ECOD: EMBEDDING CLUSTERING FOR OBJECT DETECTION IN AUTOMATING PRINTED CIRCUIT BOARD ANALYSIS

Wayne William Loo Shyan[1], Shen Bingquan[2], Lim Qin Zhi Eddie[2]
[1]Raffles Institution, 1 Raffles Institution Lane, Singapore 575954
[2]DSO National Laboratories, 12 Science Park Drive, Singapore 118225

Abstract

Printed Circuit Board (PCB) analysis is crucial for detecting hardware trojans, a vital aspect of ensuring the security of critical systems. However, current methods are largely manual, making the process labour-intensive, time-consuming, and costly. A promising method for automating visual inspection involves accurate detection of PCB components using computer vision, but existing machine learning-based object detection models, such as YOLO (You Only Look Once), require substantial amounts of labeled training data to achieve high performance. Obtaining such data, however, is also tedious and expensive. This study investigates two approaches to address these challenges: (1) a novel pipeline that integrates clustering to enhance the classification accuracy of YOLO, and (2) a method combining a pre-trained SAM (Segment Anything Model) with clustering to prioritize high recall in low-data scenarios. Additionally, we compare the performance of these methods when trained on an even smaller dataset. To the best of our knowledge, clustering has not been widely applied in an object detection context. We aim for our findings to inspire further exploration of clustering techniques for data-efficient object detection.

Introduction

Hardware trojans (HT) in printed circuit boards (PCB) are modifications made for malicious intent, such as data leakage, limiting performance and denial of service [1]. Hence, HT detection is vital to ensuring security of critical systems such as in military or aerospace. Currently, methods for detecting HTs include in-circuit testing, functional testing, Joint Test Action Group (JTAG) boundary scanning and bare-board testing [2]. Among these is manual visual inspection, which involves inspecting the PCB for malicious modifications such as additional components or modified components. Comparing it against other board-level inspection methods, it is preferred as it is non-destructive, and less costly than actively testing each board. Nonetheless, it is still time-consuming, costly and prone to human error. Automating this process using computer vision would be invaluable as it would allow for testing at scale with far less human involvement [2].

In this study, we focus on contributing to the challenging task of automating PCB component detection with a small dataset of 32 images of PCBs as a constraint. Existing attempts include both machine-learning and traditional computer vision approaches, with varying amounts of data under different constraints, and to different levels of automation.

In [3], the authors present a method for detecting PCB components, combining traditional computer vision methods and machine learning methods. While using traditional computer vision processing methods allows for better explainability and reduces model parameters, and

requires less data, the authors highlight that machine learning methods achieve better performance. In comparison, on the dataset by [4] as cited by [3], YOLO, Faster-RCNN and Retinanet-50 achieve mAP of 0.698, 0.783 and 0.833 respectively. However, the PCB dataset in [4] contains 984 images of 123 unique PCBs, which is many times larger than ours, and hence does not follow the constraints of our study. In [2], the authors focus on comparing a 'golden PCB', which is a fully trusted PCB, to test images, using a siamese neural network. However, this method is limited as it requires every test image to be taken under similar, controlled conditions, hence could be rigid in practice. While the method achieves extremely high accuracy of 95.6%, their dataset is large and contains 7500 images. Finally, the method in [5] is key to our study, as the authors presented a novel pipeline for data-efficient detection on PCBs. The authors use a graph network block to refine the extracted features of proposed objects from a region proposal network (RPN), and conduct template matching by comparing feature similarity between detections. However, this method still relies on a human to label a few instances on each circuit board to use as templates, and hence is not fully automatic.

Therefore, we propose two novel pipelines to investigate alternative methods to achieve accurate detections.

Our first pipeline, YOLO-ResNet-Cluster, aims to leverage on YOLO's reasonably high performance acting as an RPN, while improving its classification accuracy. In comparison to [5], ours also uses the concept of similarity comparison between detected objects for data-efficient classification, but tackles the main limitation of the method proposed by incorporating clustering into the pipeline. By performing clustering on all images (train and predictions combined), we obtain class-cluster mappings through a majority vote.

The purpose of our second pipeline, SAM-ResNet-Cluster, is to explore the possibility of exploiting the ability of SAM to output high-quality masks of almost any object in the image. As the final purpose of this task is to visually detect trojans on a PCB, the final model must have high recall (i.e. have few false negatives). Hence, we employ SAM, and use clustering to filter for noise (i.e. masks not containing objects) to obtain predictions.

In addition to comparing their performances on the full train split, we also obtained results using an even smaller train dataset.

In summary, few studies have been conducted investigating methods for data-efficient detection of components on PCBs for trojan detection. To the best of our knowledge, clustering has also not been widely applied to an object detection pipeline, making our pipelines novel. Furthermore, we hope that our findings inspire further exploration of clustering techniques for data-efficient object detection.

Materials and Methods

Dataset and Training

For this study, we used two datasets for training the YOLOv8x and ResNet-50 models respectively, which were derived from a main dataset in the YOLO dataset format.

The main dataset consisted of high-resolution images of PCBs, each annotated with bounding boxes. It contained 32 images and 5 classes – capacitors, resistors, chips, inductors, and others. Due to the limited amount of data, we used 26 images for training and 6 for validation. To further test the data-efficiency of each method, we also arbitrarily selected 5 images for a smaller train set (which we refer to as Train-S).

To train the YOLOv8x model, we further processed the main dataset using a sliding window of half the image width and height, and a stride of half the window's dimensions. This was done to increase the performance of the model, especially on large PCBs with small components, while capturing large objects such as chips, headers and connectors. Therefore, the dataset contained duplicates of components in different crops of the overall PCB image. The class distribution of this dataset is detailed in Table 1.

We used YOLO's [6] official implementation and trained the model for 60 and 157 epochs on Train and Train-S respectively, keeping all other parameters their default values.

To fine-tune ResNet for feature extraction, we created a new dataset to train a pre-trained ResNet-50 model as a classifier. We derived this dataset from the main dataset by cropping out an image of each labelled object by its bounding box. This dataset, containing the main five classes, was used to fine-tune ResNet. Following this, we created another dataset to fine-tune ResNet[N] by automatically generating masks with SAM on the main dataset, cropping out the box around masks whose boxes did not match a ground-truth label, and saving them under the 'noise' class. The class distribution is detailed in Table 2.

For both ResNet and ResNet[N], on both Train and Train-S, we used the PyTorch [7] implementation of a pre-trained ResNet-50 model and trained it for 100 epochs with a learning rate of 0.0001. For SAM, we used the pre-trained vit-b model from Meta's Segment Anything [8] Github page, and used their official implementations. Masks were generated with the automatic mask generator, with the input parameters being 150 points per side, cropping 2 layers, a downscale factor of 2 for the grid of points, and a minimum mask area of 100. Other arguments were left as their default values.

Table 1: Class distribution of the YOLO dataset.

Split	PCB Images	Class Distribution				
		Capacitor	Resistor	Chip	Inductor	Other
Train	26	1921	2245	315	65	1087
Train-S	5	451	911	86	23	288
Val	6	1045	1093	141	29	177

Table 2: Class distribution of the dataset for ResNet

Split	Class Distribution					
	Capacitor	Resistor	Chip	Inductor	Other	Noise
Train	722	902	142	28	598	2418
Train-S	175	357	42	9	159	469
Val	380	418	66	15	92	-

Methods

We compare 3 methods for automatic object detection on PCBs. YOLOv8x (subsequently referred to as YOLO) acts as our baseline, as it is a popular end-to-end architecture that is known to achieve high performance on a variety of tasks.

With YOLO-ResNet-Cluster, the pipeline is shown in Figure 1, and is as follows:

1. We first obtain crops from the entire image using the same sliding window technique detailed above.
2. Then, we run YOLO's inference to obtain the most confident predicted boxes with a threshold of 0.25.
3. After processing the boxes using non-maximum suppression, we combine the predictions from each window by prioritising larger boxes for boxes that are near (threshold: 10 px) the edge of each window or overlap with each other.
4. Next, we feed the cropped image of each prediction into ResNet, which outputs a feature embedding of each input image.
5. Subsequently, we perform spectral clustering together with the embeddings of the training examples. Specifically, 80 clusters were used for the full dataset, and 60 clusters for the reduced dataset.
6. Finally, we determine the predicted class of the cluster through a majority vote. Thus, by clustering the predictions with training data, we can fully automate the inference.

With SAM-ResNet[N]-Cluster, the pipeline's structure is similar to that of YOLO-ResNet-Cluster. However, as SAM is not trained to detect objects, we instead use ResNet[N] and the clustering stage to select objects. Therefore, ResNet[N] is trained with a Noise class (hence, [N] indicates the additional Noise class), and objects labelled with this are filtered out from the predictions. The pipeline is shown in Figure 2.

Figure 1: The YOLO-ResNet-Cluster pipeline.

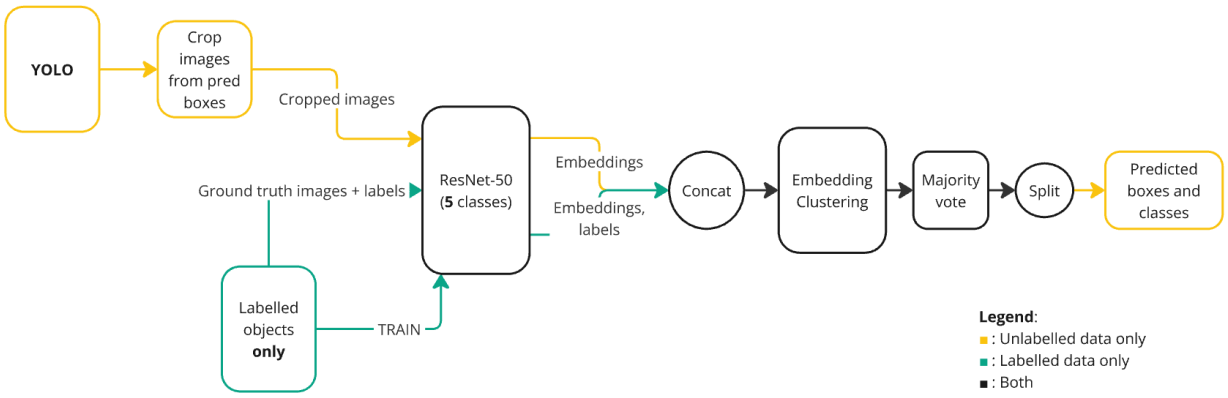
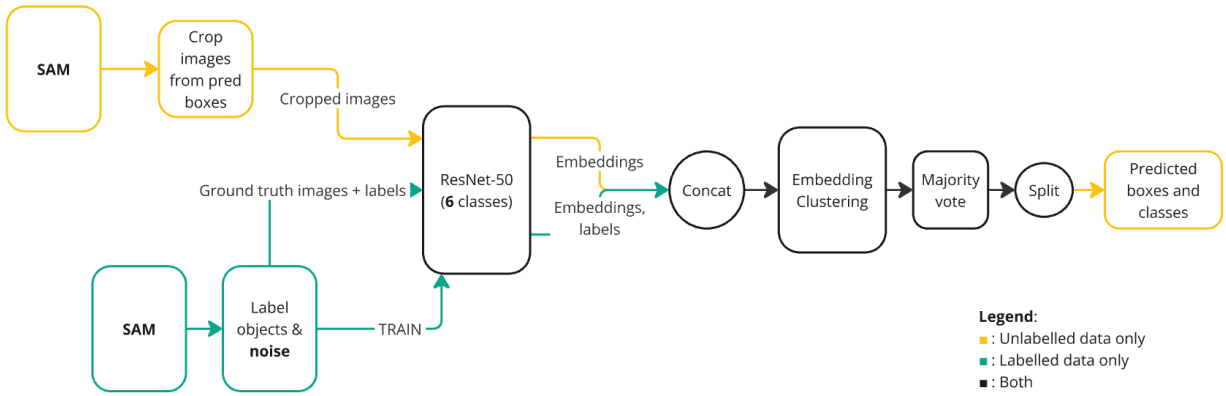


Figure 2: The SAM-ResNet[N]-Cluster pipeline.



Results

Table 3 shows the results from our study with the full and reduced datasets, and Figures 3 and 4 show qualitative results on one of the boards, which is an off-the-shelf PCB. To measure the performance of each method, we calculate precision, recall and F1 score for box predictions, using an intersection over union (IOU) threshold of 0.5, and Top-1 classification accuracy for measuring the ability of the model to classify each predicted box correctly.

Precision (P) refers to the number of true positive predictions (TP) over the sum of TPs and false positives (FP). Recall (R) refers to the number of TPs over the sum of TPs and false negatives (FN). F1 score is the harmonic mean of P and R. Top-1 classification accuracy is calculated by dividing the number of correct class predictions for TP boxes by the total number of TP boxes.

Table 3: Quantitative results from our study. Metrics used are precision (P), recall (R), F1, and top-1 classification accuracy (Cls Acc)

Method	**Full dataset (26 train)**				**Reduced dataset (5 train)**			
	P	**R**	**F1**	**Cls Acc**	**P**	**R**	**F1**	**Cls Acc**
YOLO	0.592	0.492	0.537	0.659	0.760	0.448	0.564	0.722
YOLO-ResNet-Cluster				0.847				0.825
SAM-ResNet[N]-Cluster	0.202	0.613	0.304	0.802	0.264	0.613	0.369	0.782

Figure 3: Qualitative results on one board for full training data

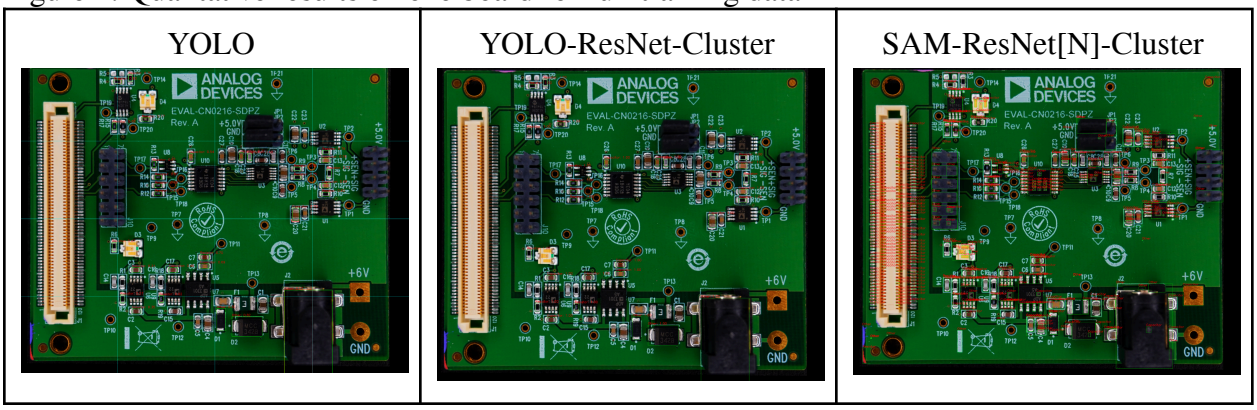
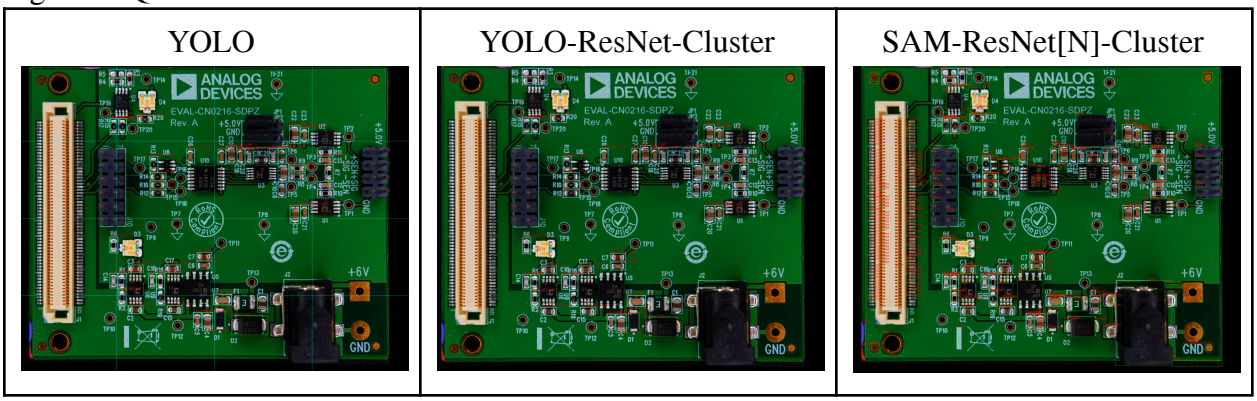


Figure 4: Qualitative results on board for reduced data



We see that our technique of clustering has improved the classification accuracy of YOLO by 18.8% on the full dataset, resulting in the best classification accuracy (and overall performance), and 10.3% on the reduced dataset. The recall resulting from using SAM is also 12.1% higher than that of YOLO on the full dataset and 16.5% on the reduced dataset. We also obtain high classification accuracy with SAM-ResNet[N]-Cluster.

Discussion

It is interesting to see that the evaluation metrics show a rough improvement in performance for areas not involving clustering when our models are trained on the reduced dataset. We mainly suspect that this is due to the fact that our reduced training dataset happens to contain images of circuit boards roughly representing each type of board used. That is, it contains boards with very large and very small components of the same class, and contains boards of each different background colour (green, black, red and blue). Alternatively, it could be that our training hyperparameters were incorrectly selected for the main train dataset, such as the number of epochs and learning rates, hence negatively affecting the resulting performance.

Overall, our pipeline YOLO-ResNet-Cluster performed better than the baseline YOLO. This result supports our hypothesis that clustering would improve the classification accuracy of YOLO on our small dataset. In SAM-ResNet[N]-Cluster, we also see how given only five training examples, the classification accuracy is also high. Hence, we see that the use of embedding clustering shows promising results in the application of data-efficient object detection.

While SAM-ResNet[N]-Cluster indeed gives higher recall, the extremely poor precision shows that even more tuning and processing must be done. Currently, the only post-processing step done by us (excluding the default mask post-processing done in SAM) is filtering out bounding box predictions whose dimensions are less than 2x2 px or whose area is greater than a tenth of the image. Following that, we depend on ResNet to learn the difference between objects and noise. Based on qualitative inspection of the visualised results, we have observed that it was largely able to filter out most text as noise. However, as shown in Figure 4, many small parts of the component on the left of the image were falsely classified. Apart from class imbalance and high intra-class variance in the 'Noise' class, we also suspect that this inaccuracy is due to how we crop and pad images which are fed in to ResNet[N], as currently images are cropped out from the bounding box, padded with 'reflect' padding, and resized to the same size (224x224). We believe that further improvements can be made to this method, as currently it does not capture the size of each detection, which can be extremely helpful in improving classification accuracy, and in the case of SAM-ResNet[N]-Cluster, the precision too.

Conclusion

In summary, this study proposes two novel pipelines for data-efficient detection of PCB components, YOLO-ResNet-Cluster and SAM-ResNet[N]-Cluster, which, while similar in overall flow, have different focuses. The former focuses on enhancing an existing model's performance on a small training dataset, while the latter focuses heavily on working with even less training data. Nonetheless, both seek to exploit the similarity in appearance between PCB components to enhance performance.

We hope that our proposition of using clustering in object detection leads to further research of its use for data-efficient methods, even beyond the application of automatic hardware trojan detection on PCBs.

Acknowledgements

I am grateful to Dr Shen Bingquan and Mr Eddie Lim for their guidance and insightful feedback during this project, and to DSO National Laboratories for this research opportunity.

References

[1] A. Kulkarni and C. Xu, “A Deep Learning Approach in Optical Inspection to Detect Hidden Hardware Trojans and Secure Cybersecurity in Electronics Manufacturing Supply Chains,” Frontiers in Mechanical Engineering, vol. 7, Jul. 2021, doi: https://doi.org/10.3389/fmech.2021.709924.

[2] G. Piliposyan and S. Khursheed, "Computer Vision for Hardware Trojan Detection on a PCB Using Siamese Neural Network," 2022 IEEE Physical Assurance and Inspection of Electronics (PAINE), Huntsville, AL, USA, 2022, pp. 1-7, doi: 10.1109/PAINE56030.2022.10014967

[3] W. Zhao, S. Gurudu, S. Taheri, S. Ghosh, Sathiaseelan, Mukhil Azhagan Mallaiyan, and N. Asadizanjani, “PCB Component Detection using Computer Vision for Hardware Assurance,” *arXiv.org*, 2022. https://arxiv.org/abs/2202.08452 (accessed Dec. 22, 2024).

[4] G. Mahalingam, K. M. Gay, and K. Ricanek, “PCB-METAL: A PCB Image Dataset for Advanced Computer Vision Machine Learning Component Analysis,” *IEEE Xplore*, May 01, 2019. https://ieeexplore.ieee.org/document/8757928?denied= (accessed Oct. 24, 2020).

[5] C.-W. Kuo, J. Ashmore, D. Huggins, and Z. Kira, “Data-Efficient Graph Embedding Learning for PCB Component Detection,” *arXiv.org*, 2018. https://arxiv.org/abs/1811.06994 (accessed Dec. 22, 2024).

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *arXiv.org*, Jun. 08, 2015. https://arxiv.org/abs/1506.02640

[7] A. Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” arXiv.org, 2019. https://arxiv.org/abs/1912.01703

[8] A. Kirillov et al., “Segment Anything,” arXiv (Cornell University), Apr. 2023, doi: https://doi.org/10.48550/arxiv.2304.02643.