MAIS 202 - Project Deliverable 2: Preliminary Results
PCB Defect Detection

Mia Valderrama-Lopez

October 21st 2025

Problem Statement

This project develops an automated PCB defect detection system using computer vision and deep learning. The goal is to classify six common PCB manufacturing defects (Missing Hole, Mouse Bite, Open Circuit, Short, Spur, Spurious Copper) to assist in quality control processes for electronics manufacturing.

Data Preprocessing

The PCB Defect Dataset from Kaggle contains 6,370 training images and 802 validation images across 6 defect classes. Key preprocessing steps include:

- **Image Resizing:** Changed from proposed 224×224 to 600×600 pixels to preserve fine defect details in high-resolution PCB images
- **Data Augmentation:** Applied rotations (±20°), horizontal flips, width/height shifts (20%), and zoom (20%) to improve generalization
- **Normalization:** Scaled pixel values to [0,1] range
- **Train-Validation Split:** Maintained original dataset split with 6,370 training and 802 validation samples

Machine Learning Model

3a. Framework and Architecture

Framework: TensorFlow/Keras with Metal GPU acceleration on Apple M2
Architecture: Custom CNN with the following structure:

```
Input: (600, 600, 3)
 Conv2D(32, 3×3) + BatchNorm + MaxPooling
```

```
Conv2D(64, 3×3) + BatchNorm + MaxPooling
Conv2D(128, 3×3) + BatchNorm + MaxPooling
Conv2D(256, 3×3) + BatchNorm + MaxPooling
Conv2D(512, 3×3) + BatchNorm + MaxPooling
GlobalAveragePooling2D()
Dense(256) + Dropout(0.5)
Output: Dense(6, softmax)
```

Total Parameters: 1.7M (1.70M trainable)

3b. Training Configuration

- **Optimizer:** Adam (learning rate=0.0001) with gradient clipping
- **Regularization:** Batch normalization after each conv layer, dropout (0.5) in dense layers
- **Batch Size:** 16 (reduced due to 600×600 image memory constraints)
- **Callbacks:** Early stopping (patience=5), model checkpointing

3c. Validation Methods

Model validation used hold-out validation with 802 samples. The model shows clear **underfitting** rather than overfitting:

- Training accuracy: 17.38%
- Validation accuracy: 17.58%
- Both losses plateau around 1.84 without significant divergence

3d. Implementation Challenges

- **M2 Mac GPU Compatibility:** Metal plugin errors resolved using legacy Adam optimizer
- **Memory Constraints:** Reduced batch size from 32 to 16 for 600×600 images
- **Training Speed:** Approx 13 minutes per epoch due to high-resolution images

Model weights saved (not on git): `models/best_pcb_model.h5`

Preliminary Results

Performance Metrics

Target (Deliverable 1): 95% accuracy, F1-score > 0.90
Actual Results: 17.58% accuracy, near-random performance

Class	Precision	Recall	Samples
Missing Hole	0.000	0.000	118
Mouse Bite	0.171	0.864	140
Open Circuit	0.000	0.000	135
Short	0.333	0.013	158
Spur	0.207	0.138	130
Spurious Copper	0.000	0.000	121

Table 1: Per-class performance showing severe prediction bias

Critical Finding: Model Collapse

The model developed severe prediction bias, collapsing to primarily predict only two classes:

- **Mouse Bite:** 86.4% recall (121/140 correct)
- **All other classes:** 0-13.8% recall
- 672/802 misclassifications were predictions of ”Mouse Bite”

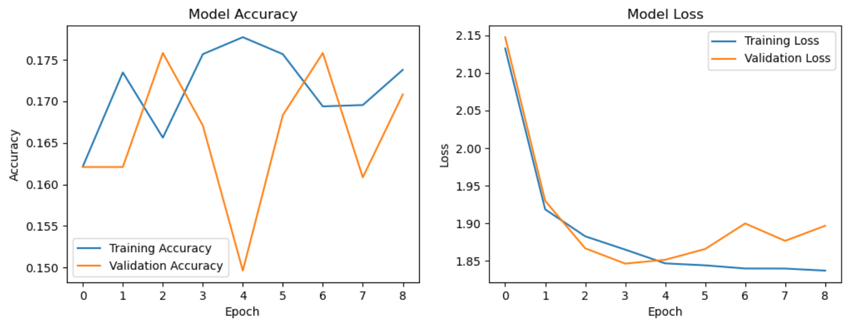
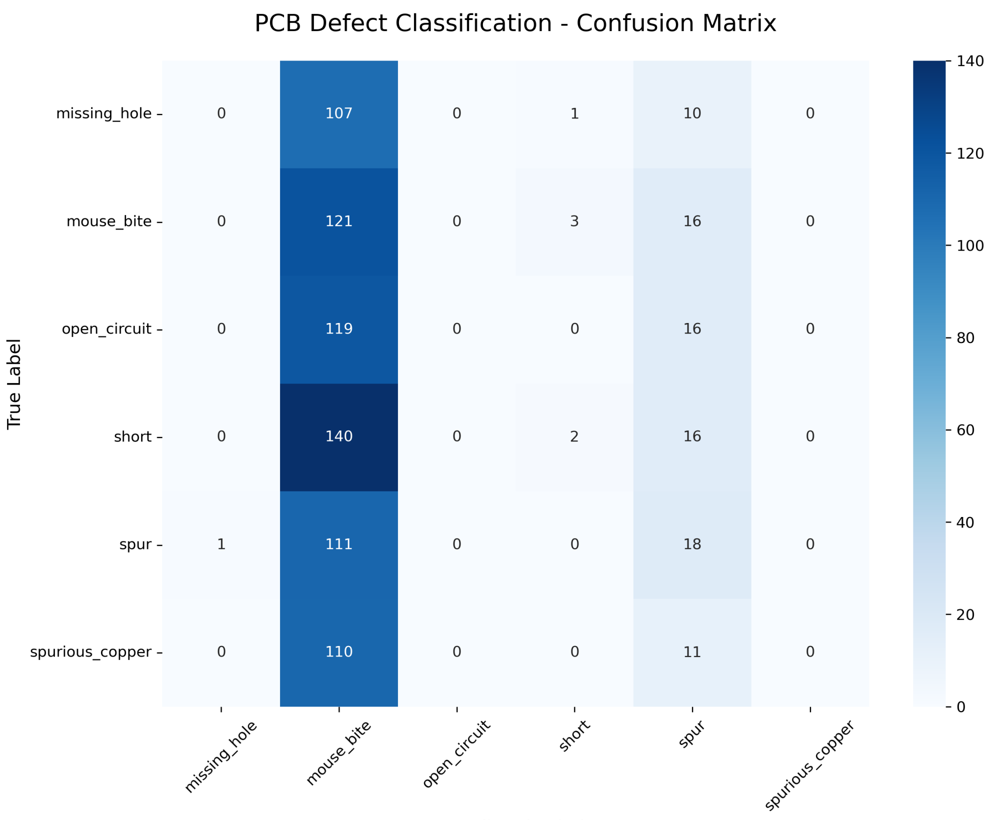


Figure 1: Training history showing underfitting - both training and validation metrics plateau at similar levels



	missing_hole	mouse_bite	open_circuit	short	spur	spurious_copper
missing_hole	0	107	0	1	10	0
mouse_bite	0	121	0	3	16	0
open_circuit	0	119	0	0	16	0
short	0	140	0	2	16	0
spur	1	111	0	0	18	0
spurious_copper	0	110	0	0	11	0

Figure 2: Confusion matrix revealing model's bias toward Mouse Bite predictions across all classes

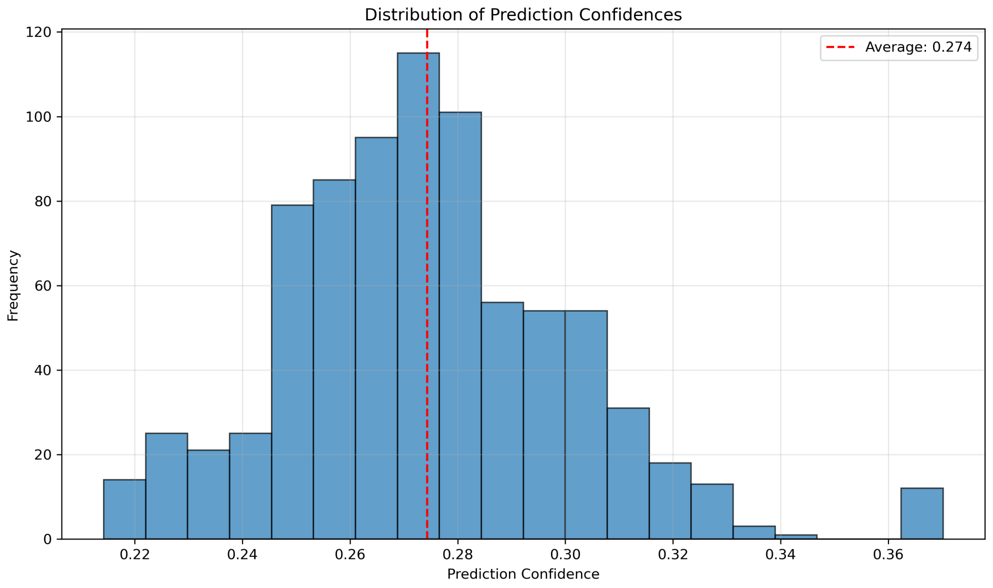


Figure 3: Prediction confidence distribution showing low average confidence (0.274)

Feasibility Assessment

Despite current performance, the project remains feasible because:

- The failure mode is clearly diagnosed (class collapse due to imbalance/insufficient capacity)
- Infrastructure (data pipeline, training framework) is fully functional
- Clear improvement path identified through transfer learning

Next Steps

Immediate Improvements (1-2 days)

- **Class Balancing:** Implement class weights to address prediction bias
- **Transfer Learning:** Replace custom CNN with EfficientNetB0 pretrained on ImageNet
- **Input Optimization:** Reduce image size to 224×224 for faster iteration

Architectural Changes

- **Pros of Current Approach:** Educational value, full control over architecture
- **Cons:** Insufficient capacity for complex PCB defect patterns
- **New Approach:** Transfer learning with data augmentation and class balancing

Expected Outcomes

- **Short-term target:** >70% accuracy with transfer learning
- **Final target:** >90% accuracy with fine-tuning and ensemble methods
- **Timeline:** 1 week for model improvement, 1 week for web app integration

The current baseline, while performing poorly, provides valuable diagnostic information and a solid foundation for rapid improvement in the next iteration.