable
- **Proposed Method:** Object detection to automate defect classification

OBJECT DETECTION TYPES



- **One-Stage Detectors**
  - Examples: **YOLOv8**, **SSD MobileNet**
  - Perform object classification and localization in a single forward pass
  - Optimized for **speed and real-time performance**
  - Ideal for applications where detection speed is critical
- **Two-Stage Detectors**
  - Example: **Faster R-CNN**
  - First stage: proposes candidate object regions (RPN)
  - Second stage: classifies and refines bounding boxes
  - **More accurate**, especially for small objects, but **slower**
  - Suited for use cases where **accuracy is more important than speed**

TRANSFER LEARNING



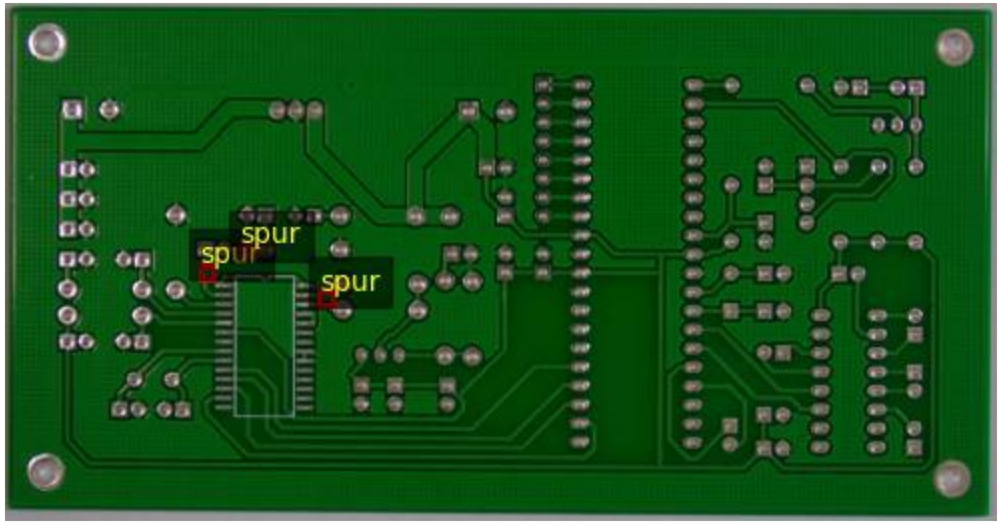
- Why use it? Saves time, data, and compute resources
- Leverages models pretrained on large datasets (COCO)
- We fine-tuned these models on our small PCB defect dataset
- Especially useful when collecting large, labeled datasets is impractical

DATASET OVERVIEW

- Source: Kaggle
- 693 original images, 7 classes (6 defects + 1 non-defective)
- Augmented to **~1,636 images**
- Annotations: Pascal VOC (XML), then converted to YOLO/TFRecord formats

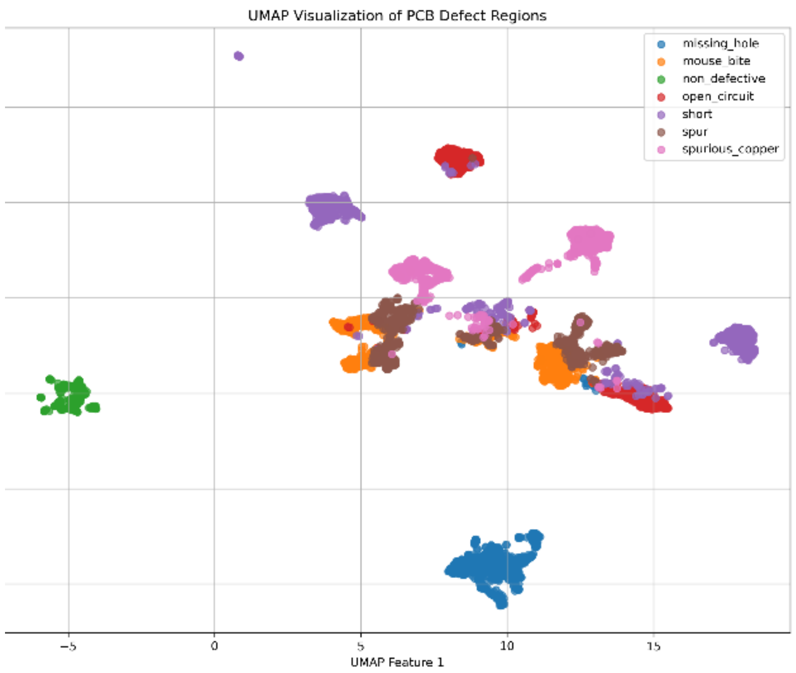
DATA PREPROCESSING

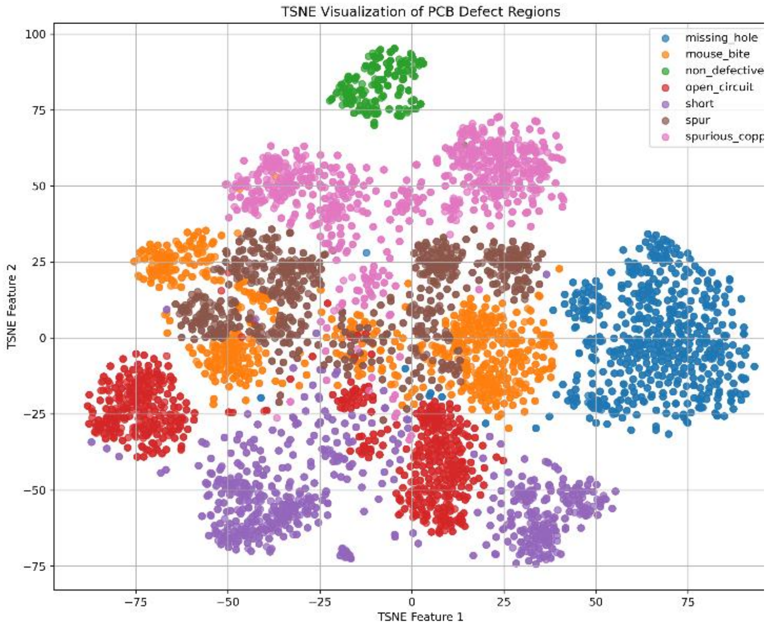
- Tools: LabelImg, Albumentations
- Augmentations: rotations, resized to 320/640/800 px
- Dataset split: 60/20/20 (Train/Val/Test)



DATA VISUALIZATION

- UMAP and t-SNE plots show clear clustering and separability between defect classes
- Indicates that feature representations learned during transfer learning are meaningful and discriminative
- Supports the effectiveness of transfer learning for low-data environments like PCB defect detection
- Visual validation that model features align well with true class labels



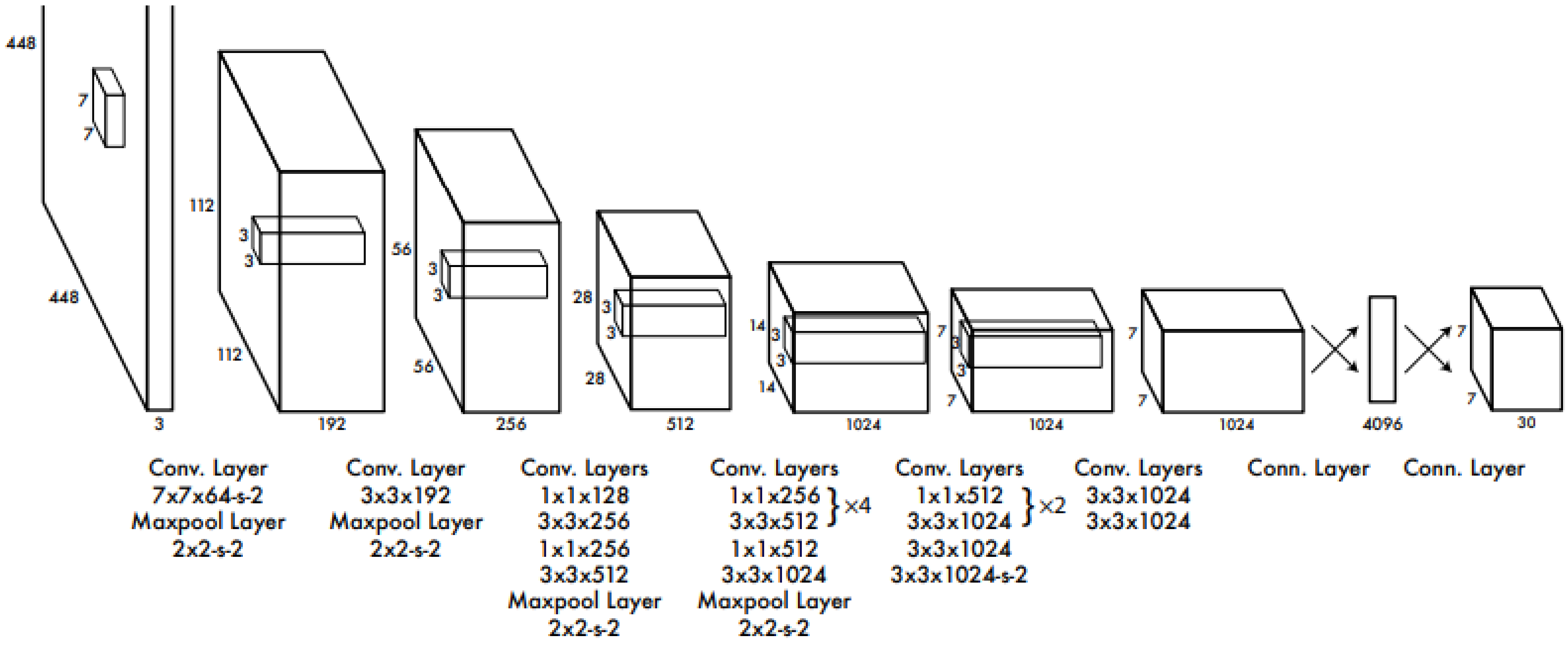


TRAINING SETUP

- Common pipeline across models:
  - Pretrained weights (COCO)
  - 100 epochs / 10k steps
  - Data formatted to match model needs
  - Batch sizes: 16–32

MODEL OVERVIEW

- **YOLOv8:** Fast, accurate, real-time, one-stage

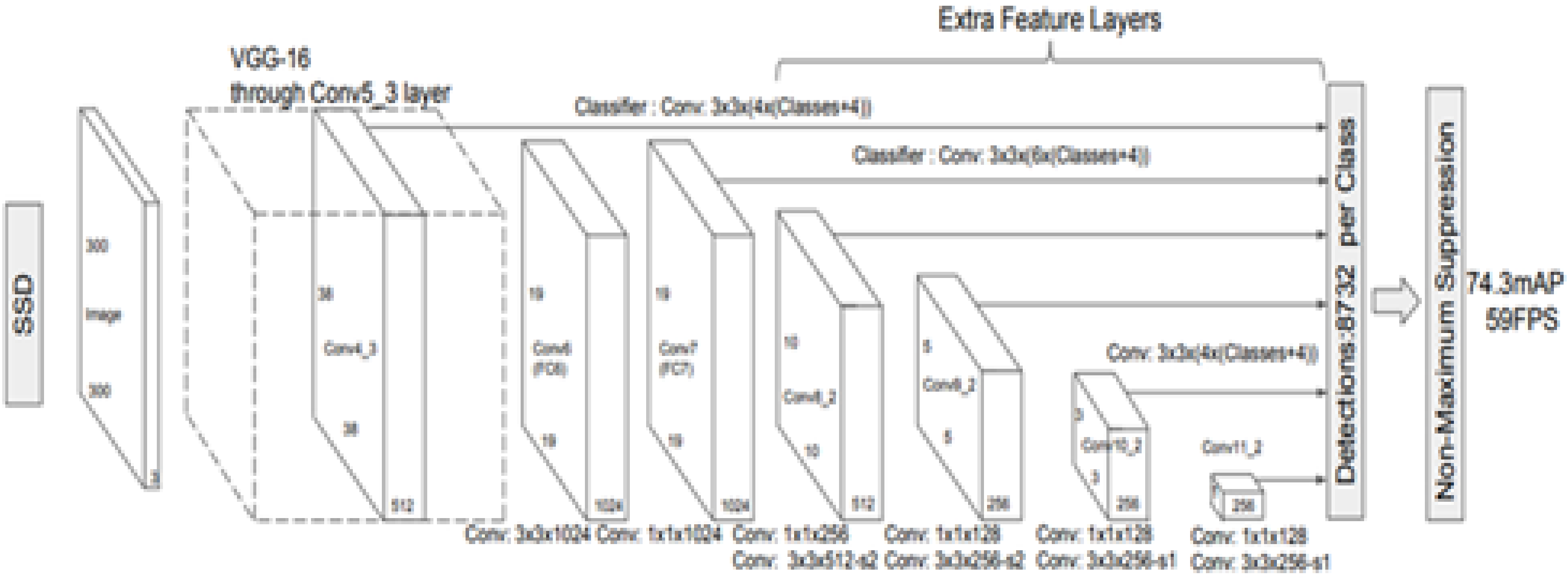


MODEL OVERVIEW

- Deep learning object detection and classification algorithm suitable for real-time applications
- Uses one stage
- Divides image into a grid where each grid is responsible for object detection within its boundary
- Untrained model or pretrained model on the COCO dataset are available
- Model with pretrained weights can be imported into a Python environment easily
- Trained and tuned on our small dataset
- Automatically generates performance metrics and results

MODEL OVERVIEW

- **SSD MobileNet:** Lightweight, embedded-friendly, low accuracy

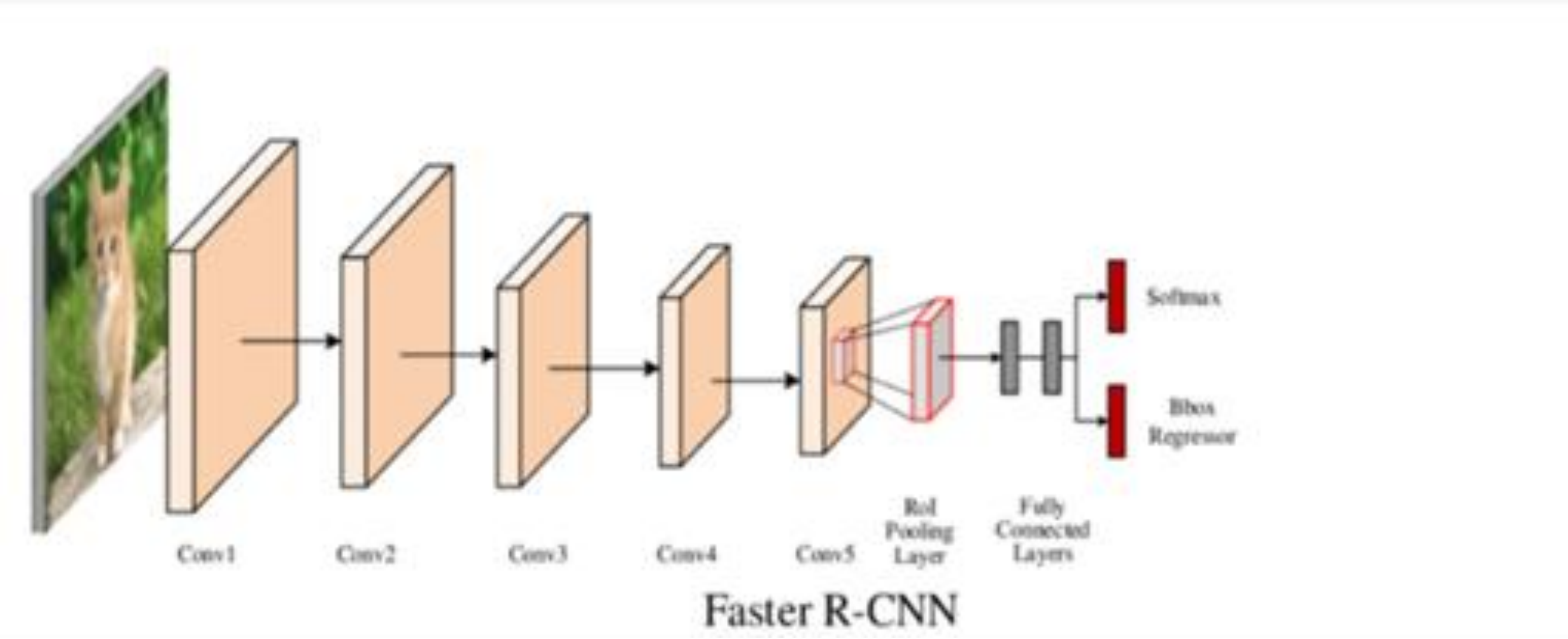


MODEL OVERVIEW

- Combines MobileNet V2 (backbone) with Single Shot Detector (SSD) head for object detection
- One-stage architecture: performs classification and localization in a single forward pass
- Optimized for real-time performance on resource-constrained devices Lightweight and fast, with reduced model size and computation cost
- Best suited for applications where speed is critical, but can struggle with detection accuracy, especially on complex datasets

MODEL OVERVIEW

- **Faster R-CNN:** Accurate but slow and resource-intensive

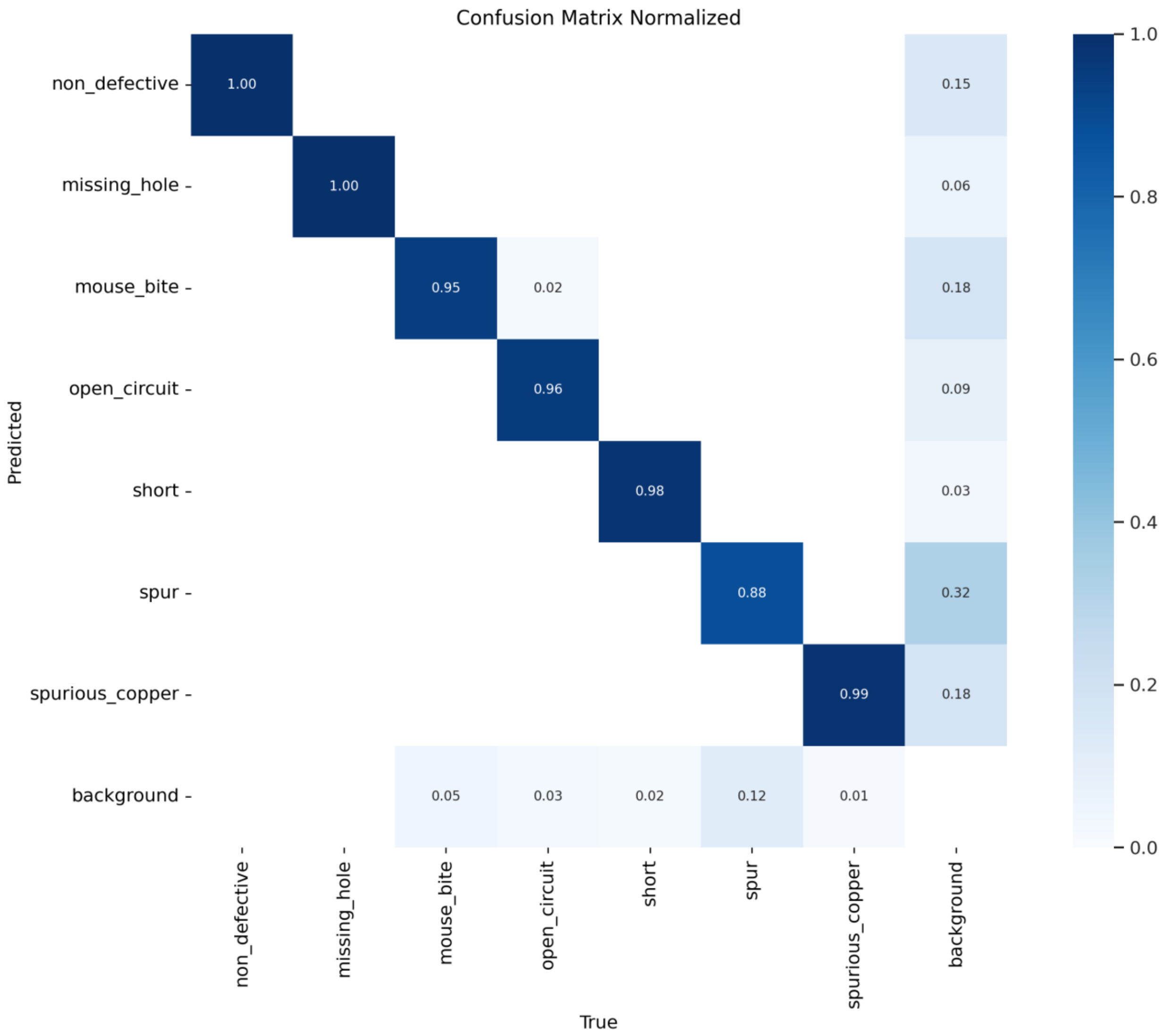


MODEL OVERVIEW

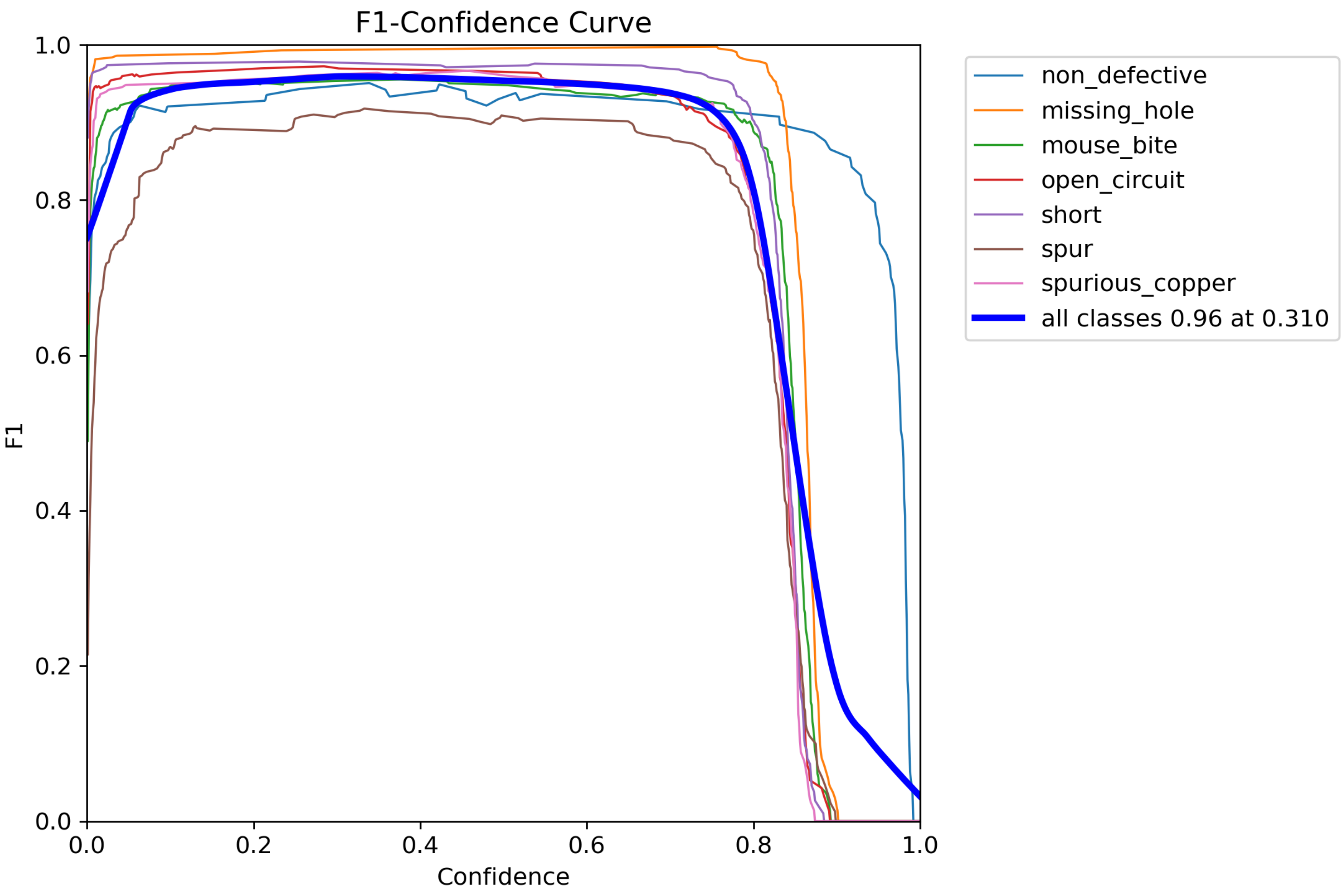
- Lightweight deep learning model for object detection on resource-limited devices
- Single-stage detector: combines SSD framework with MobileNet backbone
- Processes feature maps at multiple scales for multi-size object detection
- Pretrained models available on COCO dataset
- Can be imported and fine-tuned easily in Python environments
- Trained and evaluated on our small, augmented PCB dataset
- Faster than R-CNN, but less accurate—optimized for speed and efficiency

YOLOV8 RESULTS

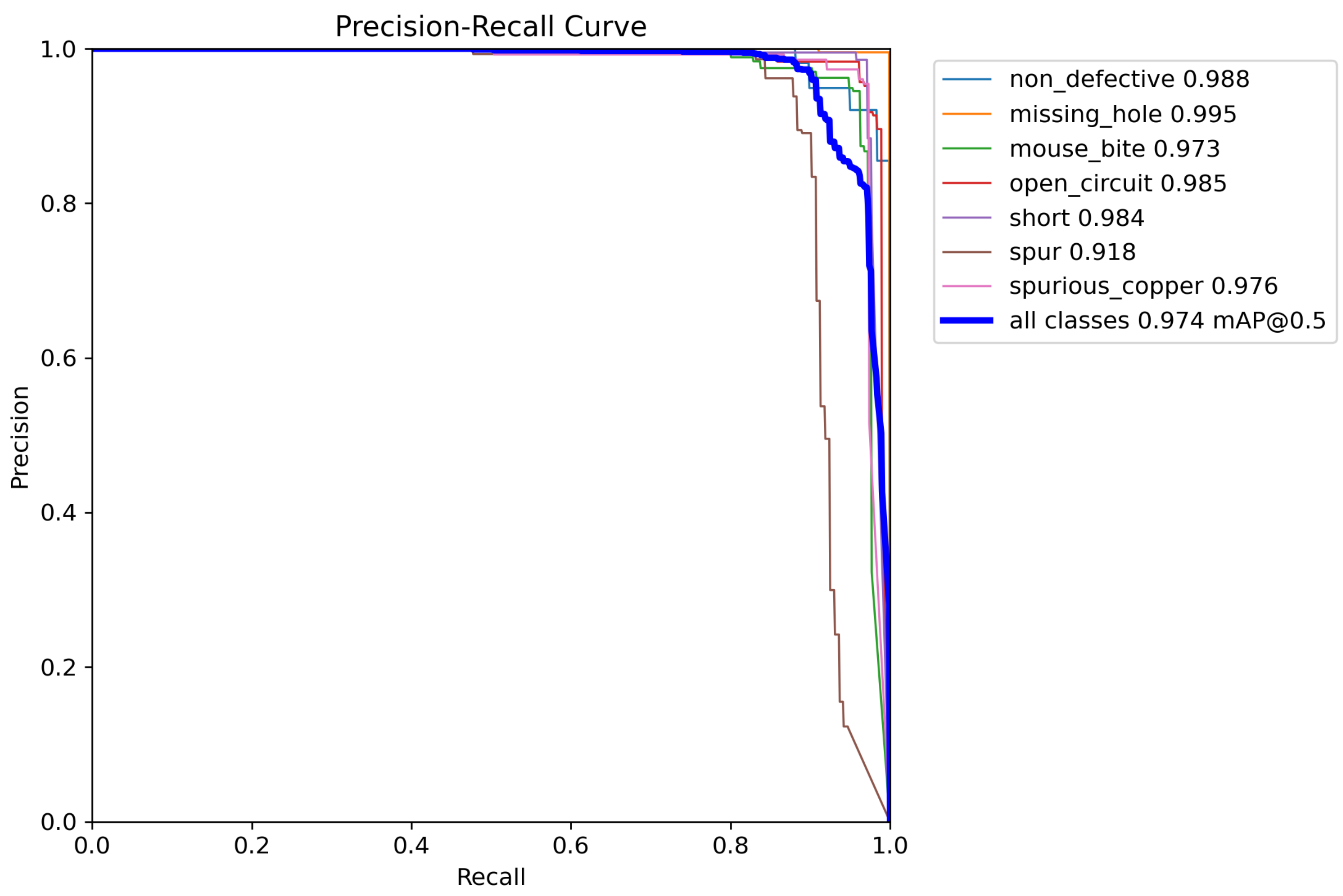
- Best performer: mAP@0.5 = **97.4%**
- F1 = 0.96 @ conf. 0.31
- Prec/Recall = 96.1% / 96%



Confusion Matrix Normalized
non_defective
missing_hole
mouse_bite
open_circuit
short
spur
spurious_copper
background
Predicted
True
1.00
1.00
0.95
0.02
0.96
0.98
0.88
0.99
0.15
0.06
0.18
0.09
0.03
0.32
0.18
0.05
0.03
0.02
0.12
0.01
1.0
0.8
0.6
0.4
0.2
0.0



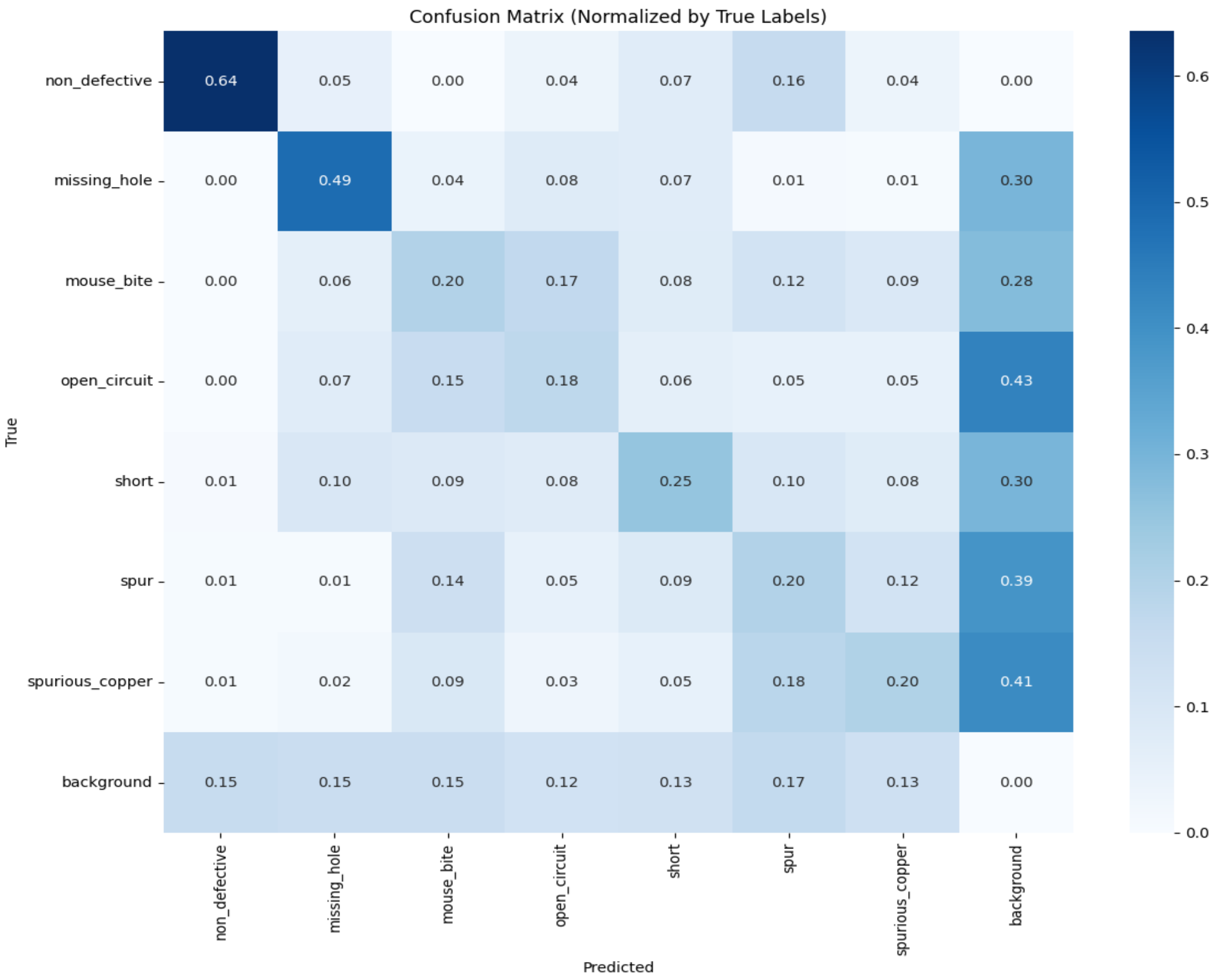
F1-Confidence Curve
F1
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
non_defective
missing_hole
mouse_bite
open_circuit
short
spur
spurious_copper
all classes 0.96 at 0.310



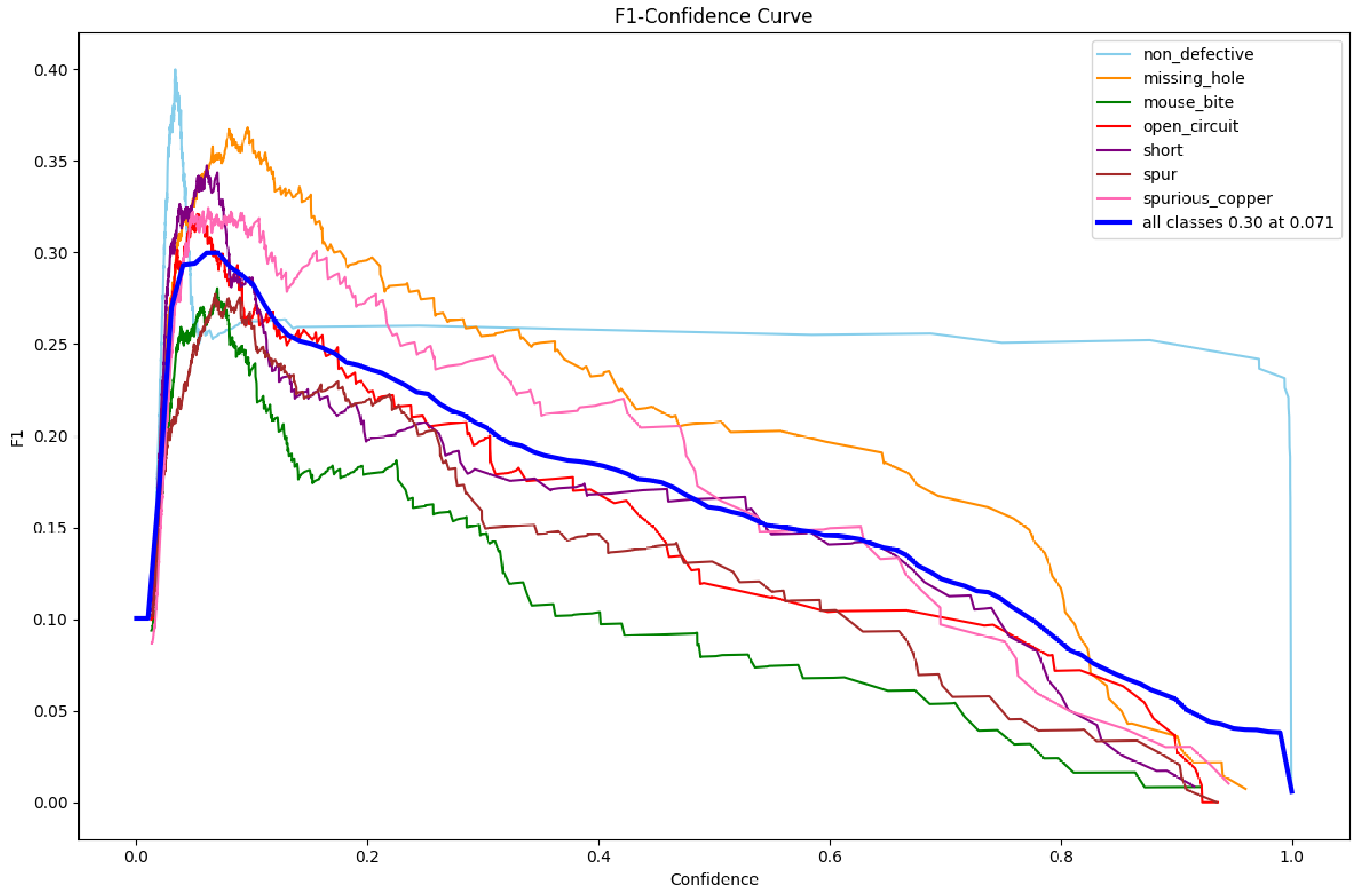
Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
non_defective 0.988
missing_hole 0.995
mouse_bite 0.973
open_circuit 0.985
short 0.984
spur 0.918
spurious_copper 0.976
all classes 0.974 mAP@0.5

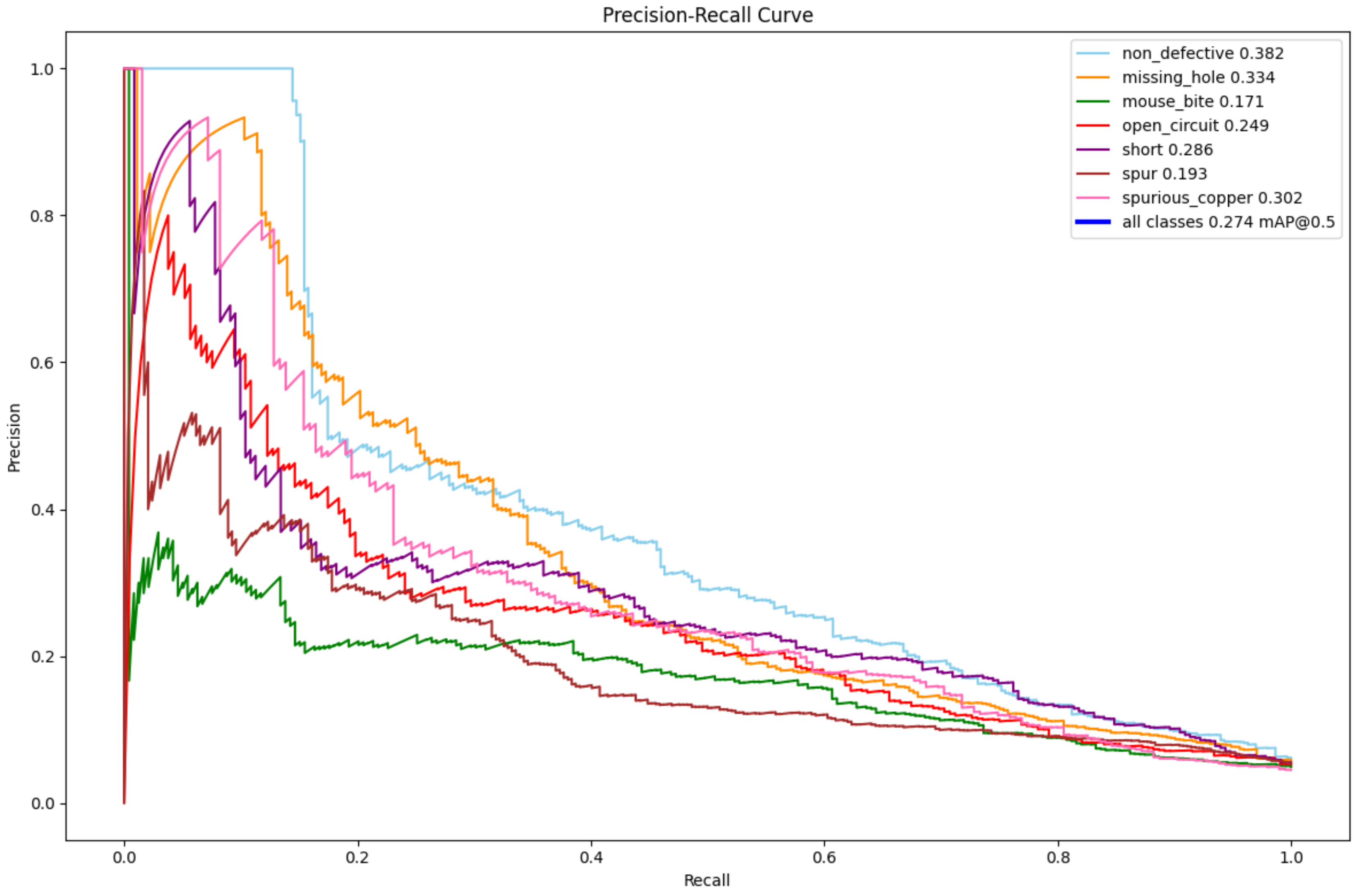
SSD MOBILENET RESULTS

- Fastest, lowest accuracy
- mAP@0.5 = **27.4%**
- F1 = 0.3 @ conf. 0.07
- Poor classification ability



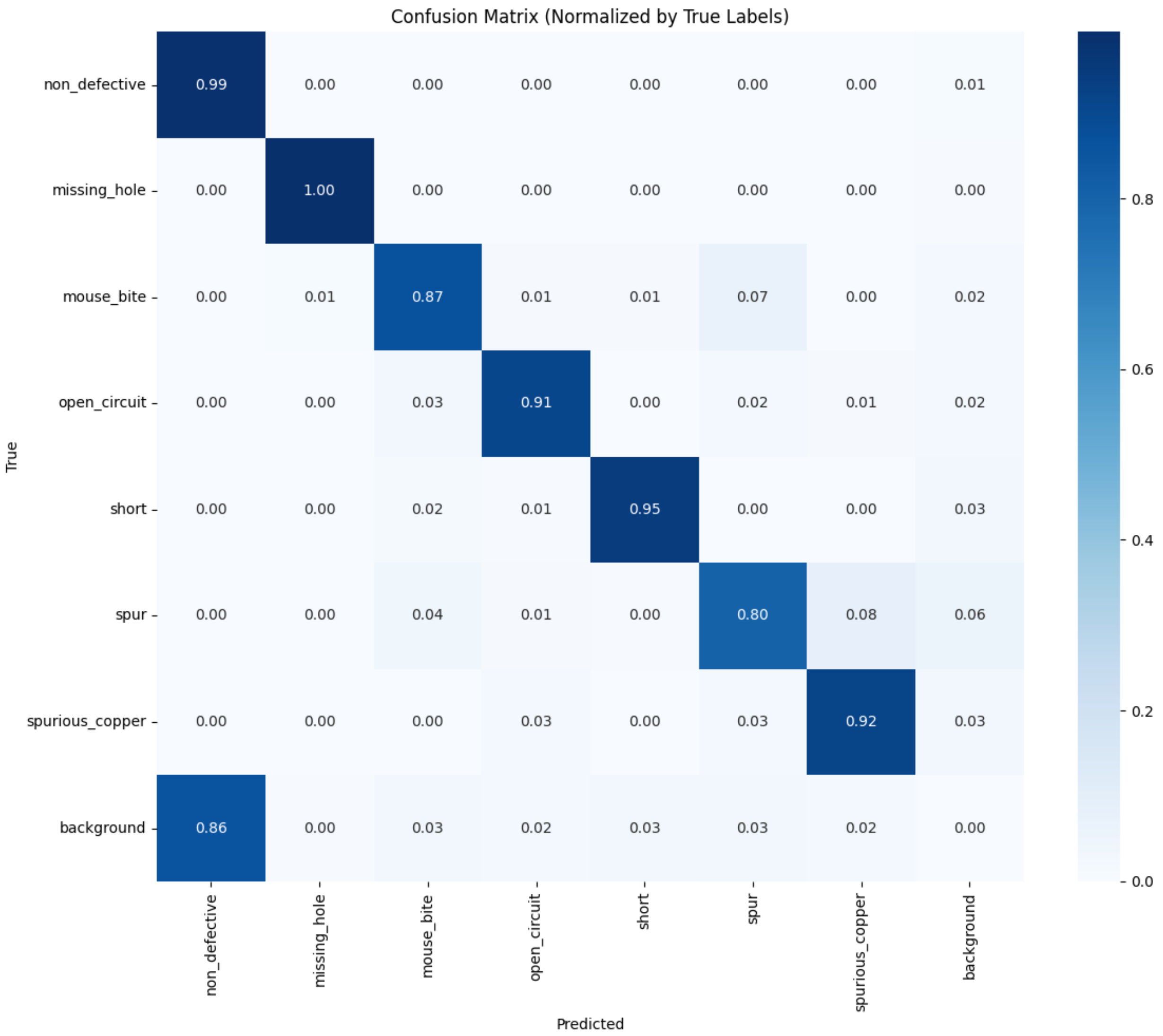
Confusion Matrix (Normalized by True Labels)
True
Predicted
non_defective
missing_hole
mouse_bite
open_circuit
short
spur
spurious_copper
background
0.64 0.05 0.00 0.04 0.07 0.16 0.04 0.00
0.00 0.49 0.04 0.08 0.07 0.01 0.01 0.30
0.00 0.06 0.20 0.17 0.08 0.12 0.09 0.28
0.00 0.07 0.15 0.18 0.06 0.05 0.05 0.43
0.01 0.10 0.09 0.08 0.25 0.10 0.08 0.30
0.01 0.01 0.14 0.05 0.09 0.20 0.12 0.39
0.01 0.02 0.09 0.03 0.05 0.18 0.20 0.41
0.15 0.15 0.15 0.12 0.13 0.17 0.13 0.00
0.6
0.5
0.4
0.3
0.2
0.1
0.0

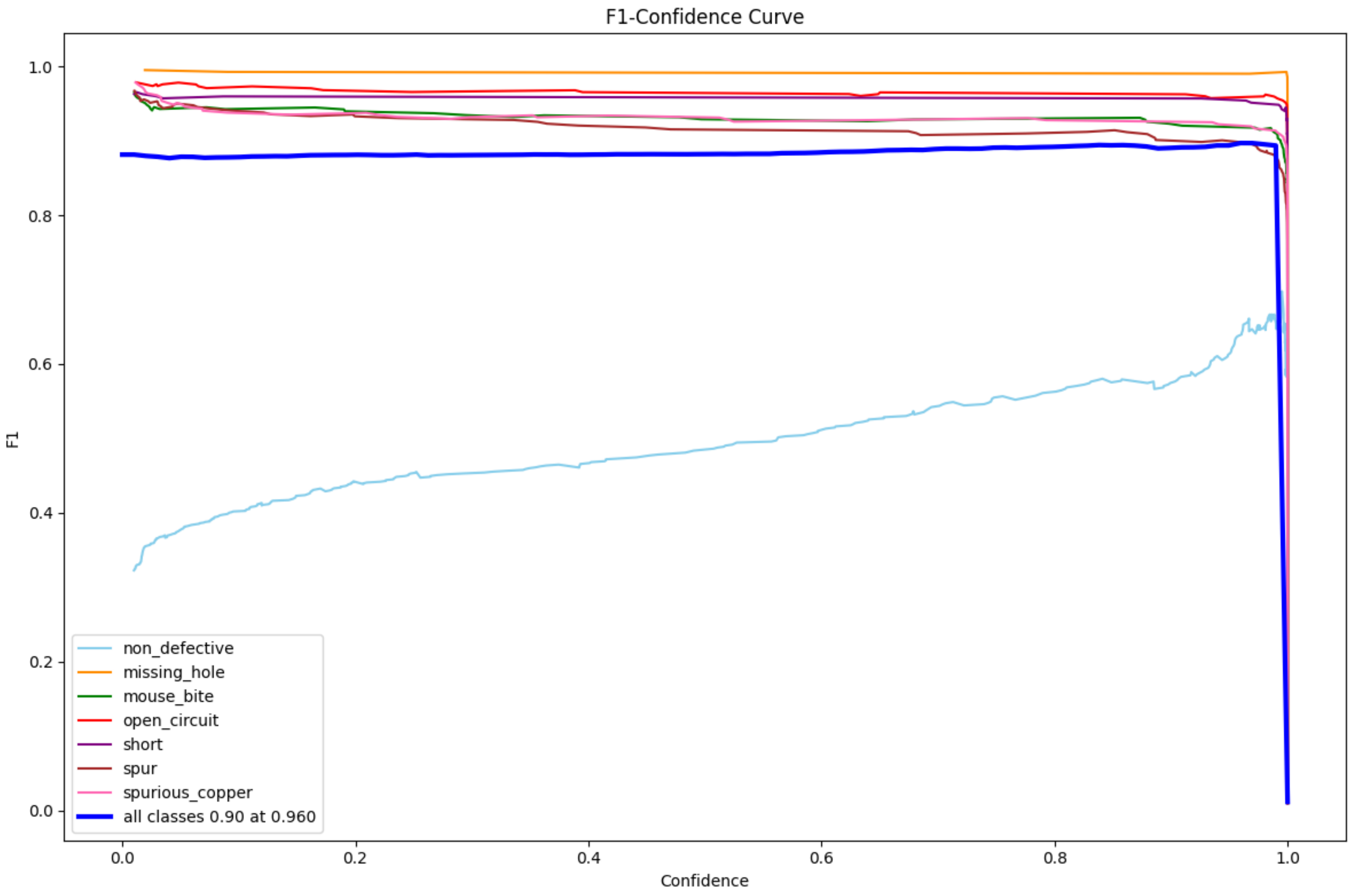

F1-Confidence Curve
F1
Confidence
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.0
0.2
0.4
0.6
0.8
1.0
non_defective
missing_hole
mouse_bite
open_circuit
short
spur
spurious_copper
all classes 0.30 at 0.071


Precision-Recall Curve
non_defective 0.382
missing_hole 0.334
mouse_bite 0.171
open_circuit 0.249
short 0.286
spur 0.193
spurious_copper 0.302
all classes 0.274 mAP@0.5
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

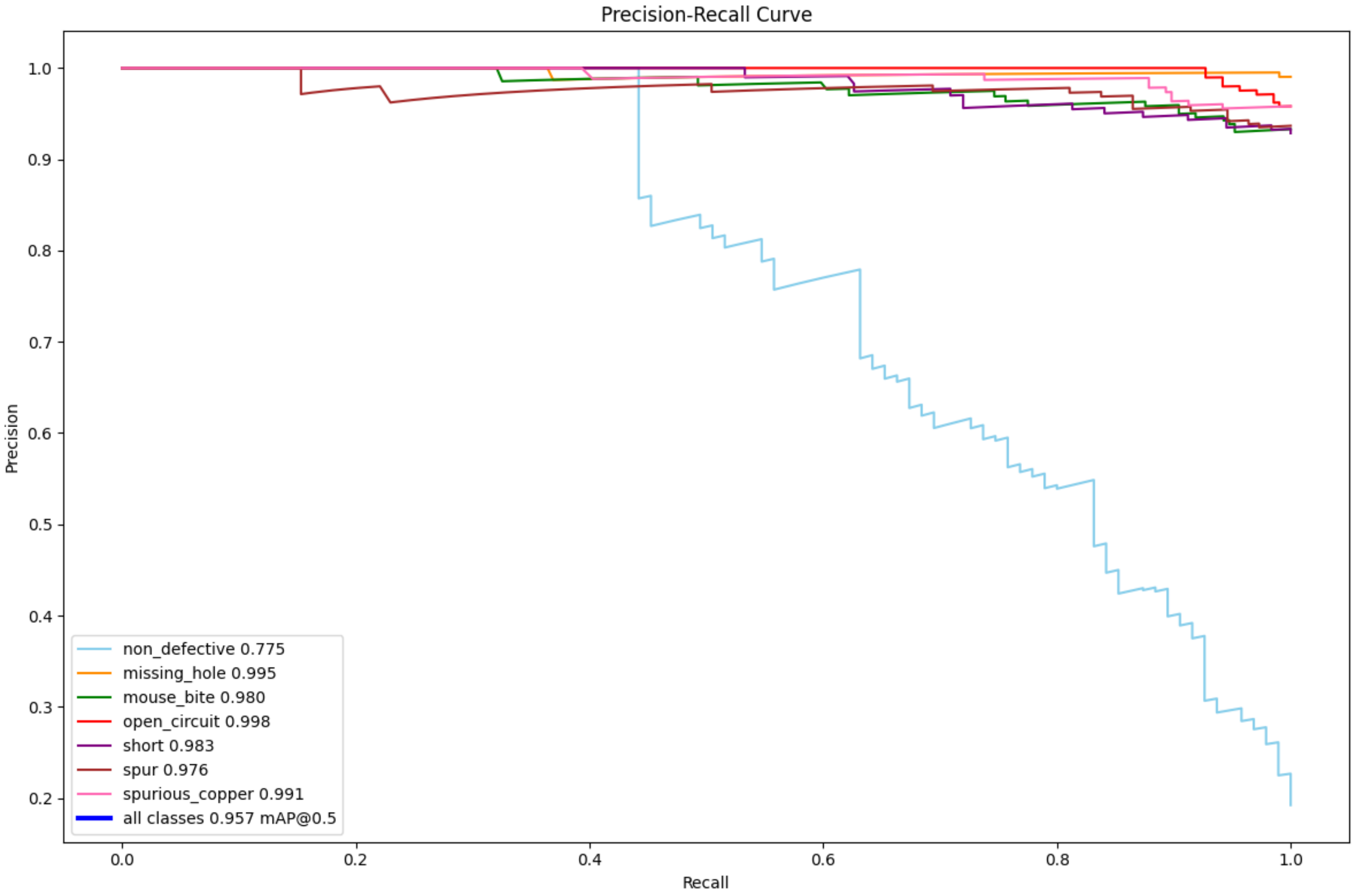
FASTER R-CNN RESULTS

- mAP@0.5 = **95.7%**
- F1 = 0.9 @ conf. 0.96
- Struggled with background class
- Took long time to train


Confusion Matrix (Normalized by True Labels)
True
non_defective
missing_hole
mouse_bite
open_circuit
short
spur
spurious_copper
background
0.99 0.00 0.00 0.00 0.00 0.00 0.00 0.01
0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.01 0.87 0.01 0.01 0.07 0.00 0.02
0.00 0.00 0.03 0.91 0.00 0.02 0.01 0.02
0.00 0.00 0.02 0.01 0.95 0.00 0.00 0.03
0.00 0.00 0.04 0.01 0.00 0.80 0.08 0.06
0.00 0.00 0.00 0.03 0.00 0.03 0.92 0.03
0.86 0.00 0.03 0.02 0.03 0.03 0.02 0.00
Predicted
0.8
0.6
0.4
0.2
0.0



F1-Confidence Curve
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
non_defective
missing_hole
mouse_bite
open_circuit
short
spur
spurious_copper
all classes 0.90 at 0.960


Precision-Recall Curve
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
0.2
0.4
0.6
0.8
1.0
non_defective 0.775
missing_hole 0.995
mouse_bite 0.980
open_circuit 0.998
short 0.983
spur 0.976
spurious_copper 0.991
all classes 0.957 mAP@0.5

MODEL COMPARISON

F1-Confidence Curve		
Model	score	@
YoloV8	0.96	0.31
SSD MobileNet	0.3	0.071
Faster R-CNN	0.9	0.96

Precision-Confidence Curve		
Model	score	@
YoloV8	1	0.877
SSD MobileNet	0.77	0.899
Faster R-CNN	1	1

Precision-Recall Curve		
Model	score	mAP@
YoloV8	0.974	0.5
SSD MobileNet	0.274	0.5
Faster R-CNN	0.957	0.5

Time To Train	
Model	Time to Run
YoloV8	00:44:52
SSD MobileNet	02:00:15
Faster R-CNN	20:35:15

CONCLUSION

- **YOLOv8**
  - Best overall performer with high F1 (0.96) and mAP (0.974)
  - Fast training time (~45 min)
  - Ideal for real-world deployment due to strong balance of **speed and accuracy**
- **SSD MobileNet**
  - Fast and lightweight, but **poor accuracy** (F1: 0.30, mAP: 0.274)
  - Struggled with class confusion and low confidence
  - Not suitable for production use
- **Faster R-CNN**
  - High accuracy (F1: 0.9, mAP: 0.957), but **extremely slow** training (20+ hrs)
  - Great for research or offline analysis, but **inefficient for deployment**