https://github.com/kikutabryan/aer850-project-3.git

AER850 PROJECT 3

Bryan Kikuta

501039179

Abstract

This project focused on the implementation of machine learning in printed circuit board (PCB) manufacturing for increased speed and performance in automated validation and verification processes of the boards. The first critical component which was developed was object masking, a means to retrieve the printed circuit board from a background which could then be used with the trained YOLO v8 nano network. The second component consisted of training a YOLO v8 nano network using provided training and validation datasets.

Project GitHub Link: https://github.com/kikutabryan/aer850-project-3.git

Table of Contents

Table of Figures

1 Introduction

The manufacturing of printed circuit boards (PCBs) is an essential component of modern electronics, requiring precise validation and verification to ensure quality and functionality. With advancements in machine learning, automation in these processes offers the potential to significantly improve speed and accuracy. This project explores the application of machine learning in PCB manufacturing, focusing on two critical tasks: object masking and training a YOLO v8 nano network. Object masking involves extracting a PCB from its background, creating a foundation for automated analysis. The YOLO v8 nano network is then trained to identify and classify various PCB components, leveraging datasets tailored for validation and evaluation. Through these tasks, the project aims to enhance the efficiency and reliability of PCB validation while addressing challenges associated with image preprocessing and model training. This report provides a detailed account of the methodology, results, and future considerations for implementing machine learning in PCB manufacturing.

2 Object Masking

2.1 Process

The object masking component of this laboratory focused on the extracting an image of a computer motherboard from its surrounding background. The image of the unaltered photo plotted can be seen in the figure below, where a computer motherboard is seen to be sitting on a table with some carpeted floor visible below.

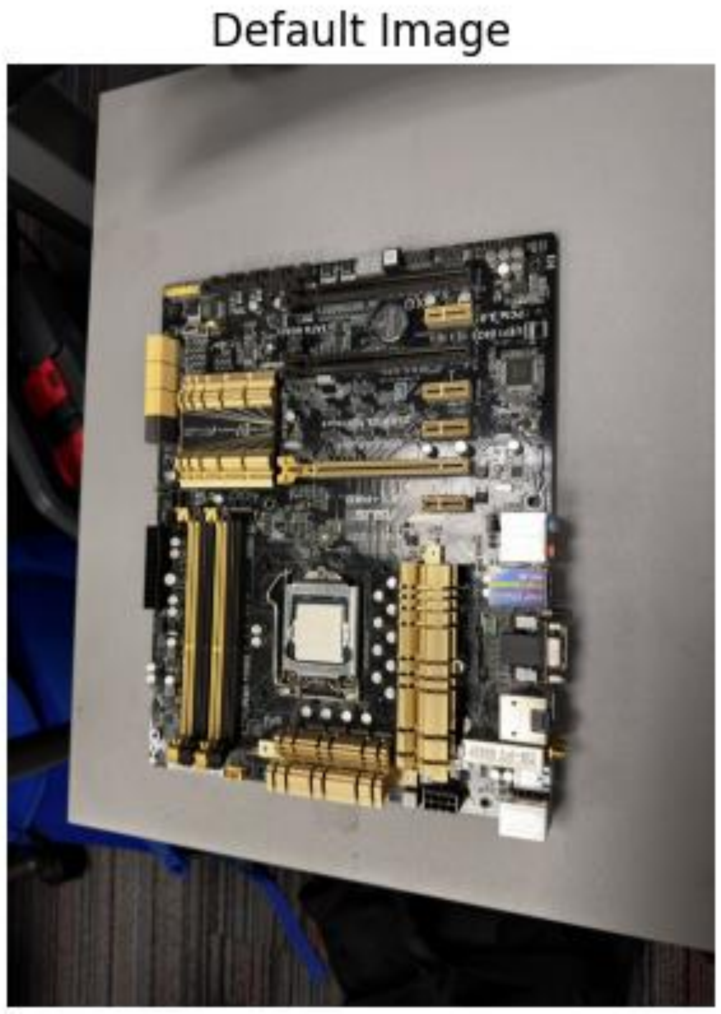

Figure 1: Plot of original motherboard image.

The first step taken in created the mask for the motherboard was to convert the image to a grayscale image, this was done as a lot of the filtering techniques applied afterwards are only available on grayscale images. Following, a gaussian blur was applied to help reduce the noise in the image, the noise could add an excessive number of edges or make the lines for the edges have many breaks due to small changes in pixel intensity. Following the blur, a binary threshold was applied to the image which can be seen in the figure below. The inverse binary threshold was used in this case as the background colour was lighter in intensity compared to the motherboard colour which is rather dark. The binary threshold forces the image to be only in two colours, either 255 for white, or 0 for black. This makes it simple to extract edges which are described as the change from white to black or vice versa.

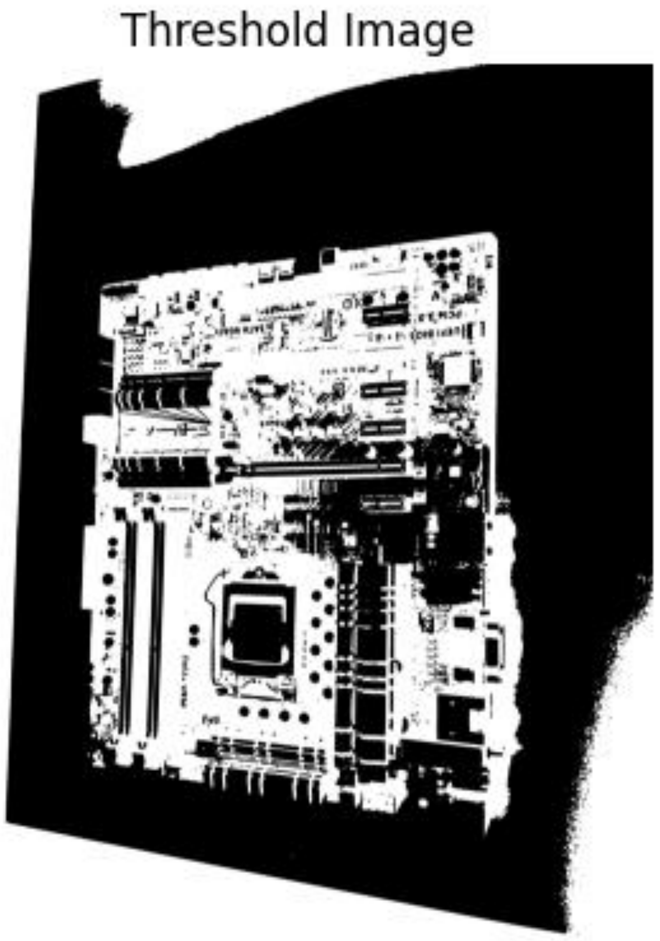

Figure 2: Plot of motherboard image after binary threshold applied.

Following the binary threshold being applied to the image, Canny edge detection was utilized. The result of the Canny edge detection alone was not satisfactory, especially with the high resolution of the image. Experimentation was conducted with lowering the resolution of the images, improving the performance of the Canny edge detection, however, when resizing the mask back to the original image, resolution was lost around the edges, and the mask would bleed beyond the motherboard and have a rough edge due to the mask being for a lower resolution image. To alleviate this issue, the mask was made for the same resolution, and OpenCV's morphology was utilized to fill in the gaps between the edges, effectively creating larger contours that could be utilized. The result of the Canny edge detector and the edge closing can be seen in the figure below.

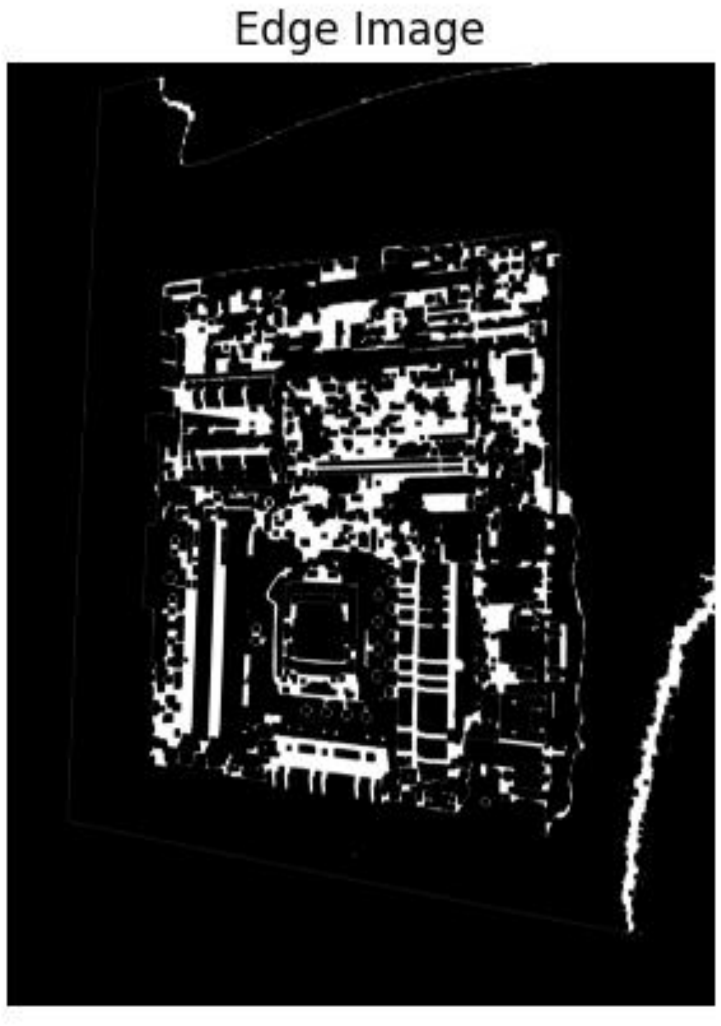

Figure 3: Plot of motherboard after edges extracted and gaps filled.

The next step was to create the mask, the mask was created by selecting the largest contour from the extracted edges, this contour was the motherboard's outline. The mask can be seen in the figure below, where the white regions are the motherboard, and the black regions are the background.

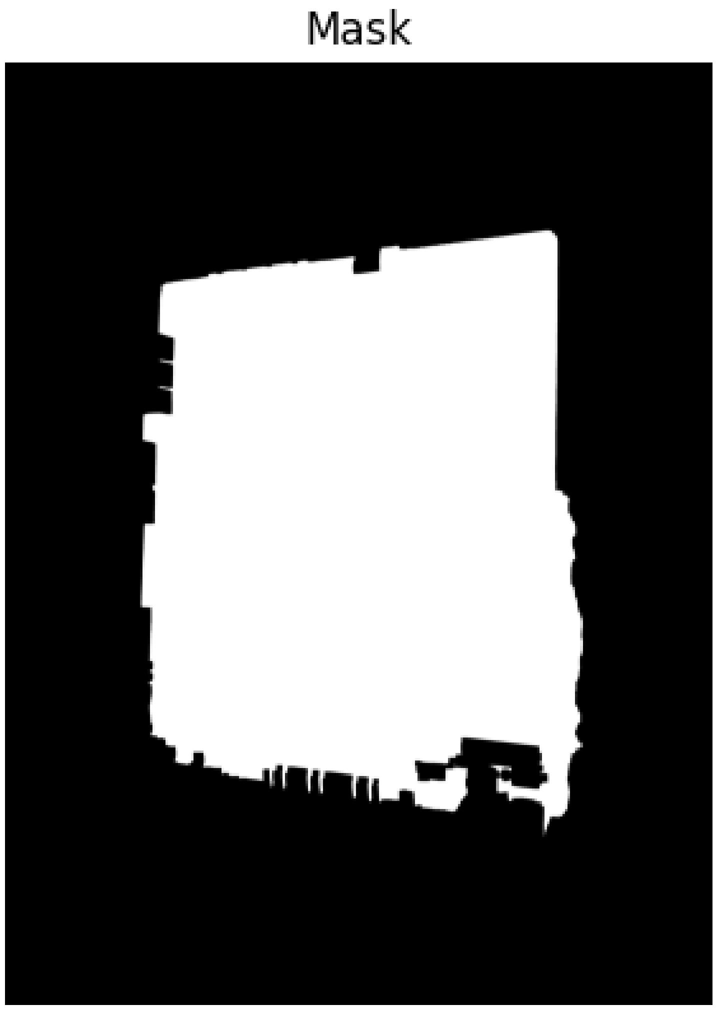

Figure 4: Plot of motherboard after largest contour used for mask.

Finally, the bitwise and operation was utilized to select the motherboard from its background by selecting the region where the mask was white and setting all the other regions to the colour black. The figure below shows the extracted image of the motherboard.

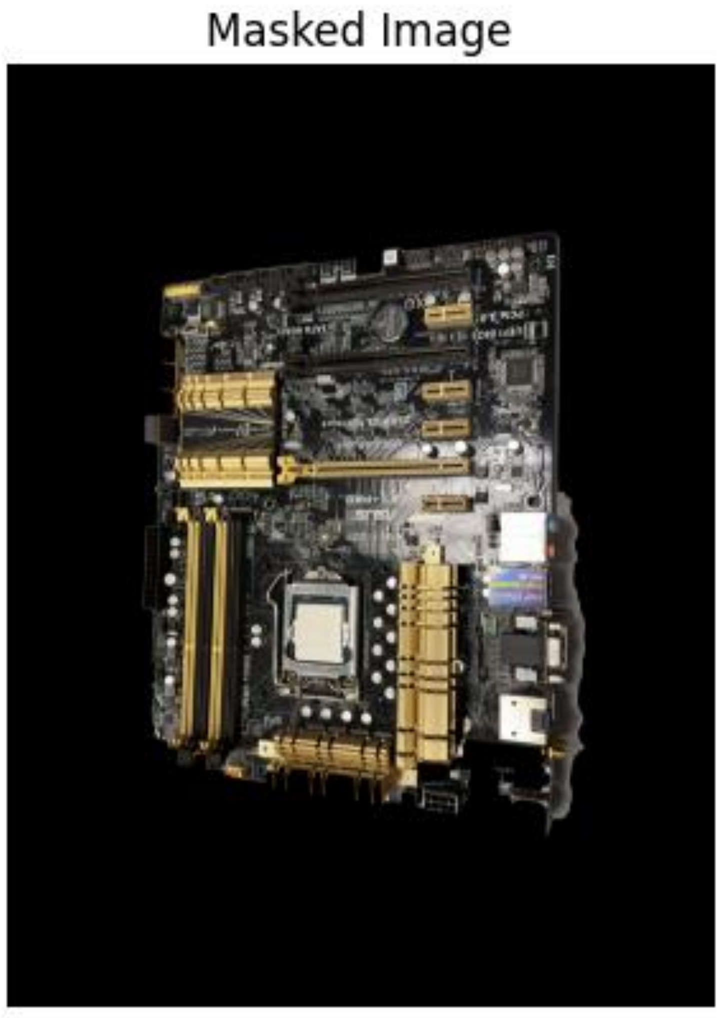

Figure 5: Plot of final extracted image of motherboard.

2.2 Discussions

From the figure showing the extracted image of the motherboard, the mask was not exact, there are some regions of the motherboard that were removed in the mask, and there are areas that show the table the motherboard was sitting on. These faulty regions appeared in areas where the colour of the background closely resembled that of the motherboard. The motherboard was composed of various components with different colours and intensities, these regions made the threshold portion of the script moderately challenging as there was no perfect threshold to entirely separate the motherboard from its background environment. Moreover, the motherboard cast shadows onto the surface it was sitting on, making the table darker and less distinct from the motherboard, making it less likely to have an edge detected after a threshold as they would appear as they same colour.

While the current process has its limitations, iterative approaches could be tested by creating a function with multiple parameters for the different methods that were utilized to generate the mask. A proper mask could be created by hand of the motherboard, and these parameters altered with the parameters that generated the best mask when compared to the proper mask being selected. This method might be good for this instance, but if the goal were to make it expandable to a plethora of different motherboards or PCBs, a different method such as image segmentation could be utilized, where a model is specifically trained on datasets of PCBs extracted from their backgrounds. This method would prove to be useful for a variety of different backgrounds and PCBs so long as it has had some exposure to differing backgrounds and boards.

3 Training & Evaluation

3.1 Metrics

The confusion matrix is a good tool for showing how well a model performed. The normalized confusion matrix seen in the figure below shows the true classes on the horizontal axis, and the predicted classes on the vertical axis. This can be used to see for a given true class the spread of predictions among the other classes and itself. From the confusion matrix, it can be seen that the model performed well for the buttons, for the connectors, electrolytic capacitors, ICs, pins, and switches. The model performed very poorly for background, making no accurate predictions for that. And moderately poorly for capacitors, diodes, pads, and resistors. Otherwise, the model performed satisfactorily well.

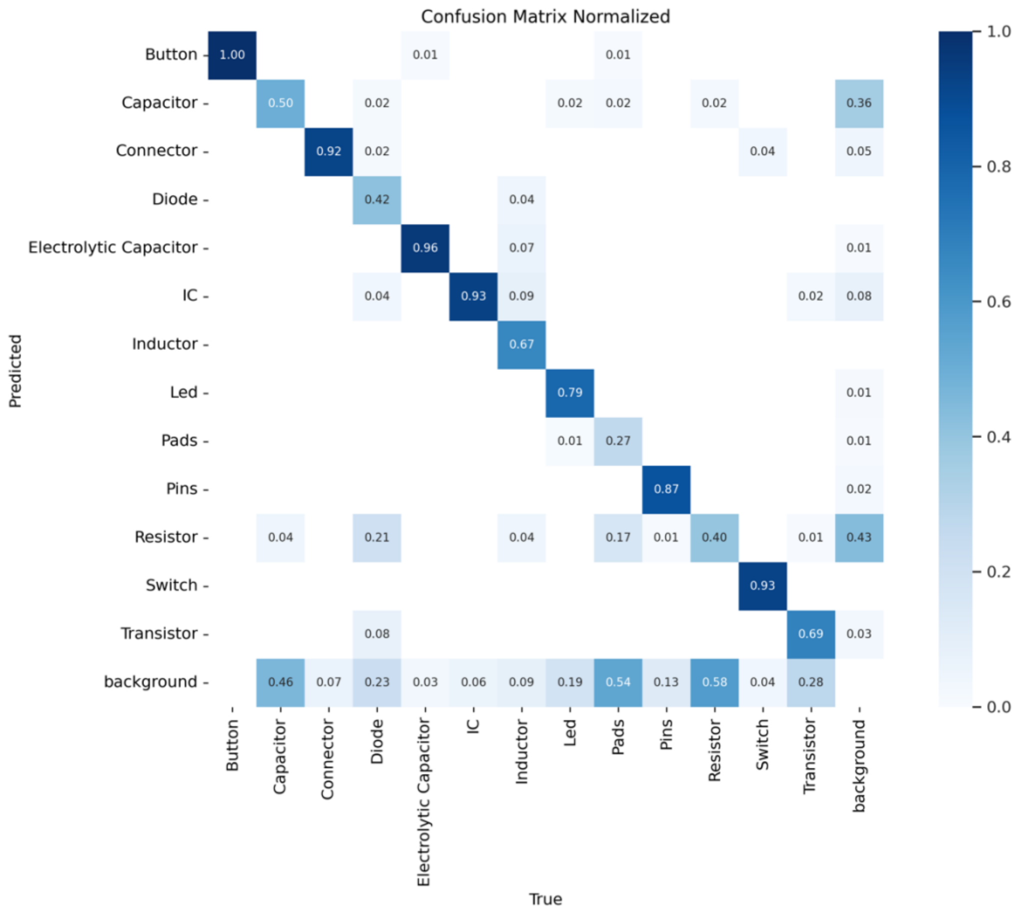

Figure 6: Normalized confusion matrix

The figure below shows the precision confidence curve of the model. From the curve, on average the curves continued to increase in value seen by the thick blue line visualizing all of the classes together. This is a good trend as it indicates that as the confidence of the predictions threshold increased, so did the precision of the model, which is what would be desirable. It would be undesirable for the model to have high confidence but perform poorly as the confidence would then be inversely related to the precision of the model.

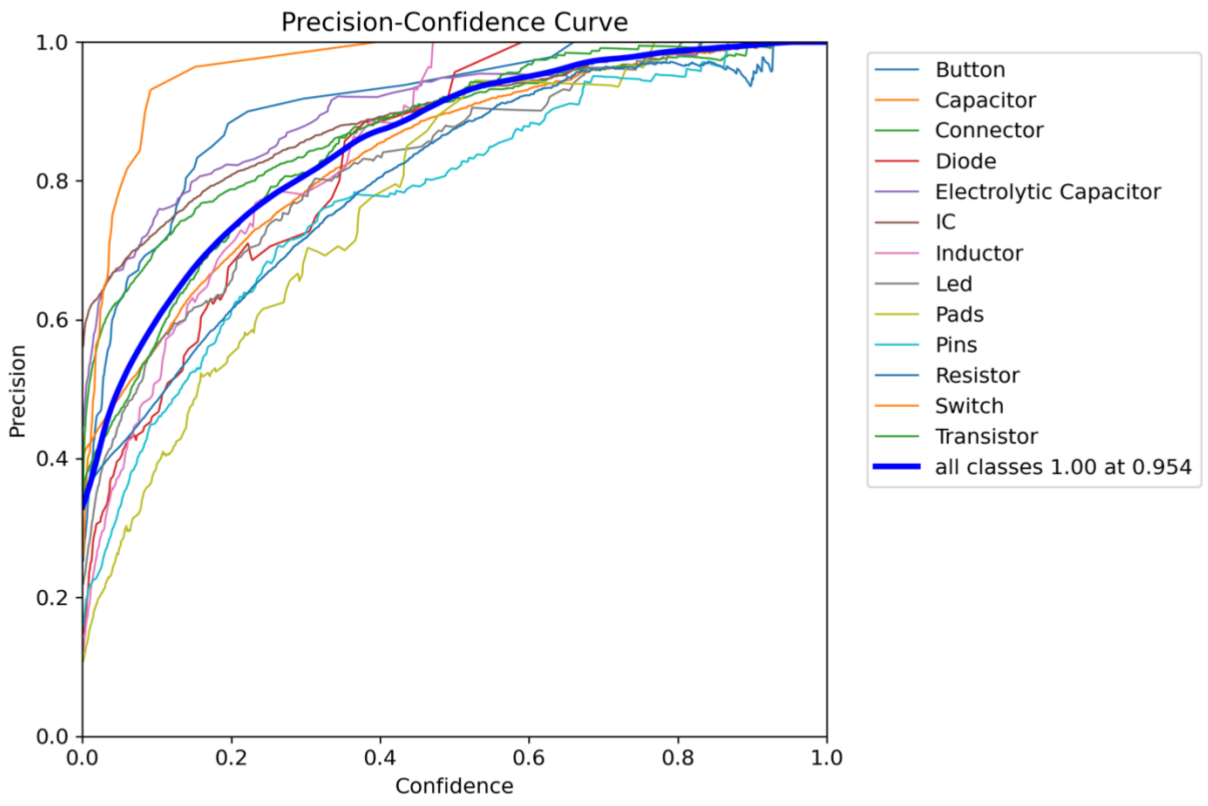

Figure 7: Precision-confidence curve

The next curve is a precision recall curve, this curve shows the relationship or the trade-off between the two with differing thresholds. The curve shows that as the recall increases, the precision will decrease which is expected as precision shows the accuracy of positive predictions, and recall shows the ability of a model to correctly identify positive instances. As the threshold is increased, the recall will typically decrease as the model is more likely to label more false negatives lowering the ratio. The precision as the threshold increases will ultimately increase as the model becomes more conservative in its predictions, and as seen in the previous figure, as the confidence of the model threshold increases, the precision increases. Thus, with an increase in precision, we should see a decrease in recall which can be seen in the curve in the figure below.

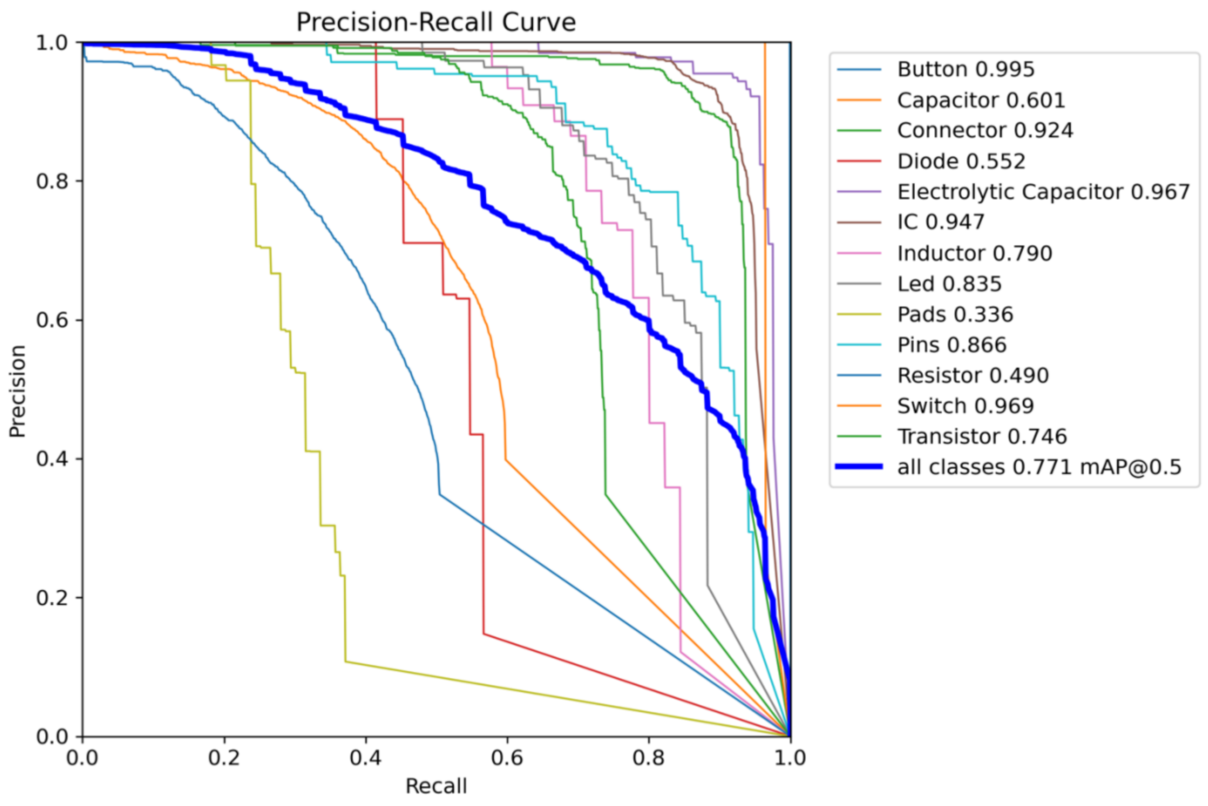

Figure 8: Precision-recall curve

3.2 Test Images

The first test image is that of the Arduino Mega board. Based on the image, the model did a relatively good job at labelling components such as the resistors, connectors, capacitors, and the button. The ICs were correctly predicted, however, there was a false positive with the main connector of the board being identified as an IC. This could be due to the rectangular shape and dark appearance in the image of the Arduino Mega. The connectors were mostly predicted, except for the ones on the top side of the board. The power jack connector was correctly predicted, in addition to the connector holes without anything soldered into them. There were some resistors missed like R17 and R18, but overall, the model did not do poorly predicting the components for the Arduino Mega.

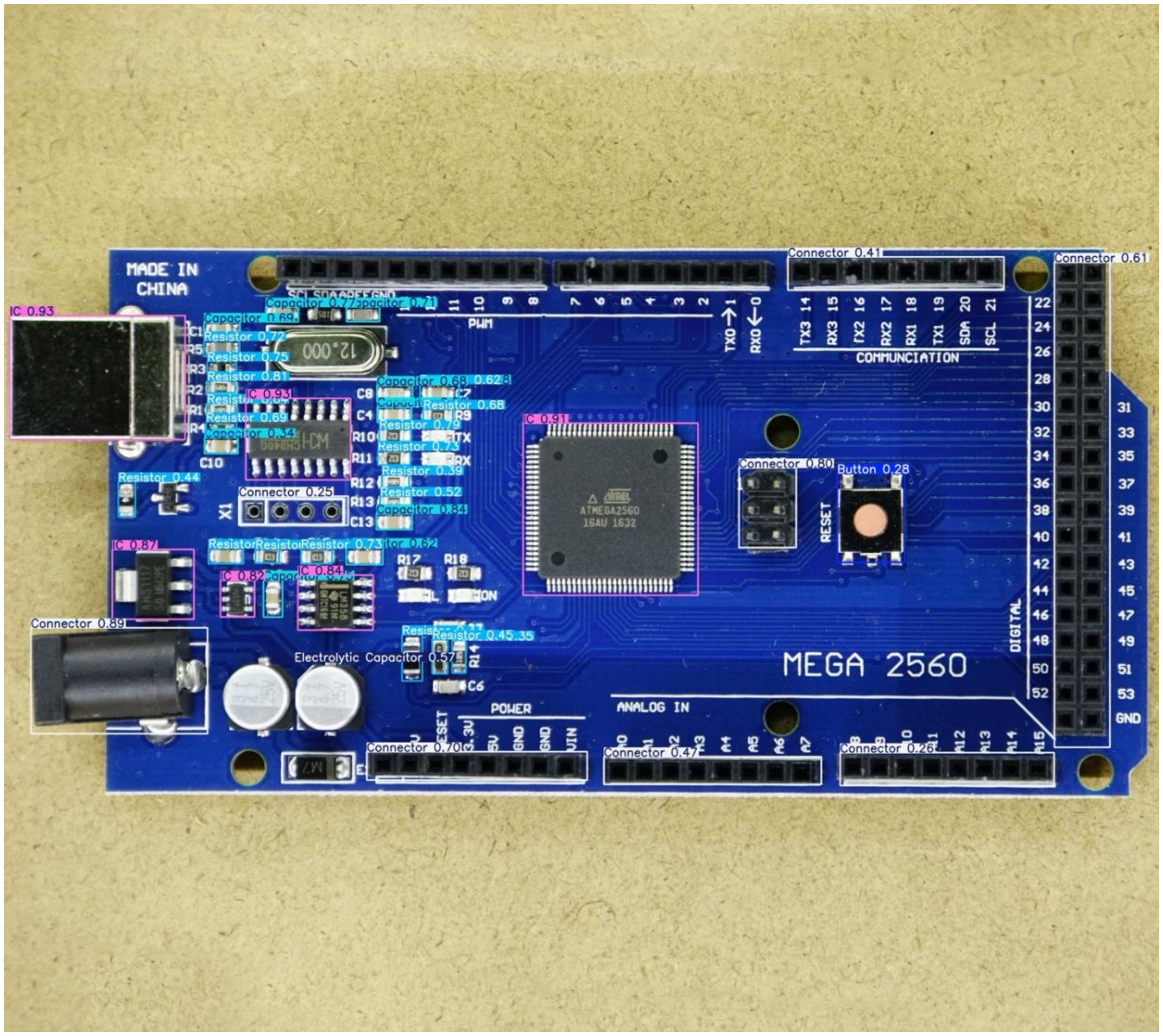

Figure 9: Arduino Mega predictions

The next board was the Raspberry Pi. There were a lot of decent predictions on this board such as the connectors, ICs, resistors, capacitors, and the connectors. While the majority of predictions were good, there were a number of incorrect ones such as the power port on the Raspberry Pi being labelled as an IC instead of a connector, the ribbon cable connector not being labelled, a handful of the smaller resistors and capacitors not being properly labelled, and arguably the SD card slot being labelled as a connector.

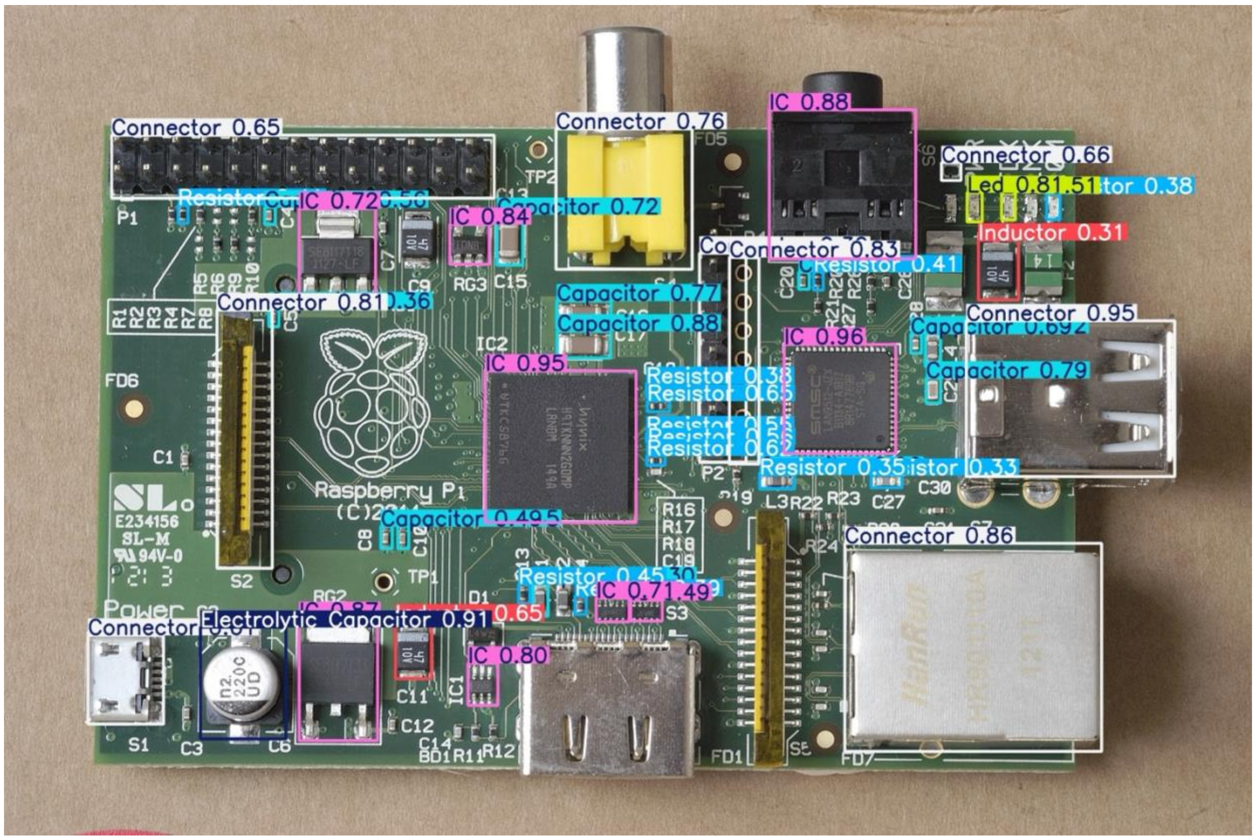

Figure 10: Raspberry Pi predictions

The final board predictions were completed on was the Arduino Uno board. On the Arduino Uno, all the connectors on the board were correctly labelled, but the main IC of the board was also labelled as a connector. This could be due to the legs of the IC being visible, and it having a long rectangular shape much like a connector might have. It appears as though the majority of the resistors and the capacitors were labelled on the image accurately, with a couple of them being missed in a few locations. Moreover, the model managed to predict the LED lights properly, but did miss one of the lights. The electrolytic capacitors of the board were accurately predicted.

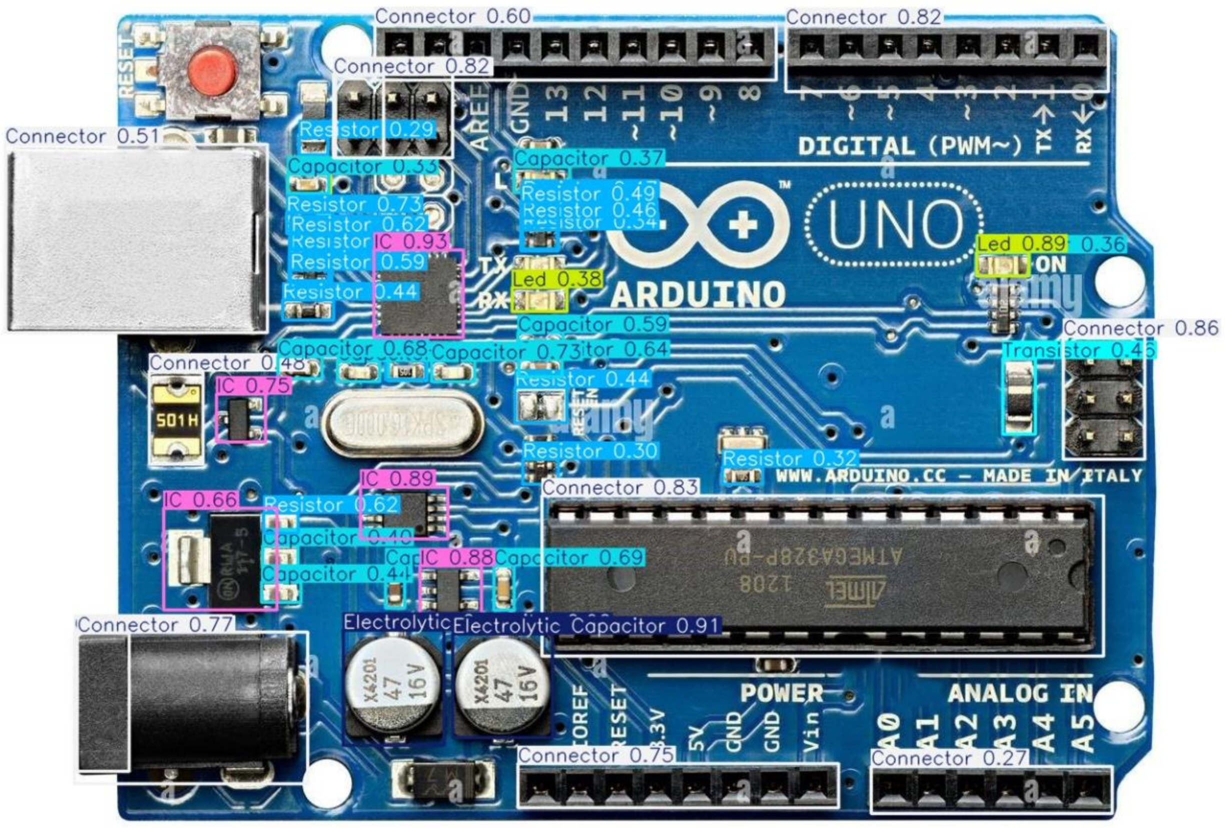

Figure 11: Arduino Uno predictions

3.3 Discussions

While a lot of predictions were accurate when the model was tested, it is evident that the performance was not exceptional. This can be seen in the model missing out on small components on the boards such as some of the very small resistors and capacitors, in addition to the incorrect labelling of some of the ICs as connectors. The parameters for training the model consisted of an image size of 928 by 928 pixels, in addition to a batch size of 8. Early stopping was set by having the value of the patience parameter set to 20 epochs. Some improvements that could be made to the model consist of increasing the resolution, possibly increasing the batch size, and increasing the number of epochs. Moreover, an increased dataset could be utilized. The changes of increasing the image size would potentially alleviate some of the challenges the model faced when predicting the smaller components of the board such as the resistors and the capacitors. This coupled with the increased batch size would run into the issue of available VRAM, as the current configuration was pushing the limits with a 16 GB graphics card, thus a GPU with more available VRAM would be required. Providing more data could help the model with the confusion of dark rectangular connectors as ICs, as it would have been exposed to similar images and be capable of predicting it better.

4 Conclusions

This project demonstrated the integration of machine learning into PCB manufacturing, focusing on object masking and the application of the YOLO v8 nano network for automated validation. The object masking process achieved reasonable success in extracting the PCB from its background, despite challenges posed by varying intensities and shadowing. Similarly, the trained YOLO model performed well in identifying key PCB components, though limitations such as missed predictions and misclassifications were observed. Improvements in training parameters, dataset size, and hardware capabilities offer potential pathways for enhancing model performance. Future work could explore advanced image segmentation techniques and broader datasets to generalize the system for diverse PCB designs and environments. Overall, this project highlights the promise of machine learning in streamlining PCB manufacturing processes while identifying areas for continued refinement and innovation.