EasyChair Preprint

№ 8174

Automated Quality Inspection of Printed Circuit Board

S D Ullas and Ravikumar Beeranur

June 1, 2022

International Conference on Sustainable Materials, Manufacturing & Industrial Engineering- ICSMMIE-2022

1st -2nd of July, 2022

Tumkur, Karnataka, India

AUTOMATED QUALITY INSPECTION OF PRINTED CIRCUIT BOARD

ULLAS.S.D [1], RAVIKUMAR BEERANUR[2].

[1] M.Tech Student, Industrial Automation and Robotics, Department of Mechanical Engineering,The National Institute of Engineering, Mysuru, Karnataka, India.

[2]Assistant Professor, Department of Mechanical Engineering,The National Institute of Engineering, Mysuru, Karnataka, India.

Abstract.

As technology gets advanced, more components depend on the printed circuit board (PCB), and the usage of the PCB layout increases. The tiniest defects on the board might cause serious system harm. PCB surface inspection and identification of defects are one of the most crucial quality control processes. We're using a new model named YOLO-v5 in this procedure, which is designed to locate and detect a variety of PCB defects. This YOLO-v5 algorithm was chosen because of the model's excellent efficacy, precision, and speed. In this paper, we used data that contain 700 images with 4 different types of defects. With a batch size of 16 and a trained epoch of 200, this model achieved a defect detection accuracy of 95.25 percent in PCB.

Keywords: YOLO-V5, Printed circuit board (PCB), Convolution neural network, Deep Learning.

1. Introduction

The major and very basic component for any electronic product is a printed circuit board (PCB). In our daily routine, we are seeing and sensing many electronic components from the beginning of the day to the end of the day. In this quick innovation time greater part of the people are relying upon electronic items from correspondence to execution, every single part is relying upon PCB, for any electronic component PCB is the core and base. Different varieties of PCB's are created and designed for various applications that must be manufactured with high precision to meet expectations and needs while maintaining acceptable quality is a difficult undertaking. PCB‘s are comprised of fibreglass, composite epoxy, and laminated materials and serve as a foundation for chips, transistors, capacitors, and other electronic components [1-3]. The PCB's quality will have an immediate influence on the electrical device's performance. As the number of uses for electrical devices grows and evolves, the PCB becomes increasingly complex. The task of defect identification, categorization, and mapping to the defect is more difficult than before. Nowadays PCB manufacturing industries use different image processing techniques and software, but still, they are facing some difficulties in identifying tinny defects in PCB, because its tedious to predict which is the error and where is the defect [4]. In this automation era, we can’t able to put a human inspector in the high-speed production line for quality inspection. To address the limitations of human discovery, for example, low precision and examination speed, automated optical inspection (AOI) based machine vision has been generally utilized in industry [5].

There are three main streams in traditional AOI methods for inspecting printed circuit boards:

a. Reference comparison approach: In this method, a standard image will be taken as a template and PCB needs to be inspected by comparing with the unknown defect. Although it is simple to use, several concerns have been considered, such as imbalanced illumination, improper registration, and so on.

b. Non-reference verification approach: This method is not limited by the reference method; yet, it may struggle to discover major faults.

c. Hybrid approach: It is a hybrid of the reference and non-reference methods, and it combines the benefits of both. However, it necessitates a large computing capacity [6].

The above detection algorithm is specific to a particular type of defect in PCB. Convolutional neural networks (CNN) are making significant progress in a variety of applications, including image recognition and object detection [7-8]. This method is used with AOI machines, apart from using the machines PCB industries train a large amount of manpower for quality inspection. To solve obstacles and issues of this nature In this project, YOLO-v5 was introduced, by applying the YOLO-v5 deep learning algorithm we can eventually reduce the errors and we can easily get to know which is the defect and where is the defect in PCB. Machine programmed deep learning algorithm is used to identify tinny defects in PCB. Machine learning is more precise and faster than a high skilled human operator. Many recent studies have shown that using machine learning to detect PCB defects and boost production with high accuracy is possible. To reach the outcome, researchers used a variety of You-Only-Look-Once techniques [9-11]. In our project, we use YOLO-v5 deep learning algorithm to find and classify the defects in PCB. YOLO-v5 performs object detection and face identification thanks to its unique features including mosaic data augmentation and adaptive anchor frame computation. In comparison to YOLO-v4, YOLO-v5 is a well-designed structure that performs operations at a very high speed and is tiny in size. The PyTorch library is utilized in this venture to send the YOLO-v5 model, which is especially easy to understand for developers. When we set batch size as 1, we can achieve 30 framerate per second (FPS) in YOLO-v5 and 10 framerate per second (FPS) can be achieved in YOLO-v4. After 300 epochs of training, we expect to obtain a mean average precision (MAP) of at least 0.90. The YOLO-v5 weight file is around 27 megabytes in size, while the YOLO-v4 weight file is 244 megabytes in size. As a result, the YOLO-v5 is approximately 90% smaller than the YOLO-v4, making it much easier to install on an embedded device[12].

2. Materials and methods

2.1 Pcb dataset

By taking reference to the PCB data set produced by Huang and Wei created a dataset for PCB defects [13], we conducted an experiment with our camera and lighting setup for our defective PCB.

2.1.1 Image acquisition

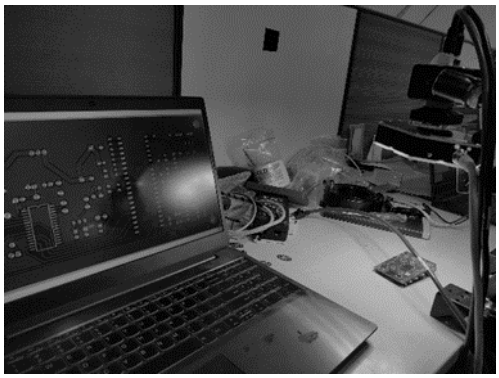

Figure 1 : A light source, workstation, support, camera, and image processing unit make up the PCB image capture system.

We create a PCB image capture system that matches the automated optical inspection (AOI) equipment to collect the image dataset, as shown in Figure 1. A 12MP Cognex industrial GEGI camera with the newest CMOS sensor captures the image of the PCB, which may be manipulated by computer software. We utilize white panel light with a specific diffuse matting board to overcome illumination disruption and eliminate specular reflections of the board, possible shadows, and the effect of unwanted illumination on the board. The resolution of the original image is 2448×2048 pixels. We are collecting four sorts of malfunctioning PCB boards: missing holes, mouse bite, open circuit, and short circuit.

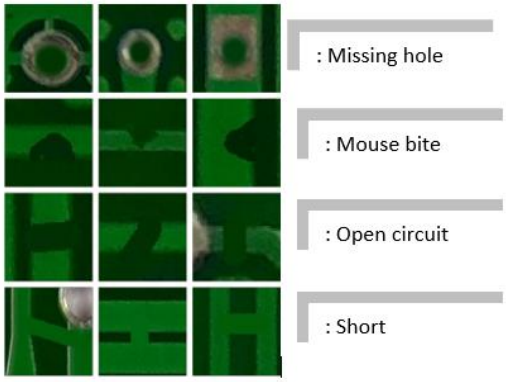

Figure 2: Types of defects in PCB

2.1.2 Image annotation

Each image in the dataset has at least 2 to 4 defects of the same category in different places of PCB. For this, we are using ROBOFLOW annotation software to create the bounding box to the defects and to label them. Then save all annotation files in XML format. The process of labeling towards the defects in the image is shown in figure 3. The annotated XML file is shown in figure 4.

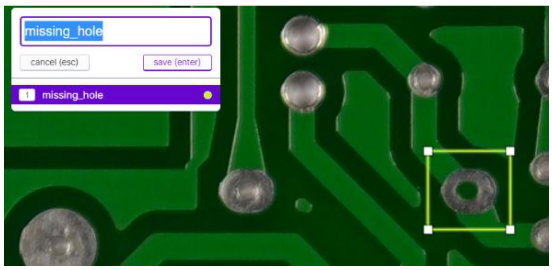

Figure 3: The labeling process towards defects in PCB

```
<annotation>
        <folder>Missing_hole</folder>
        <filename>01_missing_hole_04.jpg</filename>
        <path>/home/weapon/Desktop/PCB_DATASET/Missing_hole/01_missing_hole_04.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>3034</width>
                <height>1586</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
```

Figure 4: The annotated XML file

2.2 Statistics

We get the dataset into 3 segments, which are positioned into 3 different folders. The image folder stores PCB photos, different types of defective PCB are stored in different folders, the information of bounding boxes is stored in a .xml file that is saved in the annotation folder, and good PCB images are saved in the PCB_GOOD folder. The tree diagram in Figure 5 displays the dataset's structure.

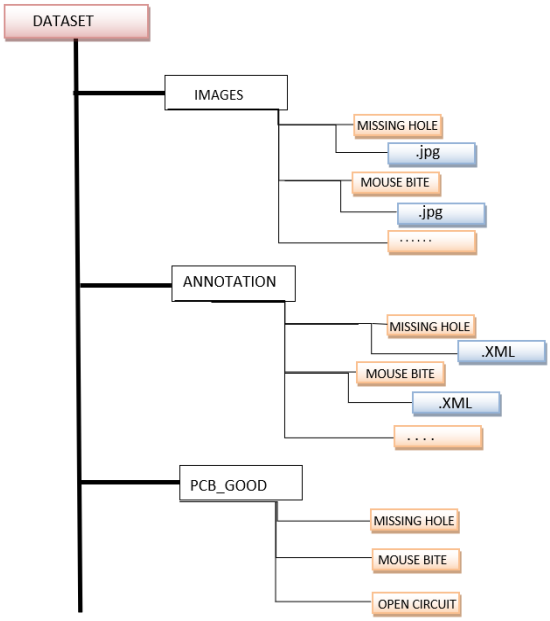

Figure5: The structure of the dataset shown in the tree diagram

3 Architecture of Yolo-v5

The YOLO-v5 network is split into 4 sections: 1. input 2.backbone 3.neck 4.prediction as shown in Figure 6.

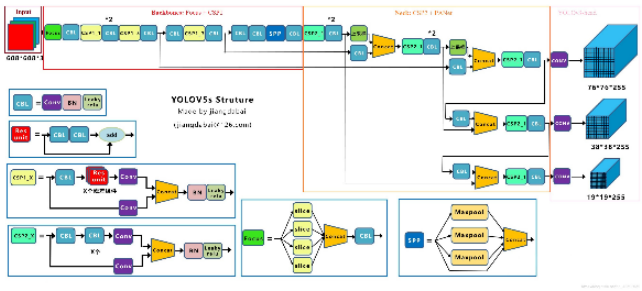

Figure6: Structure of YOLO-v5

a. **Input**: In this project, we are using the mosaic data enhancement method for data extraction because our dataset has a large count of small defects in each PCB. We have greater advantages when more data is provided because of the random use of dataset pictures, random scaling, and random distribution of slicing. These features easily refine the detection dataset, particularly random scaling adds a lot of tinny targets, making the network healthier and power full. In our project, we use the Graphics processing unit (GPU) for training data to achieve a better result. Basically, different images have different widths and lengths in the detection algorithm, so the images are uniformly scaled with respect to standard images and then fed into the detection network. Because many defects have varying aspect ratios during inspection, the black image border can vary after zooming and filling. If more filling is required, information redundancy will occur, slowing down reasoning speed [14]. As a result, we employ the YOLO-v5 code, When compared to the previous one, this one converts the letterbox function to a standard picture and reduces the black border. The black edges on both sides of images are accordingly lowered and improve our detection speed, by applying this Yolo-v5 code we got the speed ratio up to 37%, which can be said to be very effective.

b. **Backbone**: Following the example of the YOLO-v5 structure, a regular 608×608×3 is taken care of into the center design as delineated in Figure [7]. The image is then sliced to produce a 304×304×12 feature map, which is subsequently mixed using convolution operation kernels to produce a 304×304×32 feature map. Figure 7 depicts the YOLO-v5 focus structure. In the project, we are using Cross Stage Partial Network (CSP) according to [14]. This model is used to separate the base layer's feature map into two sections, then collage them using cross-stage hierarchy, which reduce computation time and ensures accuracy.

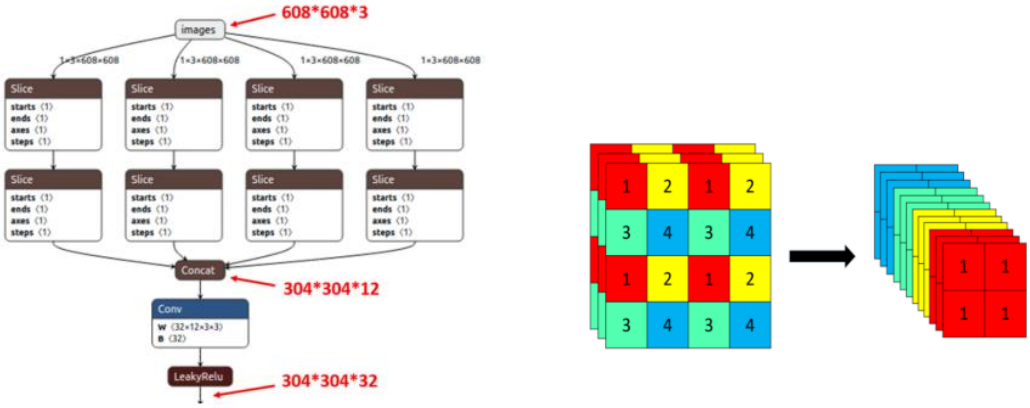

Figure 7: Focus structure of YOLO-v5

c. **Neck**: Model Neck is mainly used to generate feature pyramids. Feature pyramids help models to generalize well on object scaling. It helps to identify the same defect with different sizes and scales. Currently, we use YOLO-v5 feature pyramid network (FPN) structure and path aggression network (PAN) structure is added for latter, and another part of the network is also adjusted, CSP_2 invented by CSP_net is used to improve the capacity to extract network features, The neck design of Yolo-v5 is illustrated in Figure 8.

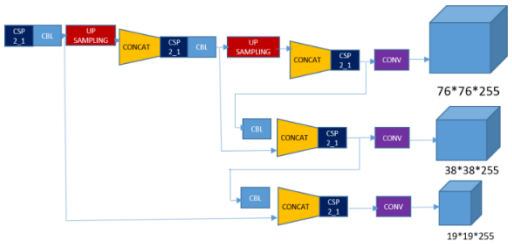

Figure 8: Neck structure of YOLO-v5

d. **Output**: Yolo-v5 use intersection over union (IOU) loss as a loss function of the bounding box, for in defects detection is a technique for determining the degree of resemblance between the anticipated and actual bounding boxes. The output region is also called as HEAD region which helps to detect the defects in PCB.

4 Training

The training procedure is completed on pc with intel CORE i5 CPU with 8 GB RAM and NVIDIA GeForce GTX GPU is taken to train the model. For this PCB defect inspection, we are training the model for 416 images of 16 batches with 200 epochs, during this procedure the datasets are divided into 3 parts namely train, test, and validation datasets. This dataset is divided in the ratio 80:10:10, 80% of the datasets are used for training purposes 10% of datasets are used for testing and the remaining 10% of the datasets are used for validation purposes. After completion of this procedure, the brain file is called “best.pt” is generated in the “weights” folder. For further analysis and testing, we use the “best.pt” file as the brain file to get the result. The training model is shown in Figure 9.

```
!python train.py --img 416 --batch 16 --epochs 150 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache
Overriding model.yaml nc=80 with nc=1

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19         [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22         [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     16182  models.yolo.Detect                      [1, [[10, 13, 16, 30, 33, 23], [30, 61, 62,
Model Summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.8 GFLOPs
```

Figure 9: Training the model in google colab

5 Result

The primary goal of this PCB defect inspection is to find and classify defects while also minimising the method's time and cost. YOLO-v5 is much easier to implement in an embedded device than the extra detection algorithm. The installation of py-torch and some basic libraries is all that is required for YOLO-v5. To conduct the experiment, three varieties of YOLO-v5 models are provided. YOLO-v5 small, medium, and large are the three models. In our project we use the YOLO-v5 medium to train the model, Up to 30 million parameters and 200 layers are generated using the YOLO model (Plain 85Mb, COCO pre-trained 170MB). The approach in this research is based on a pre-trained YOLO-v5 medium model. The performance of YOLO-v5 is superior to that of other YOLO models for each epoch, we witness increased accuracy in the training process, allowing the model to achieve higher results. When the precision is achieved, the final model is saved. In this project we train process is done by using YOLO-v5 Medium and we got an accuracy of approximately 95.25%, based on the result we come to the conclusion YOLO-v5 medium gives a better result in less amount of time. We got an accurate result of 95.25% in detecting and classification of defects and on average of 10 cross-validations the accuracy 96.56%.

Result images:

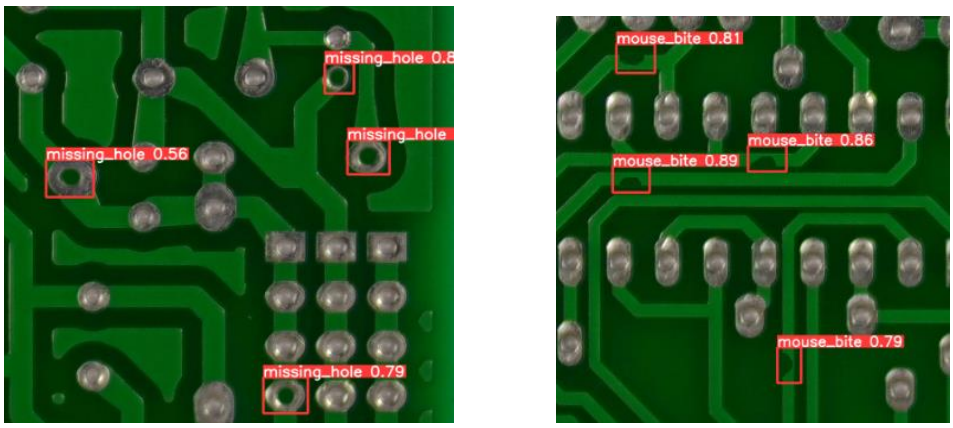

Figure 10: Missing holes and mouse bite result with true positive value

Genuine Positive's missing opening and mouse bite are portrayed in Figure 10. It very well may be seen in the example pictures over that the model can dependably distinguish defects.

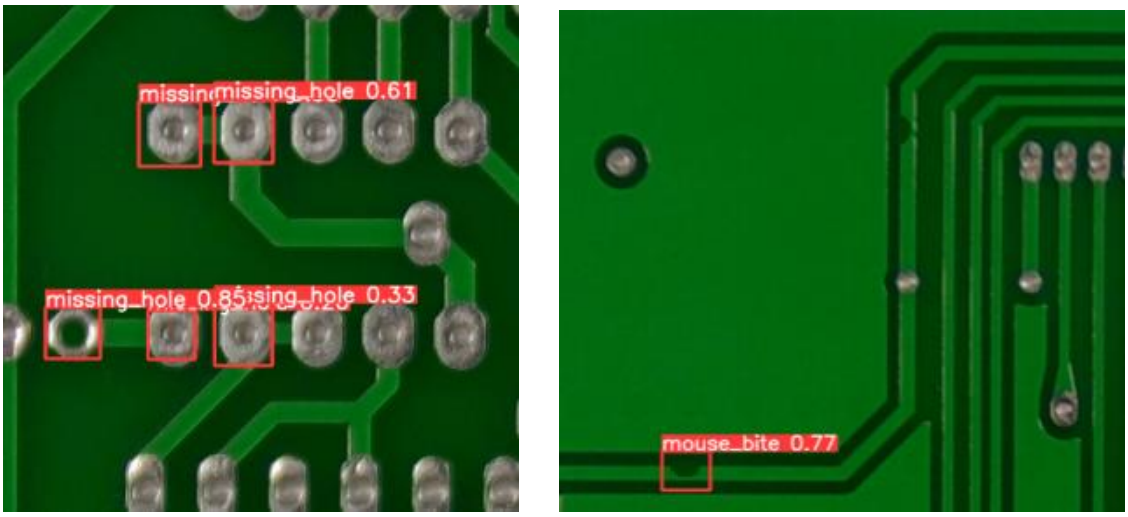

Figure 11: Missing holes and mouse bite result with false positive value

Figures 11 illustrate examples of False Negative images, which are images with a defective location that the program is unable to detect.

Table 1. For the YOLO-v5 medium model, a confusion matrix of 5 separate cross-validations was created.

Confusion matrix

BATCH SIZE	YOLO-v5 Medium		
		NG	OK
Cross-validation 1	NG	612	2
	OK	1	147
Cross-validation 1	NG	611	3
	OK	3	145
Cross-validation 1	NG	614	1
	OK	2	146
Cross-validation 1	NG	616	1
	OK	1	144
Cross-validation 1	NG	612	0
	OK	4	140

6 Discussion

One of the key advantages of the YOLO-v5 over prior versions in the YOLO series is that it was built entirely in PyTorch. YOLO-v5 is 90 percent smaller than YOLO-v4 and is significantly faster and more accurate than the previous version [15-16]. This means that YOLO-v5 will have no trouble locating the embedded device. After 200 epochs of training, YOLO-v5 is accurate and can easily obtain a mean average precision of 95 percent. The YOLO-v5 small file is 27 megabytes in size, whereas the YOLO-v5 large file is 192 megabytes. The used dataset is collected from the internet which is provided by hung and leng weik[13], in this project we detect and classify 4 types of defects in PCB by using the YOLO-v5 medium model. In our case, the model takes 1hr to 1:30hr to tarin 200 epochs and after this training, we got an accuracy of 95.25%.

7 Conclusion

The YOLO-v5 medium can distinguish defects in PCB's with a satisfactory exactness of 95.25 percent, as indicated by this task, which saved a great deal of talented human's work and time. It also enhances precision, future work can improve accuracy by considering a few more defects, class collecting must be done in a more balanced manner, and additional defects must be included. Our team will aim to construct and refine a fully automated model without the usage of humans in the next days, and will apply transfer learning to improve accuracy. Eventually, for a pre-trained YOLO model, the transfer learning strategy [17-18] can be considered.

Acknowledgment

I would like to convey my heartfelt gratitude to all of the authors and contributors of the research articles and other reports that have been referenced and cited in this study. Also, thanks to the ICSMMIE editorial for their support and review of our manuscript.

References

1. J. A. Magera & G. J. Dunn (2008), "The Printed Circuit Designer's Guide to Flex and Rigid-Flex Fundamentals", *Motorola Solutions Inc*., US 7459202, pp. 3-4.
2. A. P. S. Chauhan & S. C. Bhardwaj (2011), "*Detection of bare PCB defects by image subtraction method using machine vision*", in Proceedings of the World Congress on Engineering, London, U.K, vol. 2, pp.6-8.
3. P. S. Malge (February 2014), "PCB defect detection, classification and localization using mathematical morphology and image processing tools*", Int. J. Comput. Appl*, vol.87, pp.40-45.
4. Koch J, Gritsch A, Reinhart, G (2016)" Process design for the management of changes in manufacturing: Toward a Manufacturing Change Management process", *CIRP J. Manuf. Sci. Technol*, vol.14, pp.1-5.
5. A. J. Crispin & V. Rankov (2007), "Automated inspection of PCB components using a genetic algorithm template-matching approach", *Int. J. Adv. Manuf. Technol*, vol.35, pp.293-300.
6. Wen Yen Wu, Mao Jiun J Wang, and Chih Ming Liu (1996), "Automated inspection of printed circuit boards through machine vision.", *Computers in Industry*, vol. 28, pp. 103-111.
7. Hosseini, H, Xiao, B, Jaiswal, M, Poovendran, R (2017), "On the Limitation of Convolutional Neural Networks in Recognizing Negative Images", *IEEE International Conference on Machine Learning and Applications* (ICMLA), Cancun, Mexico, pp. 352–358.
8. Shiyang Zhou, Youping Chen, Dailin Zhang, Jingming Xie, and Yunfei Zhou (2017), "Classification of surface defects on steel sheet using convolutional neural networks", *Material in tehnologije*, vol. 51, pp.123-131.

9. J. Cong, B. Xiao (2014), "Minimizing computation in convolutional neural networks, in International conference on artificial neural networks", *Springer*, vol 8681, pp.281-290.
10. D. C. Cireşan, U. Meier, L. M. Gambardella, J. Schmidhuber (2011), "*Convolutional neural network committees for handwritten character classification*", International Conference on Document Analysis and Recognition, pp. 1135-1139.
11. Aryan garg (December 2021), "How to use yolo-v5 object detection algorithm for custom object detection", Available

    https://www.analyticsvidhya.com/blog/2021/12/how-to-use-yolo-v5-object-detection-algorithm-for-custom-object-detection-an-example-use-case/
12. Glenn Jocher (may 2020), "Comparison between Yolo-v5 and Yolo-v4", Available https://colab.research.google.com/github/pytorch/pytorch.github.io/blob/master/assets/hub/ultralytics_yolov5.ipynb
13. Weibo Huang & Peng Wei (August 2018), "A PCB Dataset for Defects Detection and Classification", *Journal of latex class FILES*, VOL. 14, pp.1-9.
14. Computer vision (July 2020), "Yolo-v5 explained and demystified", Available

    https://towardsai.net/p/computer-vision/yolo-v5%E2%80%8A-%E2%80%8Aexplained-and-demystified
15. V. Lalitha & C. Eswaran (2007), "Automated detection of anesthetic depth levels using chaotic features with artificial neural networks", *J. Med. Syst*, vol. 31, pp. 445-452.
16. Do Thuan (2021), "Evolution of YOLO Algorithm and YOLOv5: The State-of-the-art Object Detection Algorithm", *Spring*, pp. 15-35.
17. T. Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar (2017)," Focal loss for dense object detection", *Proceedings of the IEEE international conference on computer vision*. Pp. 2980-2988.
18. L. Shao, F. Zhu, X. Li (2015), "Transfer learning for visual categorization", *Neural Netw. Learn. Syst*, vol. 26, pp. 1019-1034.