National College of Ireland

Printed Circuit Board Defect Detection using YOLOv7

MSc Research Project
Data Analytics

Madhusudan Narasimhan

StudentID: 20197888

School of Computing

National College of Ireland

Supervisor: Mr. Prashanth Nayak

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Madhusudan Narasimhan
Student ID:	20197888
Programme:	MSc Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Prashanth Nayak
Submission Due Date:	15/12/2022
Project Title:	Printed Circuit Board Defect Detection using YOLOv7
Word Count:	6384
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Madhusudan N
Date:	15th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Printed Circuit Board Defect Detection using YOLOv7

Madhusudan Narasimhan
20197888

Abstract

This paper proposes using the YOLOv7 algorithm to identify defects on Printed Circuit Boards (PCBs) during the manufacturing process. Traditional methods for defect identification are costly and often result in false alarms. YOLOv7 is the fastest known object detection algorithm, making it well-suited for the high-demand environment of PCB manufacturing. The paper also discusses data augmentation techniques performed and performance improvements compared to current state-of-the-art object detection methods. The model building towards building a robust model was performed in five stages which included various levels of data augmentation and multi-stage fine tuning on the model. After considerable iterations the model built was performing well and the mAP @ 0.5 was recorded to be 95.8% with overall precision of the model at 98% percent and recall at 92%.

Table of Contents

1. Introduction

PCBs, or printed circuit boards, are crucial components of electronic circuits. They serve as foundation to establish connections between other electronic components. PCBs are manufactured with multiple layers of copper sheets on which the other components such as ICs, resistors, capacitors, relay switches are soldered. PCB manufacturing requires high precision and must be detail oriented, as any defects can cause serious problems at the product level (Zhang, et al., 2021).

There are three major challenges in detecting defects in PCBs (Ding, et al., 2019). The first is the variety of different PCB designs and wiring rules. The second major challenge is the involvement of large range of categories and characteristics of PCB defects. The third major challenge is the issue of class imbalance, as gathering samples for each type of defect is a tedious process.

PCB defects can be divided into functional defects, which affect performance, and cosmetic defects, which affect appearance. Cosmetic defects can also cause long-term problems with performance due to abnormal current distribution and heat dissipation. (Indera Putera & Ibrahim, 2010).

Some common defects within these categories include bad soldering, missing solder masks between pads, mouse bites, plating voids or missing holes, open circuits, shorts, electromagnetic issues, spurs, acid traps, starved thermals, and environmental factors. It is important to identify and fix faulty products at their source[1].

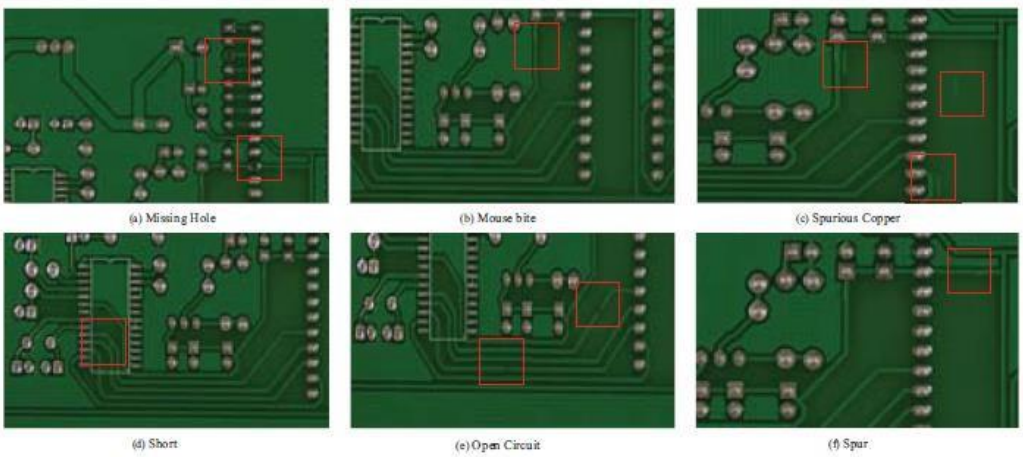

Figure 1 Common Defects in PCBs

This study focuses on identifying six common defects that occur in industrial settings: missing holes or plating voids, mouse bites, open circuits, shorts, bad soldering, and spurs. Figure 1 shows examples of these defects. It is crucial to identify and fix these defects at their source.

Figure 2 shows the current industry-standard system for identifying defects in PCBs. The process starts with the PCBs being scanned by an automatic optical inspection (AOI) system. The AOI system examines images to identify those that may have defects or issues, but it can produce a high number of false alarms because of the strict parameters it uses for classification. After being scanned by the AOI, the suspected PCBs are sent to a verify and repair station (VRS), where a human operator verifies the defective board. The cost of this process can be high because of its importance. In recent years, there have been several

[1] https://www.mclpcb.com/pcb-guide/

attempts to use computer methods, including traditional machine learning and deep learning, to improve the process (Zhang, et al., 2021).

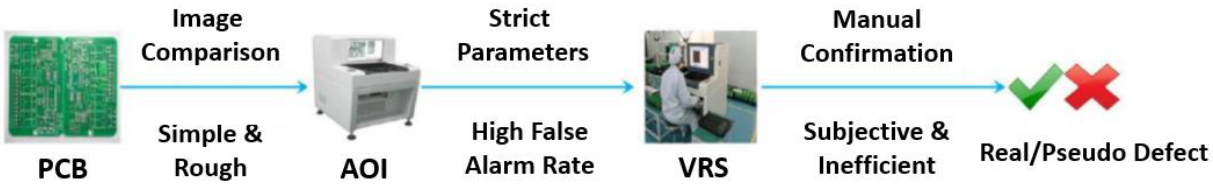

Figure 2 Traditional industry practice for defect detection in PCBs

Transfer learning is a machine learning technique where a model trained on one task is reused as the starting point for a model on a second related task. This can be useful when the second task has a smaller amount of training data, or when the second task is similar to the first but not identical. In the context of the problem statement in this paper, transfer learning would involve taking a pre-trained object detection algorithm and adapting it for use in defect detection. This could involve retraining the model on a new dataset of defective images, fine-tuning the model's hyperparameters, or both. By using transfer learning, the model can leverage the knowledge gained from the original task to learn the new task more quickly and effectively. This can save time and resources and can often lead to improved performance.

Object detection algorithms can be divided into two categories: two-stage approaches and single-stage approaches. The two-stage approach breaks down the problem into two steps: first, identifying potential object regions, and second, classifying the objects within those regions into their corresponding classes[2].

The One Stage approach uses a neural network that predicts bounding boxes and class probabilities in a single step. This contrasts with the two-stage approach, which typically involves two separate stages for predicting bounding boxes and class probabilities.

This paper presents a study on using YOLOv7, a single-stage approach, for detecting defects in printed circuit boards (PCBs). The study aims to record the performance of YOLOv7 in identifying PCB defects and compare it with state-of-the-art methods currently being used.

Research Question: "How well does the YOLOv7 Object detection algorithm perform in detecting defects on PCBs compared to state-of-the-art methodology?"

The remainder of this paper is structured as follows: Section 2 discusses the work carried out by fellow scholars and experts in the field of computer vision. Section 3 introduces to the methodology implemented in detail. Section 5 discusses the results obtained on the study performed. Section 6 concludes the study conducted and provides potential future work that can be carried out. Section 7 documents the references utilized in this paper.

2. Literature Review

There have been several approaches to solve the problem discussed in this paper, including using conventional image processing techniques, as well as machine learning and deep learning methods. Transfer learning is one way in which this problem can be addressed, and researchers in this field have explored various techniques to do so. Some of these techniques involve using deep learning approaches.

2.1. Using Autoencoders

The authors (Mujeeb, et al., 2019) have proposed a solution which uses an auto encoder for feature extraction. The authors assert that their method allows for the detection of various defects without requiring the training of a model using defective samples. The authors of this

[2] https://www.v7labs.com/blog/yolo-object-detection

study employed data augmentation to generate a replicated sample of a reference image. They then used this data to select a feature descriptor for the input image and repeated this process with a test image. The similarity of the two descriptors was then checked using a SIFT similarity matching technique. However, this method has the disadvantage of not being able to classify defects, as there are many ways that a PCB can be defective.

The authors (Khalilian, et al., 2020) employed denoising convolutional autoencoders for defect detection. The autoencoders were trained on corrupted PCB images, and the aim was to receive denoised images as output. By comparing the output image to the input image, defects could be identified. The authors (Kaviyasri & Binu, 2022) have also followed a similar approach using denoising convolutional autoencoders. However, this method has the disadvantage that as the network depth increases, a degradation problem can arise, which can lead to a decrease in the accuracy of the results.

The authors (Kim, et al., 2021) also examined a type of convolutional autoencoder called skip-connected autoencoders. In this method, the encoder and decoder layers are skip-connected, which helps to avoid the saturation problem that can occur when the network depth is increased. These skip connections allow the encoder and decoder layers to better handle image prediction tasks. However, this method may be vulnerable to input datasets with class imbalances.

The authors (Tsai & Jen, 2021) employed unsupervised autoencoder learning for defect detection. The convolutional autoencoder (CAE) they used to include a regularization technique to improve the feature distribution of defect-free samples by limiting it within tight margins. This allows the representative feature vector of all samples and the mean feature vector to be as close as possible, enabling the distinctive identification of defective samples during the evaluation stage. However, this approach can also be vulnerable to class imbalanced datasets.

2.2. Using ResNet

Multi-scale feature maps to build boundary boxes within a particular range was a method proposed by (Ran, et al., 2020). The authors applied small convolutional kernel to perform the predictions. Non-maximum suppression was used to optimize the results.

In a separate study, the authors (Tang, et al., 2019) extracted features from input images by using a convolutional backbone with max pooling. The authors calculated the differences between the features of the template and the tested images post feature extraction. A group pyramid pooling module was used by the authors to obtain features from different resolutions of the input. The authors made predictions using a backbone architecture (either VGG16 or ResNet 18) using the feature maps obtained from different scales.

The authors (Wu, et al., 2021) use a VGGNet-based SSD model with a FPN layer for multi-scale feature map prediction. The FPN layer uses a deconvolution layer and a lower sampling layer. They use a ResNet101 model as the backbone network, but their approach does not effectively handle class imbalance in the dataset, even when tested on two different datasets.

The authors (Ding, et al., 2019) have modified the Faster R-CNN network (FR-CNN) by making three key changes in the ResNet101 which was used as the backbone network. First, they trained the model on the PCB dataset and fine-tuned it by designing suitable anchors and using data augmentation. Second, they used a multi-scale feature fusion strategy to improve detection performance. Third, they employed online hard example mining during training to improve the quality of the regions of interest (RoIs).

The authors (Hu & Wang, 2020) have created a new Faster R-CNN-based network for detecting defects on printed circuit boards (PCBs). The backbone architecture is a combination of ResNet50 and Feature Pyramid Networks (FPN), which the authors say improves feature extraction and defect detection. They use GARPN for anchor prediction and combine ShuffleNetV2 residual units to reduce the load on the network. Finally, they use ROI pooling to generate object proposals and a fully connected layer to classify and produce the final defect detection results.

The authors (Zhang, et al., 2021) have developed a method for detecting cosmetic defects on PCBs that considers the issue of class imbalance in the data. They use a ResNet to extract features from the input images, which are then used by a cost-sensitive adjustment layer to calculate a weighted loss, update network parameters, and classify the images. While this approach can handle class imbalance, the large number of weight parameters (25.5 million) can make the process slow.

2.3. Using SSD

The authors (Shi, et al., 2020) use a Single Shot object Detector (SSD) for detecting small irregularities on PCBs. They enhance the SSD by adding a semantic ascending module that reduces the depth of semantic properties between deep and shallow layers by fusing features at different levels. To enable the model to learn the relationship between the fused features, they use an attention mechanism. In addition to the attention mechanism, they also use a shuffle module to eliminate the aliasing effect after fusion.

The authors (Li, et al., 2021) perform semantic labelling on their PCB dataset and then use a traditional SSD neural network for PCB defect detection. However, this method has been known to sometimes ignore important features in the image.

The authors (Tang, et al., 2019) (Jiang, et al., 2022) have used SSD by improving it. Their model uses a coordinate attention mechanism module to address the issue of ignoring important features. This modification is intended to improve the performance of the SSD network for PCB defect detection.

While SSD is a single-stage algorithm like other single-stage algorithms, it is known to be slower than some of them. In the PCB industry, speed is important because it can affect production and lead to mismatches between supply and demand. Therefore, using a slower algorithm like SSD may not be ideal in this context.

2.4. Using YOLO

The authors (Li, et al., 2020) have experimented with a combination of different models, including ResNet101, YOLOv2, and Faster RCNN with ResNet101 and FPN. They used YOLOv2 with a VGG16 backbone and ResNet101 was used as the backbone for the Faster RCNN setup. They set up the models with low threshold values in an attempt to reduce False Alarm Rates (FAR). While they were able to achieve reasonable performance, the design of the circuit seems complex and newer, faster algorithms have been developed since then.

(Adibhatla, et al., 2020) made use of the Tiny-YOLO-v2 architecture. The authors managed to obtain data from an AOI and were able to label the defective PCBs. This data was then used to train the YOLO network with around 5.7 billion operations using 12M weights.

The authors (Lan, et al., 2021) have brought in four changes to the YOLO v3 model to enhance its performance for detecting defects on PCBs. First, they combined the batch normalization and convolutional layers to improve on the training speed of the model. Second, they used the GIoU performance metric to improve the detection of small and medium targets. Third, they used K-means++ to determine the anchor boxes. Finally, they used multi-scale training to enable detection of images at different resolutions efficient.

The authors (Xin, et al., 2021) have improvised on the YOLOv4 network by replacing the Darknet53 backbone with five CSPDarknet53 modules. This change is intended to reduce memory cost, alleviate computing bottlenecks, and improve the learning ability of the convolutional neural network (CNN). Additionally, they perform downsampling several times, which produces a feature map that is the FPN layer then takes these feature maps as input. This is then passed through the PAN structure twice to output the predicted feature map. These modifications are intended to improve the performance of the YOLOv4 network for detecting defects on PCBs.

The authors (Xie, et al., 2021) have used YOLOv4 architecture that incorporates three key modifications. First, they use a multistage residual hybrid attention module (MRHAM). The MRHAM assists in enhancing feature learning by focusing on relevant features and ignoring irrelevant ones. Second, they use K-means clustering to determine the anchor boxes values. This aids in improving the accuracy of small object defect positioning. Finally, they employ data augmentation, transfer learning, and multi-scale training methods to improve the generalization of the model. These modifications are intended to enhance the performance of the YOLOv4 network for detecting defects on PCBs.

The authors (Adibhatla, et al., 2021) used the traditional YOLOv5 model. The author has cited advantages of YOLOv5 over other YOLO algorithms to justify their choice. Their results proved that they were able to take full advantage of the architecture to build a robust model. However, YOLO is known for its limitations in detecting small objects that occur in groups.

The authors (Bhattacharya & Cloutier, 2022) used YOLOv5 model that combines transformers and a convolutional neural network (CNN). The work of the CNN, in this architecture, is used to extract underlying geometrical features. The transformer is used to directly process the feature maps produced by the CNN. Around 7 million weight parameters were used to build the model. This is a low number, and this results in faster model processing. This modified YOLOv5 model is intended to improve the detection of defects on PCBs.

Researchers have compared the performance of various YOLO models on different datasets and environments.

The researchers (Jiang, et al., 2021) conducted a thorough comparison of various YOLO models, including their similarities and differences, and their performance on different datasets and environments. They reportedly focused on the YOLO family of algorithms up to version 5 and found that YOLO's grid division plays an important role in detection. In version 2, anchor and K-means were added, as well as a two-stage training with a full convolutional network. YOLOv3 added multi-scale detection using FPN (Feature Pyramid Network), and later versions added new activation functions and data augmentation techniques. Overall, it seems like the researchers found that YOLO has evolved and improved over time, with each new version adding new capabilities and enhancements.

Researchers (Nepal & Eslamiat, 2022) used YOLOv3, YOLOv4, and YOLOv5l to identify landing spots for UAVs in urban environments. Their findings showed that YOLOv5l was more accurate than the other two versions, while maintaining a slower inference speed. This suggests that YOLOv5l may be a good choice for applications that need high accuracy, even if it comes at the cost of slower performance.

Researchers (Wang, et al., 2021) used various versions of YOLOv5 to detect PPEs in construction sites. They found that YOLOv5x was the best performing in terms of mAP, and YOLOv5s was the fastest among the YOLOv5 family. This suggests that YOLOv5x may be good for applications that need high accuracy, while YOLOv5s may be better for applications that require fast performance.

These studies further iterates that the performance of YOLO over the years have been constantly improving and has set new benchmarks in terms of speed and accuracy.

3. Design Specification

Figure 3 shows the methodology being followed in this study. Dataset is sourced from Kaggle. The dataset contains 693 images distributed equally among the 6 classes to be identified in this study. The dataset is split into 70% training set and 20% test set and 10% valid set. Several data augmentation methods are also carried out to increase the robustness of the model.

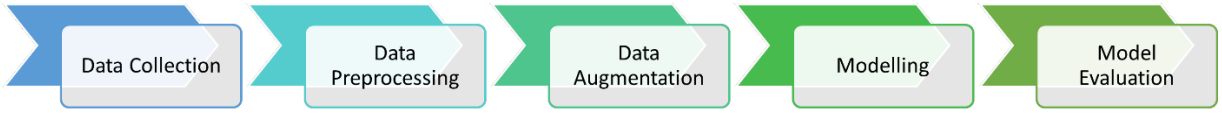

Figure 3 Process flow

4. Research Methodology

This study follows CRISP-DM methodology. The steps involved in CRISP DM include Business understanding, Data understanding, Data Preparation, Modelling, Evaluation, Deployment.

4.1. Dataset

The dataset[1] used in this research is sourced from Peking University PCB defect dataset. The dataset originally contains 693 images and 6 classes of defects namely mouse bite, missing hole, open circuit, short, spur, spurious copper. The dataset is provided with files labels of the defects in the XML format. This filetype does not meet the requirements of the experimental model and is transformed into the TXT format which is more suitable to serve as the input for the model. The conversion of formats was performed on a web platform called roboflow.com. The platform enables to annotated images into necessary format for processing. The platform also enables data augmentation and pre-processing with a single click. Figure 4 shows the various data augmentations and annotations performed on the images on Roboflow.com.

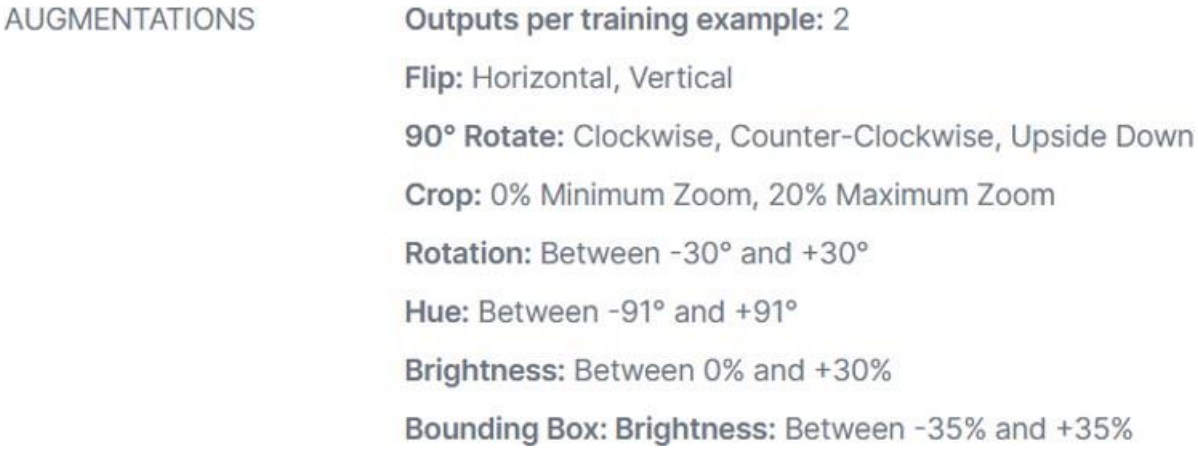

Figure 4 Augmentation operations

[1] https://www.kaggle.com/datasets/akhatova/pcb-defects

4.2. Data Pre-processing

The dataset of 693 images with almost 115 images in each class was divided into 70% training dataset and 20% testing and 10% validation dataset. Images were resized to 864 pixels on each dimension. Contrast was adjusted using Histogram Equalization technique where the contrast is boosted using images histogram to improve normalization and line detection in varying lighting conditions. Grayscale was also applied to the images to help the algorithm differentiate the defects easier.

4.3. Data augmentation

The image data was augmented post annotation. 5 levels of augmentation were performed on the images namely 1:1, 1:2, 1:3, 1:5 and 1:7. In 1:1 data augmentation, each image is augmented exactly in 1:1 ratio, which means that post data augmentation, the number of images obtained remains the same as original. In 1:2, each image is augmented twice, which means that the number of images obtained post augmentation is 2 times the original set of images and so on. For the purpose of this research the following data augmentation steps were followed: i) Horizontal flip, ii) Clockwise rotate 90 degrees, iii) Counter-clockwise rotate 90 degrees, iv) upside down rotate, v) Crop – 0% minimum to 20% maximum, vi) Rotation – between 15 degrees to -15 degrees, vii) Hue – between -91 degrees to 91 degrees, viii) brightness – between 0% and 25%, ix) Bounding box brightness – between -20 degrees and +20 degrees.

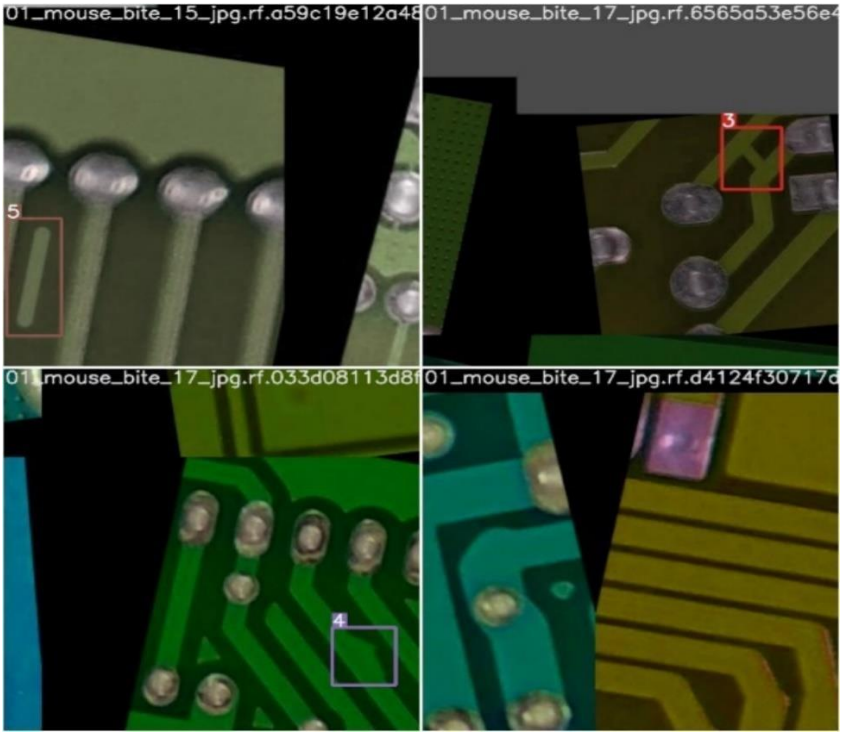

Figure 5 Augmentations performed in YOLOv7.

Apart from these data augmentations YOLOv7 also performs few unique augmentations on the data called *Mixup*, Mosaic augmentation, etc. In *Mixup* random parts of few images are mixed up with each other and are overlapped upon one another. In Mosaic data augmentation, 4 different images from are formed into a grid and these images are then trained by the model.

4.4. Model Building

YOLOv7 is the model proposed to be used for training and predicting the PCB defects. YOLOv7 is the latest of the YOLO family and is claimed to be the fastest yet YOLO

algorithm in the YOLO family and among other object detection algorithm. YOLOv7 is claimed to set new state-of-the-art for real-time object detectors. (Wang, et al., 2022) claims the model to predict video inputs ranging from 5 FPS to 160 FPS. YOLOv7 has the highest Average Precision at 56.8%. YOLOv7 outperforms both transformer-based object detectors and convolutional based object detectors such as the previous versions of the YOLO.

4.4.1. YOLO Architecture

YOLO is designed based on a Fully Connected Neural Network (FCNN). In recent times, a notable change in the architecture is the addition of transformer-based version in the YOLO family. The YOLO framework has three main components namely the Backbone, Head and Neck. The feature of an image is extracted by the backbone and is fed to the head through the neck. The feature maps extracted by the head are collected in the Neck and feature pyramids are created. The output collected is displayed in the head.

Several architectural reforms are introduced to improve speed and accuracy of YOLOv7. Some of the notable architectural reforms is the addition of E-ELAN (Extended Efficient Layer Aggregation Network) and model scaling for concatenation-based models.

E-ELAN, or the Extended Efficient Layer Aggregation Network, is a deep learning architecture for object detection developed by researchers at the Chinese Academy of Sciences. The E-ELAN architecture is designed to improve the efficiency and accuracy of object detection algorithms, while also reducing the amount of computation and memory required. The E-ELAN architecture is based on a multi-scale feature aggregation approach, which allows it to capture rich, detailed information about the objects in an image, while also reducing the computational complexity of the network. The E-ELAN architecture consists of multiple layers of neurons, which are organized into a hierarchical structure.

The first layers of the E-ELAN network extract low-level features from the input data, such as edges, colors, and textures. These features are then passed through the next layers of the network, which combine and transform them into higher-level features, such as shapes and objects. The higher-level features are then aggregated across multiple scales, allowing the network to capture detailed information about the objects in the input data.

One key advantage of the E-ELAN architecture is its ability to adapt to the size and scale of the objects in the input data. This allows the network to make more accurate predictions and to reduce the number of false positives. The E-ELAN architecture also uses a multi-task learning approach, which allows it to make predictions about multiple object classes and attributes simultaneously.

Another major addition introduced in the YOLOv7 is the concept of Trainable BoF. Trainable BoF, or Trainable Bag-of-Features, is a technique for unsupervised learning and representation learning in computer vision. The Trainable BoF approach uses a set of local features, such as SIFT or SURF, to represent an image, and then clusters these features into a fixed-size bag-of-features (BoF) vector.

The Trainable BoF approach allows the clustering of the local features to be performed in a supervised manner, using a training set of labelled images. This allows the BoF vector to be trained to capture the relevant visual information for a specific task, such as object classification or image retrieval.

The Trainable BoF approach has been shown to improve the performance of image classification and object recognition tasks, compared to unsupervised BoF approaches that do not use labeled data. It can also be combined with other techniques, such as spatial pyramids and support vector machines, to further improve the performance of the BoF vector on specific tasks.

Overall, the Trainable BoF approach is a useful technique for unsupervised learning and representation learning in computer vision. By using a supervised learning approach to train the BoF vector, the Trainable BoF approach can improve the performance of image classification and object recognition tasks.

4.4.2. Model building

Model building in this research was performed in 6 stages. Initially a baseline model was built on the dataset collected without performing augmentation. The baseline model was allowed to run for 50 epochs with a learning rate of 0.1, with a batch size of 4. The images were rescaled to 864 x 864 pixels for better training quality. Stochastic Gradient descent optimizer was used as suggested by fellow researchers as it had a better effect compared to Adam and Adam W optimizers.

Stage two was performing 1:1 data augmentation, i.e., setting the output per training sample to be 1. This increases the size of the dataset by a factor of two. The baseline model was trained on the augmented data.

Stage three was performing 1:2 data augmentation where the output per training sample was set at 2. This implies that the size of the dataset is increased by a factor of three. The weights generated from the previous model built is carried over and is trained on the newly augmented data.

Stage four was reusing the previously trained model weights on 1:3 data augmentation configuration. It was observed that the model performed best in the 1:3 configuration.

In stage five, the model was fine-tuned by introducing the model to noisy data. This step is carried out to improve robustness of the model. The model was trained only on the augmented dataset. The augmentations performed includes alterations in blur, hue, flip, rotations, brightness, bounding box brightness.

These model weights were then trained on the original data without augmentations, and it was observed that the model performed exceptionally well. The results of these six stages are discussed in the next section.

5. Evaluation and Results

In this research, the metric used for evaluation of the model is mean Average Precision (mAP). Most of the object detection algorithms use mAP and IoU (Intersection over Union) as evaluation metrics. mAP is the measure of how well the model can identify and locate objects in an image. mAP is calculated by first computing the average precision for each class, and then taking the mean of these values. In other words, mAP is the mean of the average precision scores for each class.

The model built in this study was trained with a number of changes to the dataset over a time period of 50 hours. The model built was evaluated for its performance at 5 different stages. The first stage of the model is the baseline model where the model was built on raw data without performing any augmentation or samplings. The model returned a mAP of 44% with overall Precision 74% and overall recall at 38% after a single iteration.

Figure 6 represents the baseline model results obtained after training the model on the baseline dataset. X axis in Figure 6 represents the number of epochs and Y axis represents the headings mentioned on the top of the graph. This graph is one of the outputs obtained post training the YOLOv7 model. The most important graphs to be noted here are the Precision, Recall, mAP@0.5 and mAP@0.5:0.95 graphs. mAP@0.5 denotes mean average precision at 0.5 threshold. mAP@0.5:0.95 is the mean average precision recorded at different thresholds from 0.5 to 0.95. The threshold mentioned above is the value of

Intersection over Union (IoU) of the bounding boxes. The Precision graph from the figure shows the precision recorded for each epoch. The highest precision achieved was around ~97% at the roughly the 25th epoch. The overall trend shows there is a steady rise in the precision after the 30th epoch. The graph gets even better after the 41st epoch.

The X axis of the Recall graph represents the number of epochs, and the Y axis of the Recall graph represents the recall achieved over each epoch. The Recall of the model has gradually and steadily increased over the 50 epochs after the 25th epoch. The highest recall value achieved for the baseline model is around 0.44 or 44% which is a pretty low number.

Similarly considering the mAP@0.5 and mAP@0.5:0.95 graphs over 50 epochs there has been a steady increase and the maximum mAP@0.5 was achieved at the 50th epoch which has a value around 0.44 or 44%. This is again a low number. The decrease in the rate of increment shows that more changes have to be incorporated for the model to learn.

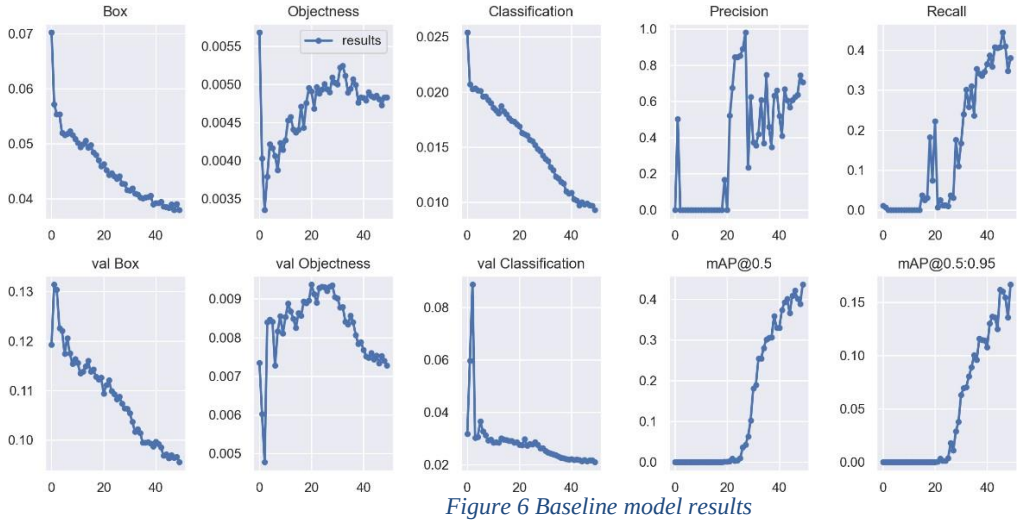

Figure 6 Baseline model results

In **Error! Reference source not found.**, PR curve for the baseline model is recorded, and it is observed that the curve lies just above the threshold line. It can be concluded that the model does not perform too well while being trained with baseline dataset.

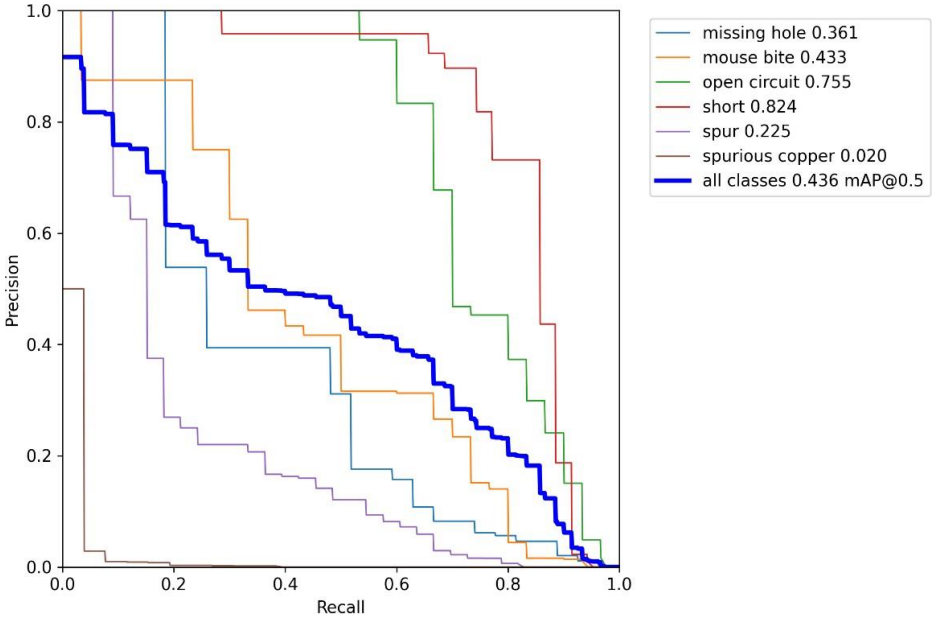

Figure 7 PR Curve - Baseline

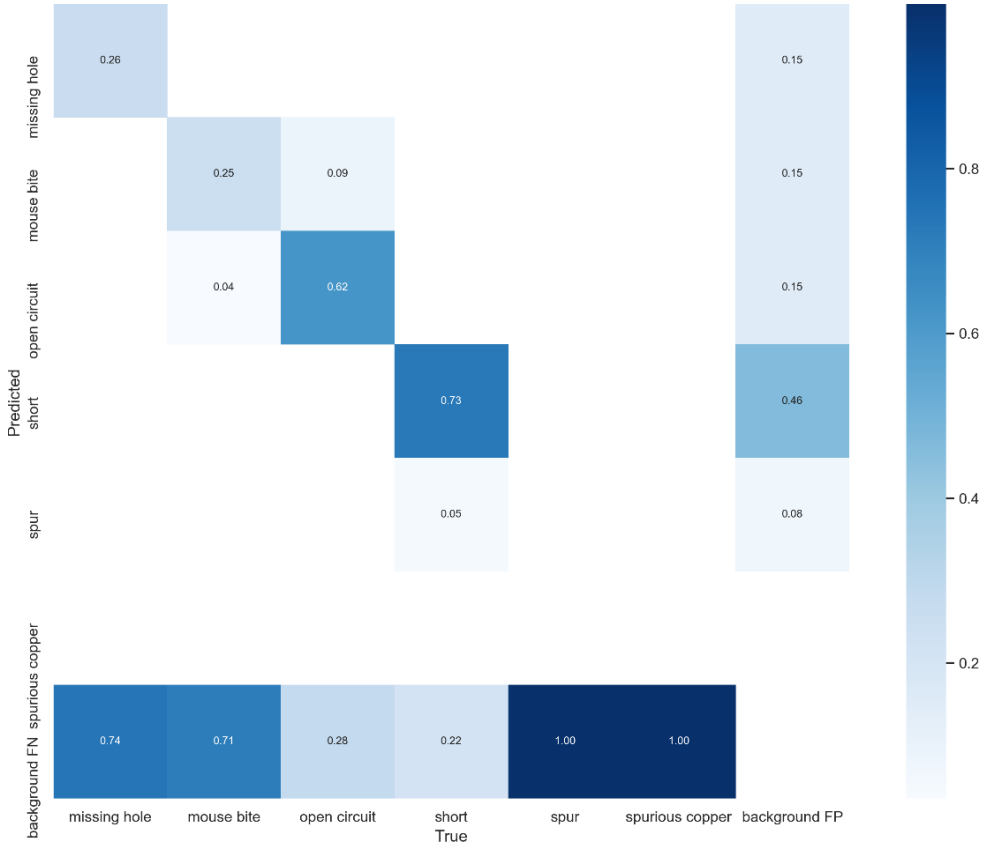

Figure 8 Confusion Matrix - Baseline

Based on results recorded as shown in Figure 8, it can be noticed that the model is not performing too well as it fails to identify spur and spurious copper classes and also performs poorly on other classes as well.

The model was saved and for the next iteration data augmentation was performed on the data with output for each example set at 1. The number of input images for training were doubled and the model was trained on the augmented data. The model showed an increase in performance. The mAP was recorded at approximately 73% which is a significant increase from the previous model. These results were achieved after training the model for a considerable amount of time with almost 250 epochs running in batches of 50 epochs each. The overall precision was recorded at 81% and the precision was at 70%.

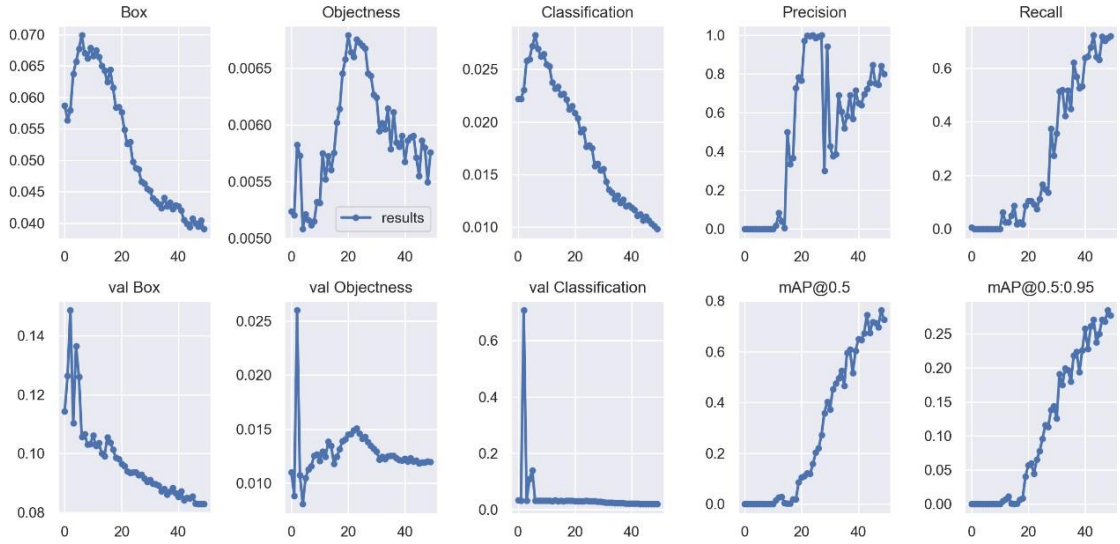

Figure 9 Results - Augmentation 1:1

Figure 10 records the PR curve for the model trained using 1:1 augmented data. Here, a drastic increase and the AUC is larger compared to the baseline model is observed. It can be concluded that the model trained using 1:1 augmented data performs better.

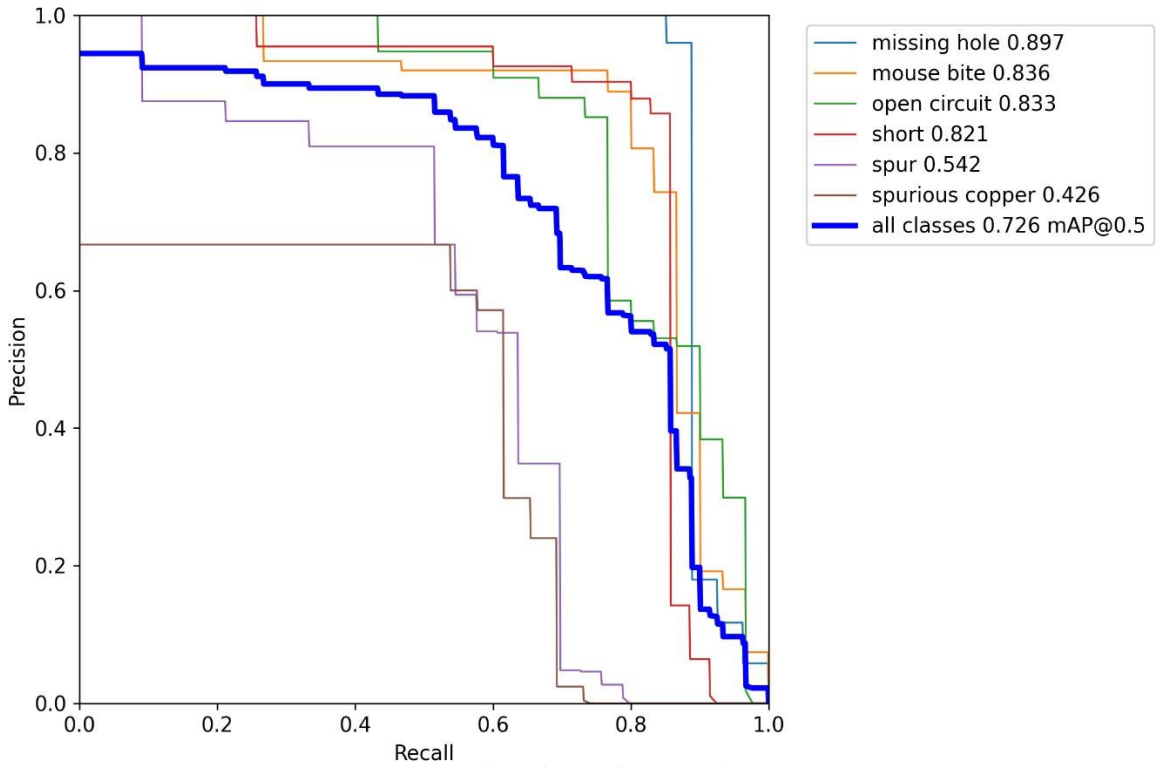

Figure 10 PR Curve - Augmentation 1:1

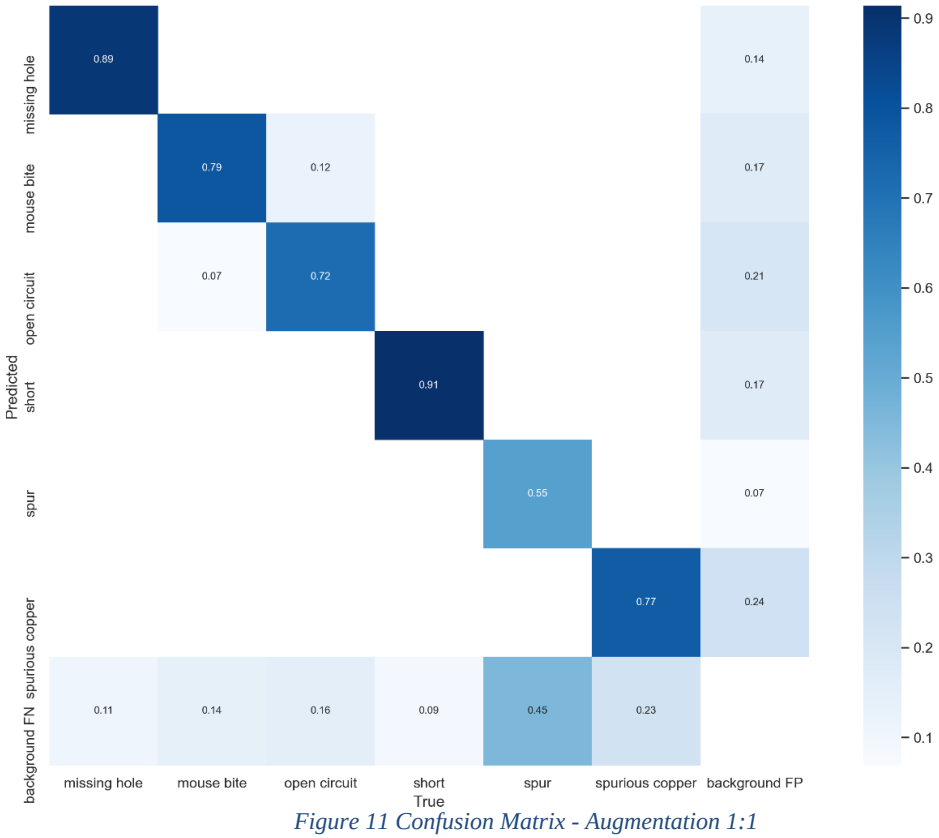

Figure 11 Confusion Matrix - Augmentation 1:1

Figure 11 represents the confusion matrix recorded while testing the model built on 1:1 dataset. A drastic increase in model performance can be noted as the model is now able to predict better compared to the baseline model. The model has performed significantly better in terms of predicting all classes.

The dataset size was further increased to 3 times the original size with output per example set at 2. The model from the previous iteration was trained on the newly augmented data. The model had a significant upward growth with mAP at 88%. The overall precision of the model was recorded at 89% and recall at 87%.

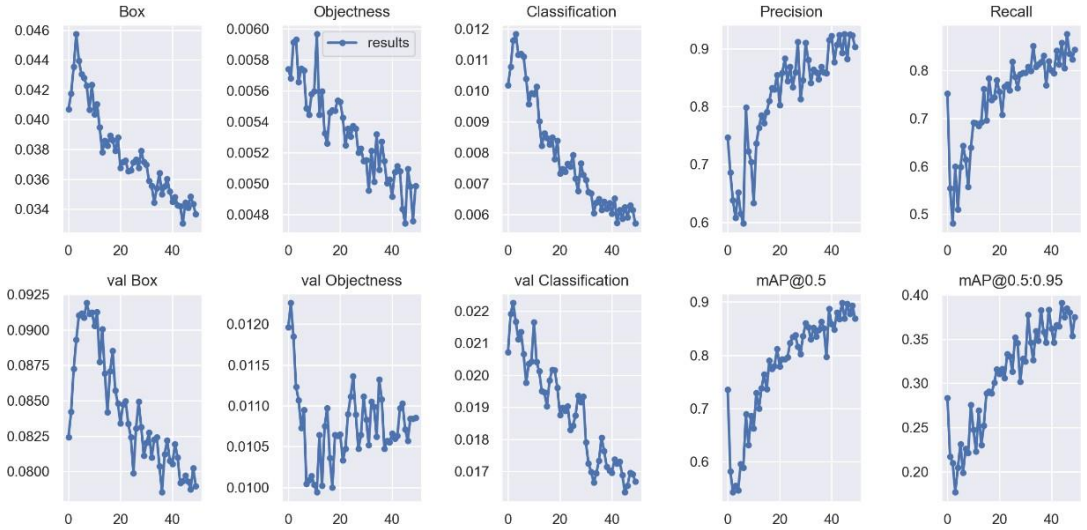

Figure 12 Results - Augmentation 1:2

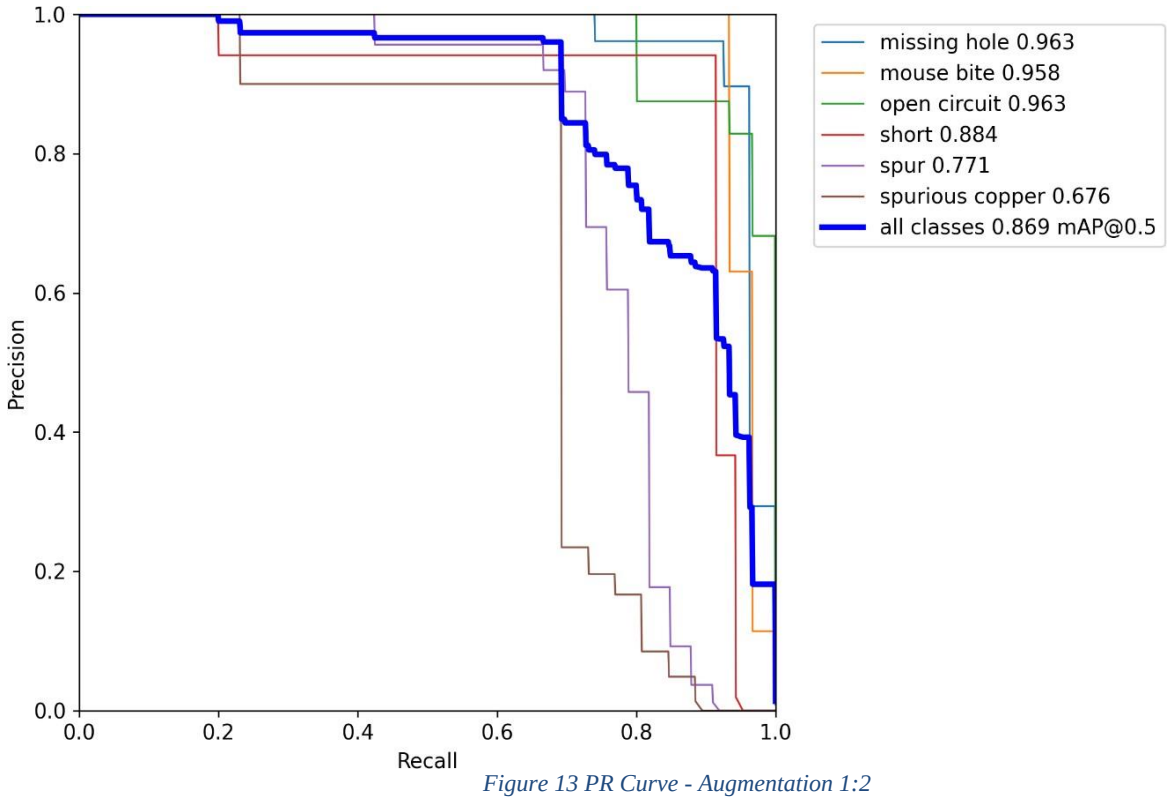

Figure 13 PR Curve - Augmentation 1:2

Figure 13 refers to the PR curve recorded when the model is trained using 1:2 augmented data. It can be noticed that this graph has larger AUC than compared to the PR curve recorded while using the 1:1 augmented dataset.

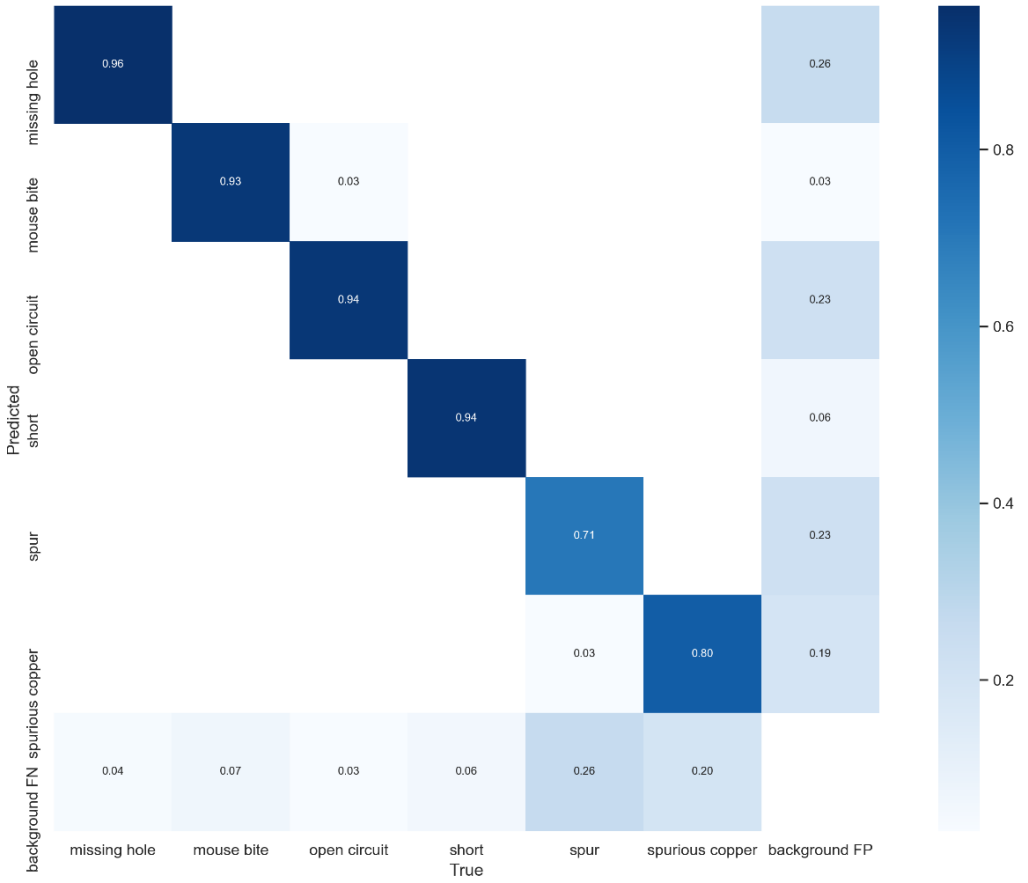

Figure 14 Confusion Matrix - Augmentation 1:2

Figure 14 represents the confusion matrix recorded while testing the model built on 1:2 dataset. It can be noticed that the model performs well on 4 classes with an accuracy of over 93% in each of the classes. However, the classes spur, and spurious copper still have room for improvement and hence the other level of data augmentation and multistage fine tuning have been performed.

The same steps were repeated for data augmentation with output per example set at 3 as well. This increased the data size to 4 times its original size. The mAP for 1:3 data augmentation was recorded at 92%. The overall precision and recall of the model were recorded at 85% and 89% respectively.

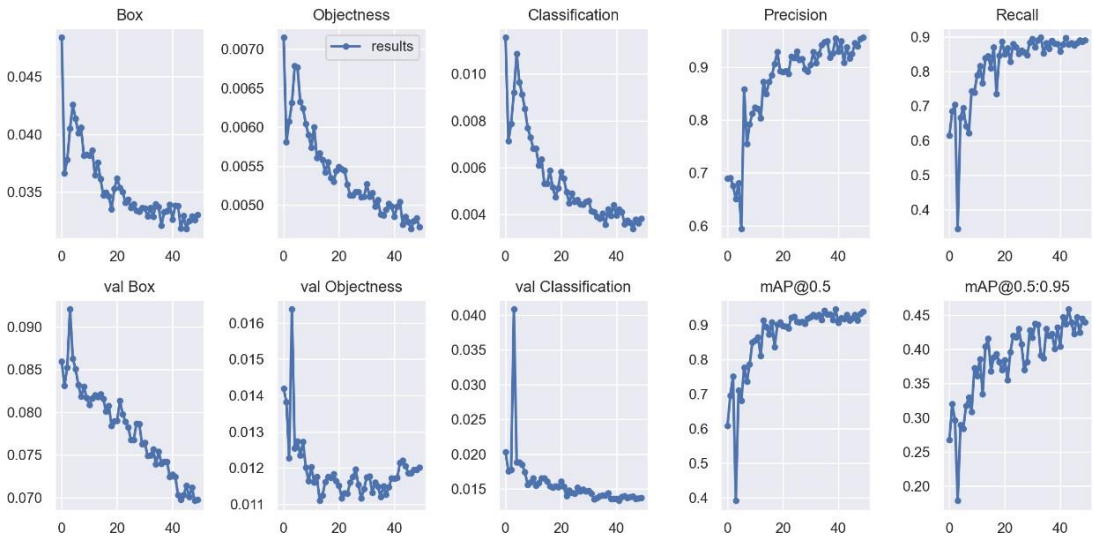

Figure 15 Results - Augmentation 1:3

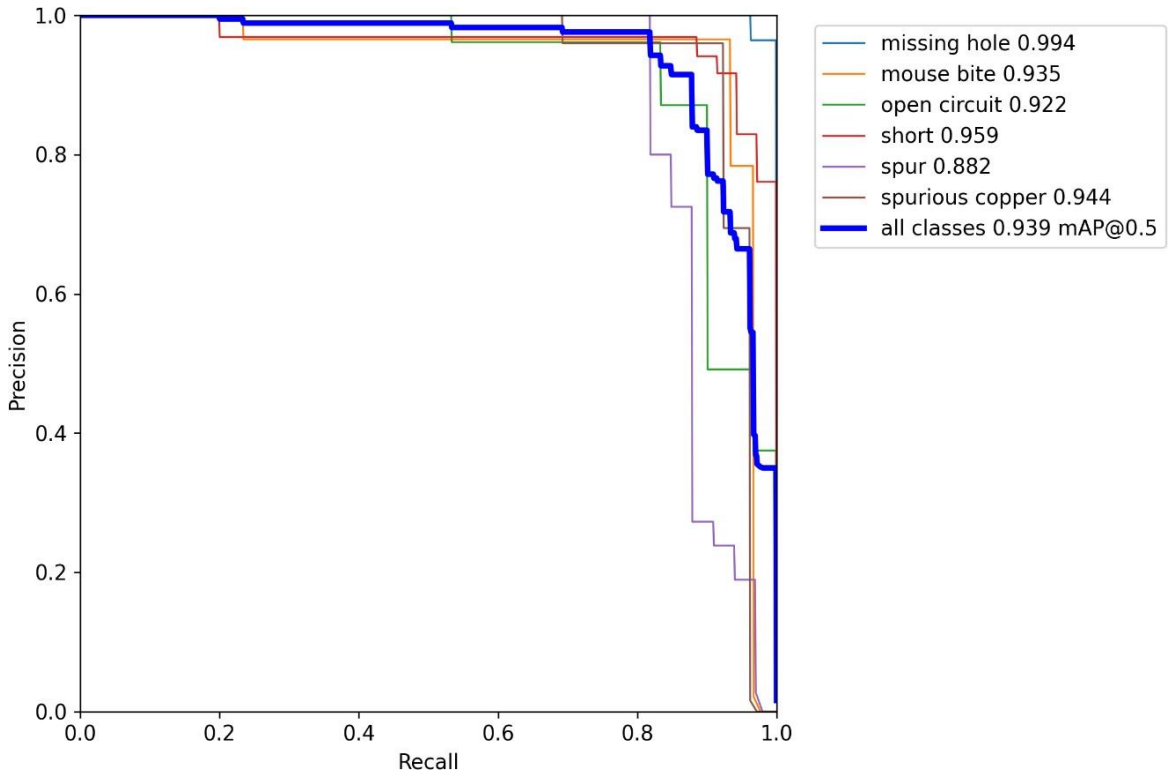

Figure 16 PR Curve - Augmentation 1:3

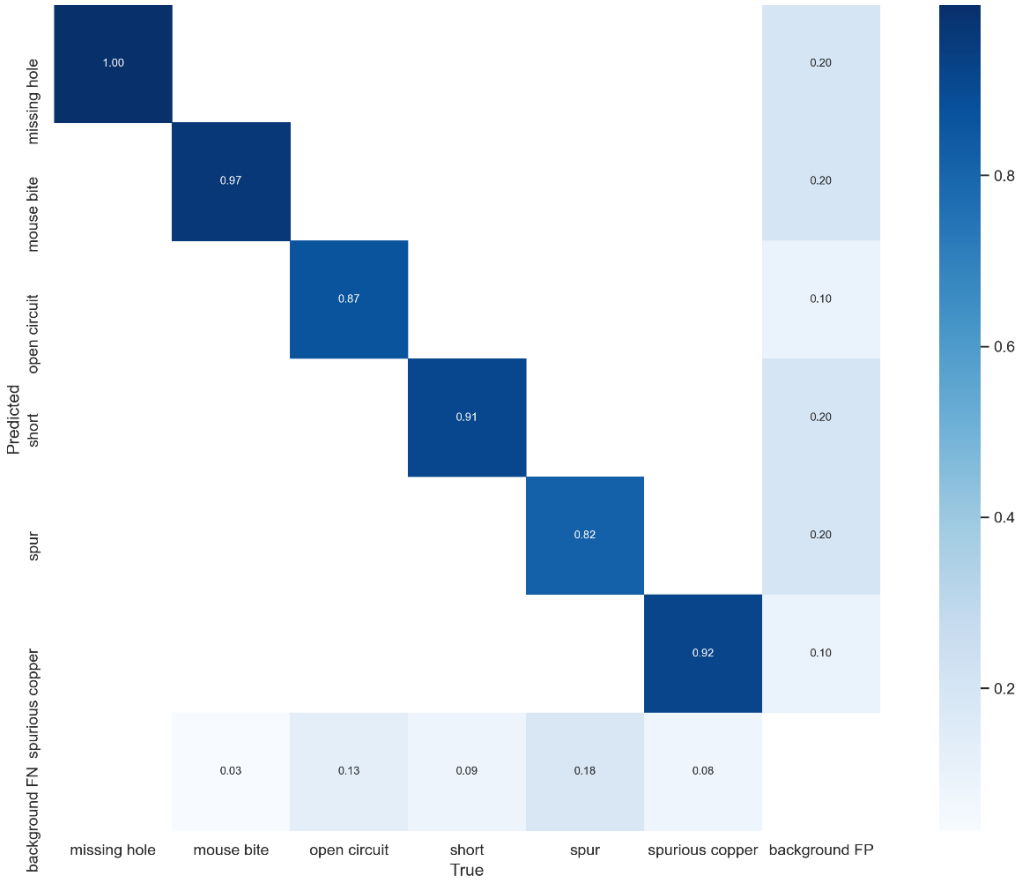

Figure 17 Confusion Matrix - Augmentation 1:3

The model trained on 1:3 data augmentation was recorded to be the best performing model. To further enhance the model performance and increase its robustness multi-stage fine tuning was performed.

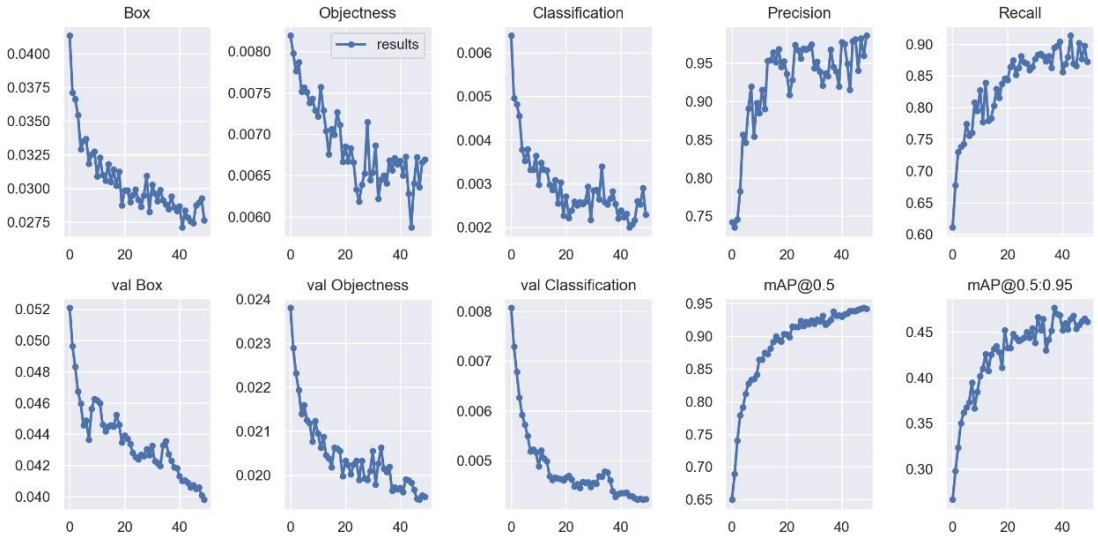

Figure 18 Results - Multistage Fine tuning

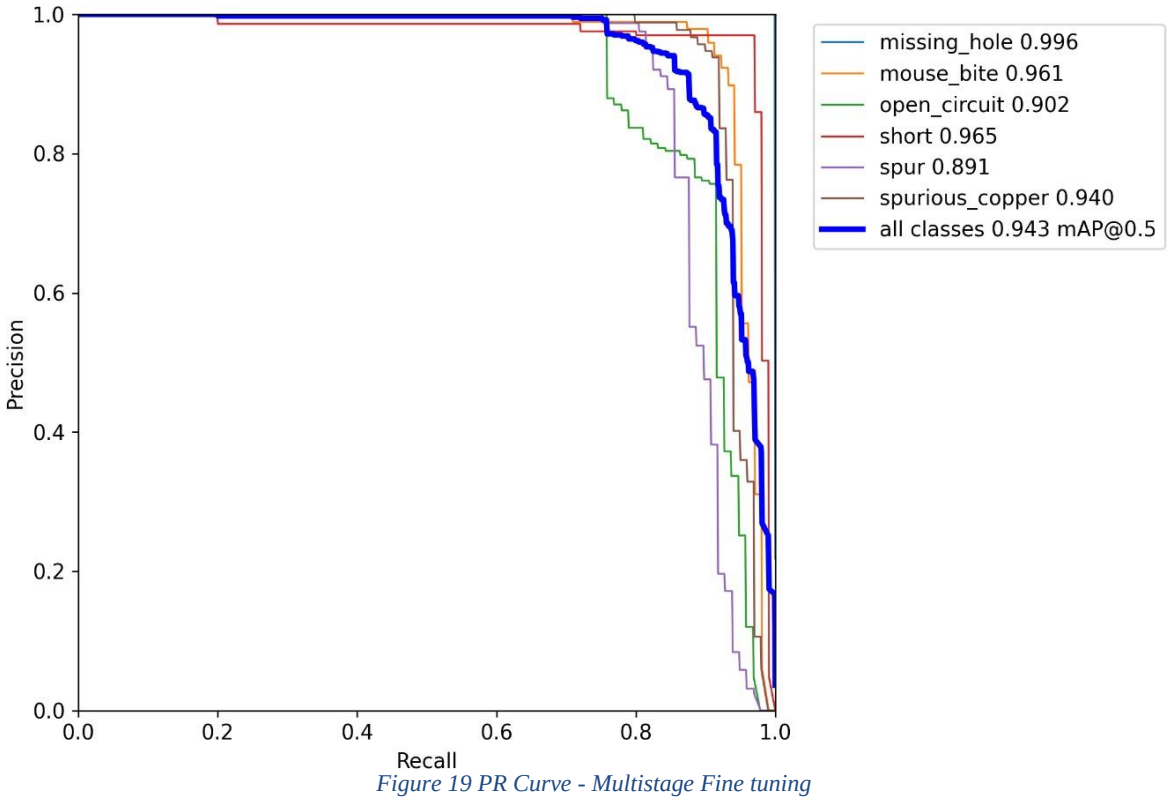

Figure 19 PR Curve - Multistage Fine tuning

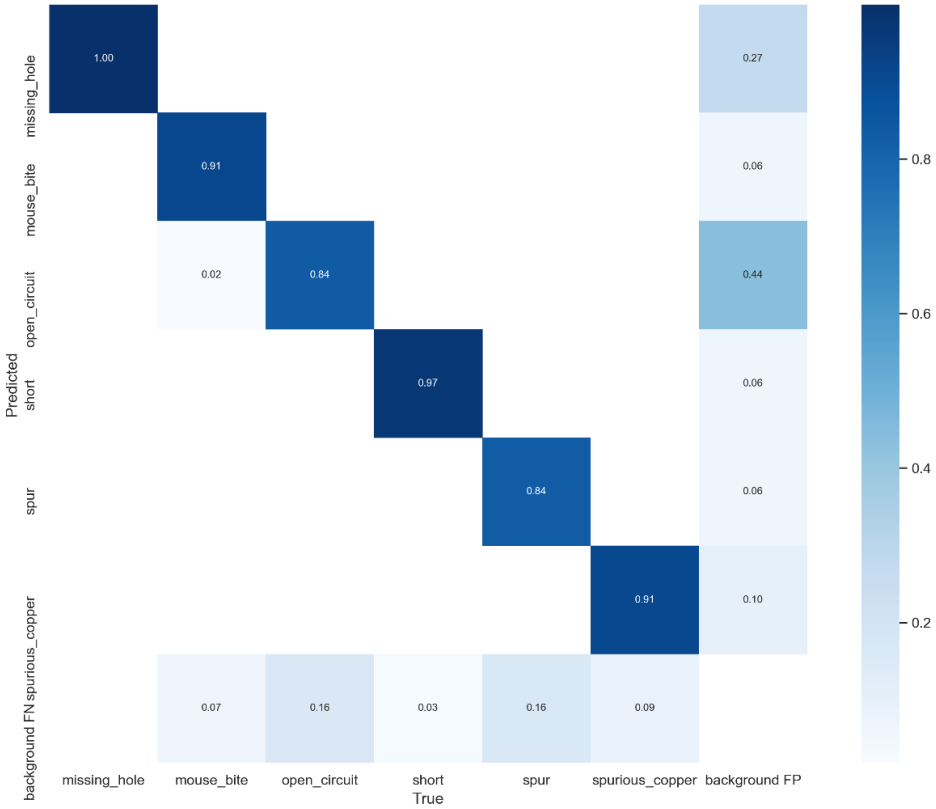

Figure 20 Confusion Matrix - Multistage Fine tuning

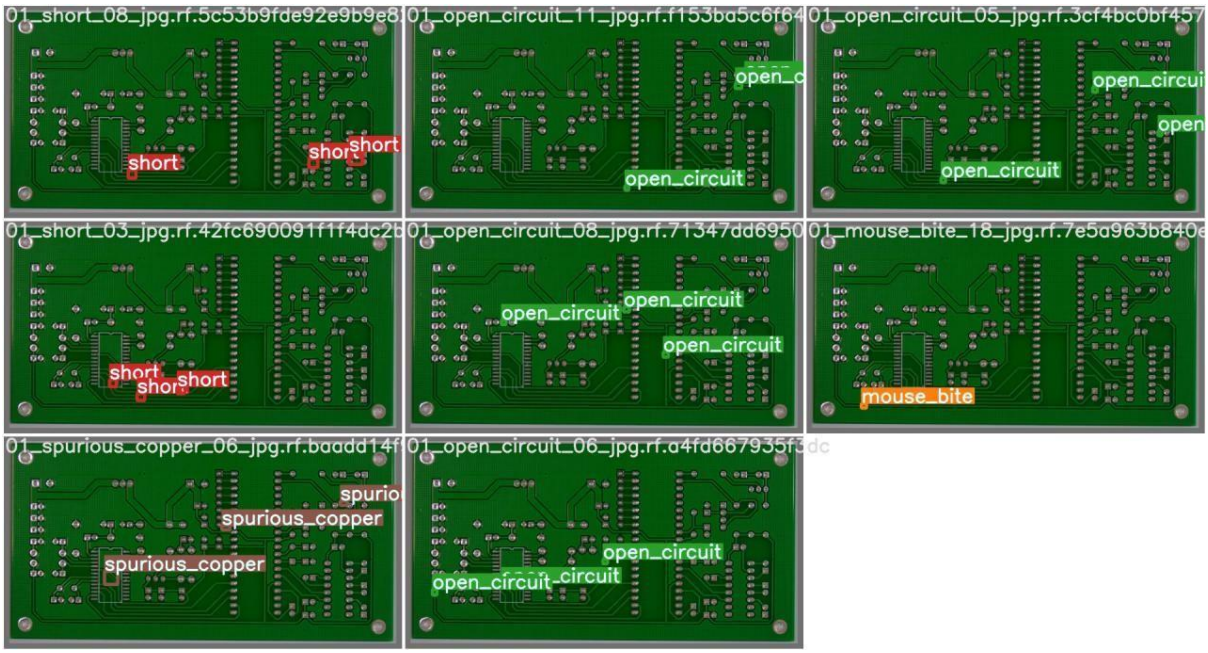

Figure 21 Sample detection results

6. Conclusion & Future Work

This research work proposes to use the state-of-the-art model in object detection to identify defective PCBs at an industrial level. As mentioned earlier, YOLOv7 is the fastest algorithm yet in object detection. They use the least number of trainable parameters compared to other standard object detection models available. The model was built based on a five-stage approach. The five stages included various levels of data augmentation to increase the size of the dataset and also improve the robustness of the model. To further enhance the model, multistage fine tuning was performed. The model was evaluated in each of the five stages to record the progress. It was identified that the model that was trained on the multistage fine tuning with actual data performed the best out of the lot with a mAP of 95.8% overall precision of 97.8% and recall at 91.6%.

The future work on this topic will focus on improving the accuracy and speed of defect detection, as well as making the technology more accessible and user-friendly for a wider range of applications. This could involve the development of new machine learning algorithms and techniques, or the integration of YOLO v7 with other technologies such as computer vision or artificial intelligence. Additionally, there may be efforts to reduce the cost and complexity of implementing YOLO v7 in PCB defect detection systems, to make it more widely accessible to manufacturers and other users.

Acknowledgment

I would like to thank, Mr. Prashanth Nayak, my research supervisor for his tremendous encouragement and clear direction and guidance that enabled me to successfully complete my research thesis and I would like to express my gratitude to my parents for their continuous encouragement and blessings, additionally my for friends who have always encouraged and supported me during the course of my research.

References

Adibhatla, V. A. et al., 2021. Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once. *Mathematical Biosciences and Engineering,* 18(4).

Adibhatla, V. A. et al., 2020. Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks. *Electronics,* 9(9).

Bhattacharya, A. & Cloutier, S., 2022. End-to-end deep learning framework for printed circuit board manufacturing defect classification. *Scientific Reports.*

Ding, R., Dai, L., Li, G. & Liu, H., 2019. TDD-Net: A tiny defect detection network for printed circuit boards. *CAAI Transactions on Intelligence Technology,* 4(2).

Hu, B. & Wang, J., 2020. Detection of PCB Surface Defects with Improved Faster-RCNN and Feature Pyramid Network. *IEEE Access,* Volume 8.

Indera Putera, S. H. & Ibrahim, Z., 2010. Printed circuit board defect detection using mathematical morphology and MATLAB image processing tools. *ICETC 2010 - 2010 2nd International Conference on Education Technology and Computer,* Volume 5.

Jiang, P. et al., 2021. A Review of Yolo Algorithm Developments. *Procedia Computer Science,* Volume 199.

Jiang, W., Zhang, S. & Chen, W., 2022. Multi-target Detection of PCB Defects based on Improved SSD. *International Core Journal of Engineering,* 8(6), pp. 319-325.

Kaviyasri, K. & Binu, D., 2022. Pcb Defect Detection Using Convolutional Auto Encoders. *International Journal of Research and Analytical Reviews,* 9(2), pp. 227-231.

Khalilian, S. et al., 2020. PCB Defect Detection Using Denoising Convolutional Autoencoders. *Iranian Conference on Machine Vision and Image Processing, MVIP,* Volume 2020-January.

Kim, J., Ko, J., Choi, H. & Kim, H., 2021. Printed circuit board defect detection using deep learning via a skip-connected convolutional autoencoder. *Sensors,* 21(15).

Lan, Z., Hong, Y. & Li, Y., 2021. An improved YOLOv3 method for PCB surface defect detection. *Proceedings of 2021 IEEE International Conference on Power Electronics, Computer Applications, ICPECA 2021.*

Li, D., Xu, L., Ran, G. & Guo, Z., 2021. Computer Vision Based Research on PCB Recognition Using SSD Neural Network. *Journal of Physics: Conference Series,* 1815(1).

Li, Y. T., Kuo, P. & Guo, J. I., 2020. Automatic Industry PCB Board DIP Process Defect Detection with Deep Ensemble Method. *IEEE International Symposium on Industrial Electronics,* Volume 2020-June.

Mujeeb, A., Dai, W., Erdt, M. & Sourin, A., 2019. One class based feature learning approach for defect detection using deep autoencoders. *Advanced Engineering Informatics,* Volume 42.

Nepal, U. & Eslamiat, H., 2022. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors,* 22(2).

Ran, G., Lei, X., Li, D. & Guo, Z., 2020. Research on PCB defect detection using deep convolutional nerual network. *Proceedings - 2020 5th International Conference on Mechanical, Control and Computer Engineering, ICMCCE 2020.*

Shi, W., Lu, Z., Wu, W. & Liu, H., 2020. Single-shot detector with enriched semantics for PCB tiny defect detection. *The Journal of Engineering,* 2020(13).

Tang, S., He, F., Huang, X. & Yang, J., 2019. Online Pcb Defect Detector on a New PCB Defect Dataset. *ArXiv,* Volume abs/1902.06197.

Tsai, D. M. & Jen, P. H., 2021. Autoencoder-based anomaly detection for surface defect inspection. *Advanced Engineering Informatics,* Volume 48.

Wang, Z. et al., 2021. Fast personal protective equipment detection for real construction sites using deep learning approaches. *Sensors*, 21(10).

Wu, X., Ge, Y., Zhang, Q. & Zhang, D., 2021. PCB Defect Detection Using Deep Learning Methods. *Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2021*.

Xie, H., Li, Y., Li, X. & He, L., 2021. A Method for Surface Defect Detection of Printed Circuit Board Based on Improved YOLOv4. *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, ICBAIE 2021*.

Xin, H., Chen, Z. & Wang, B., 2021. PCB Electronic Component Defect Detection Method based on Improved YOLOv4 Algorithm. *Journal of Physics: Conference Series*, 1827(1).

Zhang, H., Jiang, L. & Li, C., 2021. CS-ResNet: Cost-sensitive residual convolutional neural network for PCB cosmetic defect detection. *Expert Systems with Applications*, Volume 185.