VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering
FALL Semester 2022-2023

J Component Report

Title: Detect Moon Pot-Hole

Course Title	Digital Image Processing
Course Code	SWE1010
Slot	A1

Team Members

Kamna V 20MIA1053
Aman Kumar 20MIA1144
Palak Deep Kaur 20MIS1162

Faculty: Anushiya Rachel

Date: 17-11-2022

Sign:

ACKNOWLEDGEMENT

We would like to thanks Dr. Anushiya Rachel G, our professor-in-charge for their support and guidance in completing our project on the topic Detect Moon Pot-Hole. It was a great learning experience.

I would like to take this opportunity to express my gratitude to all of my group members. The project would not have been successful without their cooperation and inputs.

INDEX

Abstract

YOLO v5 is a family of compound scaled object detection models trained on the COCO dataset, and includes simple functionality for Test Time Augmentation (TTA), model assembling, hyperparameter evolution, and export to ONNX, CoreML and TFLite. YOLO an acronym for 'You only look once, is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself. YOLO is one of the most famous object detection algorithms due to its speed and accuracy. For this Challenging experiment we have used YOLO V5 for the circular object (moon surface) detection using anchor boxes. To make the image clear we use image processing filters like 2D, image sharpening, and noise filters. Noises frequently corrupt pictures. Noise can be acquired during picture catch, transmission, and so on. Noise expulsion is a significant assignment in picture handling. Overall, the consequences of the commotion expulsion impact the nature of the image-handling methods.

I. Problem Statement

Here, we are detecting potholes and extracting the information about potholes to study the moon surface and capture the changes
This helps the satellites/rover to land properly on moon surface without any discrepancy.

II. Requirements

- Image processing filters like 2D, image sharpening and noise filters. We Will use Python and its libraries like Matplotlib, NumPy, OpenCV etc. for this project.
- Jupyter Notebook IDE environment.
- For Computer Vision we will explore algorithms like YOLO

III. Introduction

Yolo architecture is more like FCNN (fully convolutional neural network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction. This the architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes. Note that bounding box is more likely to be larger than the grid itself. Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in an image. We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. First, YOLO is extremely fast. Since we frame detection as a regression problem, we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. As YOLO v5 is a single-stage object detector, it has three important parts like any other single-stage object detector.

1. Model Backbone
2. Model Neck
3. Model Head

1. **Model Backbone** is mainly used to extract important features from the given input image. In YOLO v5 the CSP—Cross Stage Partial Networks are used as a backbone to extract rich in informative features from an input image.

2. **Model Neck** is mainly used to generate feature pyramids. Feature pyramids help models to generalized well on object scaling. It helps to identify the same object with different sizes and scales. Feature pyramids are very useful and help models to perform well on unseen data. There are other models that use different types of feature pyramid techniques like FPN, BiFPN, PANet, etc. In YOLO v5 PANet is used for as neck to get feature pyramids. Understanding Feature Pyramid Networks for object detection (FPN) The model.

3. **Head** is mainly used to perform the final detection part. It applied anchor boxes on features and generates final output vectors with class probabilities, objectless scores, and bounding boxes. In YOLO v5 model head is the same as the previous YOLO V3 and V4 versions.

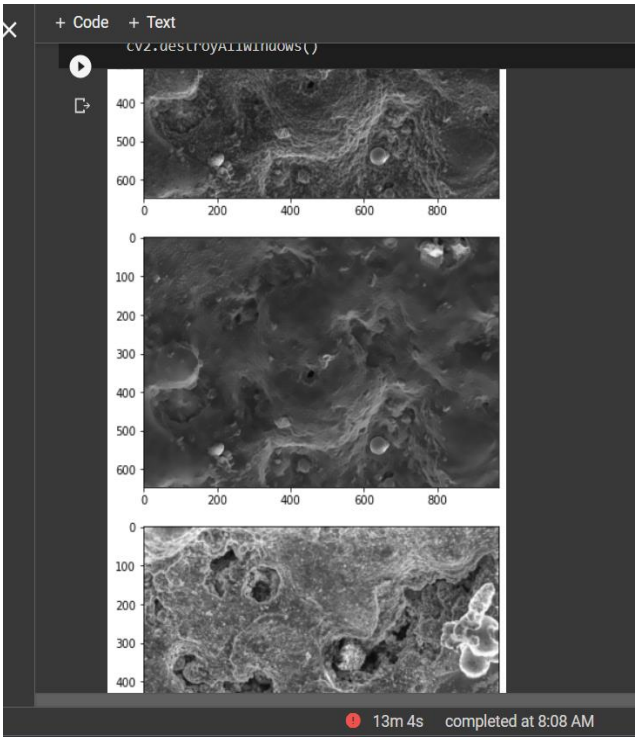
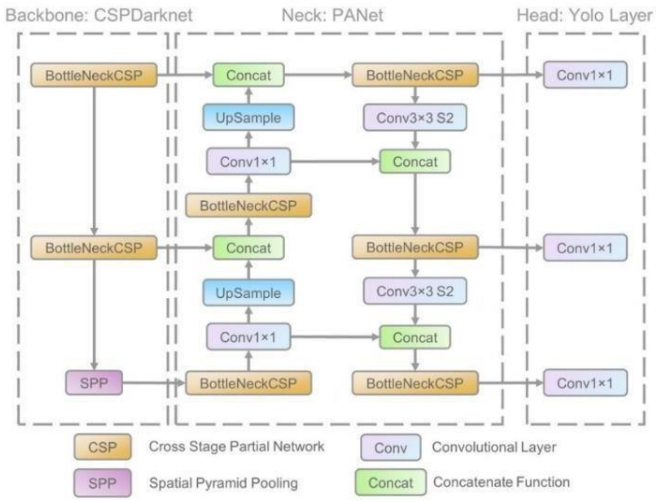


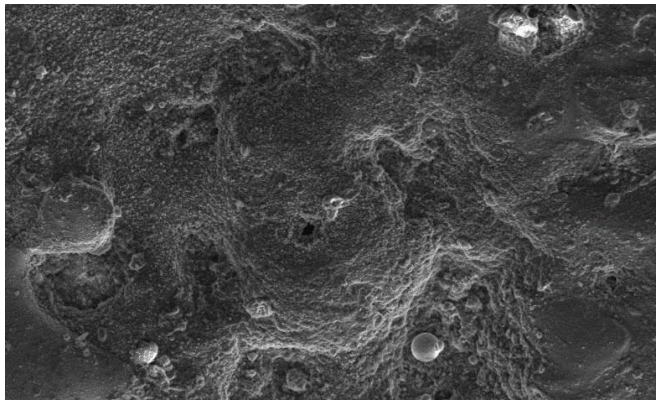
Image Noise Reduction and Filtering Techniques

A few techniques for noise expulsion are deeply grounded in a variety of picture handling. The idea of the commotion evacuation issue relies upon the kind of noise undermining the picture. In the field of picture sound decrease, a few straight and nonlinear sifting techniques have been proposed. Direct channels can't successfully kill motivation noise as they tend to obscure the edges of a picture. Then again nonlinear channels are appropriate for managing drive commotion. A few nonlinear channels in light of old style and fluffy methods have arisen in a couple of years. For instance, most old-style channels eliminate at the same time obscure the edges, while fluffy channels can join edge conservation and smooth. Contrasted with other nonlinear strategies, fluffy channels can address information in a fathomable manner. In this paper, we present outcomes for various separating strategies and we think about the outcomes for these procedures.

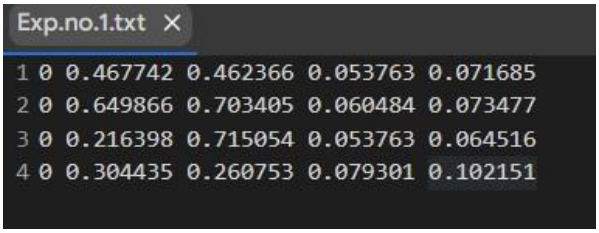


IV. Dataset

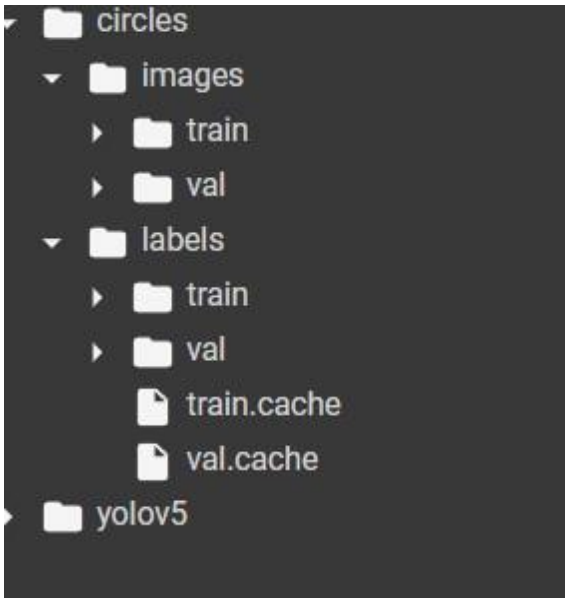
The Circular Ellipse dataset has 29 images of the moon's surface without annotations –



The images have then been split to 20 as train and the remaining 9 as validation for testing purpose. As an import phase of object detection, we have manually assigned the annotations using make sense ai and further generated the object coordinates for training our model.



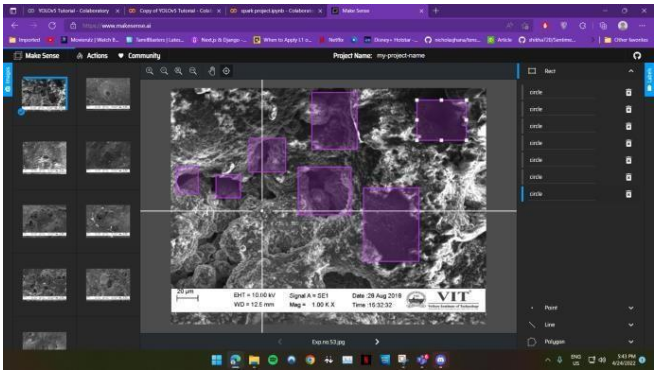
The associated coordinates of the objects in the first image. Four coordinates as there are 4 objects in the above image



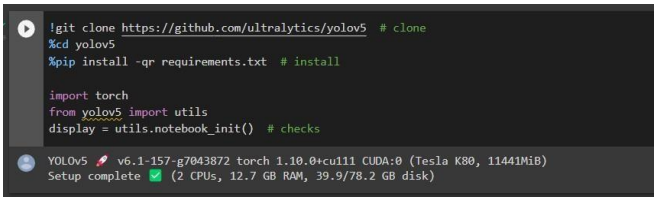
The overall data structure which includes the train validation split, labels which has the coordinates of the bounding box for both train and validation as described in the earlier images.

V. Methodology and Process

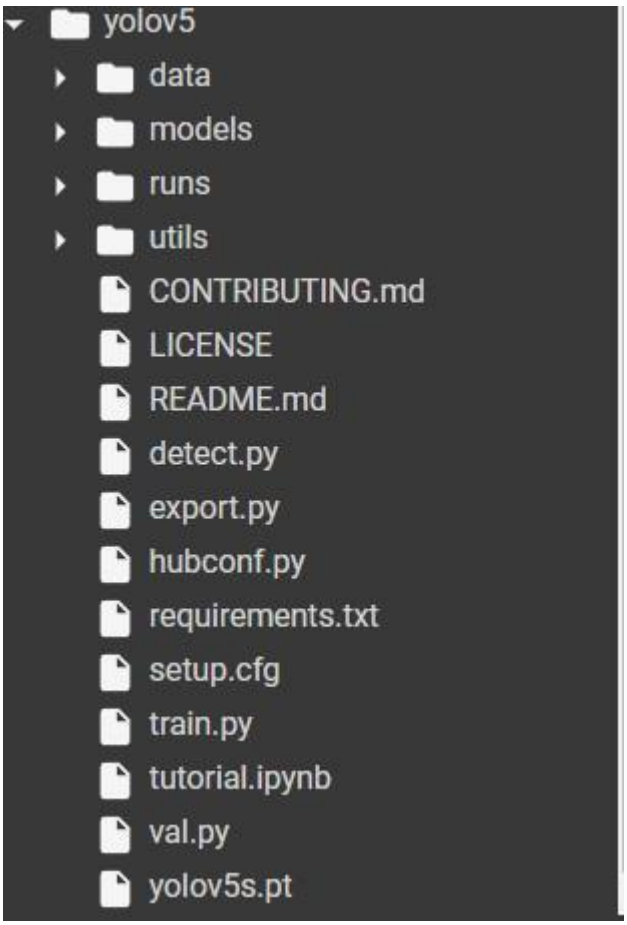
We Start with our manual annotation process using makesense.ai where we annotate each and every image's objects and export the annotations.



For this experiment the YOLO V5 predefined / pretrained is extracted and initialized.



After the execution of the previous cell, the runtime creates a library as shown in the figure below (google collab) which is nothing but the prerequisites for running our model .



As we have already Split the dataset into train and validation, we are specifying the path of train and validation located in our collab directory. The executable text which is being shown in the below figure is inside the yolov5 directory. We are modifying the number of classes and the name of the class according to our requirements which in this case is only 1 class (circular class) labelled/named as 'circle'.

```
coco128.yaml
1 train: ../circles/images/train
2 val: ../circles/images/val
3
4
5 # Classes
6 nc: 1  # number of classes
7 names: ['circle']
8
```

The final step is training our model using YOLO V5 for a total of 150 epochs without specifying any call-backs or arguments. And parallelly checking the validation using our validation

images which were defined in the beginning. The training of the images happens in a batch.

```
# Training for 150 epochs
!python train.py --img 640 --batch 16 --epochs 150 --data coco128.yaml --weights yolov5s.pt --cache
```

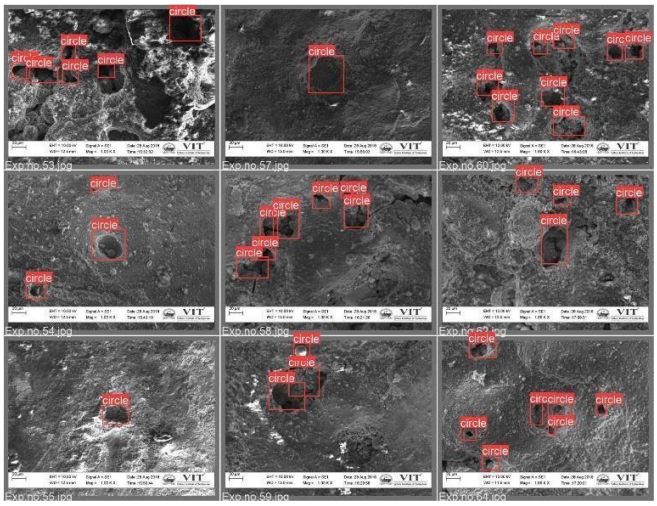
VI. Observations and Results

To evaluate object detection models like R-CNN and YOLO, the mean average precision (mAP) is used. The mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections. So, for our model the mAP value is 0.555 which is 55%. Along with the loss recorded was 0.171 which is really low for a object detection model. The below image further shows all the other metrics relevant.

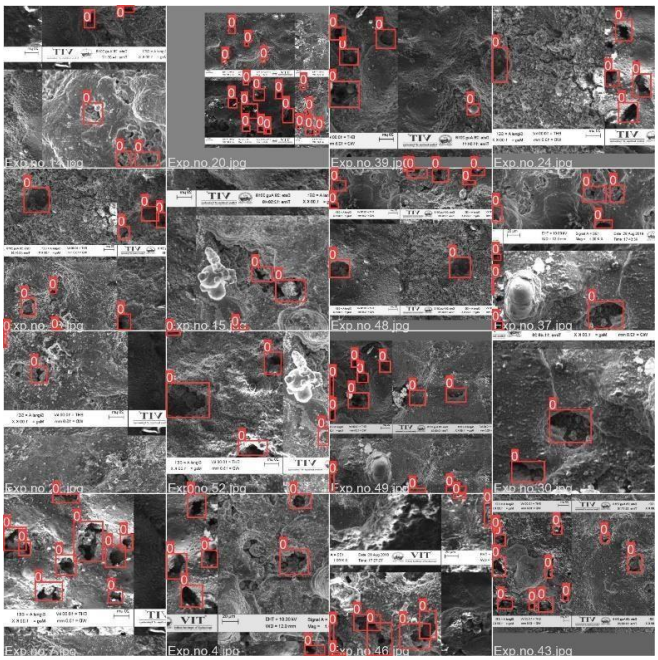
```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
   149/149     3.12G    0.0435   0.05616         0        41       640: 100% 2/2 [00:02<00:00,  1.05s/it]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00,  4.29it/s]
                   all          9         41      0.759      0.536      0.541      0.158

150 epochs completed in 0.121 hours.
Optimizer stripped from runs/train/exp5/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp5/weights/best.pt, 14.4MB

Validating runs/train/exp5/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00,  2.68it/s]
                   all          9         41      0.527      0.683      0.555      0.171
Results saved to runs/train/exp5
```

The generated annotations of the images –

Batch of 9

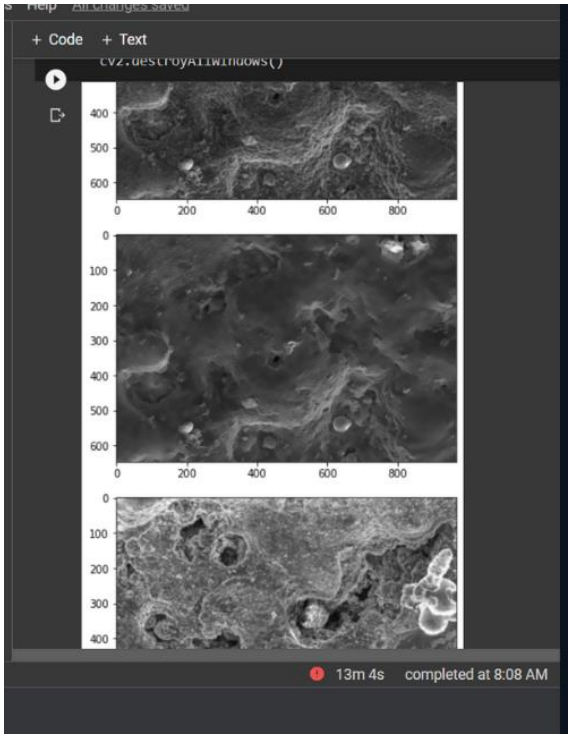


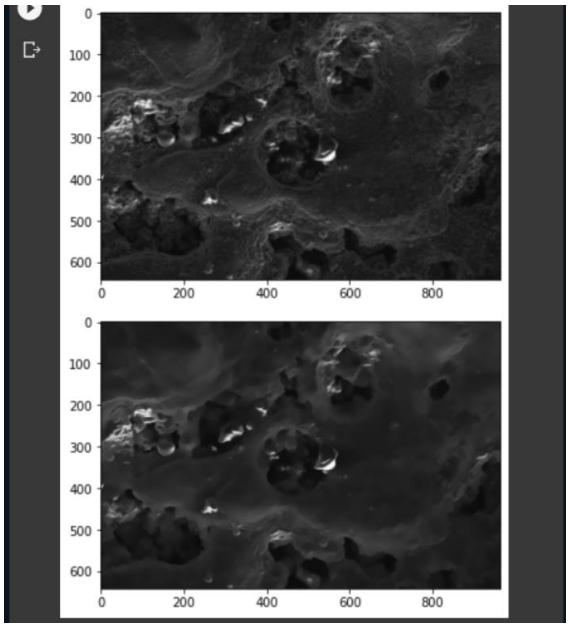
Batch of 16



For all the above images the red bounding box depicts the detected circular object and does it with high precision.

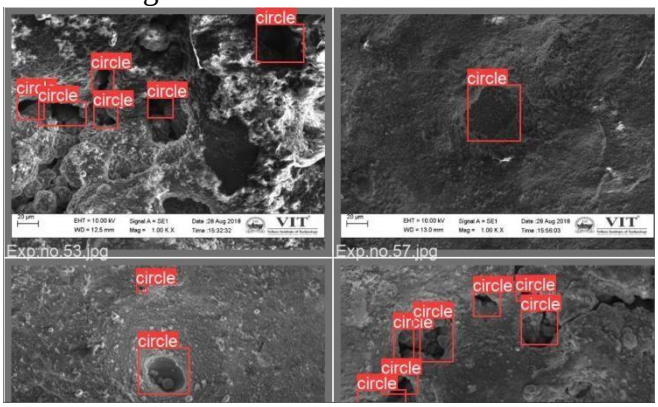
The generated images after applying image processing (noise filters) –



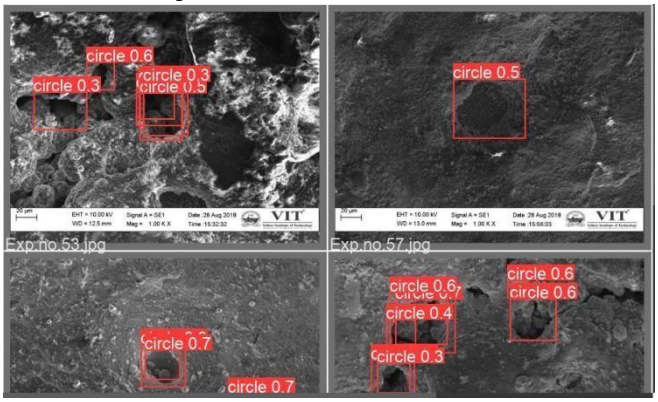


VII. Conclusion

Actual images with annotations –



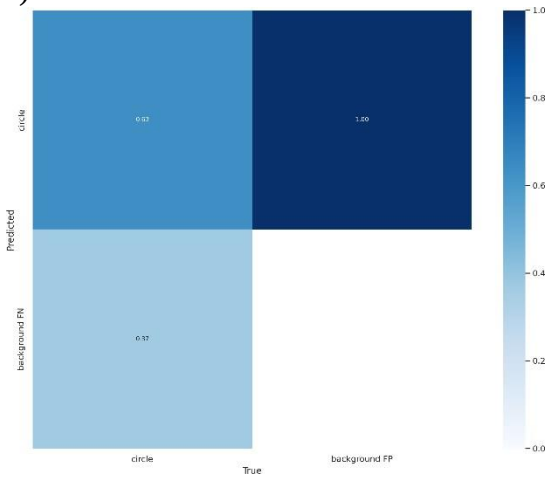
Predicted objects–



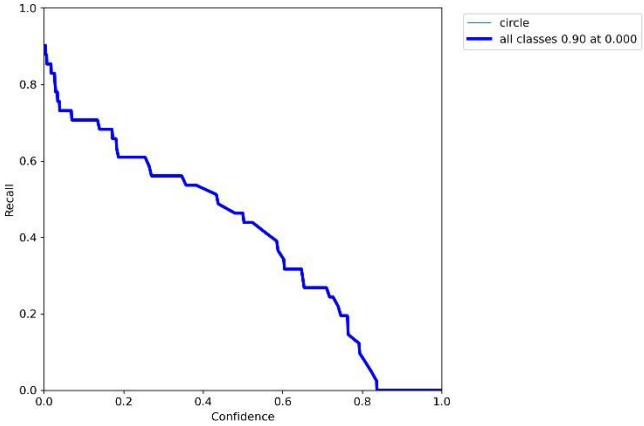
As the predicted boxes are more of less overlapping the annotated boxes , The IOU is high which says the model is well trained for detecting the desired objects.

Statistics of the Model –

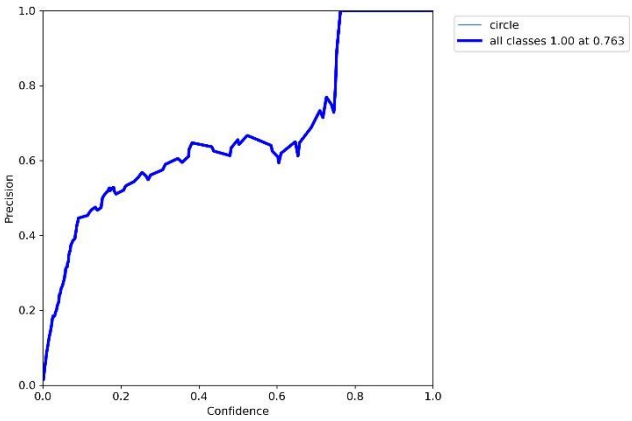
a) Confusion matrix -



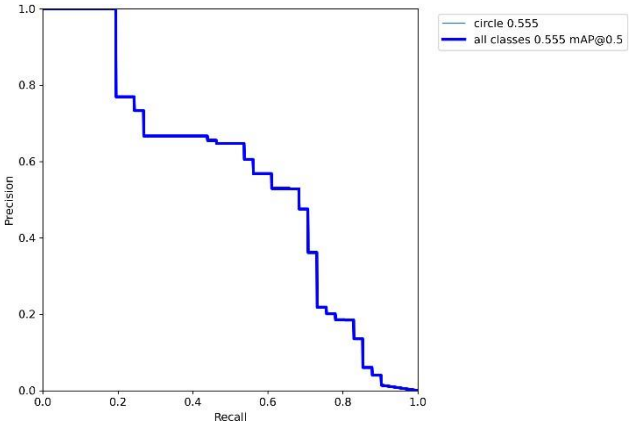
b) Recall vs Confidence graph



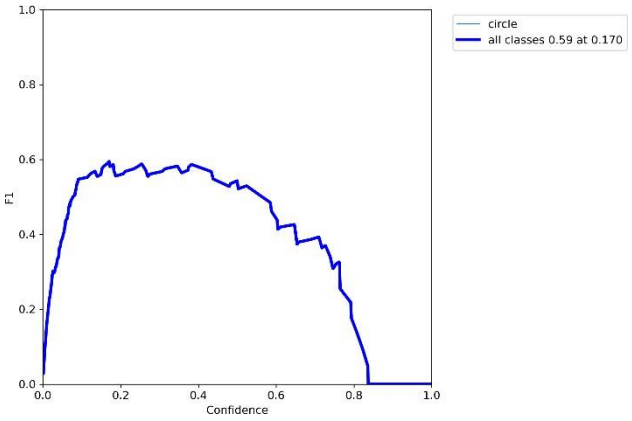
c) Precision vs Confidence graph



d) Precision vs Recall graph



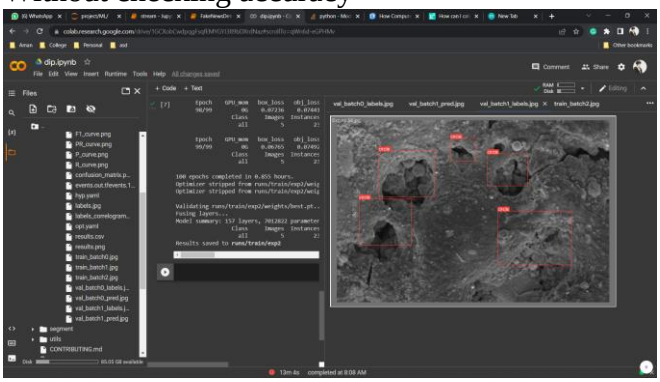
a) F1 vs Confidence graph



The statistics shows that with every increasing epoch the model trains better and gives a higher overall precision making the YOLO the best in the business out of all the available object detection models.

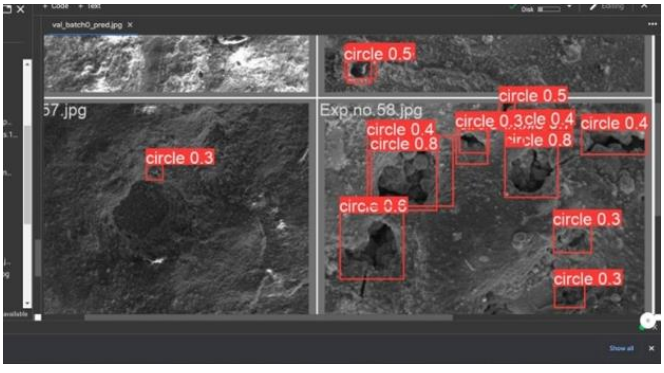
VIII. Accuracy

Without checking accuracy

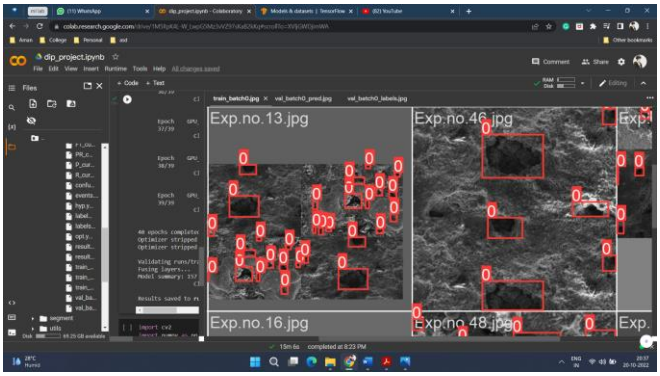


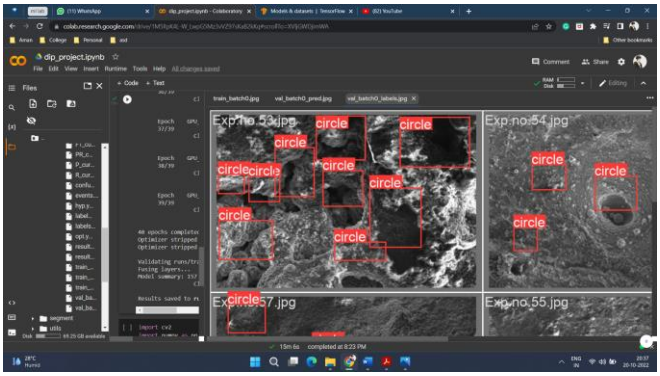
Accuracy before applying Image processing filters
In YOLO 5 is around 30% - 40%

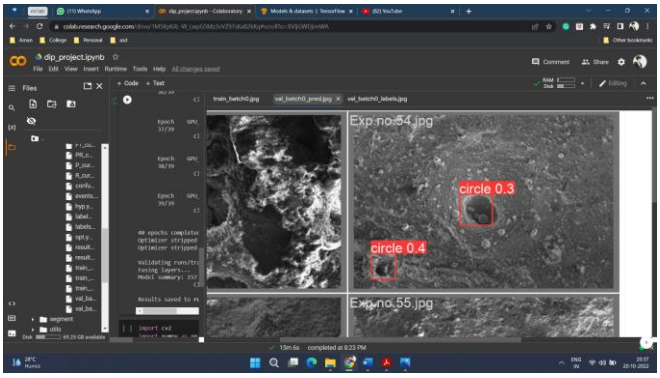
Accuracy after applying Image processing filters on result of YOLO5 is around 80%.

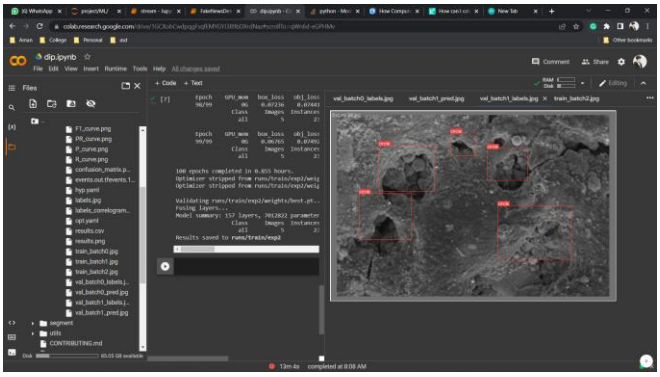


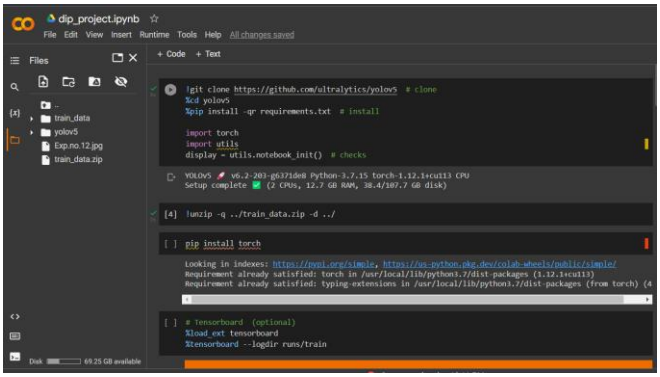
IX. Result Snapshot

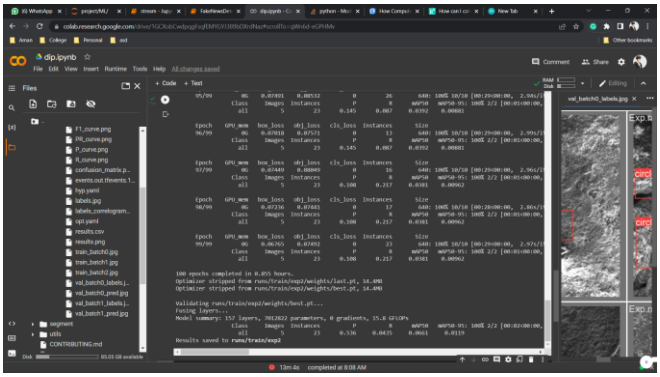


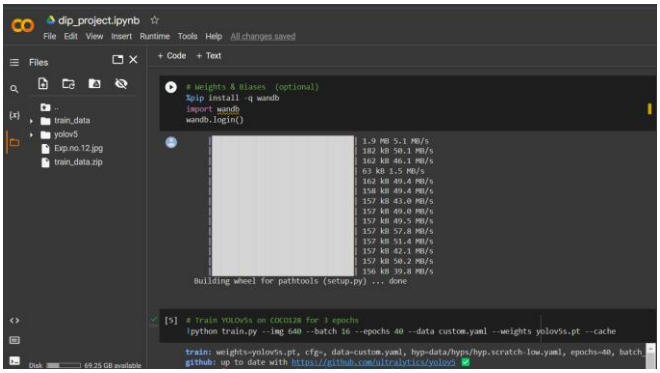


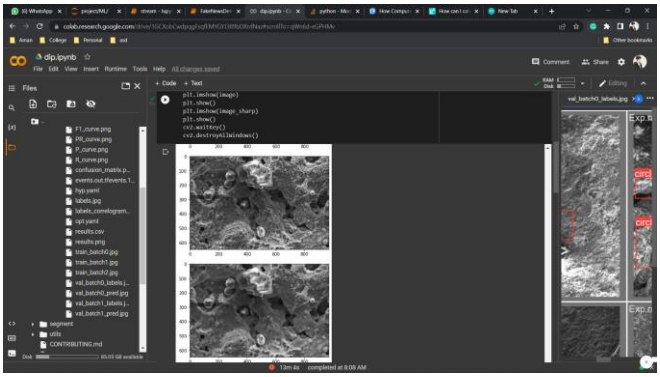


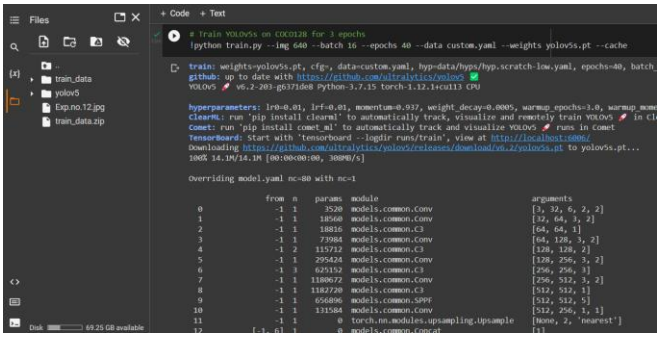


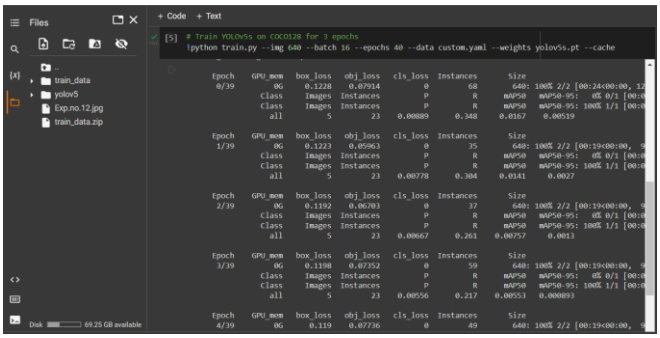


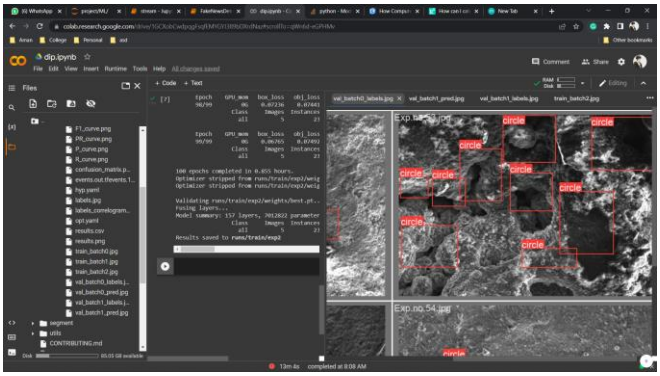












X. References

1. https://pytorch.org/hub/ultralytics_yolov5/#:~:text=YOLOv5%20%F0%9F%9A%80%20is%20a%20family,Model
2. https://github.com/ultralytics/yolov53.
3. https://towardsdatascience.com/the-practicalguide-for-object-detection-with-yolov5algorithm-74c04aac4843
4. https://medium.com/axinc-ai/yolov5-thelatest-model-for-object-detectionb13320ec516b
5. https://www.makesense.ai/