Towards One-Shot PCB Defect Detection with YOLO

Gabriele Spadaro[1], Gaspare Vetrano[2], Barbara Penna[2], Antonio Serena[2] and Attilio Fiandrotti[1]

[1]*Universitá di Torino, Dipartimento di Informatica, Via Pessinetto 12, 10149 Torino, Italy*

[2]*Optical R&D Lab - SPEA S.p.A. - Automatic Test Equipment, Via Torino, 16, 10088 Volpiano, Italy*

Abstract

Consumer electronic devices such as smartphones, TV sets, etc. are designed around printed circuit boards (PCBs) with a large number of surface mounted components. The pick and place machine soldering these components on the PCB may pick the wrong component, may solder the component in the wrong position or fail to solder it at all. Therefore, Automated Optical Inspection (AOI) is essential to detect the above defects even prior to electric tests. In this context, we leverage YOLO, a deep convolutional architecture designed for one-shot object detection, for AOI of PCBs. This architecture enables real-time processing of large images and can be trained end-to-end. We report here our experimental setup and some preliminary performance measures.

Keywords

PCB, SMD, AOI object detection, defect detection, optical inspection, YOLO

1. Introduction

Consumer electronic devices such as mobiles, TV sets, etc are usually designed around one or more Printed Circuit Boards (PCBs) that are populated by automatic *pick and place* machine. Such robots may however fail in soldering a component in the right place, or the joints may be misaligned with respect to the pads, or the component may be installed in the wrong orientation, etc. Therefore, before an electric tests are performed, a visual inspection of the PCB is required to rule out glitches like those described above. [1]

The large amount of components typically populating a modern PCB calls for highly automated, high throughput, optical inspection procedures. Traditional PCB optical testing procedures require a camera to be driven over each component according to the PCB schematics, taking a shot of the component, analyze the shot and repeat for each component. The throughput of this workflow (i.e. the number of PCBs processed over time) is limited by i) the momentum of the moving part (the camera or the PCB) during image acquisition ii) the speed of the computer vision algorithm processing the image.

This work proposes a one-shot optical inspection method for fault detection in PCBs. In detail, our goal is to detect from a single high resolution shot of an entire PCB i) missing components and ii) misplaced or wrong component types. We recasted our fault detection task into an object (the soldered components) detection and classification problem. We rely on a YOLO (You Only Look Once), a deep neural network based one-shot object detector suitable for low latency operations and trainable end-to-end. YOLO requires however large amounts of labeled data, so the first challenge we undertook was creating a sufficiently large training set of annotated whole PCB images from single component crops such as that in Figure 1. To obtain a robust component detector, we collected training images from different PCBs under different illumination conditions. We experimented with a leave-one-out approach where one board was in turn kept out of the training set and used for evaluation. Our preliminary experimental results show good component recognition accuracy especially for very popular classes of components, with headroom for further improvement as the training set grows larger.

2. Background

Deep convolutional object detection models can be grouped into two classes [1]: two-stage detectors such as the R-CNN family of architectures [2, 3, 4] and one-stage detectors.

YOLO (You Only Look Once) [5] is a one-stage detector relying on a deep convolutional feature extractor that is able to predict multiple bounding boxes and class probabilities for those boxes directly from full images in one evaluation. Using the whole image to make predictions, YOLO implicitly encodes contextual information about classes as well as their appearance. This allows it to learn

Ital-IA 2023: 3rd National Conference on Artificial Intelligence, organized by CINI, May 29–31, 2023, Pisa, Italy

*Corresponding author.
gabriele.spadaro@unito.it (G. Spadaro); gaspare.vetrano@spea.com (G. Vetrano); barbara.penna@spea.com (B. Penna); antonio.serena@spea.com (A. Serena); attilio.fiandrotti@unito.it (A. Fiandrotti)
0000-XXXX-XXXX-XXXX (G. Spadaro); 0000-XXXX-XXXX-XXXX (G. Vetrano); 0000-XXXX-XXXX-XXXX (B. Penna); 0000-XXXX-XXXX-XXXX (A. Serena); 0000-0002-9991-6822 (A. Fiandrotti)

[1]This work is the result of the collaboration with SPEA, a worldwide leader in electronic component testing solutions that made available the images sued in this work.

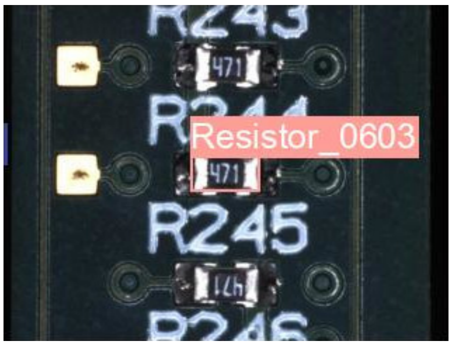

Figure 1: Example of the original 1278x958 training images we were provided: the component at the center of the image is annotated with type (Resistor) and case size (6x3 millimiters).

generalizable representations of objects.

In order to simultaneously predict bounding boxes and class probabilities, YOLO divides the input image in a grid of $S \times S$ cells, where each cell is responsible for the prediction of the object whose center falls within the cell. Each grid cell predicts *B* bounding boxes, and each bounding box consists of 5 predictions: *x*, *y*, *w*, *h*, and *confidence*. The coordinates of the box are represented by the predicted values of (*x*, *y*) for the center of the box, which is relative to the bounds of the grid cell. On the other hand, the width (*w*) and height (*h*), are predicted relative to the whole image. Instead, the confidence is a score described formally in the equation 1. This value is intended to reflect how confident the model is about the presence of an object in that box and how accurate it thinks its prediction is. It is easy to see that if no object exists in that cell, this confidence value has to be equal to zero. Otherwise it should be equal to the intersection over union between the ground truth and the predicted box.

$$Pr(Object) * IoU_{pred}^{truth} \tag{1}$$

Each grid cell also predicts *C* conditional class probabilities $Pr(Class_i|Object)$, where the condition is inherent to the presence of the object in that cell.

Dividing the image into a *S*x*S* grid and considering *B* bounding boxes for each grid and *C* class probabilities, the predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

In this work we rely on the fifth revision of YOLO architecture, known as YOLOv5. The code of this model is publicy available on the official Ultralytics repository on GitHub [6] in which YOLOv4 [7] is implemented using PyTorch. The primary distinction of this architecture, as opposed to the one discussed in the previous section, is that it employs anchor boxes instead of predicting bounding box coordinates directly from the image [8]. Anchor boxes are bounding boxes with predetermined dimensions, known beforehand. These boxes are found running a k-means clustering algorithm on training set bounding boxes to automatically find good priors. This is because the network learns how to adjust, and so starting from significant priors makes the learning process easier. Moreover, this architecture consists of a **backbone**, a **neck**, and a **head**. The backbone has the role of extracting relevant features from the input image. The combination of backbone feature layers happens in the neck. This is because the feature layers of the convolutional backbone have to be mixed and taken into account with each other. The head, on the other hand, is where detection happens. In this case we have three detection modules, in order to predict boxes at three different scales using the heads of YOLOv3 [9]

3. Methodology

This section first describes the procedure we devised to generate the annotated PCB images required to train a YOLOv5. Next we discuss the procedure to train the network and the relative performance metrics.

3.1. Dataset Generation

Towards training YOLO, all objects in each training image must be associated with a bounding box and class identifier. Figure 1 shows an example of the training images we were provided,a few thousand pictures captured from 11 different PCBs. Each image was acquired by a floating camera that would center on each component according to the CAD schematics and take a high resolution picture of thecomponet and its surroundings. For each image, only the position of the central component in absolute PCB coordinates was however known, i.e. no class label was provided. So, we parsed the CAD schematics extracting for each component the class (type and case size) from which we inferred the component bounding box.

At this point we had images where multiple components were present yet only the central component was annotated. Training YOLO on such images, our preliminary experiments showed, resulted in a large number of false negative detections at test time, i.e. verylow recall. That outcome was predictable, since only the central component was annotated whereas the other components would be presented as background to YOLO at training time.

To overcome this issue, we developed a workaround for generating whole images of PCBs where each component would be annotated. We recall that each image was acquired centering the camera onto one of the PCB components, i.e. such images represent the tiles of a jig-

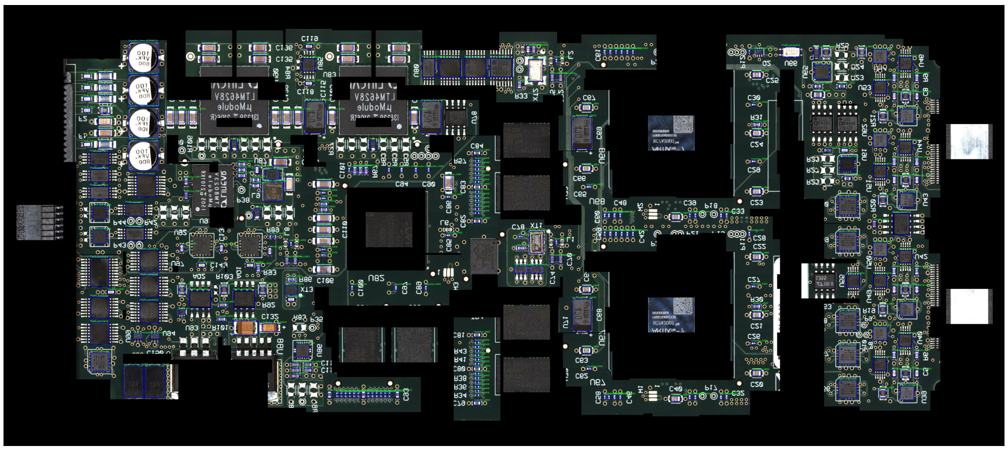

Figure 2: 60000x20000 pixels image of the CPE010 PCB reconstructed and annotated using 354 crops of single components. Black areas are due to the lack of components in those areas.

saw puzzle of a PCB. Our strategy consists in populating an empty canvas with the available images placed at the position of the central component and reconstructing the quasi-complete PCB images shown in Figure 2 (the black holes are due to the lack of components, and hence training images, in some of its parts). While solving this jigsaw puzzle, we also recomposed the previously generated annotation from each image to the reconstructed image. Following this procedure we successfully reconstructed and annotated a total of 11 PCBs where each component is annotated with a label and a bounding box, as required to properly train YOLO.

Table 1 shows the 36 classes of components considered in this work. Wepoint out that, give our specific application, the component case size has been incorporated into the class label. Therefore, for a given component (e.g.,resistor) different classes exist based on the case size (e.g. Resistor_0402, Resistor_0603 etc..). It is important to note that all components within the same class have identical dimensions, except for some classes, which were created by grouping different components, such as the class Plastic_packaging.

3.2. Training Procedure and Evaluation Metrics

YOLO is trained end-to-end minimizing the following multi-part loss function

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \tag{2}$$

where the three terms represent respectively the Binary Cross Entropy (BCE) loss related to the Classes, the Binary Cross Entropy (BCE) loss related to the Objectness, and the Complete-IoU (CIoU) loss for the Location. The objectness losses of the three prediction layers are also weighted differently, as reported in Equation 3.

$$L_{obj} = 4.0\dot{L}_{obj}^{small} + 1.0\dot{L}_{obj}^{medium} + 0.4\dot{L}_{obj}^{large} \tag{3}$$

We trained for 100 epochs a YOLO of the medium size that was was pretrained on MS the COCO dataset using the default weights of the loss terms. We used the hyper-parameters recommended for YOLO, i.e. a learning rate of 0.01 and a momentum of 0.937. Moreover, we considered 3 warmup epochs having a momentum of 0.8 and a bias learning rate of 0.1. In particular we used the SGD as optimizer and a batch-size of 32 crops over the annotated patches described above. We maintained the IoU threshold of 0.2 recommended by default for the object fusion step.

Due to complexity considerations, YOLO was trained on 5.490 patches sized 1278x958 pixels extracted from the reconstructed PCB images. Moreover, we augmented our training set of patches with random rotations, flips and a random variation of the brightness since these transformations preserve the size and aspect ratio of the components.

4. Experimental Results

This section reports some preliminary results we obtained so far using a leave-one-out approach where one board is left out of the training set and is used for testing. Namely, we trained 11 distinct instances of YOLO where in turn one PCB was left out of the training set.

Table 2 shows the mean average precision (mAP) for the board that was kept in turn out of the training set.

Performance varies significantly from board to board and our investigations showed correlation with the ratio

Table 1
Number of samples and area of components for each class over the 11 reconstructed images.

name	elements	μm^2
Resistor_0402	511	756,81
Resistor_0603	967	1.884,59
Resistor_0805	472	3.885,82
Resistor_1206	47	7.076,86
Resistor_1210	2	6.584,29
Resistor_RMINIMELF	3	7.698,42
Resistor Array	92	7.940,38
Resistor_2010	9	18.547,78
Resistor_2512	20	30.649,76
Capacitor_0402	958	794,03
Capacitor_0603	886	1.710,48
Capacitor_0805	404	3.155,46
Capacitor_1206	93	6.296,04
Capacitor_1210	39	13.096,62
Capacitor_Polar_0603	13	3.990,02
Capacitor_Polar_CMKTA	20	8.554,51
Capacitor_Polar_1411P	3	16.262,89
Capacitor_Polar_CMKTB	1	29.971,18
Capacitor_Polar_CMKTD	20	48.392,77
Capacitor_Polar_CEVPA8X10	4	69.504,67
Inductor_1210	4	13.988,71
Inductor_IND-XAL4020	4	27.742,08
Inductor_INDIHLP2525CZ01	4	67.996,82
Fuse_0603	8	2.121,57
Fuse_FUSESM	6	21.973,29
Fuse_FUSE-SMDC020	2	24.644,09
Led_0805	56	4.483,81
Led_TEKTONE_LED_1411	4	13.329,49
Connector_CMIMA4VFD_SM	2	59.097,38
Connector_CMIMA6VFD	2	76.665,90
Potentiometer_SMRVAR1	1	33.786,26
Relay_RLPICK-117-1A	52	42.563,62
Switch Array_PULSOMRON	1	56.310,86
Diode_DMELF	2	18.398,49
Cylindrical_diode	71	7.481,30 - 7.538,37
Metallic_packaging	6	23.934,04 - 52.777,34
Plastic_packaging	706	878,41 - 70.537,68

of small components on the board, small objects being harder to detect. In addition, the distribution of components at training time varies greatly among PCBs, with some classes being under-represented at training time.

In Figure 3 are reported the precision-recall curves of each component for the CPE010 board. We notice that classes with lowest scores are those less represented in the train set or with small case size. Classes Capacitor_Polar_CEVPA8X10 and Capacitor_Polar_CMKTB are a corner case further, as they are present only in the test board, and so YOLO is not able to predict something that it has not see in training. However, since the model predicts these components as Capacitor_Polar_CMKTA, this affects the precision of this class, as we can see in Figure 5.

For Metallic_packaging and Capacitor_Polar_CMKTD classes, we are in a similar situation where there are few instances per class in the dataset and about 1 out of 3 of these components are in the test board. In particular,

Table 2
mAP@0.5 for the board left out of the training set. All board images are reconstructed from patches but $*$.

Test set	mAP@0.5
CPE010	0.775
JPAMA30-256K SN	0.704
KDBRLYCMDR3	0.850
KEXANADUX70V1	0.952
LI122SM-2_CB533_009	0.819
MPSDRV608	0.882
SPE010-2	0.994
Z010500 SN	0.524
ZCPU7Z0	0.787
ZPROMEA50_SN_02680	0.783
ZPROMEA50_SN_01115	0.812
Mean	**0.808**
ZPROMEA50_SN_01115*	0.881

Capacitor_Polar_CMKTD is generally confused with the background or with the class Plastic_packaging, while the Metallic_packaging is misclassified as background.

YOLO also struggles to distinguish the background from small components, resulting in instances of the background misclassified as Resistor_0402. This leads to the fact that in Figure 5 we observe a low precision value for this class.

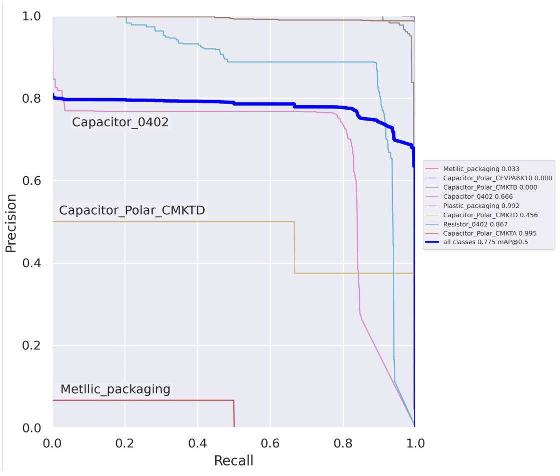

Figure 3: Precision-Recall curve of the CPE010 board

5. Conclusion

In this work we explored the possibility of using a YOLO object detector for the task of automatically detecting PCB components soldering defects. The first challenge we had to undertake was creating a training dataset of

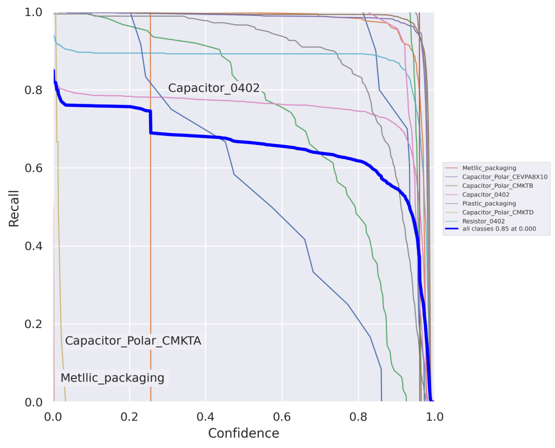

Figure 4: Recall curve of the CPE010 board

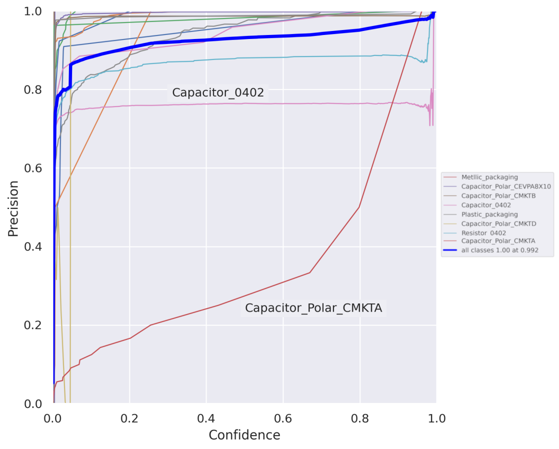

Figure 5: Precision curve of the CPE010 board

completely annotated whole PCB images we recomposed from thousands of unannotated single component images. Our preliminary experiments show performance seems to depend on component size and number of training samples available per class. Our future research includes enlarging the training set for underrepresented classes at training time and experimenting on whole PCB images rather than on the reconstructed images we had available at themoment of the writing of this article.

References

[1] Z. Zou, Z. Shi, Y. Guo, J. Ye, Object detection in 20 years: A survey, CoRR abs/1905.05055 (2019). URL: http://arxiv.org/abs/1905.05055. arXiv:1905.05055.

[2] R. B. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, CoRR abs/1311.2524 (2013). URL: http://arxiv.org/abs/1311.2524. arXiv:1311.2524.

[3] R. B. Girshick, Fast R-CNN, CoRR abs/1504.08083 (2015). URL: http://arxiv.org/abs/1504.08083. arXiv:1504.08083.

[4] S. Ren, K. He, R. B. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, CoRR abs/1506.01497 (2015). URL: http://arxiv.org/abs/1506.01497. arXiv:1506.01497.

[5] J. Redmon, S. K. Divvala, R. B. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, CoRR abs/1506.02640 (2015). URL: http://arxiv.org/abs/1506.02640. arXiv:1506.02640.

[6] G. Jocher, YOLOv5 by Ultralytics, 2020. URL: https://github.com/ultralytics/yolov5. doi:10.5281/zenodo.3908559.

[7] A. Bochkovskiy, C. Wang, H. M. Liao, Yolov4: Optimal speed and accuracy of object detection, CoRR abs/2004.10934 (2020). URL: https://arxiv.org/abs/2004.10934. arXiv:2004.10934.

[8] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, CoRR abs/1612.08242 (2016). URL: http://arxiv.org/abs/1612.08242. arXiv:1612.08242.

[9] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, CoRR abs/1804.02767 (2018). URL: http://arxiv.org/abs/1804.02767. arXiv:1804.02767.