Table of Contents

Corresponding author: Keivalya Pandya (keivalyapandya2001@gmail.com)

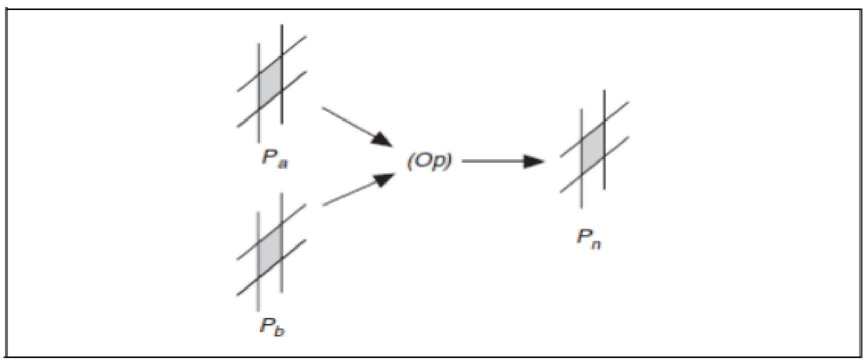
To IITD-AIA-FSM 2022,

Method

Logical operators for image

As we are provided with the template image (perfectly good PCB) and the test image (image with the faults corresponding to the template image), an initial approach towards detecting defects could be to subtract the test image pixels from the template image pixels. Pixel-by-pixel transformation is a procedure between pictures that applies logic or arithmetic. It creates a picture where each pixel's value is derived from pixels in other images that have the same coordinates. [1]

If *A and B* are the images with a resolution *XY*, and *Op* is the operator, then the image *N* resulting from the combination of *A* and *B* through the operator *Op* is such that each pixel *P* of the resulting image *N* is assigned the value *pn = (pa)(Op)(pb)*; where *pa* is the value of pixel *P* in image *A*, and *pb* is the value of pixel *P* in image *B*.



Object Detection

Classification can be understood in 3 distinguished steps:

1. Classification, in which we extract a certain type of information, with a pre-defined category or instance ID to describe the entire image.
2. Detection, in which we detect bounding boxes within which the pixels of the classified objects are observed. Compared with classification, detection gives the understanding of the picture's foreground and background. The output of the detection model is a list, and each item of the list used a data group to give the category and position of the detected target (commonly used coordinate representation of rectangular detection box)
3. Segmentation is a pixel-level description of an image, which gives meaning to each pixel category (instance) and is suitable for scenes requiring high understanding.

We are going to use a detection-based model (instead of segmentation) in order to compensate for classification’s lack of localization information, as well as segmentation’s pixel lever complexity.

One-Stage and Two-Stage

The one-stage network is represented by the YOLO series network, while the two-stage network is represented by faster-RCNN. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection.

Deep Learning Model

In this section, we first introduce the overall structure of the network. Then we introduce in detail the details of our modified classifier and evaluation metrics of the dataset.

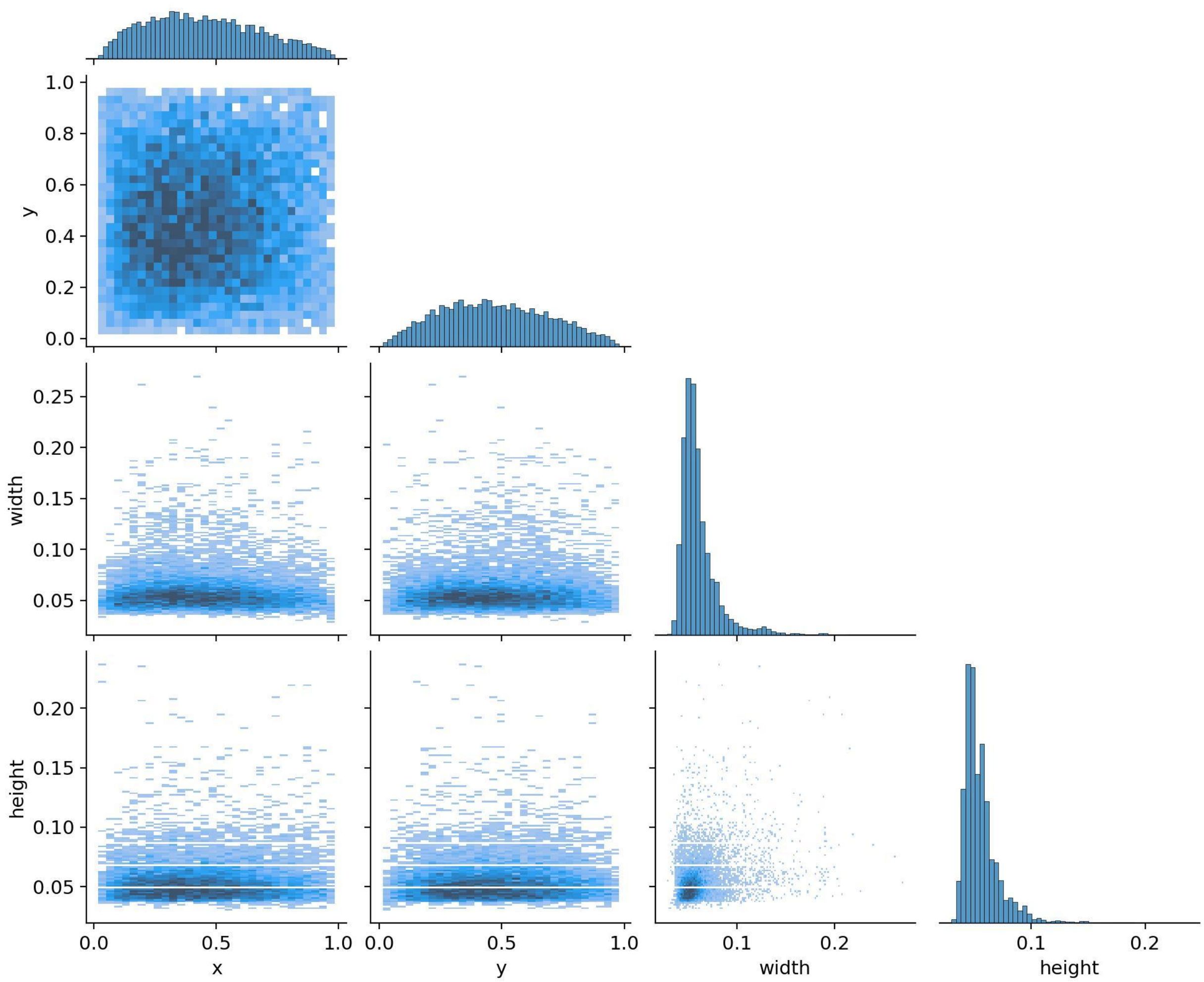
YOLOv5

YOLOv5 has inherited the advantages of YOLOv4, namely adding SPP-NET, modifying the SOTA method, and putting forward new data enhancement methods, such as Mosaic training, Selfadversary training (SAT), and multi-channel feature replacing FPN fusion with PANet.

Dataset Introduction

Our data set is 1500 images, each of which contains annotated PCB defects including positions of the 6 most common types of PCB Defects (open, short, mouse bite, spur, pinhole and spurious copper). The dataset has been adopted for research purposes from *https://github.com/tangsanli5201/DeepPCB*.

The original dimensions of each image are around 16k × 16k pixels. They are then divided into several 640 × 640 pixels sub-images using a cropping process, then aligned using template matching methods.



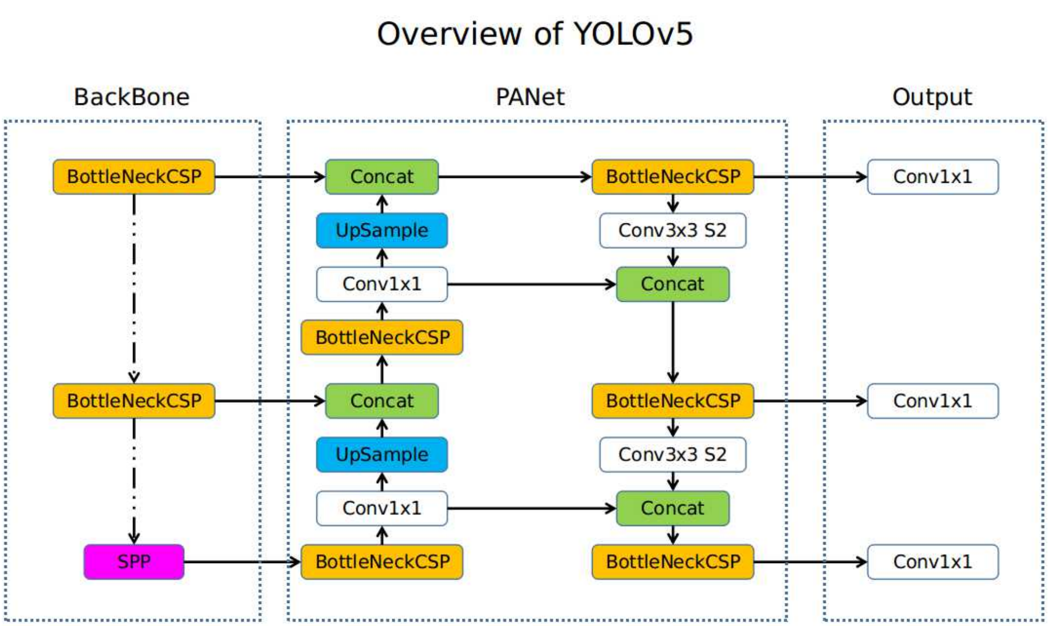
System and Hyperparameter details

Our operating system is ubuntu20.04, using the Pytorch framework, and training and testing on an i5 8.0 GB RAM, 4GB Graphics card and CUDA CPU.

Parameter	Value
Learning rate	0.01
Learning rate decay	0.999
Learning rate decay step	1
Weight rate decay	5e-4
Momentum	0.937
Batch size	16
Number of iterations	100

Model Structure

```
                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2,
'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1,
False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2,
'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1,
False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1,
False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1,
False]
 24      [17, 20, 23]  1    229245  models.yolo.Detect                      [80, [[10, 13, 16,
30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
                 Model summary: 270 layers, 7235389 parameters, 7235389 gradients
```



source: https://github.com/ultralytics/yolov5/issues/280

The network structure of yoloV5 is divided into three parts, backbone, neck, and output. In the backbone, the input image with 640×640×3 resolution goes through the Focus structure. Using the slicing operation, it first becomes a 320×320×12 feature map, and then after a convolution operation of 32 convolution kernels, it finally becomes a 320×320×32 feature map. The CBL module is a basic convolution module. A CBL module represents Conv2D + BatchNormal + LeakyRELU. [2]
The BottleneckCSP module mainly performs feature extraction on the feature map, extracting rich information from the image. Compared with other large-scale convolutional neural networks, the BottleneckCSP structure can reduce gradient information duplication in convolutional neural networks' optimization process. Its parameter quantity occupies most of the parameter quantity of the entire network. By adjusting the width and depth of the BottleneckCSP module, four models with different parameters can be obtained, namely YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The SPP module mainly increases the receptive field of the network and acquires features of different scales.
YOLOv5 also adds a bottom-up feature pyramid structure based on the FPN structure. With this combination operation, the FPN layer conveys strong semantic features from top to bottom, and the feature pyramid conveys robust positioning features from the bottom up. Combine feature aggregation from different feature layers to improve the network's ability to detect different scales' targets. At the end of the figure, output the classification results and object coordinates.

Training the model

Classifier Modifications

For the COCO dataset, there are 80 object categories, and the dimension of the output tensor is 3 × (5 + 80) = 255, where 3 represents the three template boxes for each grid prediction. And 5 represents each prediction box's coordinates (x, y, w, h) and confidence (confidence, c). [3]
We have seven types of objects in the DeepPCB dataset. Whose annotations are of the form *(x1,y1), (x2,y2), (a)*; where *x1* and *y1* are the left top coordinates and *x2* and *y2* are right bottom coordinates and *a* is the annotation corresponding to that particular bounding box ranging from *0 to 6 inclusive*.
In order to convert original annotations to the annotations that YOLOv5 supports, we use the following conversion algorithm (or python function):

```
def convert(box):
    dw = 1./640
    dh = 1./640
    x = (box[0] + box[1])/2.0
    y = (box[2] + box[3])/2.0
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x*dw
    w = w*dw
```

```
    y = y*dh
    h = h*dh
    return x,y,w,h
```

Similarly, annotations can be converted into string using the following:

```
def annotate(argument):
    switcher = {
        0: "background ",
        1: "open",
        2: "short",
        3: "mousebite",
        4: "spur",
        5: "copper",
        6: "pinHole"
    }
    return switcher.get(argument, "nothing")
```

This conversion is followed by YOLOv5 configuration file that is, `/dataset.yaml`.

Creating `dataset.yaml` file

```
path: ../Dataset/PCBData
train: images  # train images (relative to 'path')
val: images  # val images (relative to 'path')

# Classes
nc: 7  # number of classes
names: [ 'background', 'open', 'short', 'mousebite', 'spur',
'copper', 'pinhole']
```

The repository provides 4 pre-defined models to choose from. Yolov5-small, Yolov5-medium, Yolov5-large, Yolov5-extraLarge. For this detection project, YOLOv5s (-small) have been trained on 500 epochs, with W&B logging enabled. [4]

Testing the model

Matrices for evaluation

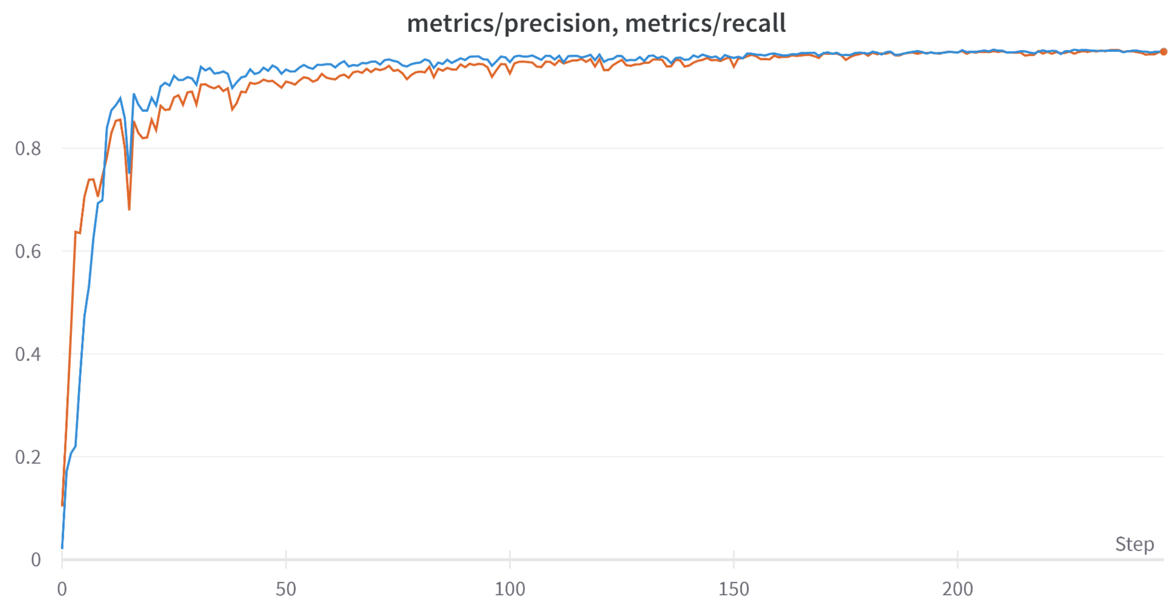
The average precision mean mAP, precision and recall are used to describe the experimental outcomes. To begin, we determine the precision rate and recall rate for each category of an object using the following formula.

$$Precision = \frac{TP}{TP + FP}$$

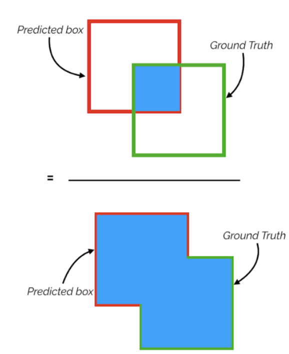
$$Recall = \frac{TP}{TP + FN}$$

$$AP = \int_0^1 precision(recall)$$

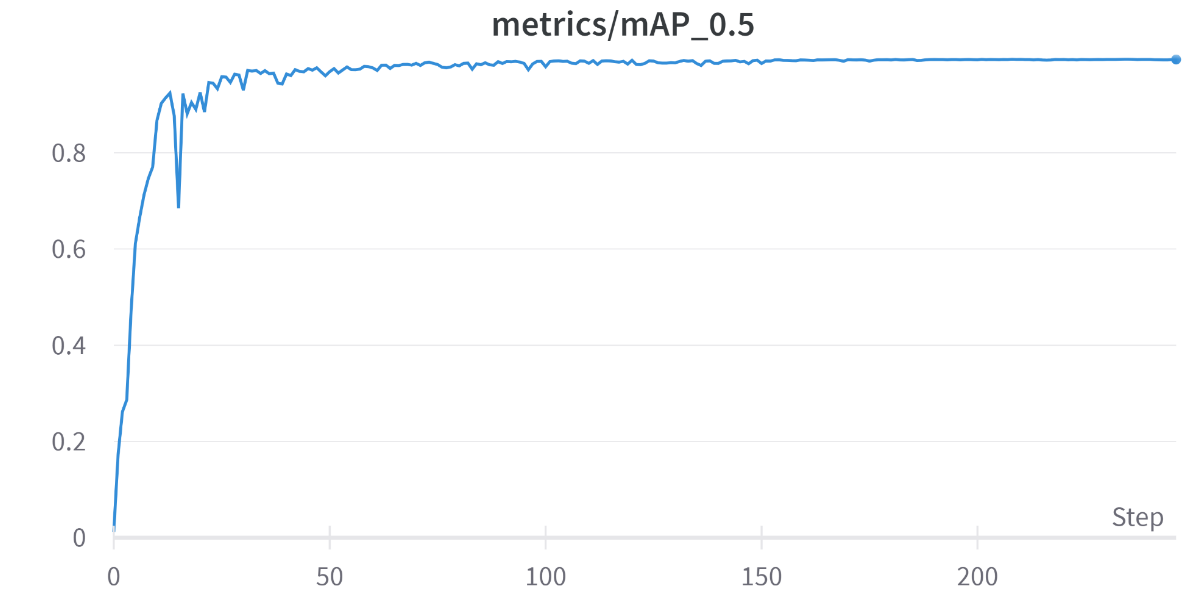
Where TP stands for true positive and FP for false positive; FN stands for false-negative; mAP is the average of all categories' APs, while AP is the average accuracy for a particular category.



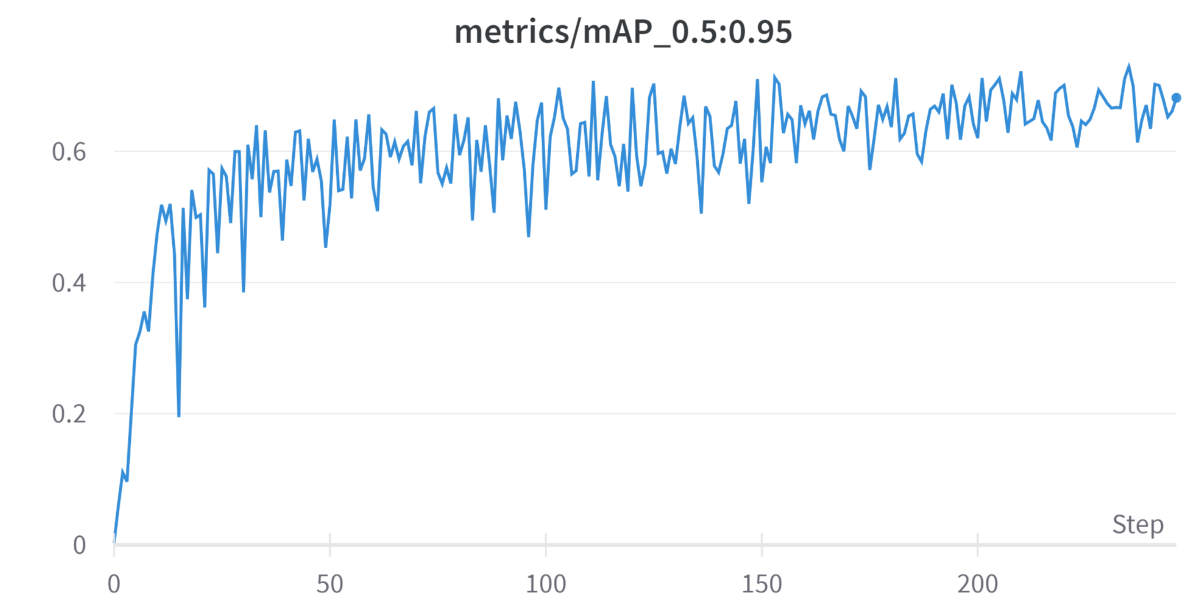
We evaluate the mAP averaged for IoU ∈ [0.5 : 0.05 : 0.95].For each bounding box, we measure the overlap between the predicted bounding box and the ground truth bounding box. This is measured by IoU (intersection over union). For object detection tasks, we calculate Precision and Recall using the IoU value for a given IoU threshold.

$Intersection\ of\ Union\ (IoU) = \frac{Area\ of\ Overlap}{Area\ of\ Union}$	

For instance, if the IoU threshold is 0.5, and the IoU value for a prediction is 0.7, then we classify the prediction as True Positive (TF). On the other hand, if IoU is 0.3, we classify it as a False Positive (FP).
In our detection model, we assume that the confidence score threshold is 0.5 and the IoU threshold is also 0.5. So we calculate the AP at the IoU threshold of 0.5.



Graphical representation of mAP:0.5 across various epochs/steps
It can be found that in the initial training phase, as the training time increases, the model increases rapidly. We can even see a significant downfall and a steep shortcoming in precision as well as recall at the 15th epoch.

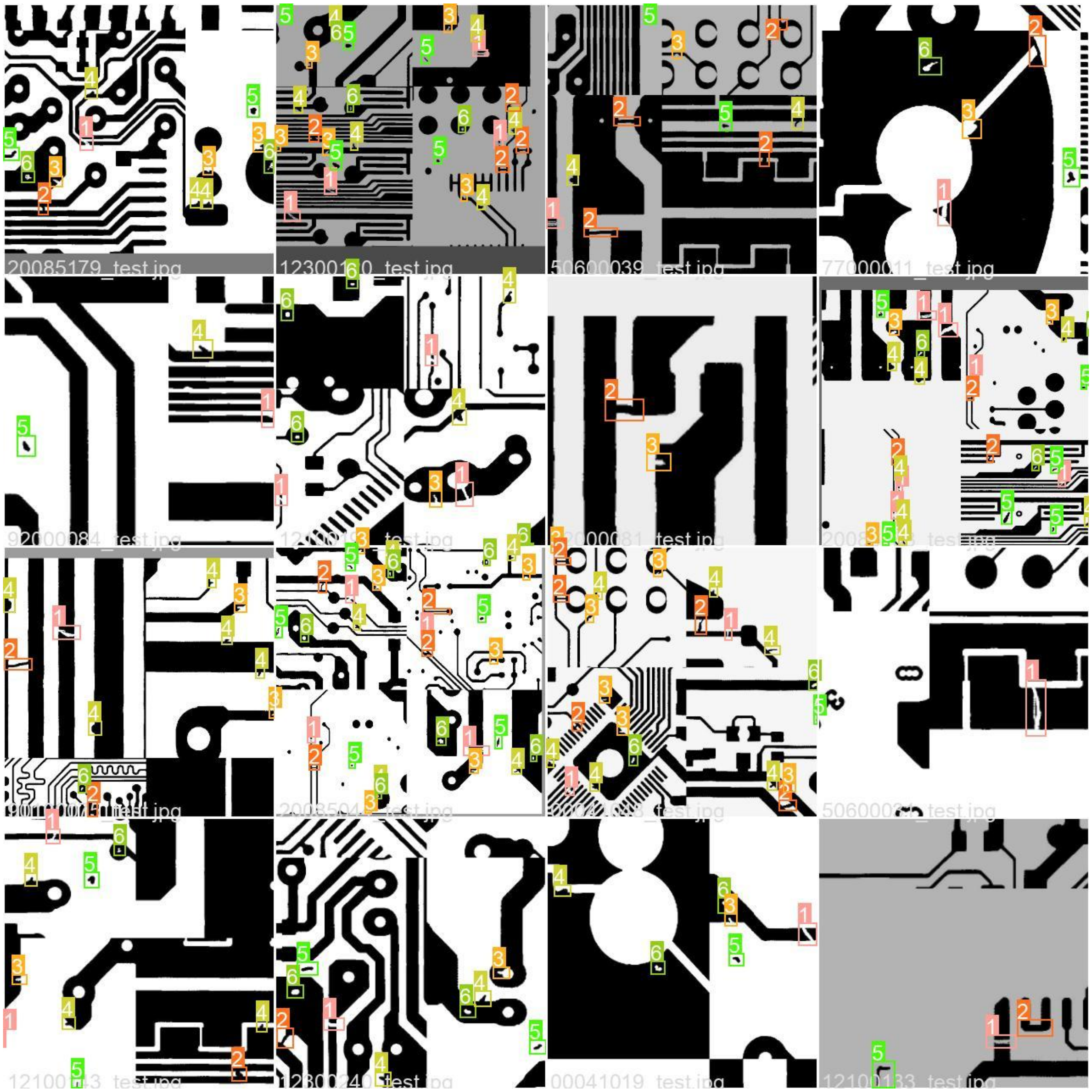


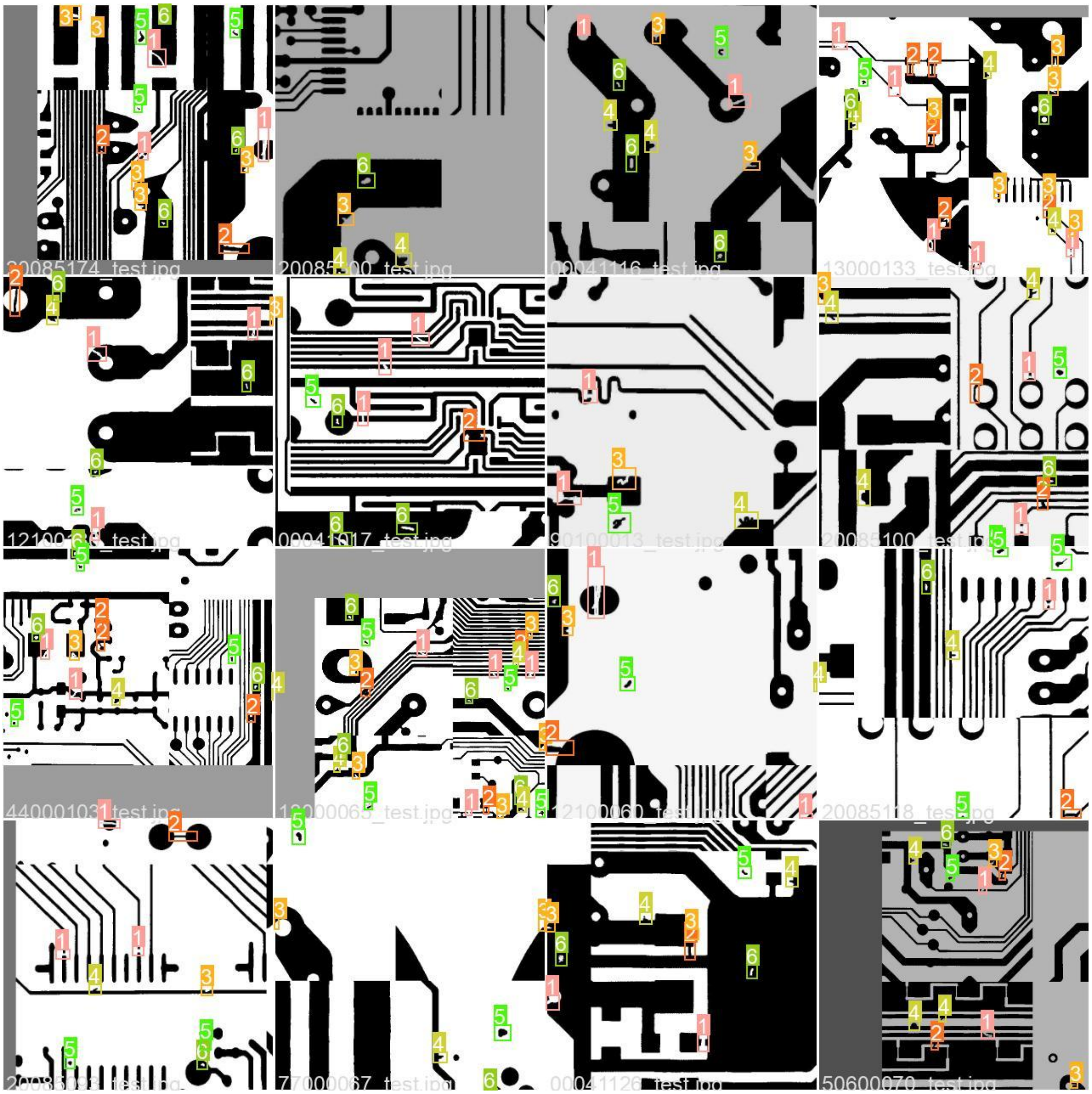
Graphical representation of mAP_0.5:0.95 across all the epochs

Here it is observed that initially, mAP0.5:0.95 is extremely inconsistent, however as the epochs increment, we can see narrower variations in the metrics. This represents the platonic convergence of the criterion and increase in its stability.

Results

After training the model for 24.058 hours and early stopping at 336 epochs due to receiving no significant upgrades over the previous 100 epochs, results can be seen as follows.


20085179_test.jpg
92000084_test.jpg
00041019_test.jpg



References

1. Pal, Ajay & Chauhan, Singh & Bhardwaj, Sharat. (2011). Detection of Bare PCB Defects by Image Subtraction Method using Machine Vision. Proceedings of the World Congress on Engineering 2011, WCE 2011. 2.
2. F. Zhou, H. Zhao and Z. Nie, "Safety Helmet Detection Based on YOLOv5," 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), 2021, pp. 6-11, doi: 10.1109/ICPECA51329.2021.9362711.
3. Lin, TY. *et al.* (2014). Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48
4. https://wandb.ai/cayush/yoloV5/reports/How-are-Your-YOLOv5-Models-Doing---VmlldzoyNjM3MTY