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	Reza Faieghi
Submission	Project
Submission No.	1
Submission Due Date	17/12/2023
Title	Image Classification Using YOLOv8
Submission Date	17/12/2023

Authors/Contributors:	Student Number (XXXX12345):	Signature*:
Shaurya Ganguly	XXXX15243	

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/

Object Masking:

The process starts by loading the motherboard image and converting it to grayscale. Grayscale images simplify edge detection by representing pixel intensities as shades of grey. Thresholding is applied to the grayscale image to segment it into regions based on pixel intensity; thresholding helps to emphasize the boundaries between the PCB and the background. Contours are identified using the cv2.findContours function which helps identify where the boundaries of objects are within the image. Significant changes in intensity, colour, or texture are highlighted, aiding in the detection of edges. To filter out small contours that may represent noise or irrelevant information, contours with an area below a specified contour threshold are dismissed. The filtered contours are drawn onto an empty mask and the resulting mask highlights the area occupied by the PCB (Figure 3). The cv2.bitwise_and operator extracts the PCB from the background, effectively masking out the unwanted regions.

Since the provided motherboard image had two separate backgrounds of vastly different colour (one dark noisy background and the clean bright desk that the motherboard is seated on) the process above had to be executed twice. The grayscale image was inverted for the second process in order for this strategy to function properly (Figures 1 and 2). In essence, the program first removes the dark background, then the light one, in order to isolate only the motherboard. There are some awkward spots on the right side and on top of the image due to the gradient of the shadows. This could have been solved by adjusting the thresholding parameters, which can improve the accuracy of pixel intensity segmentation, or by experimenting with different minimum contour area thresholds which can help filter out noise more effectively. These methods theoretically could work but they risk overriding the image of the motherboard as well since the circuit board itself is a dark black. The final extracted image using the method above is shown in Figure 4.

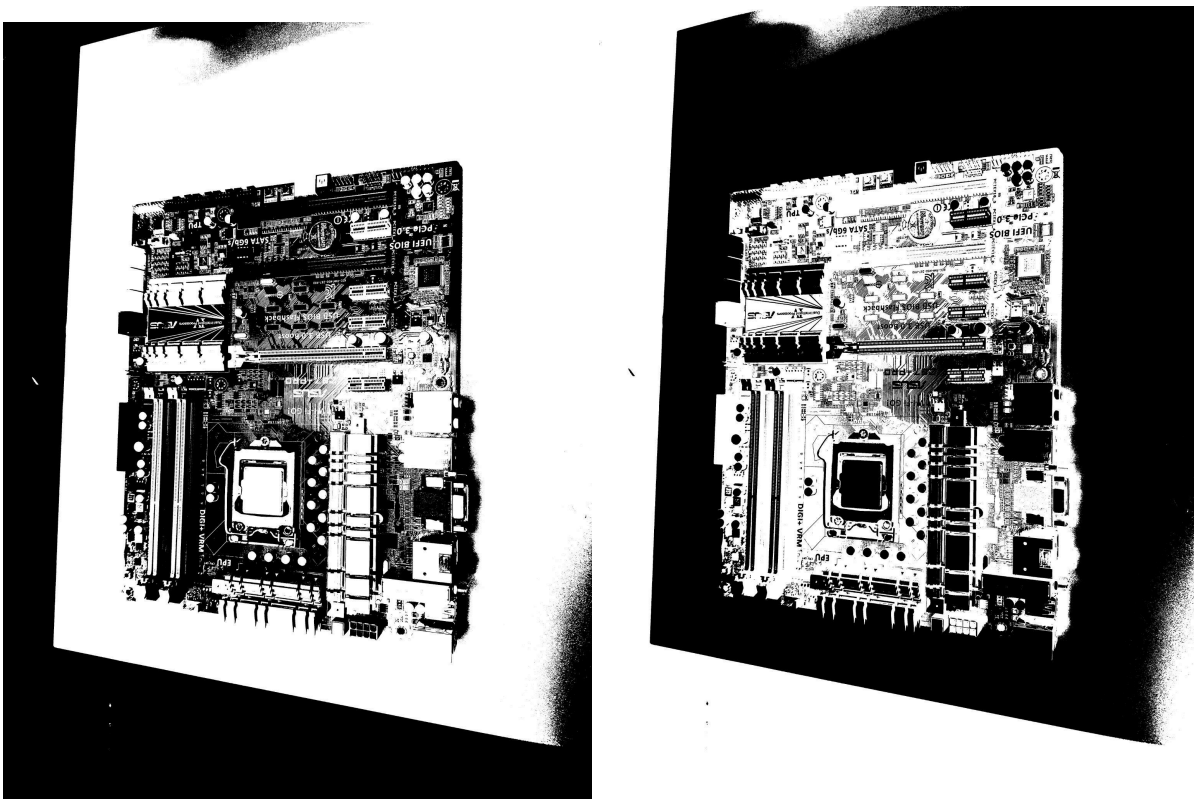

Figure 1 & 2 - Grayscale image and it’s inverted form used for edge detection

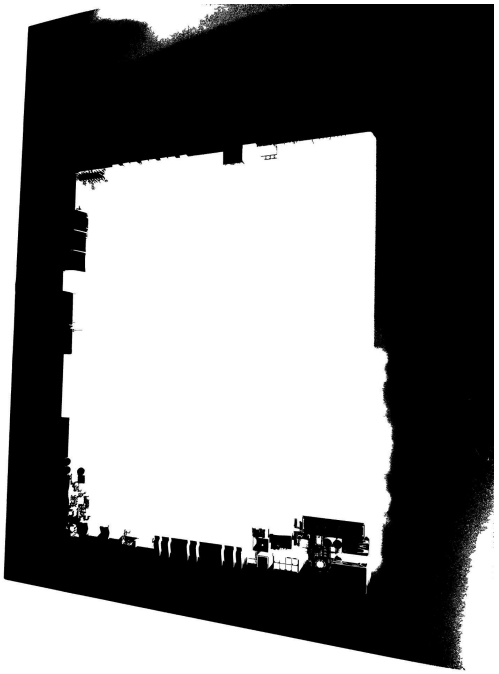

Figure 3 - Mask image

Figure 4 - Final extracted image

YOLOv8 Training and Evaluation:

The model was trained using the YOLOv8 Nano architecture as a base, and then fine-tuned to adapt it to the specific dataset provided. The model is pretrained on a large dataset and then further trained on the specific PCB dataset. Hyperparameters like learning rate, batch size, and number of epochs are tuned to achieve a balance between model convergence and avoiding overfitting. The batch size was kept low to 10 to prevent running out of memory since Google Colab only has 12GB of RAM available. The model was tested for 150 epochs in order to sufficiently train it on the dataset in a reasonable timeframe. The model was trained using the Ultralytics library, where images are input to the model, and the weights are updated based on the loss calculated between predicted and ground truth bounding boxes. The model is evaluated on a separate validation set to monitor its performance and prevent overfitting.

The model could be improved by utilizing data augmentation with transformations like rotation, flipping, and scaling. Generally using a larger dataset and fine-tuning hyperparameters like learning rate, batch size, and the number of epochs can also improve model performance.

The normalized confusion matrix provides a comprehensive view of classification performance. Rows represent ground truth classes, columns represent predicted classes, and each cell contains the percentage of instances. This shows that the model did not correctly identify nearly as many parts as it probably should have.

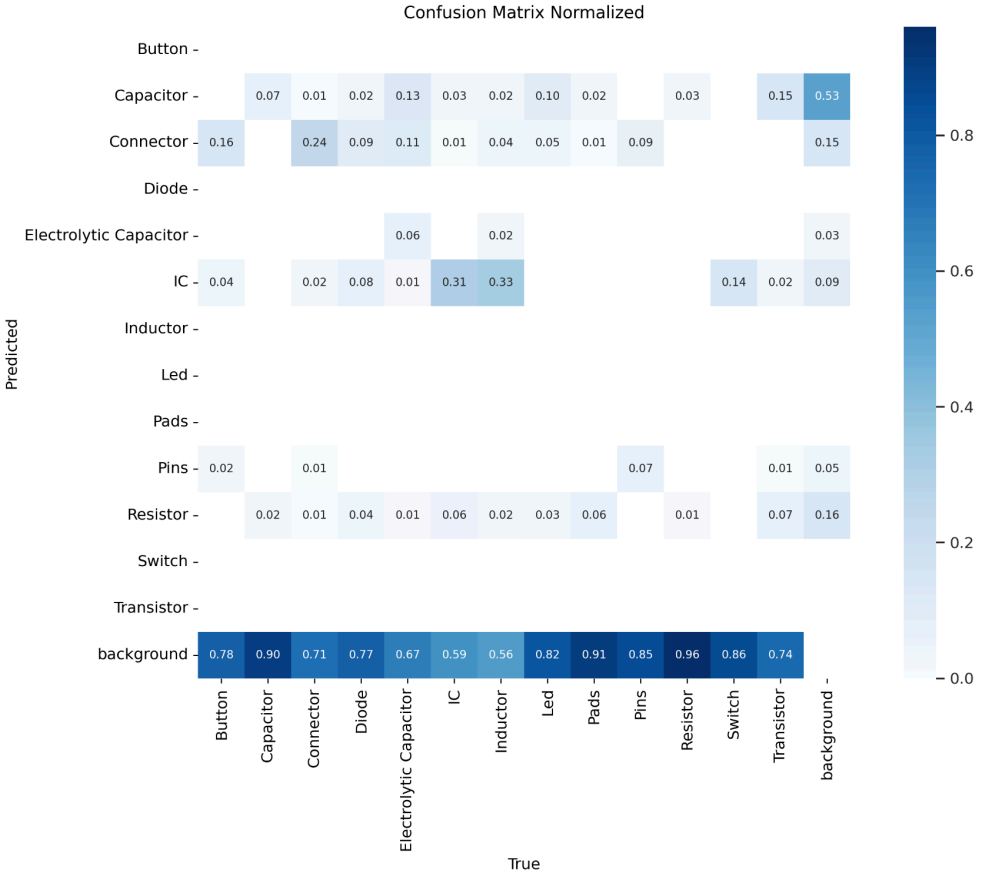

Figure 5 - Normalized confusion matrix

The precision-confidence curve shows how precision varies with confidence thresholds. It helps in choosing an appropriate threshold for deployment. This shows that as the model was more confident in identifying a part, it's precision generally increased as well (as illustrated by the blue line).

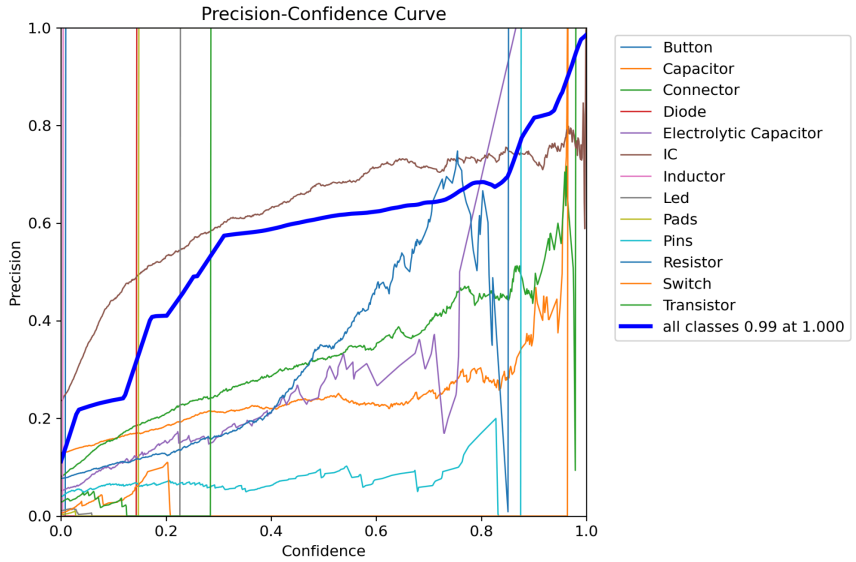

Figure 6 - Precision-Confidence Curve

The precision-recall curve illustrates the trade-off between precision and recall for different confidence thresholds. The area under the Precision-Recall curve is often used as a single-value metric to summarize the overall performance of the model across different thresholds - a higher area indicates better performance. This model trends downward which indicates low precision and low recall.

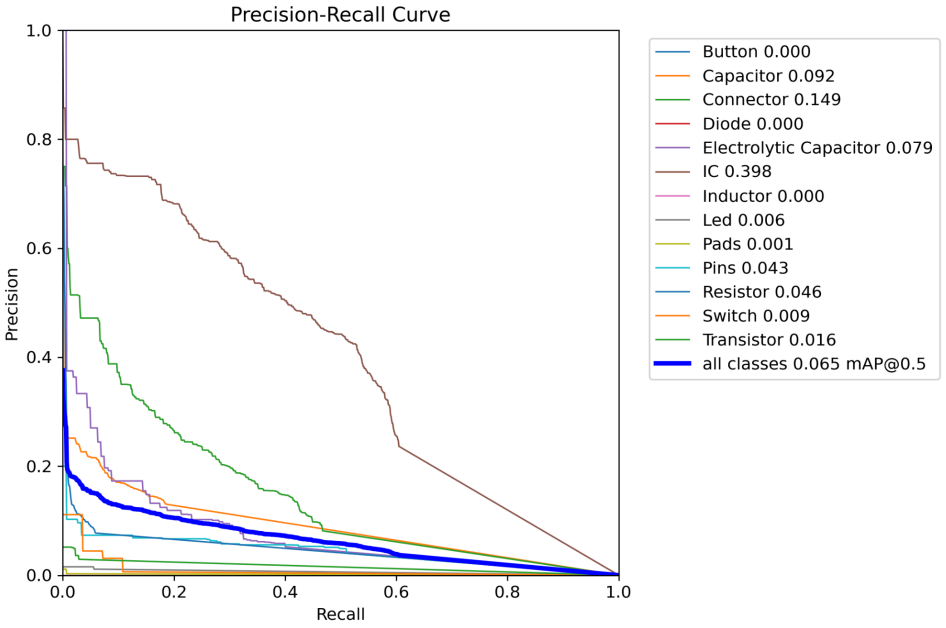

Figure 7 - Precision-Recall Curve

The following images are the results of the model evaluated on the test images. It is clear that many of the smaller parts like resistors are not detected. This could be due to a low number of epochs or an imbalanced dataset. There are not many false positives but plenty of false negatives (components that were missed):

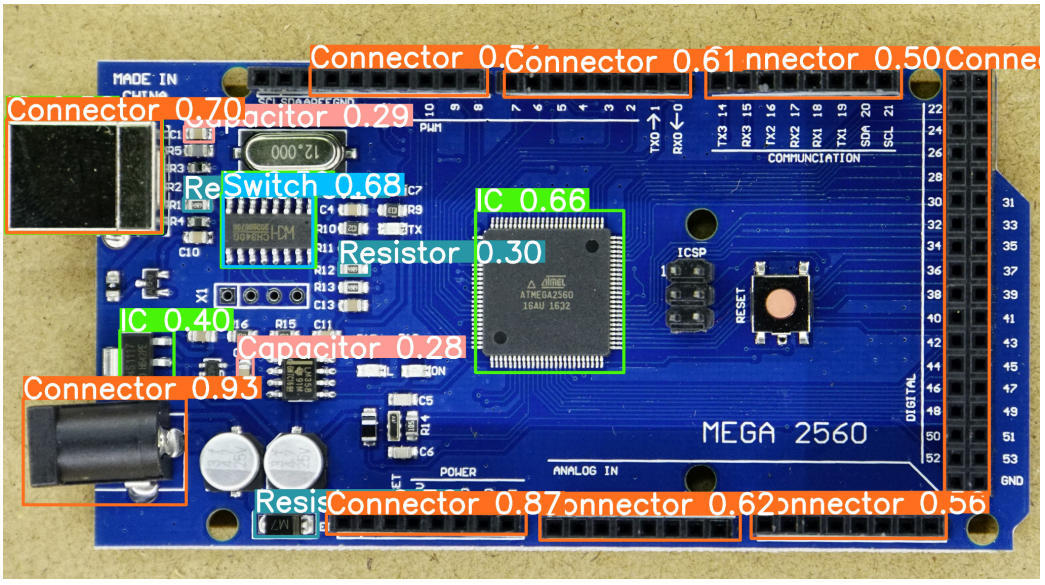

Figure 8 - Arduino Mega Board

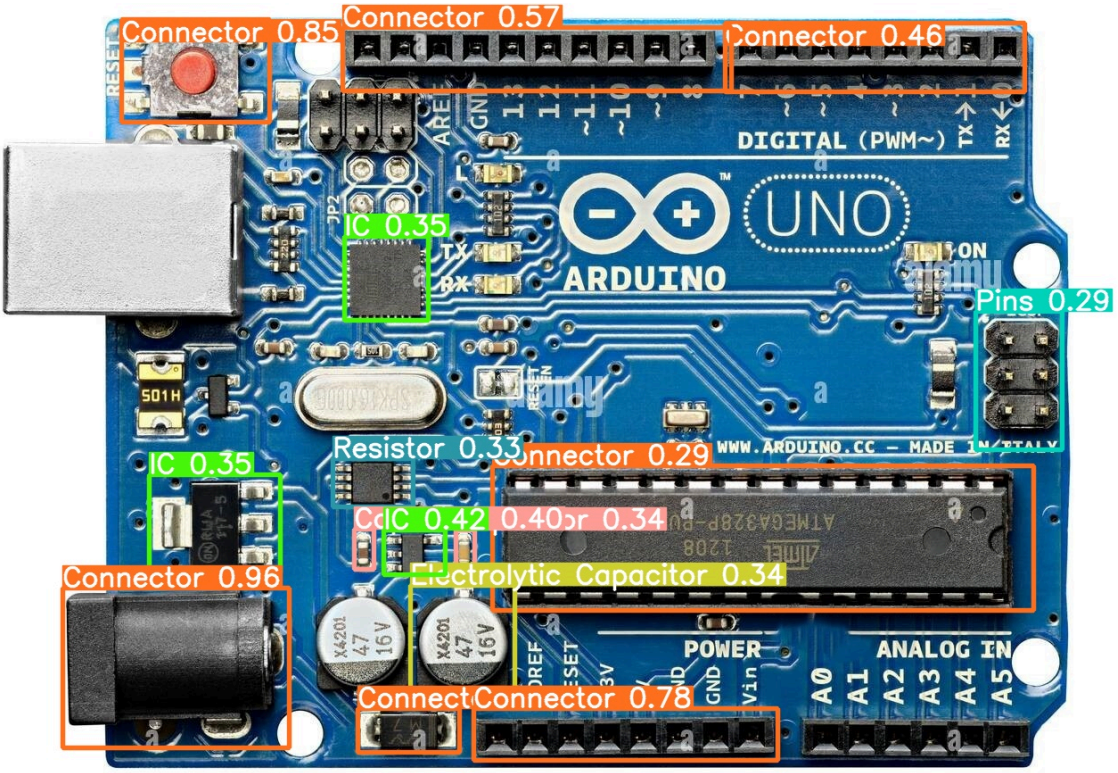

Figure 8 - Arduino Board

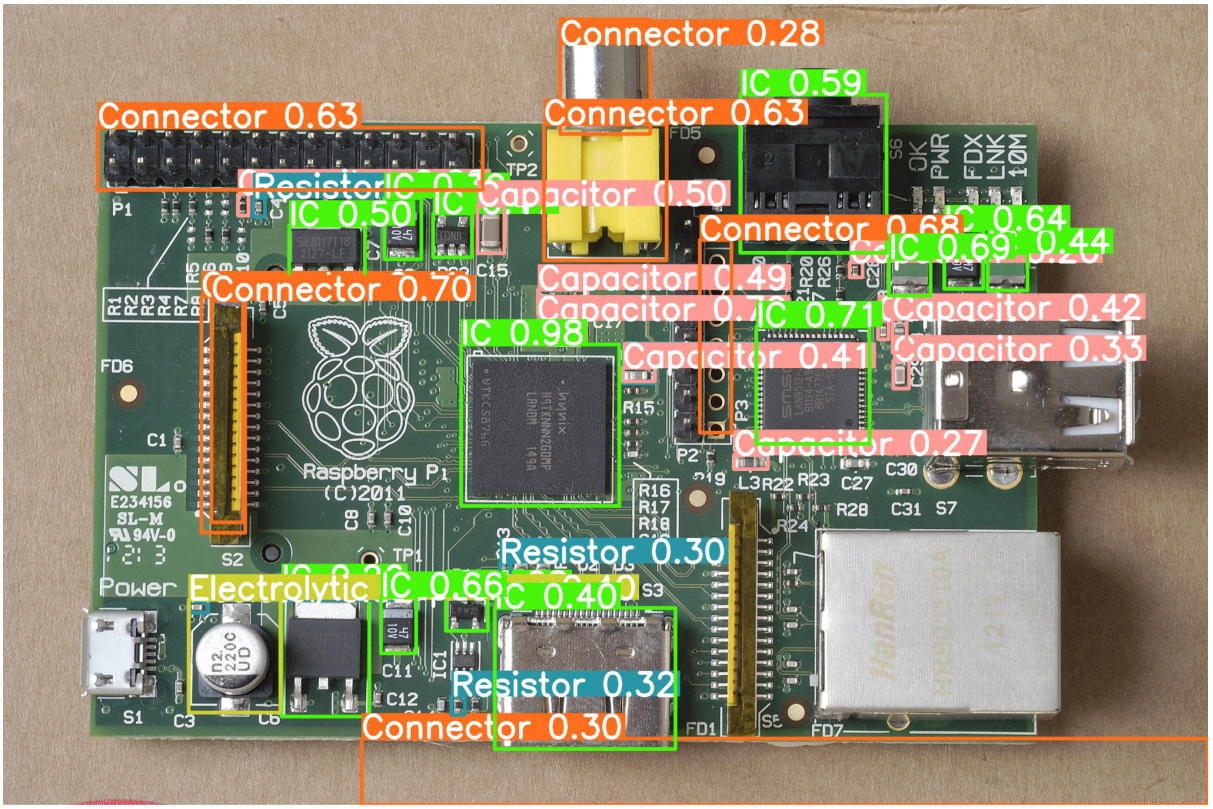

Figure 8 - Raspberry Pi

Github Link:

https://github.com/ShauryaSG/Project3

https://colab.research.google.com/drive/1kzNN4kesJ5glIuVM0fUTPK_WcODjn_Wo?usp=sharing