International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 06 | Jun 2022 www.irjet.net p-ISSN: 2395-0072

Automated Defect Classifier for PCBs using Raspberry Pi

G P V Kishore[1], R L N N Sai Abhijeet[2], B Likith[3], Sushmitha[4]

[1]Associate Professor, ECE Department, ACE Engineering College, Ghatkesar, Telangana, India.
[234]Student, ECE Department, ACE Engineering College, Ghatkesar, Telangana, India.

---***---

Abstract - *Printed circuit boards (PCBs) are an important component in the production of electronic products. A minor defect on the PCB can cause severe problems in the finished product, making PCB surface inspection one of the most crucial quality control tasks [1]. Because of the limits of manual examination, major efforts have been made to automate the process using high resolution CCD or CMOS sensors. Setting pass/fail criteria based on small failure samples has always been difficult in traditional machine vision systems, despite advanced sensor technology [2]. As a result, this paper proposes a deep learning method based on the you-only-look-once (YOLO) concept. A network of 24 convolutional layers and two fully linked layers make up the suggested approach. The model has been trained with 10,480 photos of diverse defects, including shorts, spurs, and mouse bites. In comparison to state-of-the-art works, the experimental results show that our approach detects defective PCBs with high accuracy (95%).*

Key Words: Convolutional Neural Network, YOLO, Raspberry Pi, PCB

1.INTRODUCTION

The Printed Circuit Board (PCB) is in most electronic products, mechanically supporting and connecting components along conductive tracks. Their prevalence underlies the modern electronics industry, with a world market exceeding $60 billion. PCBs are prone to a variety of defects that impede proper manufacturing, costing companies money. Defects such as shorts, spurs, mouse bites, and pinholes cause issues like current leakage and open circuits, quickly degrading performance or rendering PCBs useless. In this project we will build a full PCB Defect Classifier that automates the task of detecting and classifying defects in printed circuit boards. We will build a complete module that not only detects any defects in the given circuit board but also will segregate depending on the results. To overcome this problem, we propose an automated model that segregates the good PCBs from the defected ones. We use yolo model to detect the defects using a Raspberry Pi and Pi camera, which whose results will be transferred to an Arduino. Based on the results the PCB is either moved forward or is pushed away. The model has over 95% accuracy as this is trained with a huge data set - approximately 10, 680 images that has been divided into 80% testing images and 20% valuation images. This model does not require any input image or any other extra sources as many models out there need. This model is trained to detect the defects directly. We use YOLO – You only look once model to detect the defects.

2. METHODOLOGY

2.1 Dataset

2.1.1 Image Collection

Each image in this dataset was created using a linear scan CCD with a resolution of about 48 pixels per millimeter. The defect-free template images are manually examined and cleaned from sampling photos. The templates and the image that was tested original dimensions are roughly 16k by 16k pixels. Then, using template matching techniques, they are cropped into several 640 × 640 sub-images and aligned. The next step is to carefully choose a threshold to use binarization in order to prevent lighting disturbance. However, image registration and thresholding techniques are a common approach for high-accuracy PCB defect localization and classification. The following figure shows a sample pair from the dataset, with Figure 2 representing the defect-free template image and Figure 1 representing the defective tested image with the ground truth annotations.

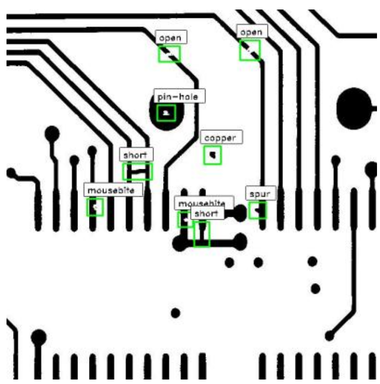

Fig -1: Tested Image

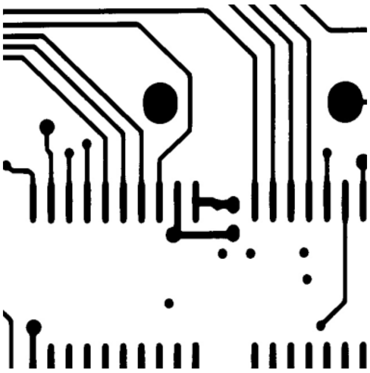

Fig -2: Template Image

2.1.2 Image Annotation

An axis-aligned bounding box with a class ID is used for each defect in the analyzed images. The figure above (Figure 1) highlights six common PCB flaws: open, short, mouse bite, spur, pin hole, and spurious copper. Since the tested image only has a few minor defects, 3 to 12 fictitious faults are added to each 640 x 640 image by adding artificial defects in accordance with PCB defect patterns to each tested image. The number of PCB faults is depicted in the following figure (Figure 3). The remaining 1,000 images are divided into a training set and a test set.

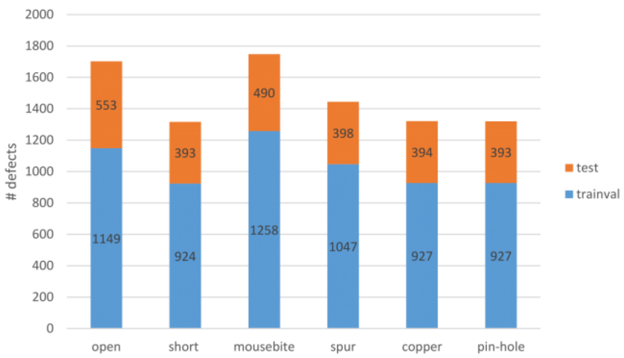

Fig -3: PCB Defects

2.2 YOLO – v5

YOLO is a regression-based target detection system. YOLO completes the prediction of the categorization and position information of the objects according to the computation of the loss function after receiving the photos or video into the deep network, transforming the target detection problem into a regression problem solution [3].

Based on the YOLO detection architecture, YOLOv5 [4] employs some of the best algorithm optimization techniques developed recently in the field of convolutional neural networks, including auto learning bounding box anchors, mosaic data augmentation, the cross-stage partial network, and others. These techniques are responsible for various tasks in various parts of the YOLOv5 architecture. The input, backbone, neck, and output are the four major components of the YOLOv5 architecture. The preparation of the data, including mosaic data augmentation [5] and adaptive image filling, is primarily done at the input terminal. The adaptive anchor frame computation that YOLOv5 includes on the input allows it to automatically set the starting anchor frame size when the dataset changes, allowing it to adapt to various datasets.

In order to extract feature maps of various sizes from the input picture via multiple convolution and pooling, the backbone network mostly employs a cross-stage partial network (CSP) [6] and spatial pyramid pooling (SPP) [7]. While the SPP structure accomplishes the feature extraction from several scales for the same feature map and is capable of generating three-scale feature maps, which helps enhance the detection accuracy, BottleneckCSP is used to reduce the amount of calculation and speed up inference.

The FPN and PAN feature pyramid architectures are deployed as the neck network. Strong semantic features from the top feature maps are transmitted to the lower feature maps through the FPN [8] structure. The PAN [9] structure simultaneously transfers potent localization features from lower feature maps to higher feature maps. Together, these two structures reinforce the features obtained from various network levels in the backbone fusion process, which enhances the detection capabilities.

The head output's primary application is to forecast targets of various sizes on feature maps as the last detecting phase. The four architectures that make up the YOLOv5 are called YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The quantity of feature extraction modules and convolution kernels at particular points on the network is where they diverge most from one another. Figure 4 depicts the YOLOv5 network structure.

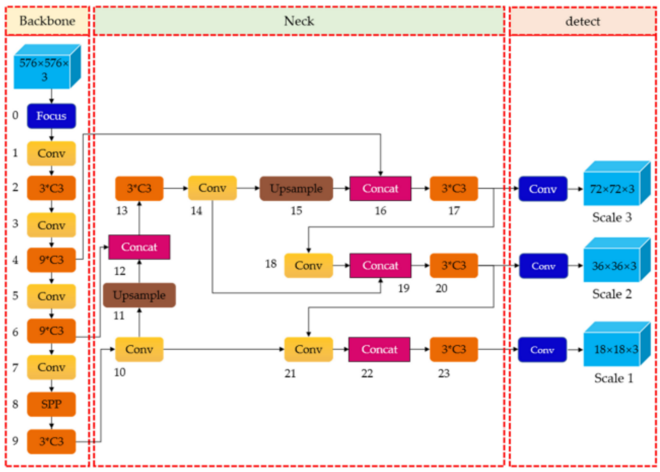

Fig -4: YOLOv5 Architecture

2.2.1 CSP Backbone

With fewer parameters and fewer FLOPS for problems of equivalent significance, the CSP solves duplicate gradient issues in other larger ConvNet backbones. For the YOLO family, where inference speed and minimal model size are crucial, this is quite essential [10]. DenseNet is the foundation for the CSP models. DenseNet was created to connect layers in convolutional neural networks with the following goals in mind: to improve feature propagation, encourage the network to reuse features, and decrease the number of network parameters; to address the vanishing gradient problem (it is difficult to backprop loss signals through a very deep network) [11].

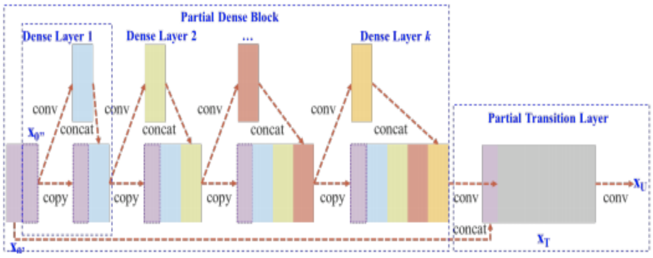

Fig -5: Cross Stage Partial DenseNet

2.2.2 PA-Net Neck

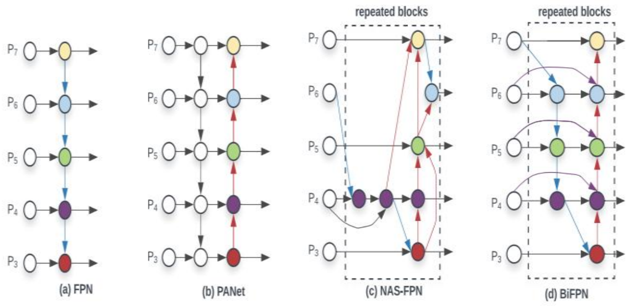

Fig -6: PA-Net Neck

3. PROPOSED WORKFLOW

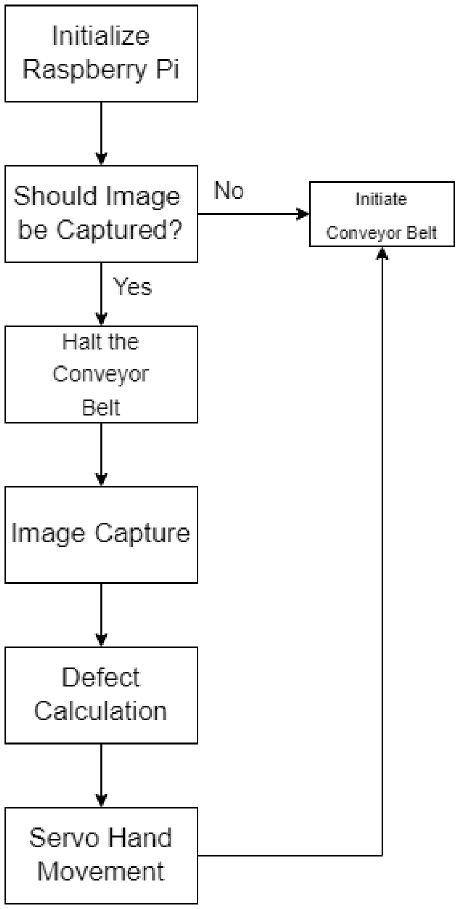

Fig -7: Workflow

The Proposed system uses yolov5. We also propose an approach to implement detection and segregation using Raspberry Pi and Arduino Nano. The accuracy of the model is above 90 %. The model will be deployed on the Raspberry Pi. The segregation of the PCB would be done by Arduino Uno. The PCB image is captured using the Pi Camera with Raspberry Pi. The yolo model deployed on the Raspberry Pi detects for any defects present on the PCB. If the model detects any defects on the PCB, the Arduino Uno move the PCB away. In the proposed system to overcome this problem, we create a new automated model, that helps to detect errors or defects on the PCBs. The model has over 95% accuracy as this is trained with a huge data set - approximately 10, 680 images that has been divided into 80% testing images and 20% valuation images. This model does not require any input image or any other extra sources as many models out there need. This model is trained to detect the defects directly. We make use of models like YOLO – You Only Look Once Py Torch vision model and some of the TensorFlow models to compare the results and to deploy the most preferrable one – that is, it depends on the cost of performance and the accuracy while being not so resource hungry. Unlike the existing system we do not make use of the template, that is we only need to capture the sample image to detect. In the proposed system we do not subtract the image rather let the computer decide if there is an error or not. This is made possible by using Artificial Intelligence. Even though AI is associated with its own drawbacks, the model is much more efficient than the existing one. We also add a belt system that makes the whole detection process automatic and much more reliable than that of the existing system. We make use of a microcontroller that gets the signals or results for the raspberry pi based on the detection results, and depending on those results the PCB is discarded.

4. RESULTS

The DeepPCB dataset, which contains roughly 10,800 images, was utilized to train the model. This dataset contains 6 different types of defects. This section evaluates the proposed approach and describes the findings. PCBs with defects and PCBs that are in good condition are used to train the network. Several hyperparameters were adjusted to improve efficiency. There were 300 training epochs with a batch size of 64. At 300 epochs, the performance of both training models is assessed using the test dataset that wasn't used during training.

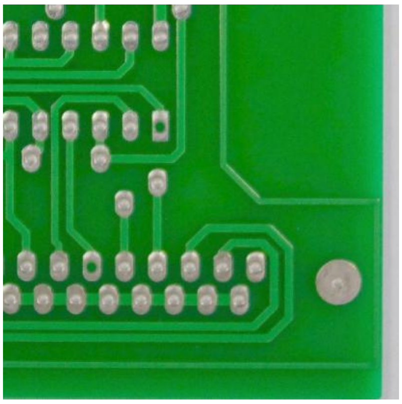

Fig -8: Input Image

Fig -9: Detected Image

5. CONCLUSION

In the proposed approach, we created a new automated model, that helps to detect errors or defects on the PCBs. The model has over 95% accuracy as this is trained with a huge data set - approximately 10, 680 images that has been divided into 80% testing images and 20% valuation images. This model does not require any input image or any other extra sources as many models out there need. This model is trained to detect the defects directly.

REFERENCES

[1] https://mdpi.com/2079-9292/9/9/1547/pdf

[2] https://ncbi.nlm.nih.gov/pmc/articles/PMC8347834/

[3] Improved YOLOV3 target recognition algorithm based on adaptive eged optimization.

[4] https://github.com/ultralytics/yolov5

[5] Approaching GoogLeNet-Level accuracy of classification and domain adaptation.

[6] Improved center and scale prediction-based pedestrian detection using convolutional block.

[7] Spatial pyramid pooling in deep convolutional networks for visual recognition

[8] Single shot multibox detector

[9] Efficient and accurate arbitrary-shaped text detection with pixel aggregation network

[10] CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN

[11] Densely Connected Convolutional Networks

[12] EfficientDet: Scalable and Efficient Object Detection

[13] https://github.com/tangsanli5201/DeepPCB