Deep Learning Based System for PCB Defect Classification

Team A-14: Winterstar

Mohnish and Faaz Muhammed

1 Introduction to the Problem Statement

- The project aims to create a deep learning based system to automatically classify PCB defects from images, focusing on high accuracy.
- PCB defects such as missing holes, open circuits,mouse bite, short circuits, spurs and spurious copper arise due to inaccuracies during manufacturing and assembly processes. Even minor defects can lead to malfunctioning circuits, reduced product reliability, or complete system failure in real-world applications.
- Manual PCB defect classification is time-consuming and inconsistent; therefore, deep learning techniques based on Convolutional Neural Networks are employed for automated and scalable PCB defect classification.

2 Dataset description

- The dataset contains six primary defect categories, namely Missing Hole, Mouse Bite, Open Circuit, Short Circuit, Spur, and Spurious Copper. The images are organized in a class-wise directory structure, where each folder corresponds to a specific defect type. This organization simplifies dataset handling and integration with deep learning frameworks such as PyTorch.
- Prior to model training, all PCB images are passed through a consistent preprocessing pipeline implemented using standard deep learning transforms. Each image is resized to a fixed spatial resolution to ensure uniform input dimensions across the dataset, which is required for batch-based training of Convolutional Neural Networks. Subsequently, the images are converted to tensors and normalized using predefined mean and standard deviation values. This normalization step stabilizes training by ensuring that input features lie within a comparable range, which is particularly important when using pretrained Convolutional Neural Network architectures.

- Following preprocessing, the dataset is split into training and validation subsets. This separation allows the model's generalization performance to be monitored during training and helps in identifying potential overfitting.

3 Model Design and Methodology

3.1 Model Selection

- A Convolutional Neural Network (CNN)–based architecture was selected due to the visual and spatial nature of PCB defect patterns.
- A pretrained ResNet-18 model was chosen as the baseline architecture.
- Compared to more complex architectures such as EfficientNet, ResNet-18 allows faster training and quicker feedback, which is beneficial for validating the correctness of the data pipeline and training methodology during this phase.

3.2 Network Architecture and Transfer Learning

- The model was initialized with weights pretrained on the ImageNet dataset.
- Transfer learning was employed to reuse generic low-level visual features such as edges, textures, and shapes.

3.3 Training Methodology

- Cross-entropy loss was used as the objective function due to its suitability for categorical classification.
- The Adam optimizer was employed for training because of its adaptive learning rate and stable convergence behavior.
- The training pipeline was implemented in a modular and reproducible manner to ensure consistency across experiments.

4 Implementation Progress and Preliminary Results

4.1 Implementation Progress

- A complete end-to-end training pipeline which includes dataset loading, preprocessing, model initialization, training, and validation was implemented using PyTorch.
- The training pipeline was tested and executed successfully under CPU-based computational constraints.

4.2 Preliminary Results

- The baseline ResNet-18 model was successfully trained on the training subset and evaluated on the validation subset.
- The observed validation performance is currently limited, indicating that the model has not yet achieved strong generalization.
- The current results serve as a baseline reference for further optimization in subsequent stages of the project.

4.3 Observations

- The limited performance can be attributed to:
  - CPU-only training constraints
  - Restricted training duration
  - Absence of data augmentation at this stage
- No critical issues such as training instability, data leakage, or implementation errors were observed.
- These observations suggest that the focus can now shift from pipeline validation to performance improvement.

5 Evaluation Strategy

5.1 Evaluation Metrics

- Model performance is evaluated using classification accuracy as a primary metric.
- In addition to accuracy, class-wise metrics such as precision, recall, and F1-score are also considered to better capture the model's performance across defect classes.

5.2 Validation Methodology

- A dedicated validation set is used to evaluate the model during training.
- Validation performance is monitored to assess the model's ability to generalize to unseen data.
- The train–validation split helps identify overfitting and guides model and hyperparameter selection.

5.3 Overfitting and Generalization

- Discrepancies between training and validation performance are used as indicators of overfitting.
- At this stage, the focus is on understanding model behavior rather than proper optimization.

6 Challenges and the Plan Ahead

6.1 Challenges Encountered

- Training was performed under CPU-only computational constraints, which limited experimentation speed and training duration.
- More complex architectures could not be fully explored at this stage due to computational limitations.

6.2 The Plan Ahead

- Introduce data augmentation techniques such as random rotations, flips, and intensity variations to improve model generalization.
- Experiment with deeper and more advanced architectures, including ResNet-50 and EfficientNet, once the pipeline is fully validated.
- Migrate training to a GPU-enabled environment to enable faster experimentation and improved performance.

Overall, the mid-evaluation phase successfully established a functional and reproducible baseline system for PCB defect classification. The insights gained from preliminary experiments provide a clear roadmap for targeted improvements in subsequent stages of the project.