Toronto Metropolitan University

Department of Aerospace Engineering
Faculty of Engineering & Architectural Science

Semester (Term, Year)	**Fall, 2023**
Course Code	**AER850**
Course Section	**1**
Course Title	**Introduction to Machine Learning**
Course Instructor	**Dr. Reza Faieghi**
Submission	**Project Report**
Submission No.	**1**
Submission Due Date	**12/17/2023**
Title	**Project 3: Machine Learning**
Submission Date	**12/16/2023**

Submission by (Name):	**Student ID (XXXX1234)**	**Signature**
Mehtab Singh	**XXXX60754**	

By signing the above you attest that you have contributed to this submission and confirm that all work you contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, and "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Academic Integrity Policy 60, which can be found at www.torontomu.ca/senate/policies/

Aerospace Assignment Cover as of May 2022

Table of Contents

List of Figures

1. Introduction and Project Outline

The project surrounds the process of detecting components in a circuit board, streamlining the automated procedure of detection and reducing the human workload required in doing such a task.

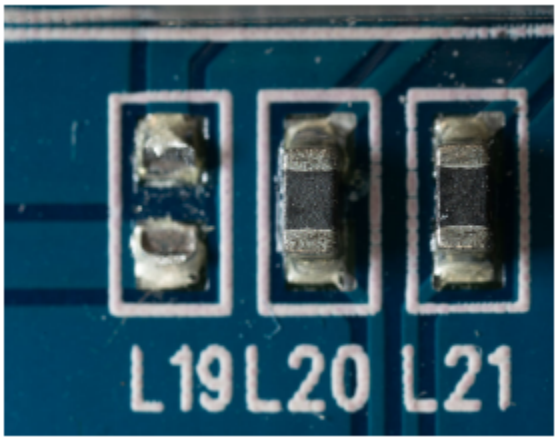

Figure 1: Missing Component in Circuit Board [1]

The following below are the circuit boards to be used for evaluation.

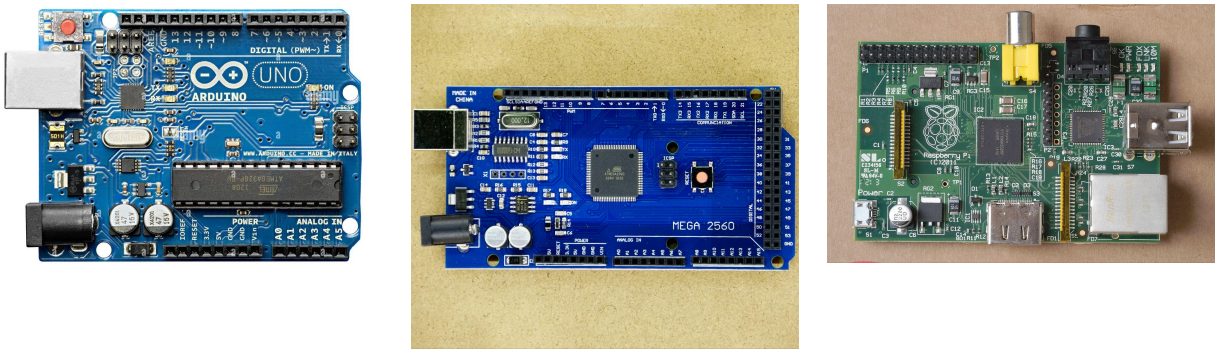

Figure 2: Evaluation Circuit Boards

2. Results & Discussion

The following are the steps taken in the project, the code for which is uploaded in the GitHub repository which is linked in the Appendix. The code was also zipped and linked alongside the submission of this report.

2.1 - Object Masking

To start off, object masking is used to remove the unnecessary part of the image when needed for training purposes. In this case, a sample motherboard is used to demonstrate the process. The *cv2.cvtColor()* is used to grayscale the image and *cv2.threshold()* command is used to assign classification to each object. The grayscaling allows for the object to be turned into a black and white image with shades of gray, this makes it simple for the detection of objects and backgrounds in a given image set. Once that is done the *cv2.threshold()* command will create an image from the given data, in the case with this code it was done both regularly and inverted to provide accuracy in the final cropped image. After this is done, *cv2.findCounters()* is used find edges in the object image and trace accordingly. Afterwards, the *np.zeros_like()* is used to cut the outer edges to mask the image. The two thresholds are then added together by layering on top of one another using the *cv2.bitwise_and()* command. The images are then output on each step as shown below.

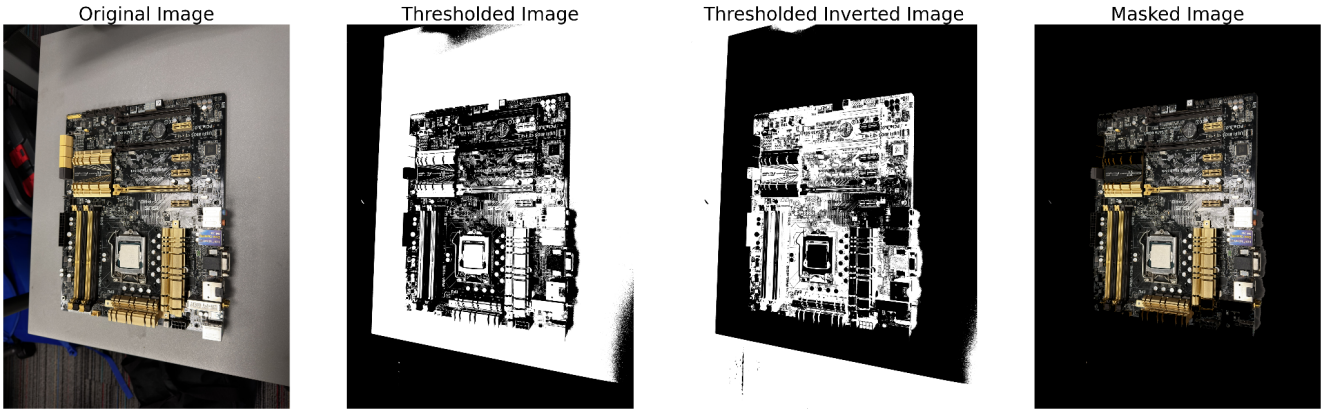

Figure 3: Motherboard Image Masking

The results show that the masking worked utilizing the inverted and regular method, with the exception of the parts at the bottom of the motherboard being darkened.

2.2 - YOLOv8 Training and Evaluation

YOLOv8 was used as the base model throughout this part of the code, with the addition of training to be done on the model. The *data.yaml* file was used as the basis where each part of the components are to be taken from. The *yolov8n.pt* file held all the model data that was trained.

2.2.1 Training

Training the model utilized the folders within the data file, using the *.train()* command. Running the program through 1 epoch for a low epoch output then 100 epochs for a high epoch output. The results are shown below.

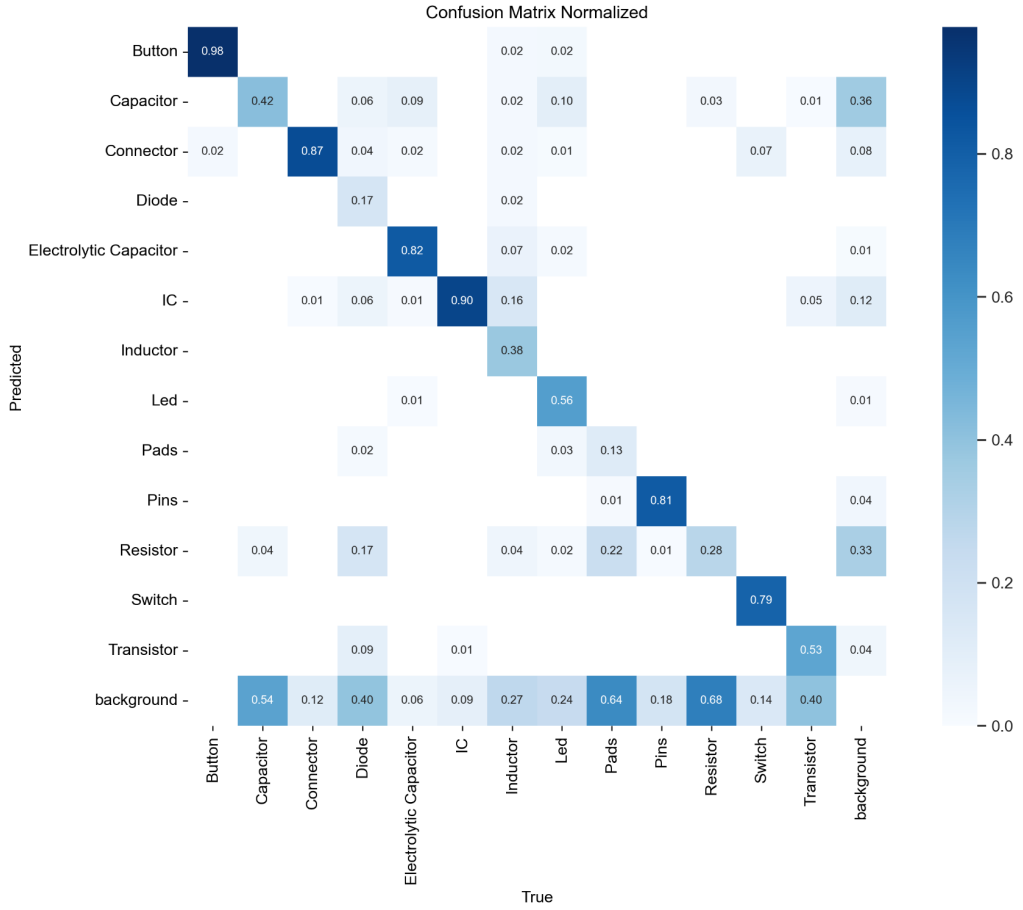

Figure 4: Confusion Matrix for YOLOv8 Trained Model

The confusion matrix showed somewhat accurate results as per standards. The places it had the least confidence were the pads and diodes.

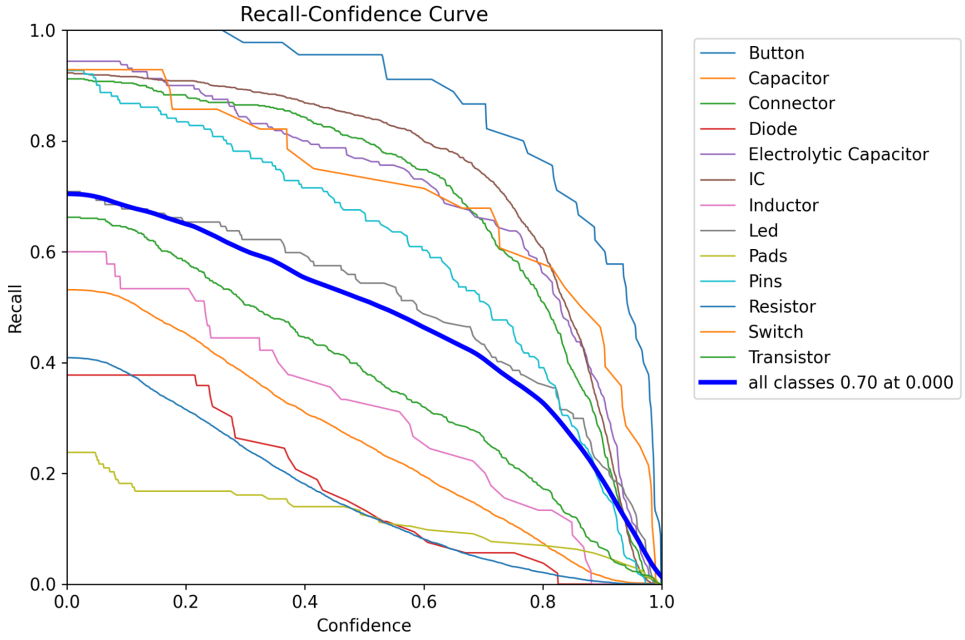

Figure 5: Recall-Confidence Curve for YOLOv8 Trained Model

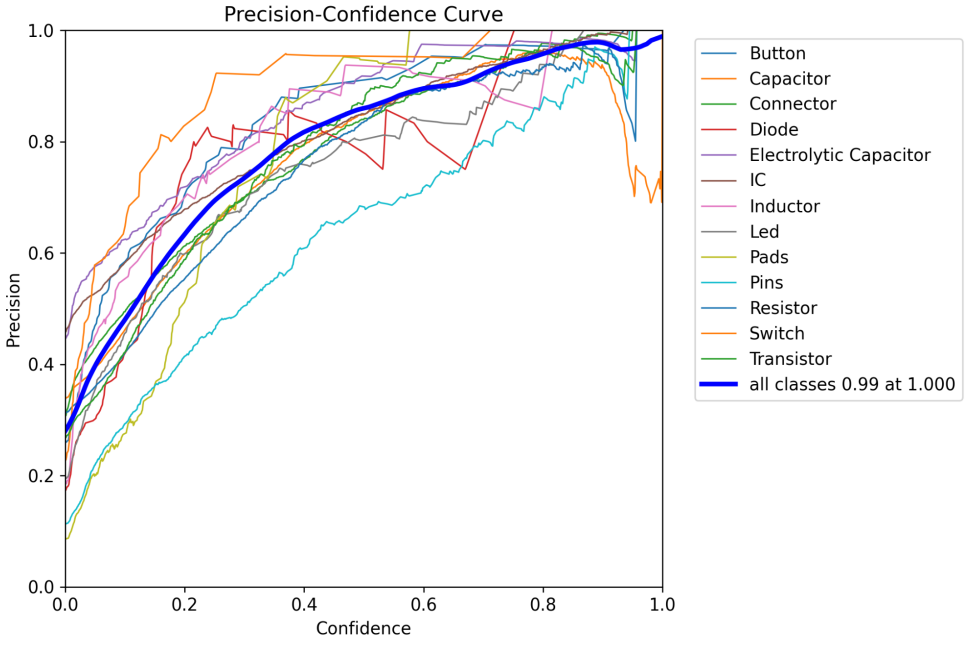

Figure 6: Precision-Confidence Curve for YOLOv8 Trained Model

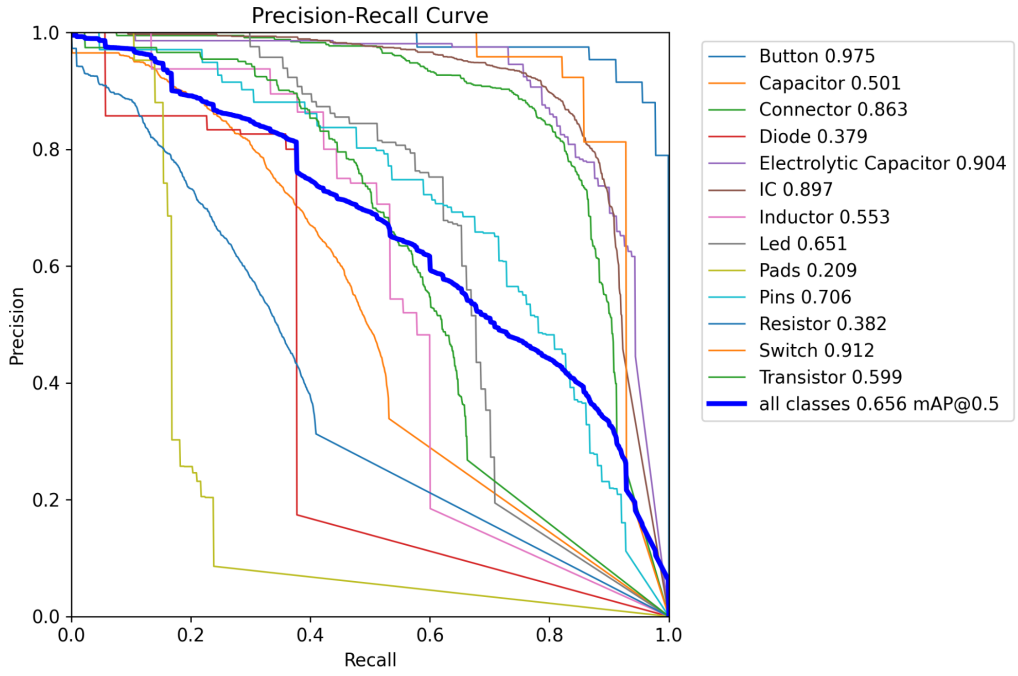

Figure 7: Precision-Recall Curve for YOLOv8 Trained Model

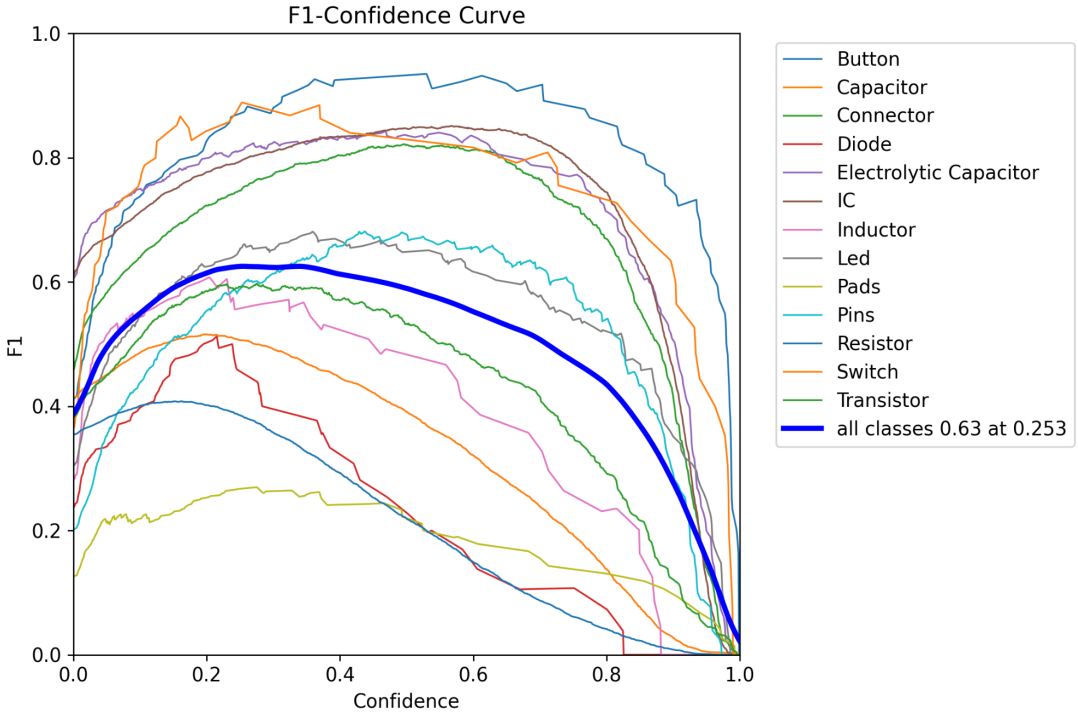

Figure 8: F1-Confidence Curve for YOLOv8 Trained Model

Each of the confidence and precision curves showed that button was the least confident and the capacitor was the most in detection. This indicates that there will be a correlation when seeing the final labeled output images. The following below shows the final results of the trained model.

```
Model summary (fused): 168 layers, 3008183 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 18/18 [00:21<00:00,  1.18s/it]
                   all        105      19108      0.736        0.6      0.656      0.457
                Button        105         45      0.803      0.978      0.975      0.763
             Capacitor        105       7251      0.702      0.377      0.501       0.25
             Connector        105        659      0.702      0.865      0.863      0.646
                 Diode        105         53      0.821       0.26      0.379      0.264
Electrolytic Capacitor        105        160      0.808      0.842      0.904      0.604
                    IC        105       1322      0.743      0.893      0.897      0.653
              Inductor        105         45      0.786      0.444      0.553      0.452
                   Led        105        127      0.687      0.622      0.651      0.426
                  Pads        105        143      0.727      0.161      0.209      0.143
                  Pins        105        151      0.506      0.781      0.706      0.473
              Resistor        105       8600      0.665      0.243      0.382      0.183
                Switch        105         28      0.921      0.833      0.912      0.766
            Transistor        105        524      0.702      0.504      0.599      0.316
Speed: 3.5ms preprocess, 100.8ms inference, 0.0ms loss, 54.2ms postprocess per image
Results saved to runs\detect\AER850_Project3_Trained_Model16
```

Figure 9: Model Results for YOLOv8 Trained Model

2.2.2 Evaluation

The following below are the results for each of the evaluation images in figure 2.

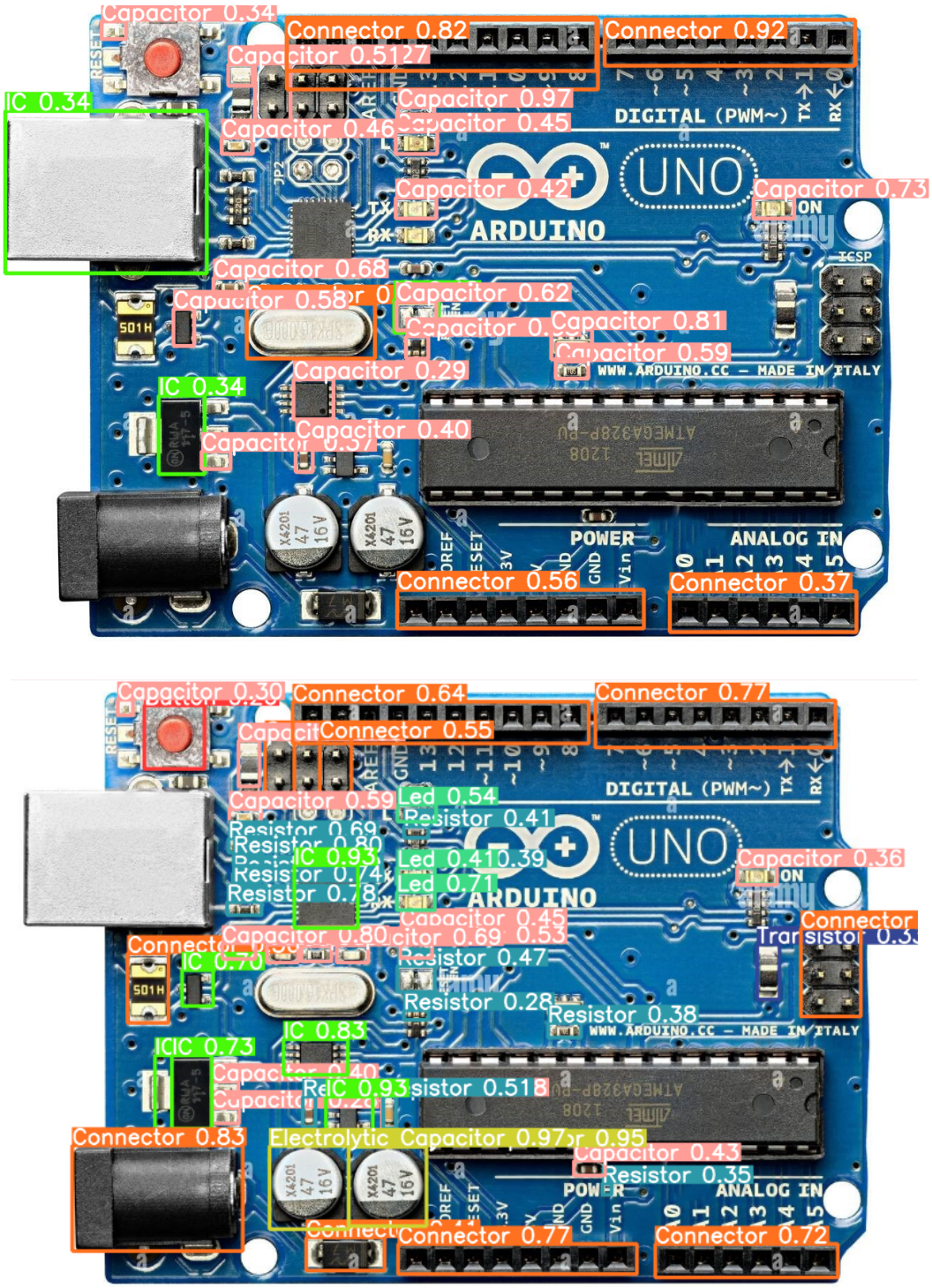

Figure 10: Low Epoch (Top) and High Epoch (Bottom) YOLOv8 Trained Evaluations for Arduino UNO Board

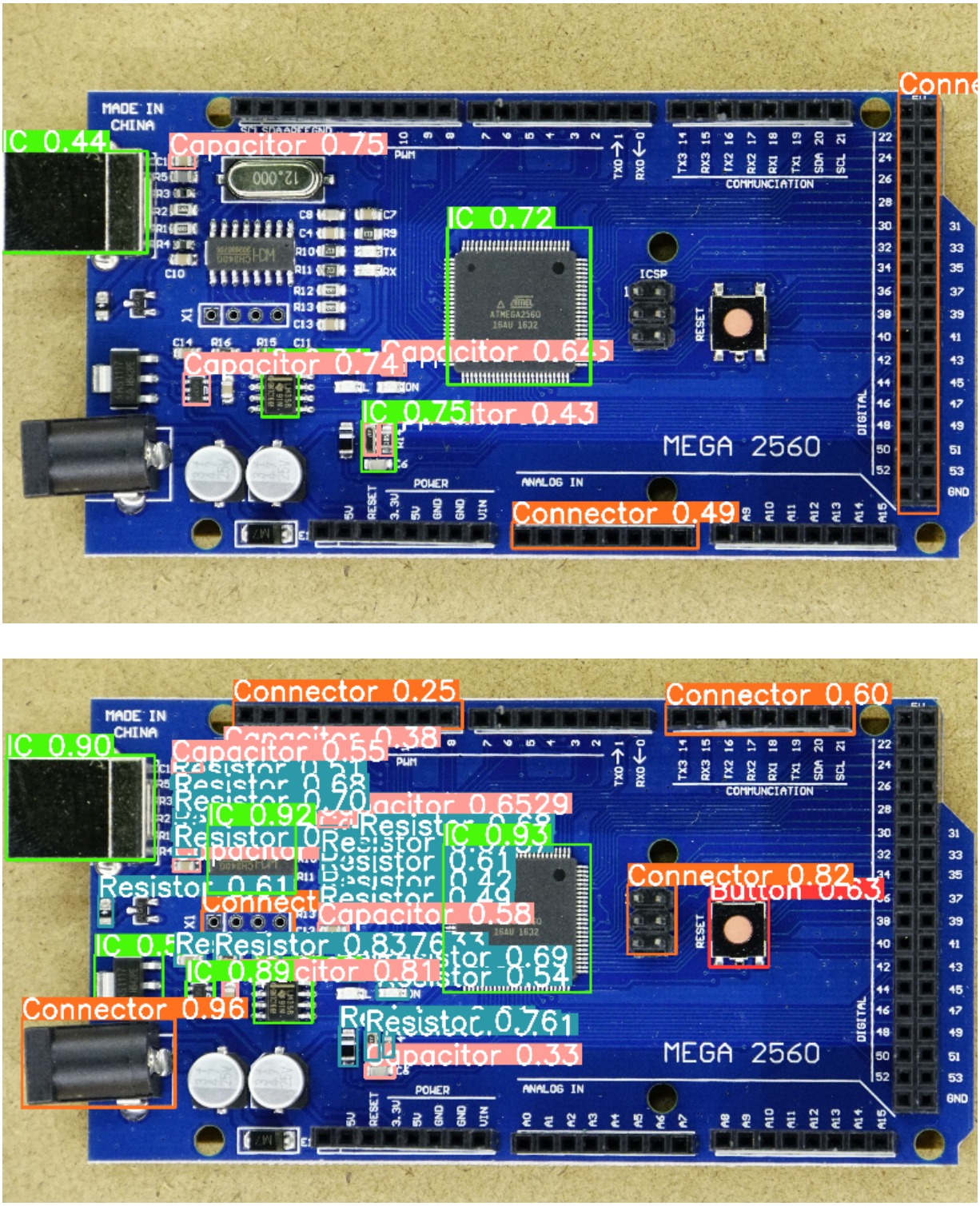

Figure 11: Low Epoch (Top) and High Epoch (Bottom) YOLOv8 Trained Evaluations for Arduino MEGA 2560

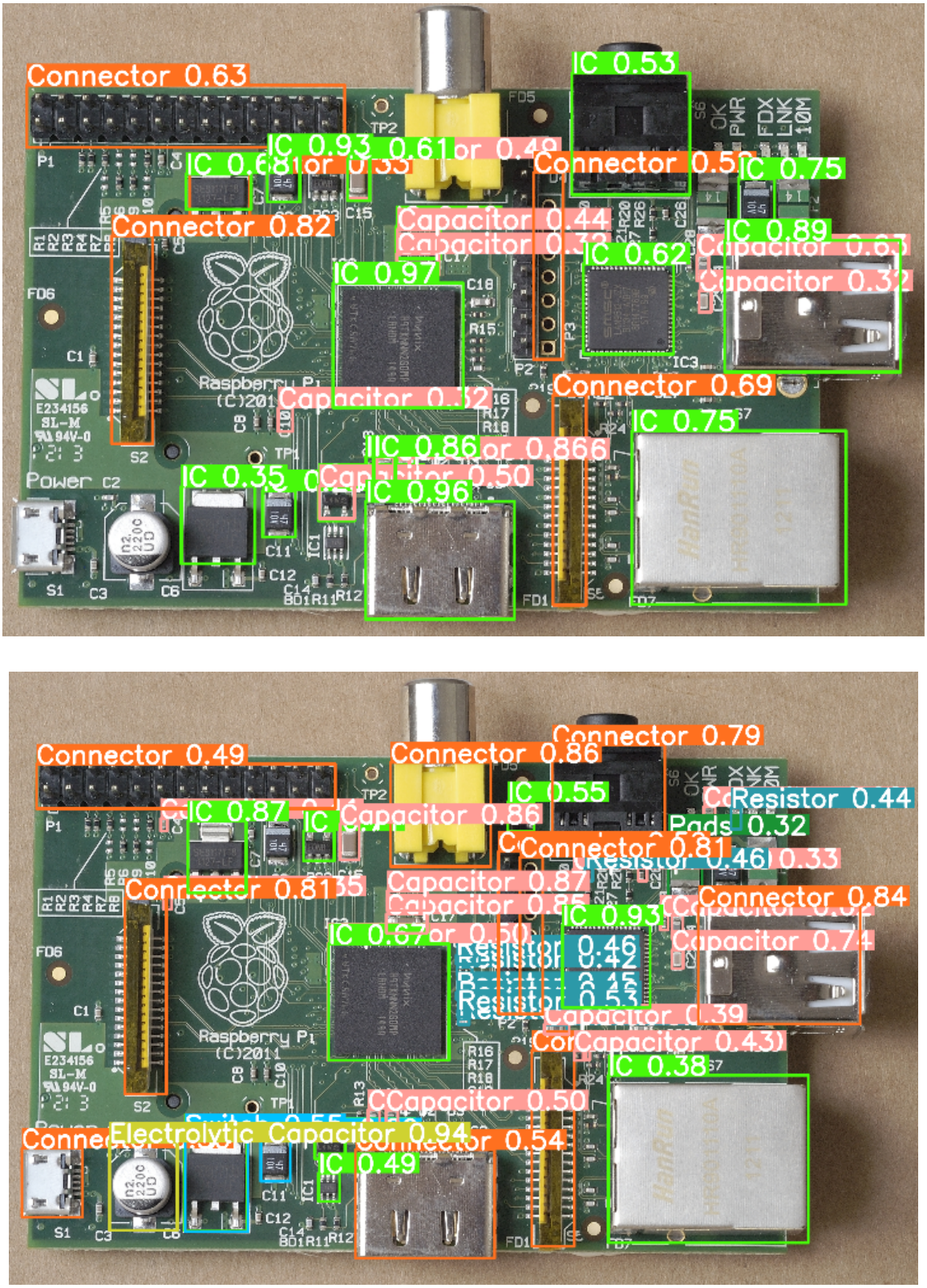

Figure 12: Low Epoch (Top) and High Epoch (Bottom) YOLOv8 Trained Evaluations for Raspberry PI C

In terms of the comparison between low and high epochs, the high epochs allows for more detection of components. The only output image that suffered a lot of detection loss was the arduino mega circuit, the connectors and capacitors were not fully detected. The reason for this is because more fine tuning is required to allow for more detection in the output. The *batch* size was 3 which could be increased to 5, while the *imgsz* size was put to the minimum 900 which could be increased to 1200. Computer requirements also lead to longer run times, as such lower parameters were used as the output.

3. Conclusion

This project helped in object masking and utilizing formed models to be trained. The process that goes into image masking allows for the analysis of circuit component detection. Utilizing this allows for the automation of circuit detection in factory work.

References

[1] Dr. Reza Faieghi “AER850 Project #3” https://courses.torontomu.ca/d2l/le/content/804341/viewContent/5432841/View [Accessed 12/9/2023]

Appendix

The following link is for the GitHub repository, all the code and folder designations are found in the zip file linked alongside the submission of this report:
https://github.com/Mehtab0Singh/AER850_Project_3_MehtabSingh