To,
IITD-AIA Foundation of Smart Manufacturing

Subject: **Weekly Progress Report for Week 2**

Dear sir,

Following is the required progress report to the best of my knowledge considering relevant topics and project objectives covered.

What happened last week - W1 :

- OpenCV advance implementation
- Fault analysis
- Classification - identifying different types of defects in PCBs
- Binary image Morphology and Image subtraction
- Sobel Edge Detection and Canny Edge Detection

What's happening this week - W2:

- Studying Dataset
- Experimentation with **group00041** dataset
- Exploratory Data Analysis (Primitive draft)
- Dataset size reduction for faster computation
- Working with Pillow (or PIL) library for image manipulation
- Working on the algorithm and implementation sequentially
- Extracting Defects using OpenCV
- Defect extraction by cropping the particular segment mentioned in notation files

Weekly Progress:

June 06:

(Monday) After getting and exploring the dataset, my main objective was to understand, if the given dataset is of optimum size or needs to be optimized in accordance with the optimum speed requirement and quality too.

- Learnt to reduce the size of data image into something that will be much more helpful and a lot more efficient.
- Doing so will not only help us with the processing speed and parameterization but also, make our work much faster.

Implementations:

- Implemented a size reduction algorithm on every single image.

Using the PIL library in order to reduce images without compromising on quality.

June 07:
(Tuesday) Continuing the previous day's work. Reducing the size of every image from 640x640 to 128x128. It will be better if we reduce the dimension of them so that it will be easy for training the data, but to make sure the quality of the image didn't get affected, we will compress the image using the Anti-Aliasing technique.
After resizing the image to 128x128, we save them into a different folder. I saved many images of equally mixed good and bad PCBs. So, that folder is going to be your Training Data set.

June 08:
(Wednesday) After processing the images, our main motive is to extract defects out of them. In order to do that, I applied defect extraction algorithms that can take up all the defects from a few images.

- First, we need to build the defect extraction pipeline.
- Then I am using an Open Source defect detecting library known as tetryon_ai
- Hence, extracted defects in the differenced images using contours, and created the 2 files we need to use in the labelling application we create next.

The extract_contours_from_image function uses contours to extract the highlighted parts of our differenced images.

June 09:
(Thursday) After extracting the features and defects, I realised that it is not possible to just extract defects from each image, in the way I was doing. In order to change my approach, I am planning to now crop every defect at the individual level and then operate a classification and detection algorithm.

- I applied defect cropping algorithms that can take up all the defects from a few images.
- Cropping and then teaching machines can be a much more efficient way.

June 10:
(Friday) I applied defect cropping on an entire group of images. About 1642 images have been extracted.

- I learnt how to create my own Image Cropper from scratch as well as from the libraries.
- Used PIL or Pillow library for making the pipeline run fast and efficiently.


Open circuit defect

Short circuit defect

Mousebite defect

Spur defect

Copper defect

Pin-hole defect

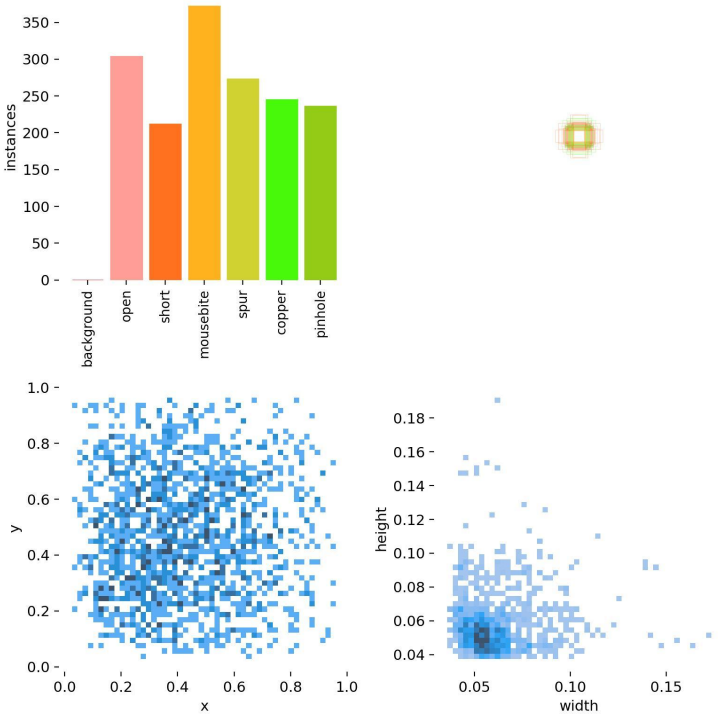
June 11:

(Saturday) As I thought I was prepared with the extracted dataset yesterday, and I'll just go on and train a random model today out of it, just to check the working of my algorithm. Turns out, the approach is best suitable for classification problems and NOT for segmentation.

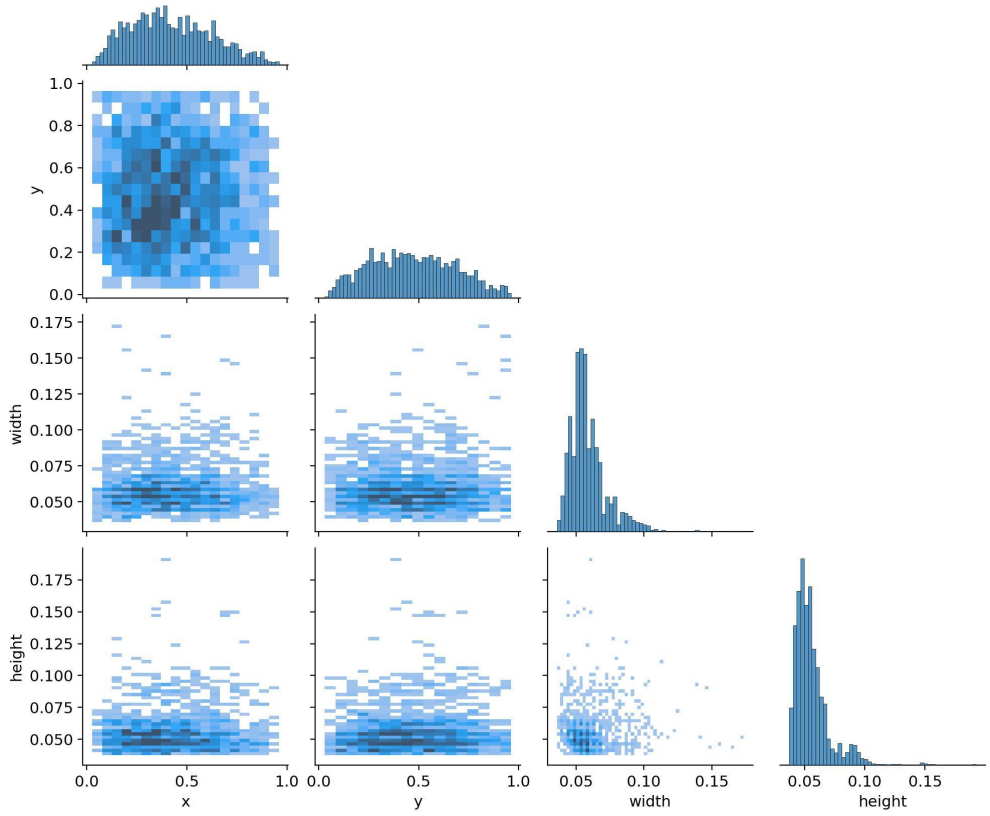
- Turns out, we need to segment each image, not just for classification but to determine the exact location on where that particular defect is observed.
- Traditional ML algorithms, that I am acquainted with, will not solve the problem.

Implementations:

- Switched the approach. I am starting with YOLOv5 architecture in order to solve the problems.
- However, my computer is not capable enough, on the hardware side, that it can undergo such a complicated and heavy workload.



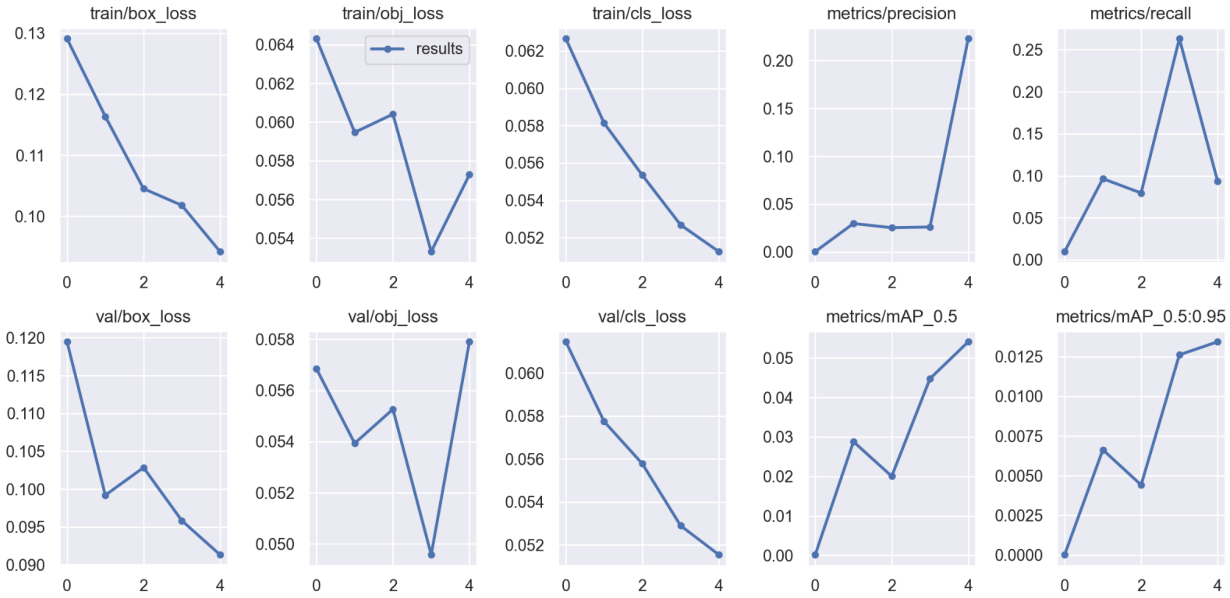
Labels and their density



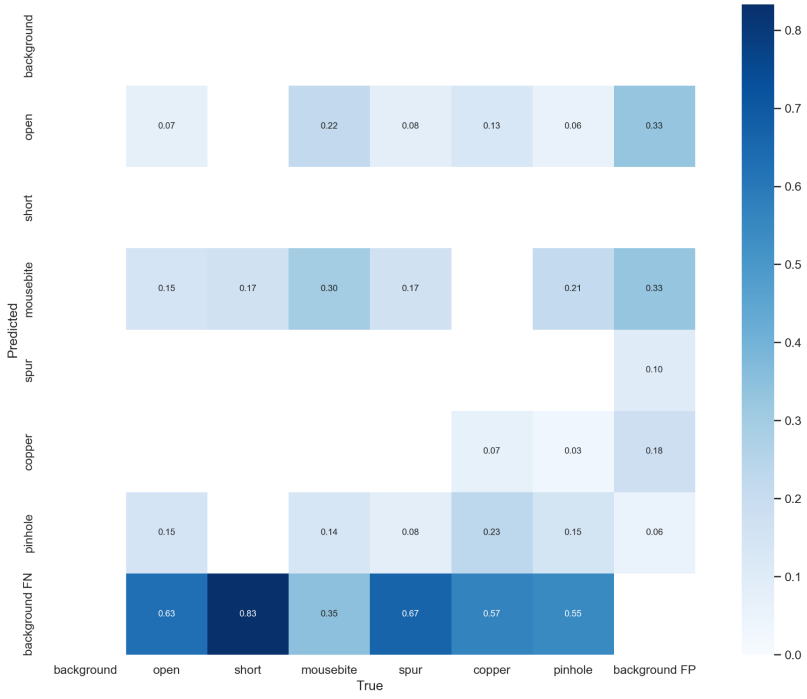
Data-correlation graph

June 12:

(Sunday) Continued my experimentation with primitive neural networks that can undergo minute detections. However, the model performed terribly, as it was trained upon only 5 epochs (recommended 300~1200) due to my computation constraint, I realised that this approach can turn out to be perfect if invested a little more time.



The above figure represents the **Experimental results**



Confusion Matrix for 5 epochs (ill-trained dataset)