

Detection of PCB Defects Using Deep Learning

Marina Miladinović, Miljana Milić

Abstract - Printed circuit board (PCB) form the foundation of almost every electronic system, and its defects can lead to many problems in functionality. Manual inspection of those defects is time consuming and prone to human error, since not all of them can be spotted immediately. This paper represents an efficient solution for automated PCB defects detection using a pre-trained Faster R-CNN deep-learning model. The model was trained on a Kaggle dataset, with YOLO annotated images for six most common defects. Training was finished after 12 epochs, and model had great achievements of around 90% of accuracy.

Keywords – Printed circuit board, defect detection, Deep Learning, Faster R-CNN.

I. INTRODUCTION

Printed Circuit Boards (PCBs) are the backbone of modern electronics, providing a platform for mounting and connecting various components to create complex electronic circuits [1]. To ensure proper device performance, PCBs must meet high quality standards. However, PCBs may still become damaged during production or handling, and even small defects can cause serious functional failures. The most common PCB defects are: spur, mouse bite, missing hole, open circuit, short circuit and spurious copper.

Manual visual inspection is challenging, especially for small or densely populated boards, and is both time-consuming and prone to error. Deep learning, a subfield of machine learning and artificial intelligence, employs powerful tools for automating this task. Convolutional Neural Networks (CNNs), particularly the Faster R-CNN architecture with a ResNet-50-FPN backbone, are well-suited for detecting defects in PCB images.

Training Deep Learning model requires large dataset with accurately labeled defects. During training, the network learns to recognize defect patterns and generalize its knowledge to new, unseen images.

Automating a process like this will save time and resources needed to detect defects manually, but most importantly it will significantly improve reliability of PCB.

II. THEORETICAL BACKGROUND AND RELATED WORK

Researchers have spent years improving computer-vision algorithms to support automated inspection.

Marina Miladinović, and Miljana Milić are with the University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 4, Niš, Email: personanongrata@elfak.rs, miljana.milic@elfak.ni.ac.rs

The core of this system relies on deep-learning and Neural Networks, the most powerful and prevalent methodologies in modern science and technology, that affecting common applications and almost all electronic fields.

A. Deep Learning and Neural Networks

Deep Learning is a type of Machine Learning, that enables computers to process data much like human brain does. It relies on multilayered artificial neural networks that consist of connected nodes (neurons) organized into input, hidden and output layers.

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. [2]

Artificial Neural Networks, mimic the human brain by extracting abstract features from the data and by combination of inputs, weights and bias. These elements work together to accurately recognize, classify and describe objects within the data [3].

Deep neural networks consist of many layers of interconnected nodes as shown in Fig. 1, each adding on the previous layer to tune and optimize the prediction. Input layer receives the data and passes them to hidden layers who then perform series of operations that help the output layer to make a final prediction or classification.

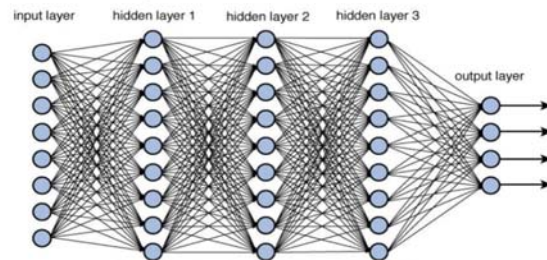


Fig. 1 Architecture of Deep Neural Network

Important processes that enable neural networks to make predictions and correct errors are feedforward propagation and backpropagation. In feedforward propagation input data moves through each layer of neural network where each neuron applies weighted sum, adds bias, passes the result through an activation function in order to make the predictions. It determines the output of neural network with a given set of inputs and current state of model parameters (weights and biases) [4].

The logic of backpropagation is that the layers of neurons are essentially a series of nested mathematical functions. During training, those interconnected nodes are nested into yet another function called a "loss function" that measures

the difference (or “loss”) between the $f(x) = \max(0, x)$ (or “ground truth”) for a given input and the neural network’s actual output. Mathematically speaking, backpropagation works backward from the output to efficiently calculate the “gradient” of the loss function: a vector of derivatives for every equation in the network. The gradient points out which parts of the model are causing most errors and which way to adjust it to get better results.

An activation function, in a neural network is a mathematical function applied to the output of a neuron. It introduces non-linearity, enabling the model to learn and represent complex data patterns. Without it, even a deep neural network would behave like a simple linear regression model. Activation functions decide whether a neuron should be activated based on the weighted sum of inputs and a bias term and is shown in Fig. 2 [5].

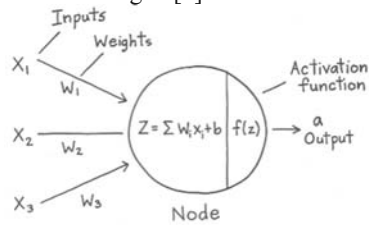


Fig. 2 Activation function of Neural Network

There are several activation functions to choose from and one of the most commonly used are *Sigmoid* and *Tangent* functions.

Throughout the years researchers developed the Rectified Linear Unit (ReLU) functional block, which also proved as computationally efficient, mainly since the significant performance improvement. The form and graph for this function can be seen in Fig. 3.

The derivative of ReLU, with respect to its input, is always 1 for positive input values. Therefore, it solves the problem of “vanishing gradients”, making it suitable for deep neural networks. [6] Vanishing gradient problem occurs when gradients became small during backpropagation, causing early layers to learn slow or not to learn at all.

This means that it does not alter the scale of positive inputs and allows the gradient to pass through unchanged during backpropagation. Even though ReLU is linear for half of its input space, it is technically a non-linear function because it has a non-differentiable point at $x=0$, where it abruptly changes from x .



Fig. 3 ReLU activation function of Neural Network

B. Convolutional Neural Networks and Faster R-CNN

Designed to process data and extract features from grid-like topology such as images, Convolutional Neural Networks are a foundation for modern computer vision applications. Consisting of multiple layers as it can be seen in Fig. 4. they are able to automatically learn the features from raw data that are most useful for a particular task. For this reason, it's common to consider CNN layers as feature extractors: the early layers (those right after the input layer) extract low-level features from raw data, and the later layers (often fully connected layers like in a multilayer perceptron) use these features to predict a continuous target value or class label [7].

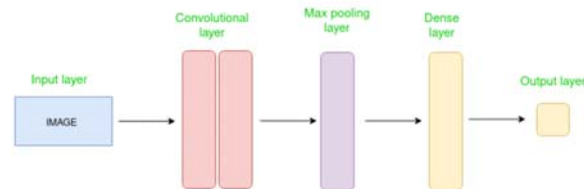


Fig. 4 Architecture of Convolutional Neural Network

Convolutional layer is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are small matrices usually 2x2, 3x3, or 5x5 dimensions. A kernel slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is known as feature maps. Activation function layer which inserts non-linearity to the network is added to the output of this layer.

Main function of Pooling layer is to reduce the size of volume as it makes data processing faster by reducing memory and also prevents overfitting.

Dense layer takes feature maps from convolutional and pooling layer, flattens them in to one-dimension vector, links them together and computes the final classification or regression task.

In this present moment artificial intelligence and deep learning are developing minute by minute, using only CNN for this paper will not give wanted results. Considering that Faster R-CNN is chosen, as pre-trained model with crucial ResNet-50-FPN structure for more accurate and efficient outcome.

The Faster R-CNN (Region-based Convolutional Neural Network) model, consists of two stages of object detection, and combines Region Proposal Network (RPN) and classification in one deep neural network. The first stage proposes potential locations where defect might be. After resizing and processing (Region of Interest Pooling) the second stage then classifies those regions as one of six defects, as shown in Fig. 5.

ResNet50 (Residual Network with 50 layers) serves as a feature extractor and helps training deep neural networks in early stages of learning. In this way, “vanishing gradient” problem can efficiently be solved. In combination with FPN

(Feature Pyramid Network) this model allows detection of both tiny and large defects.

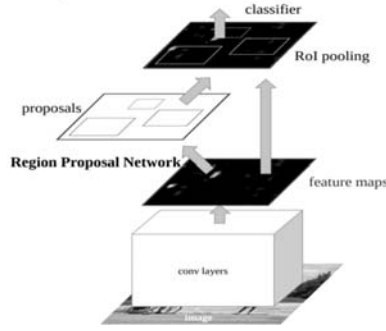


Fig. 5 Faster R-CNN architecture

C. PCB Defects

PCBs have a critical role in many applications, and maintaining their high manufacturing quality is essential. As modern electronic systems grow more complex, the margin for error becomes smaller. Reliable, well-made PCBs ensure stable device performance and minimize failures in the field. For this reason, improving PCB quality control remains an important topic in both research and industry [8] [9]. Quality tests and defect detections are still primarily done manually by humans and missing some of defects does not come as a surprise. Automatization of such processes with a computer vision can be very helpful.

Producing PCBs with low quality materials to cut costs leaves risk of malfunctioning boards that sometimes cannot be seen during manual inspection. Design errors, thermal issues, scratches, mechanical damage are also well-known causes of defects that are difficult to notice.

Six of them are displayed in Fig. 6.

- 1) *Spur* is inflexible projections of conductor along linear trace that can result in short circuits when in contact with other elements.
- 2) *Mouse bite* is a defect where conductor's path is thinned or narrowed along its length which can cause potential overheating.
- 3) *Missing hole* defect happens when a hole that is specified in the design is not drilled on the final board.

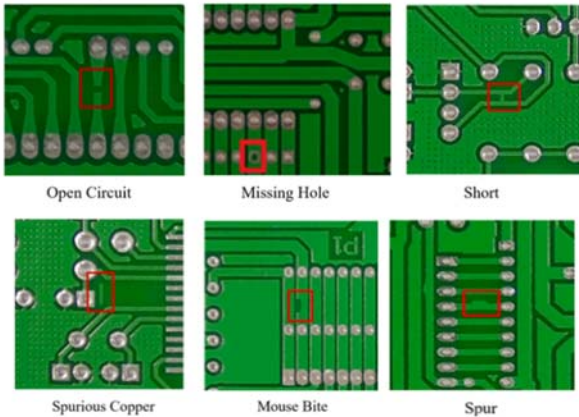


Fig. 6 Common defects on printed circuit boards

- 4) *Open circuit* occurs when conductor between two points is not connected and can disable the function of the board.
- 5) *Short circuit* is present when conductor connects two tracks that should not be connected.
- 6) *Spurious copper* is an isolated region of conductor appearing in a location not shown in design.

Main focus in this research is given to the detection of listed types of defects.

III. IMPLEMENTATION

A. Dataset and framework

Dataset chosen for this project can be found at Kaggle database and website and has 693 images with classified in six types of defects. They can be used for detection, classification and registration tasks.

Every image in dataset had different dimension so to standardize the input they were converted to fixed size of 512x512 pixels and gray-scale using custom Python script. Only images with defects 'spur' and 'spurious copper' were converted to 800x800 pixels to get better training of these small defects.

Dataset is divided into three folders, one for training, one for validation and one for testing. Along with images, there are text files with labels of raw defect annotations, provided in YOLO (You Only Look Once) format showing normalized center coordinates, width and height (x_c, y_c, w, h). Since Faster R-CNN model requires absolute pixel coordinates ($x_{min}, y_{min}, x_{max}, y_{max}$), it is required to perform a coordinates conversion.

The custom class was implemented to handle the coordinates conversion: `PCBYoloDataset`. This class ensures that the data was correctly translated for the model to use.

The proposed system was implemented using PyTorch framework in Python environment. It is flexible and modular to design with and has strong support for object detection through `torchvision` library.

The core of the model was defined using `torchvision.models.detection` module and that includes pre-trained architectures, in this case Faster R-CNN with a ResNet50-FPN backbone.

This structure allowed the model to switch between training and evaluation modes with `model.train()` and `model.eval()`. During the evaluation, predictions were obtained as tensors that contain class ID, bounding box and confidence score which is then processed and visualized on the PCB images.

B. Data loading and pre-processing

Using custom class `PCBYoloDataset` images were loaded and preprocessed, while the coordinates were converted. Each image was transformed to a tensor with values scaled between 0 and 1. Data loaders were created for training, validation and testing with a batch size of 4 for

training and 2 for evaluation. The class also includes `__len__()` function that returns total number of images and `__getitem__()` that provides image-annotation pair for the given index. This allows PyTorch DataLoader to combine multiple samples into batches without errors, and guarantees consistent and structured input for the model.

C. Model training

Training process was conducted using Faster R-CNN architecture as the core model with pre-trained ResNet-50 structure. Model was made to train for 12 epochs, where one epoch took approximately 3 minutes. Each epoch consisted of complete pass through the training dataset using custom function `train_one_epoch()`, with training loop executing in the following order:

- 1) Loading one batch of images with their annotations.
 - 2) Moving tensors to the CPU.
 - 3) Performing a fast forward process through the model to calculate the loss.
 - 4) Summarization of all losses into a single value.
 - 5) Performing backpropagation process to update the weights using optimizer.
 - 6) Record average loss statistics after every 20 batches to monitor the training.
- After each epoch, a validation was performed using `quick_infer_check()` function to verify the model was producing right predictions.

Also a checkpoint mechanism was implemented with `maybe_save_checkpoint()` to overlook model and ensure that progress was safely stored in case of any interrupt. Another important implementation was the implementation of the optimizer that updates the network's weights and minimize the loss.

IV. EVALUATION AND RESULTS

After the training process was completed, model was tested for the images from the test folder. The evaluation demonstrates that implemented model achieves strong and confident defect detection on PCB images even in limited environment. Running training on CPU decreased epoch durations and restricted the total number of epochs. Although it had some limitations, model showed accuracy of around 90% with minimal losses. Model was able to recognize all of six defects, and demonstrated its ability on a complex and challenging problem. PyTorch framework results can be seen in Fig. 7.

V. CONCLUSION

This paper proposed the system for defect detection on PCB images using Deep Learning as one of the most powerful tools in computer vision. It enables automatization of the detection process, and saves time and expenses that defective PCBs can cause. Faster R-CNN model was used, based on deep learning architecture, along with ResNet50-FPN structure.

While the developed system had promising results, there are several issues that need to be explored in future research. They are data augmentation, fine-grade classification and higher accuracy.

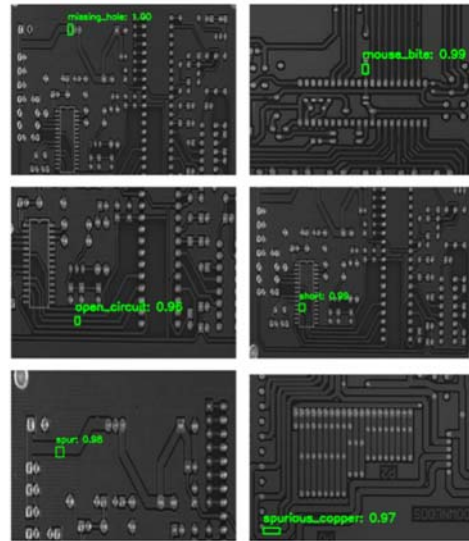


Fig. 7 Results

ACKNOWLEDGEMENT

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

REFERENCES

- [1] Neha Kardam, "Introduction to Printed Circuit Board Desing and Layout using Ki-CAD", *Lake Washington Institute of Technology*, 2024.
- [2] Kenji Suzuki, *Artificial Neural Networks - Methodological Advances and Biomedical Applications*, *InTech*, 2011.
- [3] Available at: <https://www.ibm.com/think/topics/deep-learning>, accessed on: 22/09/2025.
- [4] Available at: <https://www.geeksforgeeks.org/deep-learning/what-is-forward-propagation-in-neural-networks/>, accessed on: 01/10/2025.
- [5] Available at: <https://www.geeksforgeeks.org/machine-learning/activation-functions-neural-networks/>, accessed on: 09/10/2025.
- [6] Sebastian Raschka & Vahid Mirjalili, "Python Machine Learning Third Edition", *Packt Publishing*, 2019.
- [7] Sebastian Raschka & Vahid Mirjalili, "Python Machine Learning Third Edition", *Packt Publishing*, 2019.
- [8] Predrag M. Petković, Miljana Milić, Dejan Mirković, "VHDL i VHDL-AMS podrška projektovanju elektronskih kola i sistema", *Univerzitet u Nišu Elektronski fakultet*, 2009.
- [9] Vančo B. Litovski, *Osnovi testiranja elektronskih kola*, *Univerzitet u Nišu Elektronski fakultet*, 2009.