Deep PCB Defect Detection Project

Abstract

This document describes the development of a deep learning-based system for detecting defects in Printed Circuit Boards (PCBs) using image data. The system involves data preprocessing steps to prepare the training data, creation of a suitable dataset, and a Convolutional Neural Network (CNN) architecture for automated defect classification (unet architect).

Data Preprocessing

- **Data Path Definition:** The file paths for the training data directory, normal image directory, and defect image directory are defined.
- **Parsing Label and Image Paths:** Text files containing lists of image paths and corresponding labels are parsed to create separate lists for normal and defect images with their associated labels.
- **Image Loading and Preprocessing:** Images are loaded, converted to grayscale format, and resized to a uniform size (128x128 pixels).
- **Label Processing:** Label text files are parsed to generate a list of bounding boxes representing defect regions within each image.

Dataset Creation

- **Combined Dataset:** A combined dataset is formed by merging the preprocessed normal and defect images with their corresponding labels and bounding boxes.
- **Data Shuffling:** The dataset is shuffled to randomize the order of data points, mitigating potential biases during training.

- **Train-Test Split:** The shuffled dataset is divided into training and testing sets using an 80/20 ratio. The training set is further batched for efficient model training.

Model Building

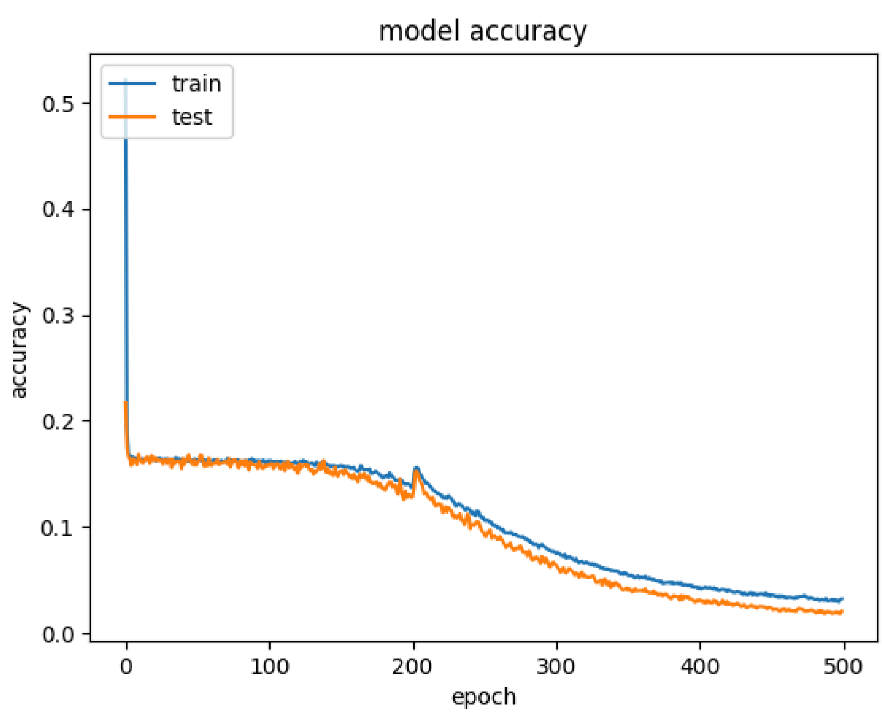
- **Convolutional Neural Network (CNN) Architecture:** The core of the system is a CNN architecture designed for defect detection.
- **He Normal Initialization:** He Normal initialization is employed for weight biases in the convolutional layers to address the "dying ReLU" problem and improve training efficiency.
- **ReLU Activation:** The ReLU (Rectified Linear Unit) activation function is used throughout the convolutional layers to introduce non-linearity into the network.
- **Encoder-Decoder Structure:** The network follows an encoder-decoder structure:
    - **Encoder:** This part performs feature extraction through convolutional and pooling operations. The image size progressively reduces while the number of feature channels increases, capturing informative features from the input images.
    - **Center Cropping:** In the encoder path, center cropping is used to focus on relevant defect areas within the image.
    - **Decoder:** The decoder upsamples the extracted features and combines them with corresponding cropped features from the encoder path, enabling precise defect localization.
    - **Concatenation:** Feature maps from different parts of the network are merged using concatenation to improve the model's ability to detect defects.

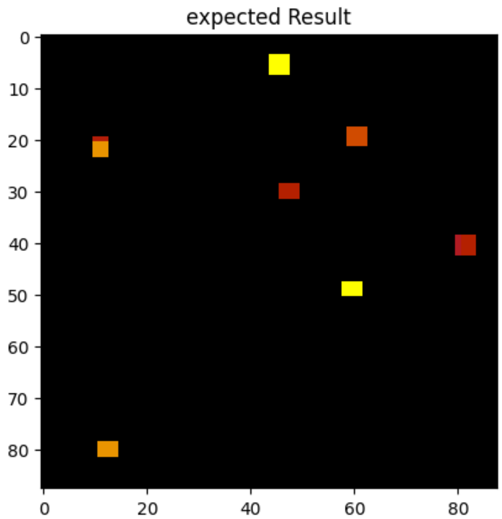
- **Output Layer:** The final layer of the CNN has 7 filters, corresponding to the 7 different defect classes the model aims to identify.

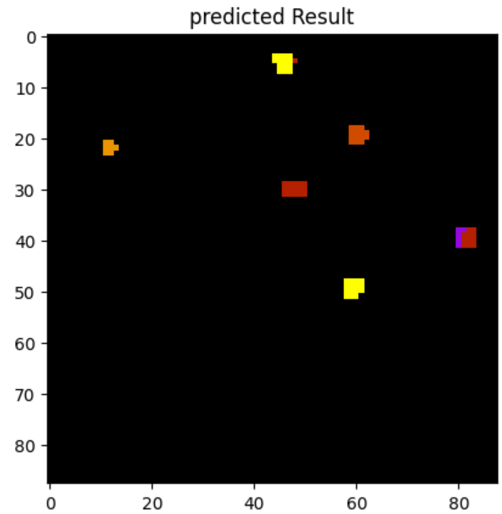
Model Training and Evaluation

- **Adam Optimizer:** The Adam optimizer is used to train the model, which efficiently updates the network weights based on the calculated loss.
- **Sparse Categorical Crossentropy Loss:** The Sparse Categorical Crossentropy loss function is employed to measure the difference between the predicted defect classes and the ground truth labels during training.
- **Validation Dataset:** A validation dataset is used to monitor model performance during training and prevent overfitting.

Result:




expected Result
0
10
20
30
40
50
60
70
80
0
20
40
60
80


predicted Result
0
10
20
30
40
50
60
70
80
0
20
40
60
80