Cross Analysis of Transfer Learning Models for PCB Defect Detection

Brandon Markham
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
uzy7@txstate.edu

John Gellerup
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
xqv6@txstate.edu

Josh Muniga
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
ghh22@txstate.edu

***Abstract*—This paper describes a project exploring the use of machine learning principles such as object detection for detecting PCB defects in commercial manufacturing applications. Transfer learning was applied to three existing object detection models, YOLOv8, SSD MobileNet and Faster R-CNN. These models were trained on a small, augmented dataset consisting of PCB Images. These images included defective PCBs as well as non-defective ones. The trained models were evaluated on metrics such as precision score, recall score, training time, and mean average precision. YOLOv8 performed the best of all the models achieving a mean average precision score of 97.2% and precision and recall scores of 96% and 96.1% respectively. SSD MobileNet, while being fast and resource friendly, scored the worst of the models with a precision score of 2% and recall score of 27%, making it far too inaccurate for this manufacturing application. Faster R-CNN was much more accurate than SSD MobileNet with a precision score of 68% and recall score of 71%. Although it was an improvement compared to the SSD MobileNet model, it fell short of the YOLOv8s scores and required much more time to train and more computational resources. The results of this paper demonstrate the feasibility of object detection and transfer learning in the application of PCB defect detection systems.**

I. Introduction

The demand for electronics is the highest it has been in history, and at the heart of these electrical systems are Printed Circuit Boards (PCBs). Ensuring PCBs are free of defects is a vital step in the manufacturing process of digital electronics, as even slight defects within the PCBs fabrication can compromise the functionality of the entire electrical system.

The standard, for a long time, for PCB defect detection has relied on technicians to visually inspect the board, which consumes a vast amount of time, can be inconsistent and is not scalable. A more reliable and automated solution for PCB detection is needed especially as the demand continues to grow.

This paper researches the use of object detection and other machine learning techniques to automate the defect detection process in PCBs. This paper applies transfer learning to use three different pretrained models, YOLOv8, SSD MobileNet and Faster R-CNN to tune and train these models on a significantly smaller dataset of images than would be traditionally needed to create accurate models for object detection. The goal of this paper is to compare each model's performance using metrics such as F1 score, speed and accuracy to determine the best model for detecting PCB defects in a manufacturing setting.

II. Background

A. Industry Challenges

Printed Circuit Boards (PCBs) are essential in creating the modern electronic devices we enjoy today. One of the main challenges faced in the electronics manufacturing industry is ensuring the functionality and correctness of these PCBs before they are tested and used in their respective products. The smallest defects in the fabrication of a PCB such as shorts, open circuits, lifted pins on ICs or mouse bites (a small notch or indentation on the PCB) can compromise the functionality of the board. As a result, these defects cost companies money and can drastically increase the lead times of their products. The main method of checking for PCB defects today is visual inspection by technicians or other personnel. However, this method is slow, wastes manpower and introduces human error to the defect detection process. An alternative to this method is to use computer vision and machine learning to detect and classify these defects.

B. Object Detection

Object detection can be used to scan images taken by cameras, detect a certain defect and locate the area in which the defect occurs using bounding boxes. This removes the human component by allowing a system itself to detect defective boards and locate exactly where and what defect is causing the board to fail the inspection.

There are two main categories of models found in Object Detection. There are One-stage detectors and Two-stage detectors. The One-stage models which are found in faster and more lightweight applications only pass over the image one time simultaneously determining where the object (defect in our case) is and what the object is. Examples of One-stage models are YOLOv8 and SSD MobileNet. Two-stage models such as Faster R-CNN make two passes over the image instead of one. In the first pass, the model makes a broad scan of the image to determine which regions of said image contain the object. In the second pass the model examines the regions that were determined to possibly contain the object and determines whether it actually detects the object. The addition of the second step in this model makes it slower than the One-stage detectors but often times more accurate.

C. Transfer Learning

Building and training Deep learning models from scratch takes a long time, and an enormous amount of data. The idea of Transfer Learning allowed us to take a general model that was pre-trained in object detection on an extremely broad and vast dataset, and train it to detect defects in PCBs. Using Transfer Learning we were able to drastically reduce the amount of labeled data needed as well as the time it took to successfully train the models.

D. Models Implemented

Three models were selected for this project to simulate different applications found in the real world manufacturing industry.

1) YOLOv8

A new model from the YOLO family that's main purpose is speed, making it suitable for real-time object detection applications. This model uses a single neural network that divides the input image into a grid. Each cell within the grid array is responsible for detecting and classifying that object within the cell itself. Because this model uses only one stage, it is less computationally expensive than models such as Faster R-CNN which use two stages to detect and classify objects.

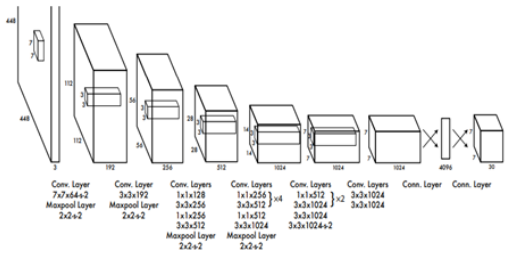

Figure 1 YOLO Model Architecture

2) SSD MobileNet

A lightweight CNN that uses Single Shot Multibox Detector (SSD) to predict the bounding boxes and classification of the images. Mobile Net, unlike traditional CNNs that apply a single filter across every output channel per input channel, uses depth wise and pointwise channels. While depth wise applies a filter across every input channel, pointwise convolution then mixes the output channels. The SSD which sits on top of the CNN is responsible for interpreting features and their predictions by predicting "anchor boxes" which attempt to fit the object. These form the basis of the models confidence score. The architecture of this model is lightweight and suitable for devices with limited resources such as embedded systems.

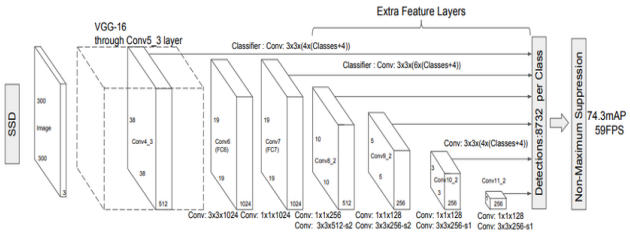

Figure 2 SSD MobileNet Model Architecture

3) Faster R-CNN

A Two-stage detector which operates in two stages. In the first stage, the model scans the image and identifies regions that may contain the object using a Region Proposal Network (RPN). In the second stage, the Faster R-CNN examines these regions identified using the RPN, tunes the size as well as the position of the bounding box, determines whether the target object is within the region and its confidence score. The addition of the second step makes the Faster R-CNN more computationally expensive and slower but more accurate than One-stage detection models.

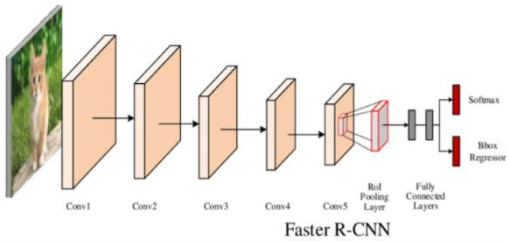

Figure 3 Faster R-CNN Model Architecture

E. Purpose of Analysis

Our goal is to determine the speed and accuracy of these models in detecting and classifying PCB defects. By comparing these three models we can determine how well they work, what tradeoffs come with each, and which would be the best for this particular application.

III. DATASET

The PCB Defects dataset was obtained from kaggle.com and was originally released from the Open Lab on Human Robot Interaction of Peking University. The data set consisted of 10 PCB images that were modified using Adobe Photoshop to synthesize 6 different classes of defect. The defects defined in the dataset are: missing hole, mouse bite, open circuit, short, spur, and spurious copper. The set consisted of 693 images with each class of defect having approximately 115 images. Each image has a corresponding annotation file in Pascal VOC XML format that contains bounding box coordinates for each defect in the image. In addition to the defective images there is also a set of rotational metadata for each class of defect containing image filenames and corresponding angles of rotation. This dataset was augmented and preprocessed for the purpose of training, validating, and testing all three models.

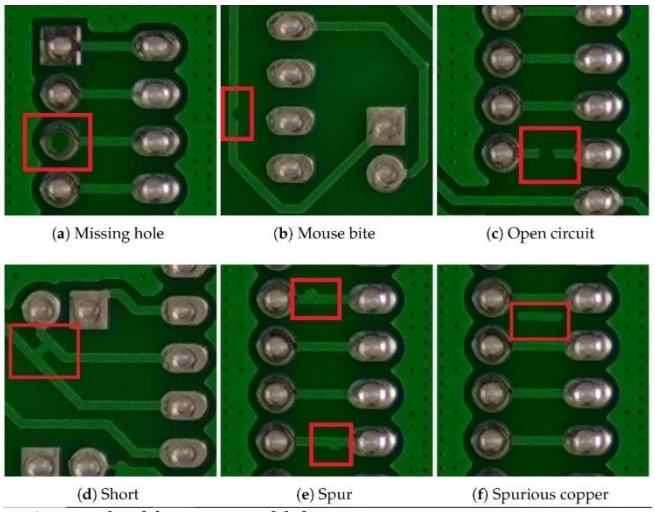

Figure 4: example of the six types of defects

IV. Procedures

A. Data Augmentation

The first step of data preprocessing entailed using a program called LabelImg to create annotation files consisting of bounding box coordinates that encapsulated the entire circuit board for the 10 non-defective PCB images. After the annotations were created a python script containing the albumtations library was used to augment the non-defective images by rotating and saving each image to a new file 25 times for a total of 250 images and creating corresponding annotation files for the augmented set by creating copies of the original annotation files and updating the bounding box coordinates using the designated angles of rotation.

After creating the non-defective class, the images in the defective classes were augmented with similar tactics by using the rotational metadata provided in the original dataset to rotate each image and save it to a new file along with corresponding annotations with updated bounding box coordinates. These augmentation methods resulted in a final dataset consisting of 1636 images distributed between 7 classes and their corresponding annotation files.

B. Visualization

After the dataset was complete the first step towards visualizing the data was taken by cropping the defective regions within the bounding boxes specified in the annotation files. The cropped images were resized to 128 x 128 pixels before feeding them through the convolutional base of a pretrained VGG16 model to generate high-dimensional feature vectors. The feature vectors then underwent transformation to two dimensions through the application of t - Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP). The 2D plots demonstrate distinct class clusters that help evaluate dataset separability and the efficacy of transfer learning.

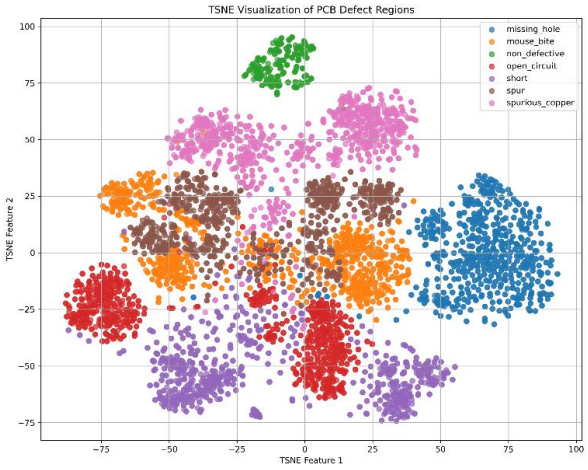

Figure 5 TSNE Plot of Dataset

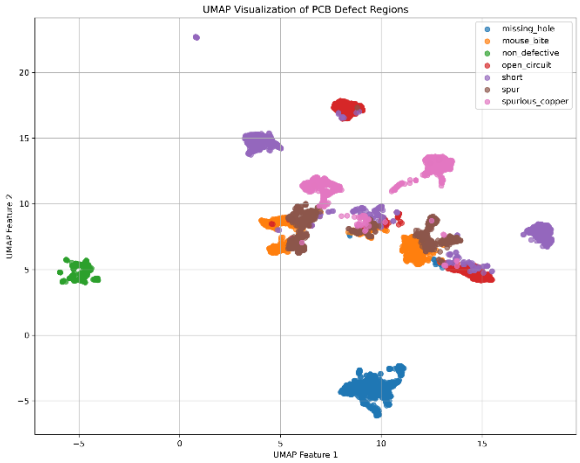

Figure 6 UMAP Plot of Dataset

C. Test/Train/Val Split

After class separability and transfer learning effectiveness had been confirmed by the visualization of the data, the dataset was merged into one directory and split using three ratios (60/20/20, 70/15/15, and 80/10/10) into three directories containing training, validation, and test samples. After the splits had been performed further preprocessing was required to configure the data into the proper format required by the different models. All models were trained and tested on the three data splits with the best results being selected and cross analyzed for the report

D. TFRecords

SSD MobileNetV2 and the Faster R-CNN ResNet 50 V1 are models built on the TensorFlow architecture. This architecture requires that the data set be converted into a format known as TFRecords. A TFRecord is a binary file format that sequentially stores data and is used to streamline data into a TensorFlow training pipeline.

To facilitate this data conversion a label map was created assigning a numerical value to each class. Then a conversion script was used to read the annotation files, parse the bounding

box coordinates, read the corresponding images and normalize the coordinates, designate classes using the label map, and write serialized binary files in TFRecord format. These files are titled test.record, train.record. and val.record.

E. YOLO Annotation Format

The YOLOv8m model requires a different annotation style and file structure. This consists of two folders titled "labels" and "images". Each folder in the file structure contains 3 sub folders titled "test", "train", and "val". After the folder format was restructured a conversion script was used to convert the Pacal VOC XML annotations to the required annotation format. Similar to the TensorFlow architecture, a label map was created assigning numerical values to each class in the form of a data.yml file.

F. Model Configuration and Training

With all of the data preprocessed and formatted for each respective model, all of the required dependencies for each model were downloaded into their respective environments on the LEAP2 cluster.

1) YOLOv8m

The YOLOv8m model was run using a script that tells the yolov8m model to run object detection in train mode while specifying the data path. The script then tells the model to run for 100 epochs and resize the images to 640 x 640 pixels and runs the training in batches of 16. The results get saved to a specified file path along with the results allowing either a new file to be created or a file with the same name to be overwritten.

2) SSD MobileNet

The SSD MobileNet model was trained with the pipeline.config file modified to declare 7 unique object categories. All of the input images were then resized to a resolution of 320 x 320 for consistent input. The anchors were set to declare anything with an intersection over union (IOU) between the predictions and the actual bounding boxes to be greater than or equal to 50% meaning that if the prediction overlapped more than half of the actual defect coordinates it was declared successful in detecting the defect. The training configuration portion of the file was set with batch sizes of 32 and trained for 10,000 steps total. There was a momentum optimizer employed with a cosine decay learning rate that started with a 0.001, peaked at 0.01 and decayed over 10,000 steps. The model was loaded with the pretrained MobileNet weights from the COCO-trained checkpoint located in the model's file path location. With all of the training configurations completed the model was trained and then evaluated using an evaluation script that extracted the saved model, and the model was evaluated using the test portion of the data split.

3) F-RCNN

The F-RCNN model was trained with the pipeline.config file modified to declare 7 unique object classes and have the images resized to 800 x1333 pixels with the pad to max dimension equal to true. The feature extractor settings were kept in the default settings with the training configuration modified for a batch size of 16 and 10,000 steps. A momentum optimizer was used with a cosine decay learning rate set to 0.01 and decayed for 10,000 steps. The model was loaded with the pretrained weights from the COCO- trained checkpoint located in the model's file path location with all of the data augmentation options removed from the default configuration. With all the training configurations completed the model was trained and then evaluated using an evaluation script that extracted the saved model, and the model was evaluated using the test portion of the data split

V. RESULTS

A. YOLOv8m

The YOLOv8m model performed the best with the non – defective and missing hole class being detected with perfect accuracy. The rest of the accuracies and their misclassifications can be seen in the normalized confusion matrix depicted in Figure 6. The F1 curve depicted in Figure 7 reached its average maximum value between all classes when the confidence threshold was at 0.3 indicating a great balance between precision and recall at that confidence threshold. The precision-recall curve has all classes mAP @ 0.5: 0.974. This indicates desirable precision – recall tradeoff across the object types. These results indicate that the optimal confidence threshold is 0.31 to yield the highest F1 score which is suited well for the object detection required to detect and classify the PCB defects.

1) YOLO Figures From 60/20/20 Split

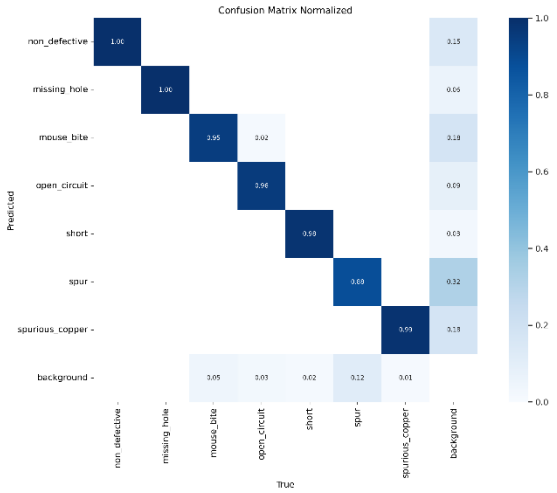

Figure 7: YoloV8 Confusion Matrix

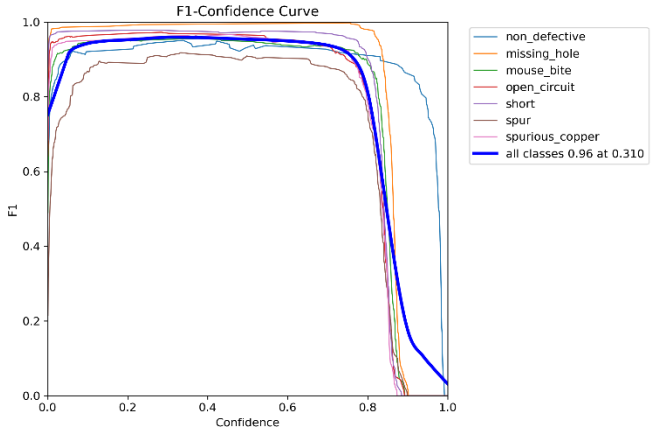

Figure 8 YOLOv8 F1-Confidence Score

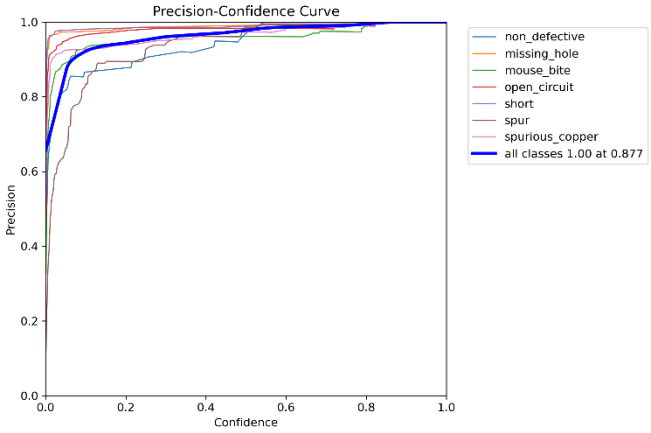

Figure 9 YOLOv8 Precision-Confidence Curve

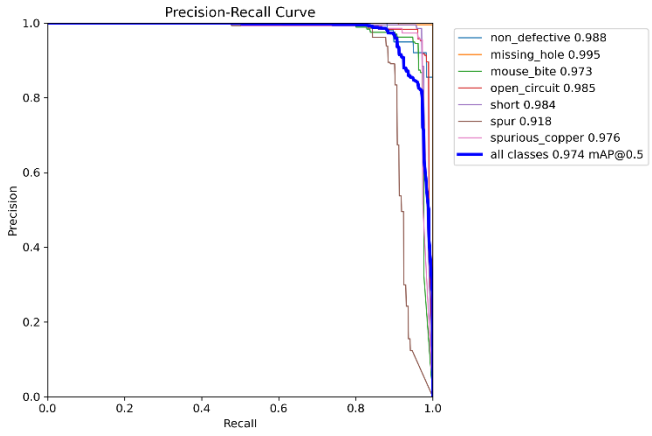

Figure 10: YOLOv8 Precision-Recall Curve

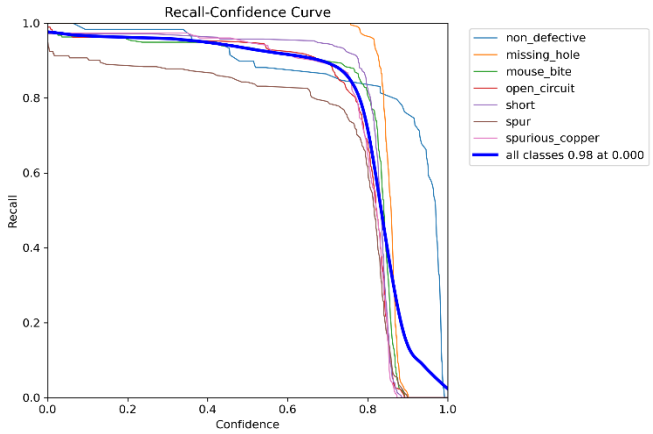

Figure 11 YOLOv8 Recall-Confidence Curve

B. SSD MobileNet

This model by far showed to be the least accurate with several of the classes misclassified as background as depicted in the normalized confusion matrix seen in Figure 12. The F1 score maxed out at approximately 0.3 at a confidence threshold of 0.071. This indicates a low precision and recall tradeoff with the model lacking confidence in its predictions. The precision-recall curve has an overall mAP@0.5 = 0.274 which is quite poor. The model is flagging many false positives and misclassifying most true objects.

1) SSD MobileNet Figures From 60/20/20 Split

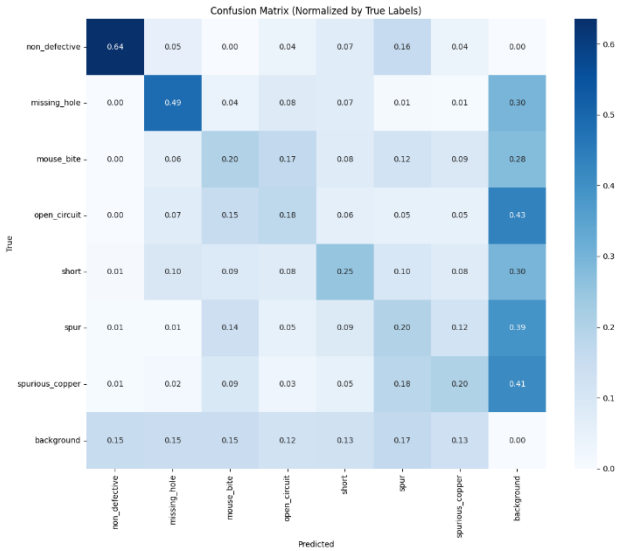

Figure 12: SSD MobileNet Confusion Matrix

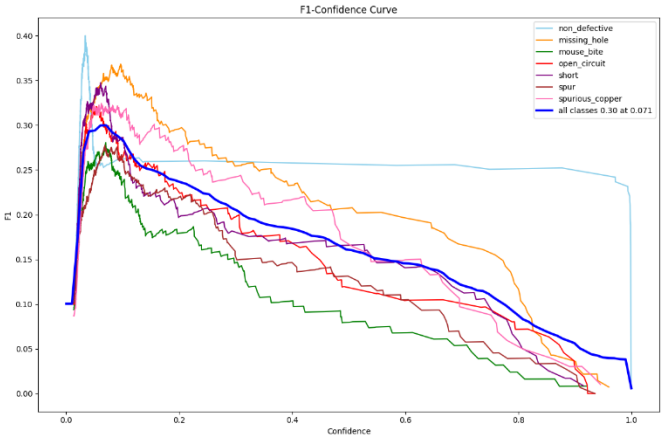

Figure 13: SSD MobileNet f1-Confidence Curve4

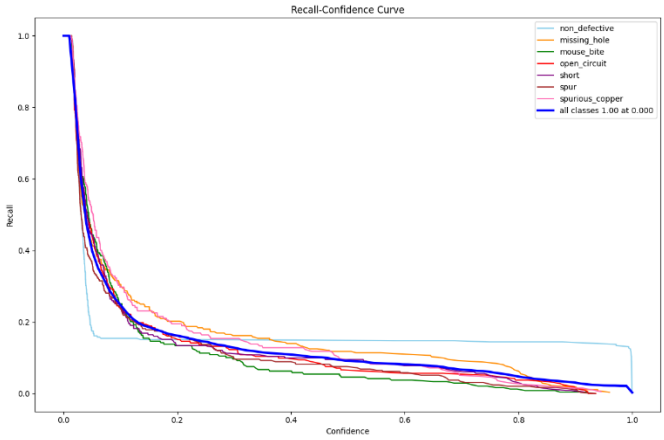

Figure16: SSD MobileNet Recall-Confidence Curve

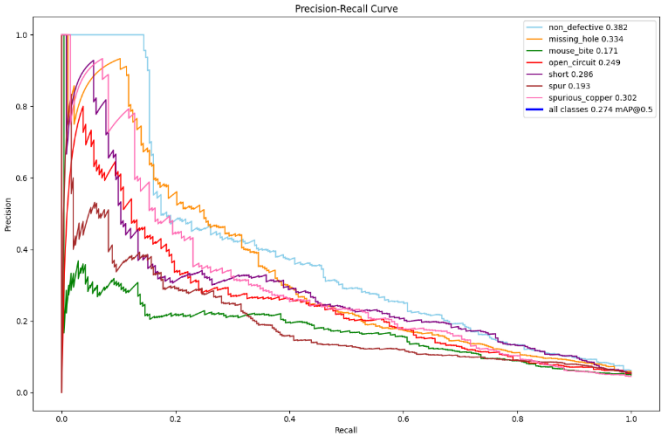

Figure 14: SSD MobileNet Precision-Recall Curve 5

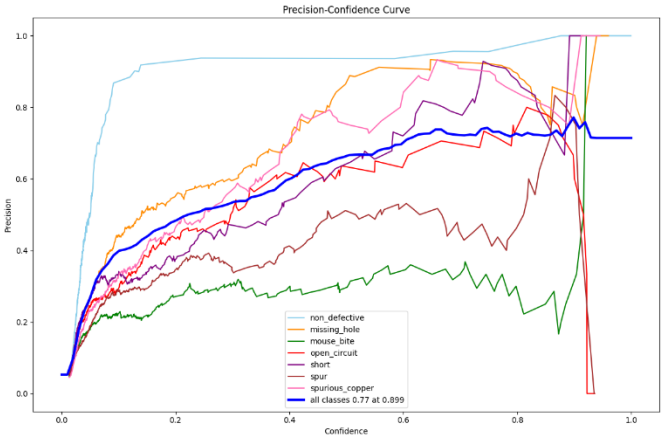

Figure 15: SSD MobileNet Precision-Confidence Curve 6

C. Faster R-CNN

The F-RCNN model was the second-best model as far as accuracy was concerned, however the amount of time it took the model to train was drastically longer than the YOLOv8m making it a less ideal solution to detect PCB defects. The background was often misclassified as the non-defective class as seen in the normalized confusion matrix depicted in Figure 17. The F1 score reached an average of .90 between all classes at a confidence threshold of 0.96 indicating that the model performs reliably at higher confidence levels. Only the non-defective class had a rising F1 trend improving proportionally with confidence. The precision-recall showed an overall mAP@0.5 = 0.957 which is very high indicating that the tradeoff between the two is well balanced and robust under different operating points. The performance metrics indicate strong classification with a clean ability to separate the classes. If the model was less computationally intensive and was able to yield responses in a timelier manner it would be a strong candidate for real-world PCB defect detection.

1) Faster R-CNN Figures From 60/20/20 Split

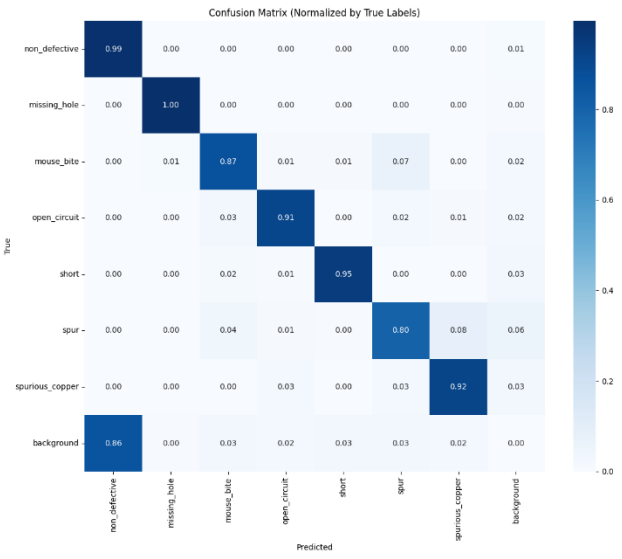

Figure 17: Faster R-CNN Confusion Matrix

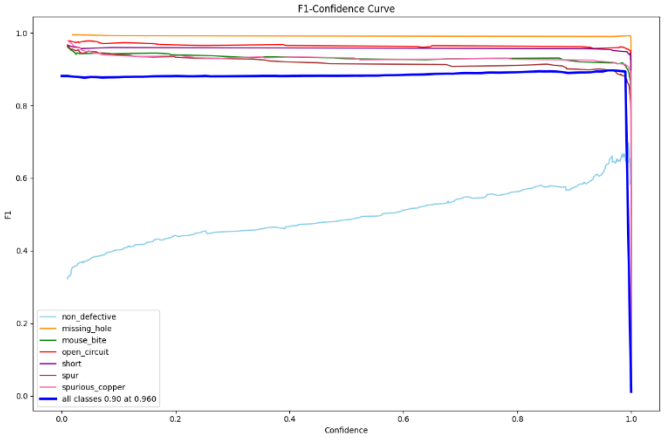

Figure 18: Faster R-CNN F1-Confidence Curve

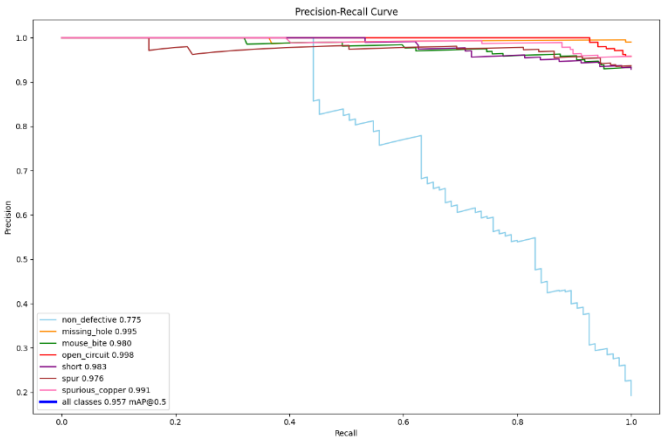

Figure 19: Faster R-CNN Precision-Recall Curve

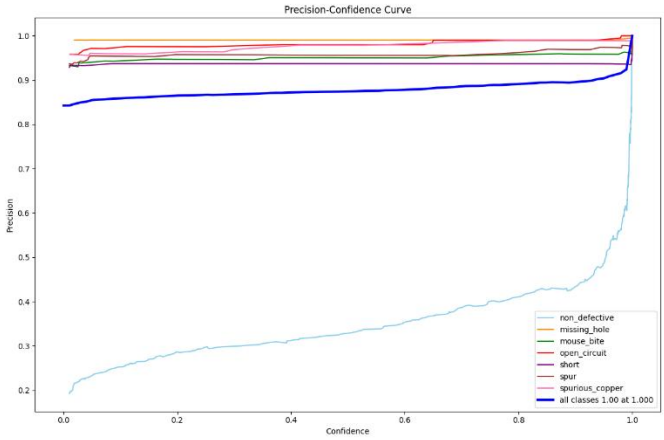

Figure 20: Faster R-CNN Precision-Confidence Curve

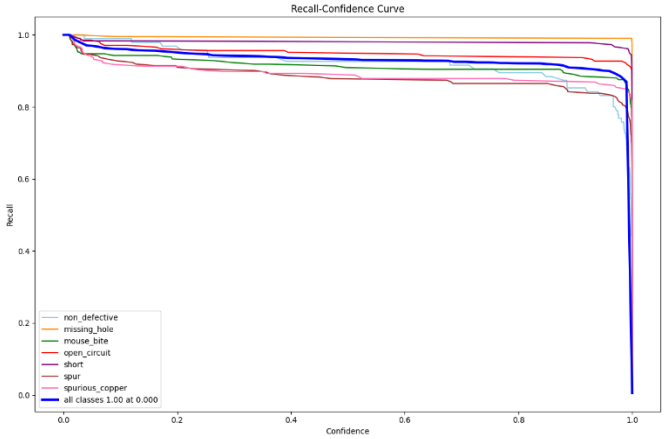

Figure 21: Faster R-CNN Recall-Confidence Curve

D. Tables Comparing Model Metrics

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

Table 1: F1-Confidence score for each model

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

Table 2: Precision-Confidence score for each model

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

Table 3: Precision-Recall Curve score for each model

VI. Conclusion

The goal of this project was to train, test and evaluate three different object detection models, these models being YOLOv8, Faster R-CNN and SSD Mobile Net for the purpose of detecting defects in PCBs during manufacturing.

We found that the YOLOv8 model had the best overall performance of all the models. YOLOv8 was determined to be the best overall model based on its high accuracy, speed and consistency. SSD MobileNet provided quick results at a great loss to accuracy whereas Fast R-CNN was nearly as accurate as YOLOv8 but took a much longer time to run and was more resource intensive.

Model Conclusions:

YOLOv8

YOLOv8 provided the best performance of all three models. The model, after 100 epochs, had a mean average precision score of 97.2%. This model trained in less time than the Faster R-CNN model and provided significantly better results. The precision score of the model was 96.1% and its overall recall score was 96%, far outperforming the other two models. The model's ability to train and evaluate well on a small data set demonstrates its practicality for automated PCB defect Detection systems.

SSD MobileNet

SSD MobileNet required the fewest computational resources and trained the fastest of all the models, roughly three hours. Its performance in detection was poor, as demonstrated by its 2% precision and 27% recall scores. This may in large part be due to the architecture of the model in its attempt to be a lightweight option for embedded systems. The lack of depth is not optimized for complex tasks such as object detection and classification of defects that are very similar. This model is not a good option for high accuracy detection of PCB defects.

Faster R-CNN

Faster R-CNN performed well =, having high precision and recall scores across the defect class, with F-1 scores from 0.8 to 0.99. The model itself however struggled in the background class, which means the model struggled to classify normal and defective boards to some extent. The model achieved an overall accuracy of 68% and due to the poor performance in the non-defective class had an average F-1 score of 0.71. This combined with the fact that Faster R-CNN took the longest to train, makes the model not well suited for the conditions found in real-world manufacturing environments.

An important aspect of this project was Transfer Learning, the data set that was sourced was small compared to most that are used in training deep neural networks. Even with this limitation we were able to further train and tine these pre-trained models and achieve successful results. This demonstrated the practicality of implementing these types of models in a real-world manufacturing setting, without the need for a large, labeled dataset.

This project serves as a demonstration of a machine learning based alternative to manual PCB inspection that reduces the manpower needed to validate PCBs and minimizes the risk of human error during the inspection process.

VII. Future work

The project proved the feasibility of using object detection and machine learning, however there are a few ways that these models can be improved to work better in a real manufacturing environment:

Dataset Expansion

One of the major limiting factors of this project was the lack of a robust dataset. In the future work could be done to collect more PCB images, with and without defects, to give the models more training data which would improve the performance of the models.

Class Balancing & Noise Handling

The Dataset that was used for this project had a clear dataset imbalance, with some of the defect classes having far fewer images than others. So, balancing the number of defect images in each class could improve the model's recognition of those defects along with introducing noise into the images such as shadows, or blur.

Model Optimization

In the future, we could try to shrink down the models and make them run more efficiently, especially if we want to deploy them on embedded systems like a Raspberry Pi. Another idea is to try combining multiple models using an ensemble approach. This would combine the strengths of multiple model's object detection, so where one model lacks accuracy, the other can compensate.

Comparison with Newer Architectures

There are newer and more recent models for object detection such as YOLOv9 and EfficentDet. Future work could incorporate these newer models into the testing to see if there is an increase in defect detection and a reduction in false positives.

Future work has an enormous amount of potential to continue this study and improve upon the current results, making these models more reliable, accurate and faster, all of which lead to the end goal of a fully autonomous PCB defect detection system.

REFERENCES

[1]T. Ahmad, “Object Detection using MobileNet SSD - Tauseef Ahmad - Medium,” Medium, Aug. 25, 2022.

[2]A. S. CHOUDHARY, “Object Detection Using YOLO And MobileNet SSD Computer Vision -,” *Analytics Vidhya*, Sep. 22, 2022.

[3] W. Huang and P. Wei, "A PCB dataset for defects detection and classification," arXiv preprint arXiv:1901.08204, 2019. Available: https://www.kaggle.com/datasets/akhatova/pcb-defects

[3] You Only Look Once: Unified, Real-Time Object Detection; Joseph Redmon et al, 2016

[4] Y. -C. Chiu, C. -Y. Tsai, M. -D. Ruan, G. -Y. Shen and T. -T. Lee, "Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems," 2020 International Conference on System Science and Engineering (ICSSE), Kagawa, Japan, 2020, pp. 1-5, doi: 10.1109/ICSSE50014.2020.9219319.

[5] TensorFlow, "TFRecord and tf.train.Example," TensorFlow Tutorials, [Online]. Available: https://www.tensorflow.org/tutorials/load_data/tfrecord.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," arXiv preprint arXiv:1506.01497, 2015. Available: https://arxiv.org/abs/1506.01497.

[7] TensorFlow, "Transfer learning and fine-tuning," TensorFlow Tutorials, [Online]. Available: https://www.tensorflow.org/tutorials/images/transfer_learning.