Toronto Metropolitan University

Department of Aerospace Engineering
Faculty of Engineering & Architectural Science

Semester (Term, Year)	F2024
Course Code	AER 850
Course Section	2
Course Title	Intro to Machine Learning
Course Instructor	Dr. Reza Faieghi
Submission	Project Report
Submission Due Date	Dec. 15th, 2024
Title	Project 3 Report
Submission Date	Dec. 14th, 2024

Authors/Contributors:	**Student Number (XXXX12345):**	**Signature*:**
Ahad Razack	8108	[illegible]

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/

Introduction

The following report summarizes 2 major efforts. The first is an image processing task in which OpenCV is used to mask out an image of a motherboard and separate it from its background. The second involves training a YOLOv8 model to detect and classify various hardware components from a PCB. These techniques have the potential to greatly enhance the manufacturing efficiency of these components, as they can replace time-consuming manual inspections.

Object Masking

In order to generate the image mask, OpenCV tools were extensively utilized. After loading the image and converting it to grayscale, the first major step was to apply a gaussian blur to the image. This has the effect of reducing sharp details in the image, ensuring only the large "structural" features remain. The next step was to threshold the image, creating distinct regions based on pixel intensity. For this step, OTSU thresholding was used, meaning the threshold pixel value was automatically generated. Presented below is the thresholded image.

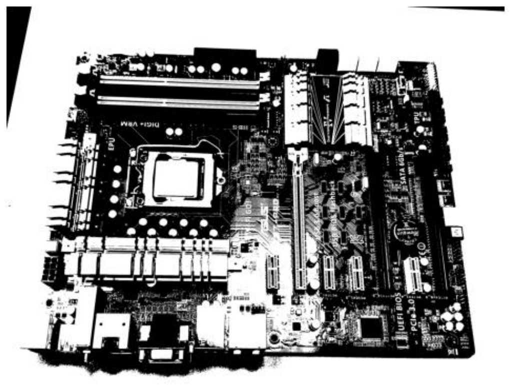

Figure 1: Thresholded image

From this, it can be seen that certain regions of the mother board, like the heatsinks on the left-hand side, are not sufficiently distinct from the background and as such appear as white. This adversely affects the edge detection. Additionally, as the image was taken at a slight angle, there is a shadow casted on the lower left-hand side. This creates a region that is interpreted as part of the motherboard. It is recommended that future images be taken on a distinct background, from an overhead angle, and with sufficient lighting. This would ensure that a cleaner mask can be created, to better extract the motherboard.

After thresholding the image, Canny edge detection was used, generating the following.

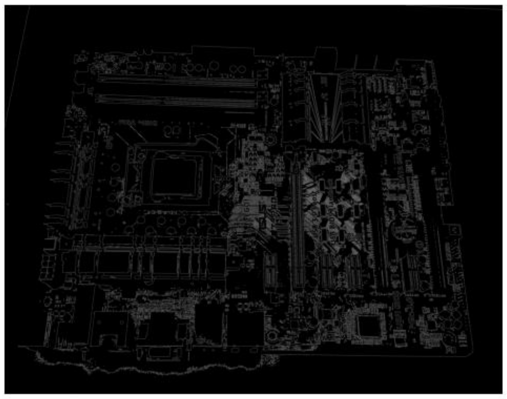
Figure 2: Detected Edges

Once the edges were detected, a morphological operation was applied to close the edges. A large kernel size was used, as it had the effect of connecting the jagged edges around the motherboard, leading to a more complete mask.

After closing the edges, contour detection was performed, and the largest contour was filtered out in order to draw the mask presented next.

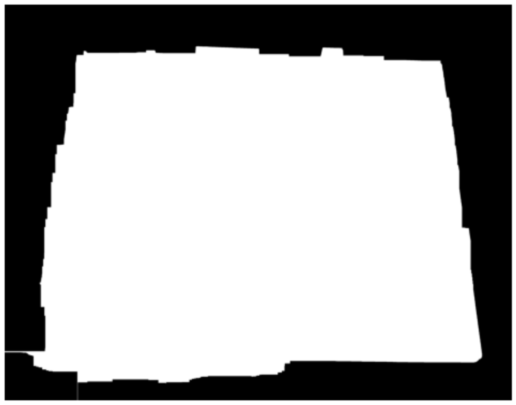
Figure 3: Mask Image

Lastly, the mask was used to extract the final image of the motherboard.

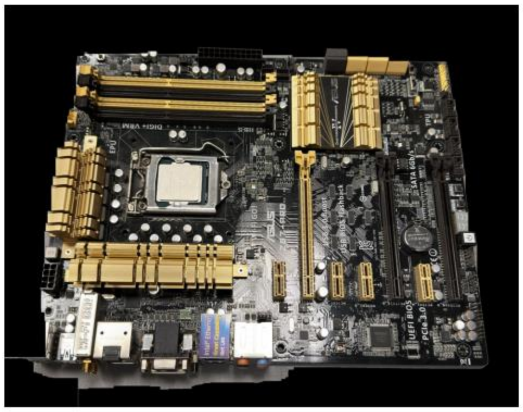
Figure 4: Final extracted image

As seen in figure 4, the mask obviously isn't exact. The two main areas of improve as mentioned above would be using a distinct background to better detect components at the edges and using lighting and angles to eliminate shadows.

Model Training – Setup

For this portion of the project, a YOLOv8 nano model was trained on a dataset of labelled PCBs to be able to detect components on the PCBs. An image size of 700 was used, as any size above this would lead to the runtime crashing due running out of memory. In an attempt to remedy this, the auto-batch argument was used. This would calculate the optimal batch size given the memory available for training. Although it delayed the memory error, it still occurred after a few epochs of training. As a final attempt, a batch size smaller than the calculated auto-batch was used. Despite not crashing, training was extremely slow and would time out due to inactivity. As recommended, 200 training epochs were used.

Model Training – Results

Presented first is the normalized confusion matrix of the trained model.

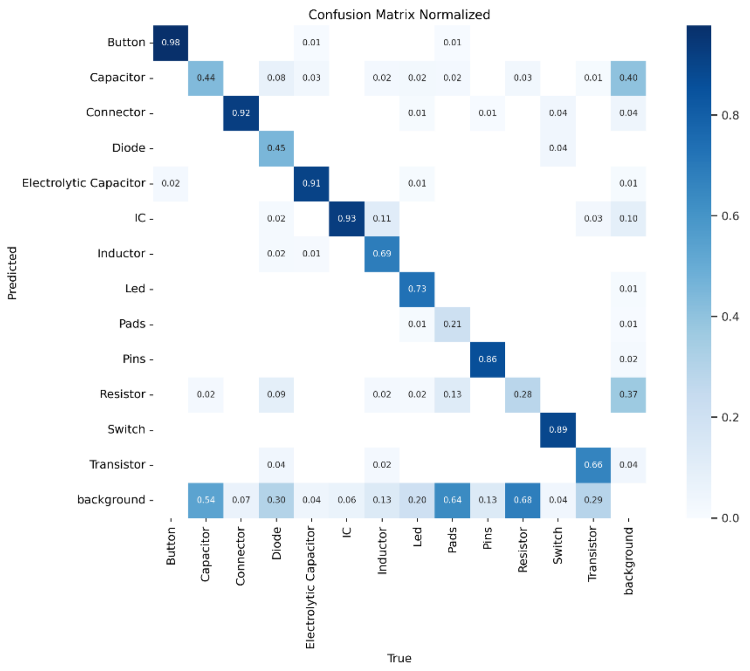

Figure 5: Normalized confusion matrix of trained model

The normalized confusion matrix presents the prediction accuracy of the classifier. The main diagonal represents the correct class predictions, and the other values are wrong predictions. The results of this classifier vary greatly. Certain components like buttons and connectors have very high accuracy, meaning the classifier has no problem identifying them and doesn't confuse them for other components. On the other hand, certain components like pads and resistors have really low accuracy. The most predicted class for these cases is background. This implies that the model isn't confusing components for each other, but rather struggling to distinguish them from the background. This makes sense, as the misclassified components tend to be the smaller ones. To remedy this, a higher image resolution would be necessary. This would ensure that these small components are recognized as structural features and not background artifacts. Another solution could be to increase the frequency of these components in the training data, allowing the model to better learn what these components look like and thus being better able to recognize them.

Presented next is the precision-confidence curve of the model. Precision is the proportion of true positives to all positive predictions. The confidence, or threshold, represents the point that distinguishes a positive prediction from a negative one. The general shape of the precision-confidence plot is that as the required confidence increases, the precision also increases, as the model is stricter about what it classifies as positive.

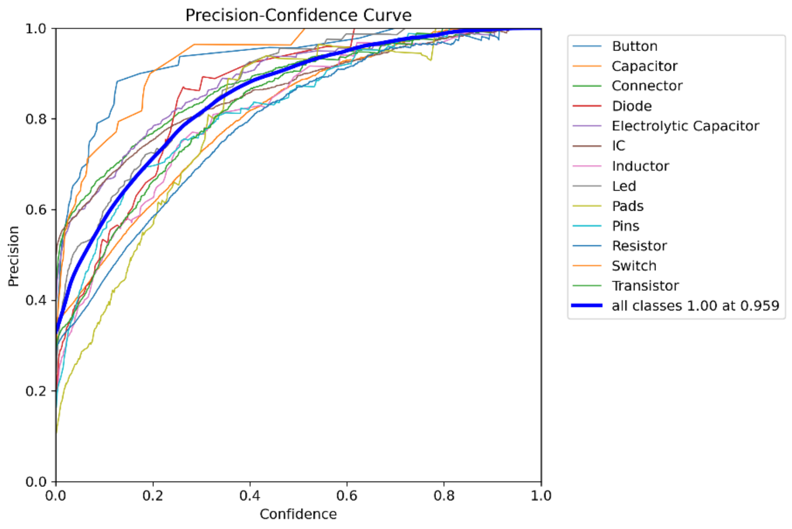

Figure 6: Precision-confidence curve

From figure 6, the model has a reasonable precision-confidence relationship. It's precision rapidly increases between 0 to 0.6 confidence, and then sees marginal increases from there. All classes demonstrate this overall trend.

The precision-confidence curve provides incomplete information without considering recall. Whereas precision indicates how many positive predictions are actually positive, recall indicates the true positive rate, meaning how many true positives are actually identified. This is critical, as a model can have a superficially high precision by only prediction positive cases it is very confident about, while ignoring the vast majority of actually positive cases. This is where the precision-recall curve can provide additional insight, by displaying this trade-off relationship. Figure 7 presents this relationship for this model.

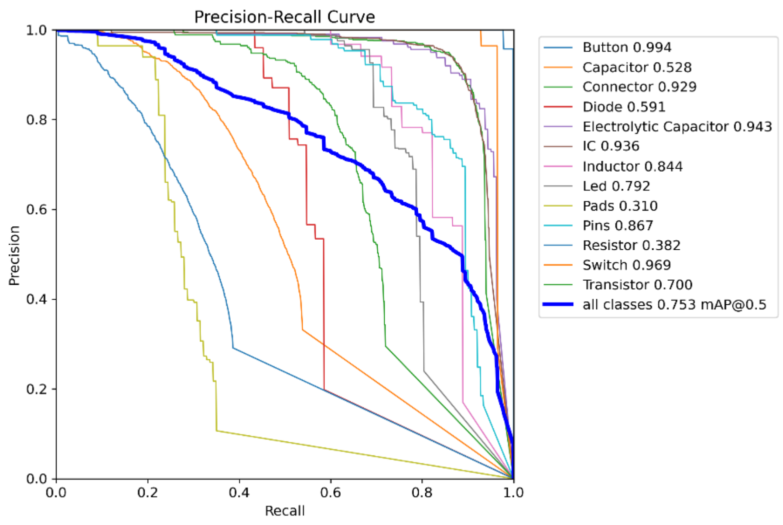

Figure 7: Precision-recall curve

Figure 7 demonstrates that this relation varies greatly for the different classes. This is consistent with the results observed in the normalized confusion matrix. For some classes, like pads and resistors, the recall is extremely low at high precision. This indicates that for the model to accurately predict these classes, it must do so very sparingly. This leads to the result seen in figure 5, where the model often misses these components and misclassifies them as part of the background. On the other hand, some classes like switches and connectors, show very little decrease in precision as recall is increased.

Model Evaluation

After training was complete, the model was used to classify components on 3 evaluation images. These results are presented next.

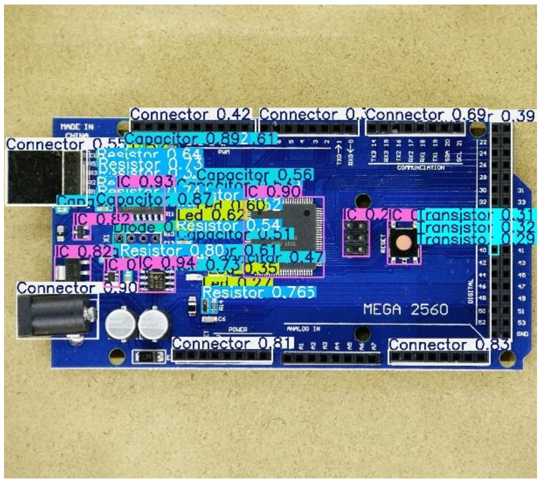

Figure 8: Ardmega evaluation results

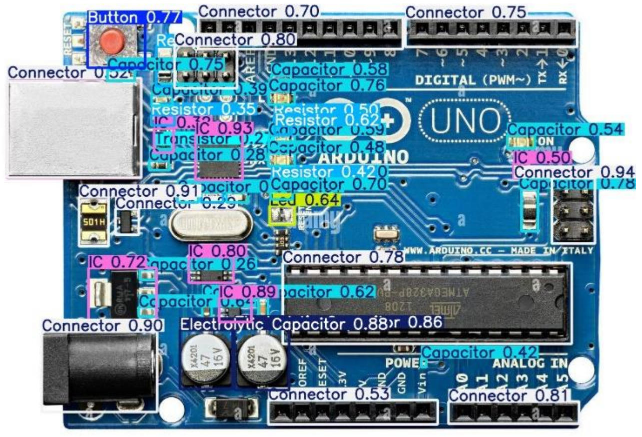

Figure 9: Arduno evaluation results

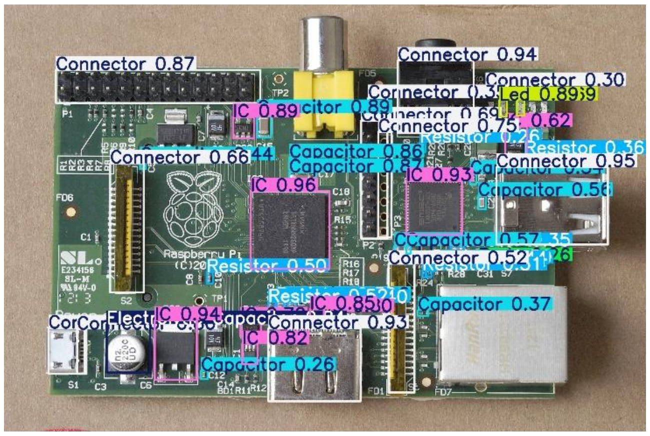

Figure 10: Rasppi evaluation results

Figures 8-10 confirm the results seen in the metrics analyzed earlier. The larger components such as connectors and buttons are labelled more frequently and with a greater degree of certainty than the smaller components. Another key result is the accuracy of the bounding boxes. When the model does make accurate predictions, the boxes it draws for the components line up really well with the placement of the components on the PCBs.

Conclusion

In conclusion, this project served as a great example of the utilization of computer vision for manufacturing applications. It first showed how computer vision tools could be used for processing images, making them more suitable for further analysis. It then used a YOLOv8 model to classify components on PCBs. Although a great first step, there were some issues with the approaches. In the first case, the mask generated for the PCB wasn't exact. It is recommended to improve the lighting conditions, angle, and background of the image, to create a more accurate mask. Secondly, the model trained to classify components struggled with smaller parts, often ignoring them and classifying them as part of the background. To improve this aspect of its performance, it is recommended to use higher-resolution images, to prove the model with more clues in order to correctly identify these smaller components.

GitHub Submission

https://github.com/CaptainAhad/Project-3