Detection of Defects in PCB Images by Numbering, Measurement of Chain Features and Machine Learning

ROMAN MELNYK, PAVLO VIAZOVSKYY
Software Department,
Lviv Polytechnic National University,
12 Stepan Bandera St., Lviv, 79013,
UKRAINE

Abstract: - The approach contains algorithms for determining connection and resistance defects. They are thinning, numbering, comparison of corteges, and measurement of the trace resistance. Imposing a tolerance on the concentrated resistance changes it is possible to mark the suspicious printed circuit boards and calculate data for application of Machine Learning. All software procedures solve the task of data preparation for multi-layer neural networks Brain.js which divides printed circuit boards into two classes: defective and working.

Key-Words: - Printed circuit board, chain, trace, defects, flood-filling, thinning, cumulative histogram, numbering, resistance, area and width of trace, tolerance, Machine Learning.

Received: May 26, 2024. Revised: October 28, 2024. Accepted: November 23, 2024. Published: December 31, 2024.

1 Introduction

Many publications discuss various approaches to detecting defects in printed circuit boards. Some of them are collected in multiple reviews of journal articles and presentations at conferences, for example in [1], [2], [3], [4]. The surveys contain many references to publications describing approaches and methods for detecting and classifying defects in printed circuit boards. Some of them are implemented in powerful complex control systems to prevent damage to manufactured printed circuit boards and surfaces, others are described theoretically. For example, in [5], edge detection using the Sobel operator, image enhancement, and a cross-correlation pattern matching method were developed and described to meet the detection requirements. In the mentioned work and [6], defects are extracted using differential processing between the gray template and image samples.

Other groups of approaches contain traditional image processing algorithms including subtraction algorithms, grayscale statistical matching methods, and division into sub-block images [7], [8]. For inspection of inaccuracies in PCB production, traditional image processing-based, machine learning-based, and MATLAB-based defect detection methods are discussed in [9], [10], [11].

A lot of publications consider Neural networks of various architectures and the approaches on their base. In the article [12] a full PCB Defect Classifier that automates the task of detecting and classifying defects in printed circuit boards was developed and shown. Machine learning exists in combination with a full application. In the work [13], a deep model that accurately detects PCB defects from an input pair of a detect-free template and a defective tested image. To train the deep model, a dataset is established which contains 1,500 image pairs with annotations including positions of 6 common types of PCB defects.

The second group of works with Artificial intelligence represents publications [14], [15], [16], [17]. In these works, a deep learning algorithm based on the you-only-look-once (YOLO) approach is proposed. Also, accurate deep learning algorithms, such as convolutional neural networks (CNNs), were used. Machine learning algorithms also process data and predict results in many other tasks [18], [19].

All mentioned approaches differ between themselves by complexity, input data, and characteristics of their implementation.

In the presented work, two groups of approaches are used to solve the given problem: 1) numbering of chains on reference and sample images, finding the coordinates of pixels belonging to the chains, finding and marking chains with communication defects, data preparation to apply machine learning and machine learning to detect defective circuits; 2) detection of edges and determination of the area of circuits, calculation of circuit characteristics and tolerances, calculation of deviation from the standard, preparation of data for the application of machine learning, machine learning for detecting defective circuits.

For mass production, the advantages of the proposed method are the pre-processing of images by algorithms followed by the division into two classes by machine learning software. No special PCB data sets are required.

2 Types of defects in PCB image

Structural elements of printed circuit boards can be manufactured with defects of various sizes and shapes. If they are available, they are located in different parts of the board and affect the functionality of the circuit in different ways. It is difficult to determine what effect they have on the performance of the device. The defect type, shape, size, and location are random and do not match even on two PCBs. Some examples of different types of defects are shown in Figure 1. Four open defects are marked and circled in red. They come in different sizes and shapes. Two short defects are marked and circled in blue.

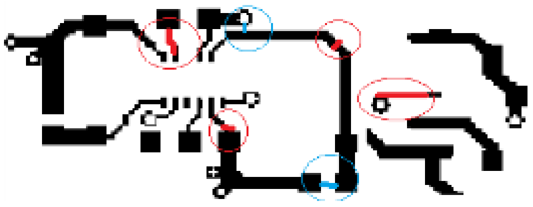

(a) (b)

Fig. 1: Short (blue) and open (red) defects

The problem of determining the type of defects is to select each structural element separately and compare it with the standard. In a simpler version, the element comparison can be replaced by a comparison of their characteristics. In particular, for example, comparing the resistances of circuits. The problem of determining the type of defects is to select each structural element separately and compare it with the standard. In a simpler version, the element comparison can be replaced by a comparison of their characteristics. In particular, for example, comparing the resistances of circuits.

3 Detection of Connection Defects by the Numbering of Chains in Two Images

3.1 Assigning the Identification Numbers

Each printed circuit board has a geometry such that all pins assigned to one circuit are connected, and two or more pins assigned to different circuits are not connected. As a result, they are characterized by a planar topology of wires or traces. Automatic access to each chain is possible only if the coordinates of the pixels in the traces are known. With their help, one chain can be selected and separated from the entire set of black traces on the board. This can be done by changing the color of the selected chain.

Most thinning algorithms work with binary (black and white) images and are reliable and easy to program. A lot of the thinning algorithms are known. Hilditch's algorithm [20] was programmed to hold experiments in this work. Being chosen and programmed it is modified to find the pixel coordinates of specific points (endings and switches). For the PCB fragment in Figure 2(a), the resulting skeleton with endings and switches is shown in Figure 2(b).

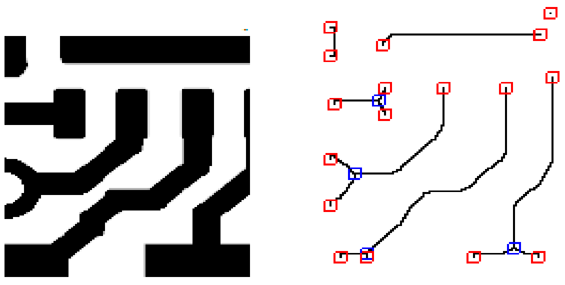

(a) (b)

Fig. 2: PCB image fragment (a), its skeleton with red endings and blue switches (b)

The specific points are combined into the set P_r (r) marks the reference image). They have only coordinates and are located in the order of their definition. There is no information about their relationship with chain numbers, just like there are no chain numbers on the circuit board.

Then the following problem is to be solved: for each element $p_i \epsilon P_r$ to find the serial number of the chain to which it belongs, that is relationship:

$$R: P_r \rightarrow E_r,$$

where E_r is a set of chains, at the beginning E_{r0} is empty, that is, without the identification numbers.

The coordinates of the points in P_r are sorted from the upper left corner of the image to the right and further down and placed into the set S_r:

$$S_r\{p(i)\}=\{\min(x_k), \max(y_k)\}, k=1,\ldots,N, x_0=0, y_0=0,$$
$$i=1,\ldots,N, \quad x_i \in X(s), y_i \in Y(s)$$

where S_r is the sorted set, k is the sequence number of the point in the unsorted P_r.

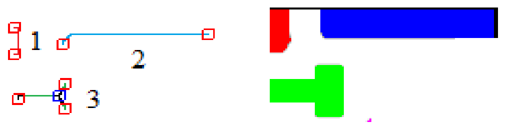
The highest element from the set is accepted as the starting point for the flood-fill algorithm to fill the chain and its tree with colors. For the separation of components, their filling colors are different.

The flood fill algorithm is sometimes called initial fill. The seed is the starting pixel and then other seeds are planted in the area to change the color of the pixels. The algorithm replaces the internal color I_o of the object with the fill color I_n.:

$$\text{If } I(x_i, y_i) = I_o, \text{ then } I(x_i,y_i) = I_n$$

When there are no more pixels of the original interior color, the algorithm terminates. This algorithm is based on a four-join or eight-join method for pixel filling. It searches for all adjacent pixels that are part of the interior. The standard flood-filling algorithm for work requires a uniform color surface when all pixels are of equal intensity. The results of applying the flood-filling algorithm to the chains and skeleton in the PCB image are shown in Figure 3 and Figure 4.

All specific points belonging to the tree change their color, are selected, and get ID=1. In addition, from the chosen starting point, the circuit in the reference image is filled with the same color and receives the same identification number. Then individual points with a changed color are excluded from the list of candidates for assigning identifiers. The next iteration continues with the next coordinate as the starting point. The filling algorithm is applied as many times as there are trees in the skeleton image of the PCB. The first three filled trees (enlarged) are shown in Figure 3(a), where red dots and lines have ID=1, blue ID=2, and green ID=3. For illustration purposes, the surrounding squares remain in the image. The corresponding chains are shown in Figure 3(b).



(a) (b)

Fig. 3: Filled trees with IDs (a) and their chains (b)

when the procedure is completed, each chain and its specific points have the identification numbers. Thus, the set D is divided into subsets of points belonging to only one chain:

$$P(p_1, p_2,\ldots,p_N)=P(P_1,P_2,\ldots,P_n).$$

where p_1, p_2,…,p_N are coordinates of N specific points found after the thinning algorithm, P_1, P_2,…, P_n are n sets of points assigned to the chains.

The extended version of the points decomposition is formulated as follows:

$$P_1=\{(x_{11},y_{11}, c_{11}), (x_{12},y_{12}, c_{12}), (x_{1n(1)},y_{1n(1)}, c_{1n(1),})\}\rightarrow 1,$$

$$P_2=\{(x_{21},y_{21}, c_{21}), (x_{22},y_{22}, c_{22}),.., (x_{2n(1)},y_{2n(1)}, c_{2n(1)})\}\rightarrow 2,$$

$$\ldots,$$

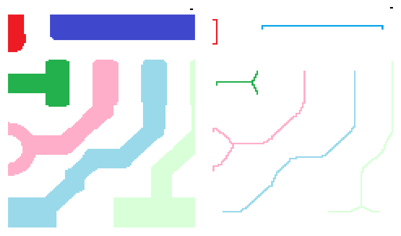
$$P_n=\{(x_{n1},y_{n1}, c_{n1}), (x_{n2},y_{n2}, c_{n2}),.., (x_{nn(1)},y_{nn(1)}, c_{nn(1)})\}\rightarrow n,$$

where c_{i1} denotes a color, and the number of all points is a sum of the corresponding components:

$$N=n(1)+n(2)+\ldots+n(n).$$

The trees of the skeleton in Figure 1(b) and the corresponding chains have the following IDs (with the number of specific points): 1(2), 2(2), 3(4), 4(4), 5(2), 6(4).

Thus, the IDs such as 1, 2, …, n are assigned to the chains and their trees. Figure 4 demonstrates the results of the flood-filling algorithm (without inscription IDs).



(a) (b)

Fig. 4: Filled chains (a) and trees without IDs (b)

The IDs of corresponding chains in the reference image create the cortege of the chain numbers:

$$ID(I_r)=(1, 2, 3, 4, 5, 6).$$

A similar cortege should be formed for the manufactured sample image. In this case, the scheme may contain another number of circuits due to possible short circuits and breaks.

3.2 Determination of Connection Defects

A similar sequence of the considered algorithms is applied to the printed circuit board sample: thinning algorithm, filling, and decomposition of individual points.

The task is to fill the chains of the defective sample with the same colors as the reference image. For this, specific points of the defective sample are sorted. The target color is taken as the color of a pixel in the reference image with the coordinates of the starting point on the defective sample:

$$set\_target\_color\{I_d\{(p(x_i,y_i)\}=get\_color\{I_r\{(p(x_i,y_i)\}.$$

where $I_r\{(p(x_i,y_i)\}$ is a pixel in the reference image, $I_d\{(p(x_i,y_i)$ is a pixel in the defective image.

It is not difficult because the function *get_color*() works with a filled chain on the reference image. All color numbers are stored and checked to avoid double use of the same color.

Figure 5(a) and Figure 5(b) show a fragment of the PCB layout with an open defect and its skeleton. After filling, in Figure 5(c) the number of chains (trees) is seven with two new chains (new colors) compared to six chains in the template image.

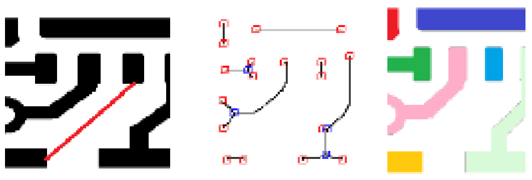

(a) (b) (c)

Fig. 5: PCB fragment with an open defect (a), its skeleton (b), and filled chains (c)

The resulting number of chains indicates wiring defects: a higher number indicates an open defect and a lower number indicates a short circuit defect. It is demonstrated in two cortèges with identifiers of the chains (for images in Figure 4(a) and Figure 5(a)):

$$ID(I_r)=(1, 2, 3, 4, 5, 6, 0, 0),$$
$$ID(I_d)=(1, 2, 3, 4, 0, 6, 7, 8).$$

Here in the first cortege, zero values are added to equalize the sets. This estimate is highly unreliable because the two different types of defects cancel each other out. This case is shown in Figure 6 for two defects.

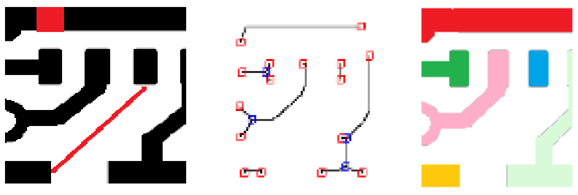

(a) (b) (c)

Fig. 6: PCB fragment (a) with short and open defects, its skeleton (b), and filled chains (c)

Identifiers of the chains in Figure 6 and for the chains in the reference image are given below in corteges:

$$ID(I_d)=(0, 0, 3, 4, 0, 6, 7, 8, 9),$$
$$ID(I_r)=(1, 2, 3, 4, 5, 6, 0, 0, 0).$$

Although the number of chains remained the same, three new chains were generated and three new colors were added. Their positions are new.

The new identifiers form a space for specific subtraction operations and serve to select dissimilar elements. They operate on two variables a, b, and zero. The variables belong to two operands $O_1=(a, 0)$ and $O_2=(a, b, 0)$. Subtraction operations on variables in corteges are only of the following types:

$$a\text{-}a=0,\ a\text{-}0=a,\ 0\text{-}b=0,$$
$$a\text{-}a=0,\ 0\text{-}a=0,\ b\text{-}0=b.$$

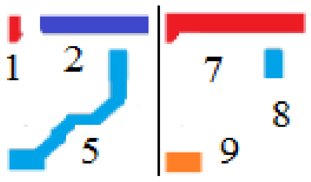
where a denotes the variable of the ID of a correct chain and b denotes the variable of the ID of a defective chain. The first cortege contains only type a, and the second cortege has a and b.

In the space of identifiers, two subtraction operations are formulated as follows:

$$ID_t(r)=ID(t)\text{-}ID(d),\quad ID_d(r)=ID(d)\text{-}ID(t)$$

where $ID(t)$ and $ID(d)$ are identifiers of all chains in the template and sample, $ID_t(r)$ are identifiers describing only those correct chains that are defective in the sample, and $ID_d(r)$ are chains with defects.

In two experiments the first operations give two cortèges: (0, 0, 0, 0, 5, 0, 0, 0) and (1, 2, 0, 0, 5, 0, 0, 0, 0) with numbers of correct chains. The second operations give the following corteges: (0, 0, 0, 0, 0, 0, 7, 8) and (0, 0, 0, 0, 0, 0, 7, 8, 9) with numbers of defective chains. In the case of two defects, the resulting cortèges are represented by the images in Figrue 7.



(a) (b)

Fig. 7: Numbered chains: correct (a), with defects (b)

The following example of a printed circuit board contains four types of defects: missing pad, extra pad, thinner trace, and misaligned traces. All defects are encircled in Figure 8.

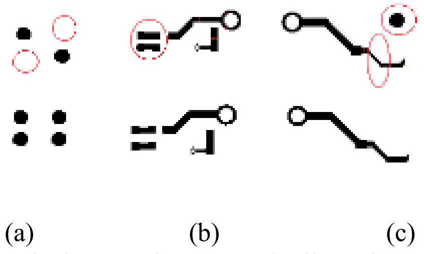


Fig. 8: Missing pads (a), misaligned traces (b), thinner trace and extra pad (c)

The sequence of the algorithms for assigning the identifiers for each chain detects missing and extra pads. The skeleton built on the first step illustrate specific points responsible for these defects in Figure 9.

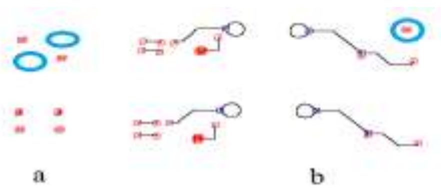


Fig. 9: Missing pads (a), extra pad (b)

Thus, after numbering and comparing the corresponding corteges, missing or redundant pads are detected and displayed as they are marked in Figure 10.

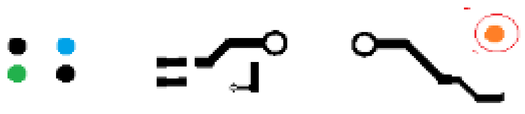


Fig. 10: Missing and extra pads

In summary, skeletons, specific points, filling, and numbering are useful tools for detecting connection defects, missing components, and redundant elements on a PCB. Other types of defects, such as misalignment of traces, and lack or excess of metal in pad traces, require the other approaches to their detection.

3.3 Application of Machine Learning for Classification

When the production process covers a large number of printed circuit boards, the developed approach is ready to prepare data for training and testing artificial neural networks. They will provide an answer to the question of whether the circuit contains connection defects such as a short circuit or a break. Later if needed, the suspected circuit defects are determined and located by the developed software.

Instead of images from two classes, the following approach uses extracted features from two fragments shown in Figure 11. The first image shows the reference PCB, and the second illustrates the defective PCB with two damaged chains. For the application of ANNs, the correct chains are marked with black and defective with red or different colors. In the last case, the input data for ANNs are larger.

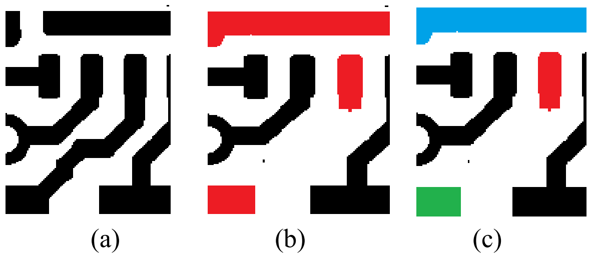


Fig. 11: PCB images: correct (a), with defects (b, c)

Cumulative histograms for the reference and sample images are calculated (Figure 12). The straight blue line corresponds to pixels of all black chains. The red charts are for images with red chains and multi-color chains.

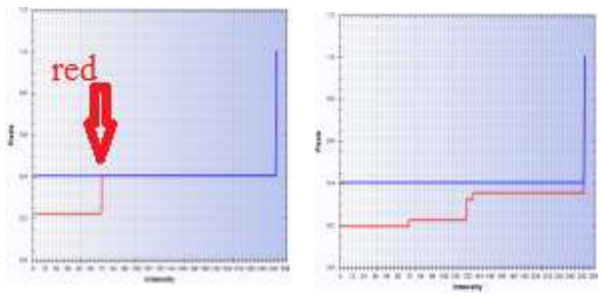


Fig. 12: Cumulative histograms for the reference (blue) and sample image (red)

At first glance, the given diagrams are acceptable and good data for training the ANN to divide printed circuit boards into the class of correct circuits and those with connection defects. Only the pixel count levels may vary from a sample to a sample in the test dataset. It is not known in advance what the size of the schemes will be in the data for training and data for testing. This is a weak point for choosing the level of cumulative histograms.

The red vertical lines of the histogram steps have constant OX coordinates that reflect the intensity of the colors of the defective tracks. Therefore, the color intensity levels can be adopted for ANN training: zero (plus-minus small values) for correct circuits and 80 (plus-minus small values) for circuits with connection defects marked with red. More colors require more intensity values.

So, the PCB images are not required but only their colors of correct and defective chains are used for training and testing of the ANN.

There are some known developed ANN involving Machine Learning on the Internet allowing to be held experiments. Among them, the libraries [21], [22] facilitate the understanding of Neural Networks. The main object responsible for Machine Learning in the Brain,js JavaScript library is *NeuralNetwork*() which has two main methods: *train*() to get the data and build the model, and *run*() to define and present the results.

Machine Learning from JavaScript does not need to enter any additional data such as the number of input data, iterations, accuracy, and others. If any options are not specified, Brain.js will use the default values, which are: *hiddenLayers*: 3, *activation*: 'sigmoid', *learningRate*: 0.3, the maximum number of iterations to train a network is 20000, *errorThresh*: the error threshold to stop the training (0.005), *log*: a boolean value that indicates whether to log the training progress (false), *logPeriod*: the number of iterations between each log (10), *learningRate*: the learning rate to adjust the weights (0.3).

The number of parameters in the methods *train()* and *run()* must coincide.

For two colors of defective chains, the following values are chosen for training and testing:
correct chains - 0, 2, 1, (black colors),
defective chains - 60, 90, 60 (red and blue colors)
testing - 60, 90, 90 (red and blue colors).

Then the JavaScript code snippet looks like this:

```
 const net = new brain.NeuralNetwork();
net.train([
{input:[0, 2, 1], output:{correct:1}},
{input:[0, 0, 0.], output:{correct:1}},
{ input:[60, 70, 80], output:{defect:1}},
{ input:[90, 80, 90 ], output:{defect:1}},]);
// What is the expected output of defect?
let result = net.run([80, 85, 90]);
```

The classification result is as follows:

```
defect: 0.9249435663223267,
correct: 0.07526996731758118.
```

If the input data for the result are within the valid interval, the result will be the opposite:

```
result = net.run([0.1, 0.15, 0]);
defect: 0.06544135510921478,
correct: 0.9348098039627075.
```

In conclusion, this simple experiment confirms that circuit colors indicating the chains with connection defects, along with Machine Learning tools, allow us to classify PCB images into defective and correct samples without involving the user in the process.

4 Detection of Defects by Measurement of Chain Features

4.1 Separation of Chains

In many cases, the traditional subtraction operation can not be applicable for the reasons of the shift in tracks, small resolution, or not an exact match between the reference and produced samples. For example, three fragments of one large printed circuit board contain a large number of elements. After subtracting the reference and manufactured samples of these images, the resulting differences are shown in Figure 13. The pictures show how the fragments differ from each other. The total number of pixels indicating a mismatch can be quite large in various cases.

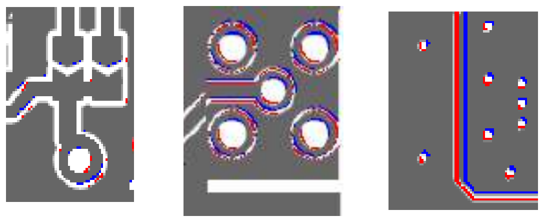
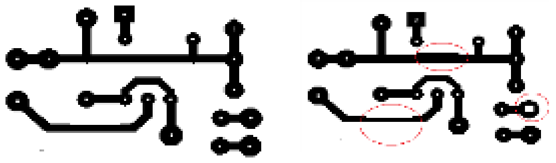


Fig. 13: Three examples of a difference between PCB fragments

It is better to consider an approach based on the analysis of the characteristics of the chain. Figure 14 shows two small images of the printed circuit board: reference and manufactured.


(a) (b)
Fig. 14: PCB images: correct (a), with defects (b)

It is impossible to measure the characteristics of an individual circuit on the image of all black circuits. To do this, you need to consistently allocate one image of a separate circuit for experiments or mark each circuit with a separate color to have access to pixels of this color. In both the first and second cases, it is necessary to use the filling algorithm to mark the chains.

In order to reduce the volume of the article, the second approach is considered here. The result of applying the filling algorithm to the printed circuit board in Figure 14(a) is shown in Figure 15(b).

The filling algorithm requires starting points for its operation. Thus, a thinning algorithm is applied to the PCB image to find the endpoints and switch points of the skeletons. After that, the filling algorithm fills the chains with different colors. These obtained images are shown in Figure 15.

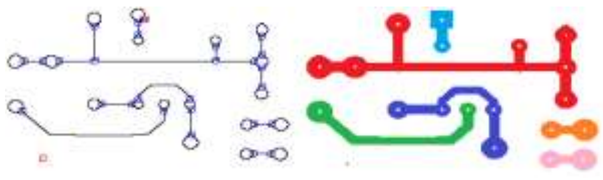

(a) (b)

Fig. 15: Skeleton (a) and filled PCB image (b)

In this experiment with a small PCB image, there is no problem of assigning colors but for large circuits, a special procedure is required.

4.2 Defects of Tracks and Pads

The skeletons and chains are numbered by the previous procedure. Now all chains and their edges are objects for processing. Taking into account the need to construct an edge at the boundary pixels of the trace and simultaneously calculate the length of the edge and the area of the trace, a modified algorithm was developed.

To determine the area of each color and the lengths of the perimeters of the various shapes, a procedure for scanning an image with a quadrate of 2×2 pixels has been developed (Figure 16). In each row, the quadrate moves in steps of one pixel to the right. When the right border of the quadrate reaches the right border of the image, a scanning rectangle becomes with a size of two pixels: a rectangle with dimensions 2×1. Such scanning is carried out *m*-1 times (*m* is the number of image lines). In the *m*-th line, scanning is carried out by a 1×2 rectangle. Figure 16 illustrates the scanning process, as well as the transformation of the scanning areas for the final column, row, and pixel.

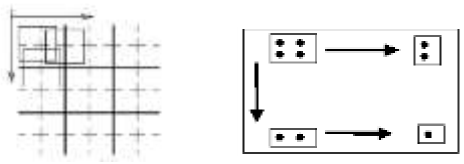


Fig. 16: Moving of the scanning area

A value of the horizontal and vertical components and borders between different colors are accumulated during the scanning.

There are two ways to receive required filled edges: if the edge detection algorithm builds a binary image then all edges are filled with corresponding chain colors; if the chains are in full color the edges are in full color too. Two examples of the binary and colored edges are shown in Figure 17.

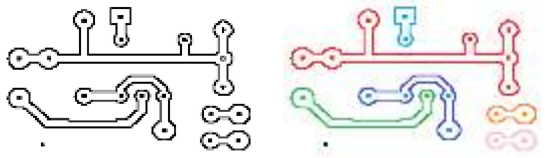

(a) (b)

Fig. 17: Edges: black (a), in full color (b)

The edges are helpful for the calculation of the chain features: the areas of the circuits, and the length of the border (the number of pixels forming it). In addition to them, some derivative features are also calculated: circuit resistance and its deviation from the nominal value.

For the filled PCB image in Figure 15(b), the areas are as follows:
S_1=4977, S_2=2100, S_3=717, S_4=2088, S_5=852, S_6=852.

The edge lengths (the number of pixels) are the following:

L_1=2161, L_2=980, L_3=343, L_4=1047, L_5=406, L_6=406.

The results of the measurement are shown in Figure 18.

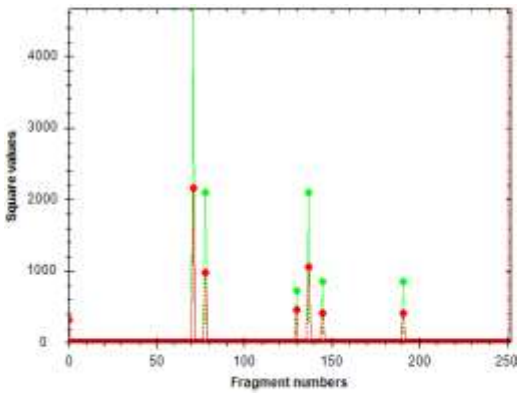


Fig. 18: Areas of chains (green) and values of the edge lengths (red)

A trace's length is twice as short as a border's (edge's) length and the approximate average width of the chain is as follows:

$$W=(S\times 2)/L.$$

For the filled PCB image in Figure 15(b) the widths are as follows:

$$W_1=4.6,\ W_2=4.3,\ W_3=4.2,\ W_4=4.0,\ W_5=4.2,\ W_6=4.2.$$

The values of the widths are slightly different from each other due to the different number of contacts in them. More contacts cause longer average length.

The trace conductance resistance R_s is proportional to the length L of the trace and inversely proportional to its width W:

$$R_s \approx (L/\ W).$$

For the calculated values of widths and lengths the trace resistance coefficients are as follows:

$$R_1=235,\ R_2=114,\ R_3=41,\ R_4=130,\ R_5=48,\ R_6=48.$$

Then these calculations are applied to the printed circuit board image of the manufactured sample.

Figure 19 shows the PCB image with defects and its edges with the same colors.

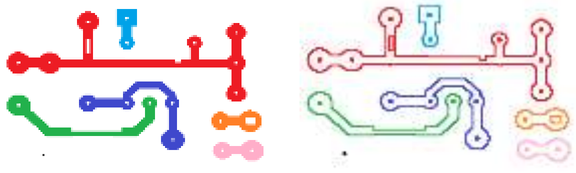

(a) (b)

Fig. 19: Filled chains (a) and edges (b) of the PCB image with defects

Skipping the auxiliary calculations, the following values of resistances of circuits with defects were obtained:

$$R_1=262\ (235),\ R_2=114,\ R_3=41,\ R_4=143\ (130),$$
$$R_5=68\ (48),\ R_6=48.$$

The obtained results show a decrease in the width of the traces and an increase in the resistance in the 1st, 4th, and 5th circuits by 7 (12), 9 (10), and 18 (42) percent, respectively.

The change in resistance of the entire track, caused by the change in the average width of this track, has a distributed character. The larger the trace (area and length), the smaller the difference between the trace values. That is, defects are more clearly visible in smaller traces (by the increased value).

Thus, to check the efficiency of the chain, it is necessary to use the resistance tolerance of a variable value, which depends on the length of the track. The weighting factors consider the spread interval from the largest to the smallest length and the length of the track itself:

$$tol(R(i)) = tol\times[1-(L(i)-L_{\min}))/L_{\max}]$$

where $L_{\max}$, $L_{\min}$ are the largest and smallest lengths, $tol(R(i))$ is the tolerance value for the i-th chain, and tol is the tolerance value from the user.

For example, for the first chain $tol(R(1))$ =1.5 and the 5-th chain $tol(R(5))$=10 when the input tolerance value equals 10.

The two types of defects affect resistance diametrically oppositely: a lack of metal increases, and an excess of metal decreases the resistance in the trace. Therefore, criterion functions for comparison of the features are formed separately for each case, for example, for resistance:

$$|R_r\ (i) - R_s^-(i)| <= tol(R),$$
$$|R_r(i) - R_s^+(i)\ | <= tol(R),$$

where $R_r(i)$ is the resistance of the i-th chain in the reference image, $R_s^-(i)$, $R_s^+(i)$ are the smaller and larger of the resistance of the i-th chain in the manufactured scheme.

The last stage of the chain elaboration algorithm is the determination of the defect chain location. All chains in both images are numbered with the same numbers. Thus, if a difference between their deviations is within the $tol(R)$ tolerance, the circuit is considered correct. In the opposite case, two chains are selected from the template image and the sample image.

This is illustrated in Figure 20 where three correct chains and three chains with smaller resistance are selected.

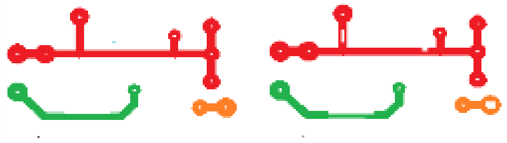

(a) (b)

Fig. 20: Three chains: correct (a) and with defects (b)

In conclusion, the conducted experiments with chains and their edges showed that the resistance of the chains is a criterion function for the selection of defective chains and checking the functionality of the printed circuit board.

4.3 Data for Application of Machine Learning

A resistance of chains is a base to form data for ANN to divide the printed circuit boards into two classes: correct and with resistance defects.

For the reference image, a set of resistances for all chains is formed:

$$R=\{R_1, R_2, \ldots, R_n\}.$$

All chain resistances are presented in relative form:

$$R_i= R_2(a)/ R_i(s) \quad i=1,2,\ldots,n,$$

where $R_2(a)$, $R_i(s)$ are available and standard values of the chain resistance, and $R_i(a)=R_i(s)$ in the reference image.

Then a set of n units is formed:

$$R=\{1, 1, \ldots, 1\}.$$

This is the first set for learning ANN according to the correct schemes. The second should take into account the resistance tolerances at which the circuit is considered operational. They can be positive, negative, or both. In general, they can also have different values. For example, $D=\{1.2, 0.8, \ldots, 0.7\}$ etc. As a rule, in practice, no one plans different tolerances for different circuits but indicates the maximum tolerance value for all circuits (or two: positive and negative). Two intervals of the resistance values: with tolerances acceptable for the correct PCB and those belonging to the defective resistances are shown in Figure 21.

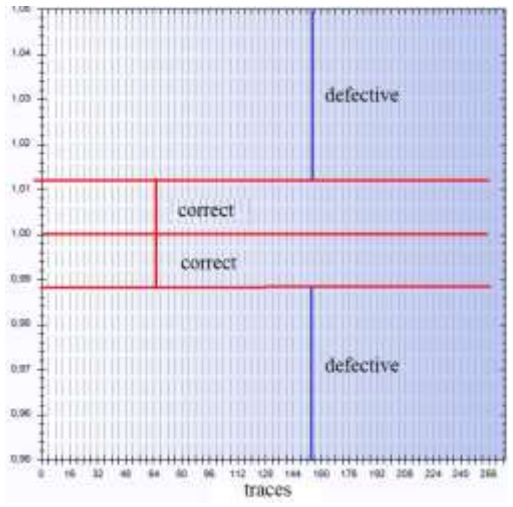


Fig. 21: Intervals of tolerance for correct and defective chains of the circuit

The OY coordinates are randomly selected as representatives of the correct patterns for learning the ANN. The OY coordinates from the two outer intervals are randomly selected as representatives of the circuit with resistance defects. So, the PCB images are not required but only their features of chains are used for training and testing of the ANN.

There are some developed ANNs including Machine Learning on the Internet to support experiments. Among them, Brain.js is a JavaScript library that makes it easy to understand Neural Networks. The main object responsible for Machine Learning is *NeuralNetwork*() with two principal methods: *train*() to obtain input data and *run*() to obtain results.

Input data such as the number of input data or iterations, accuracy, graphs of losses, achieved accuracy and other information are in default.

Now, instead of images from two classes as in the work [23], the following approach uses extracted tolerance features. For the three tolerance intervals, the following values are chosen for training and testing:
correct chains-1, 1.2, 0.8, 1.0, 1.1, 0.9 (middle interval),
defective chains-1.4, 1.5, 1.45 (higher interval),
defective chains-0.5, 0.6, 0.7 (lower interval),
testing-1.3, 1.4, 0.7 (from two intervals).

Then the JavaScript code is formed like this:

```
 const net = new brain.NeuralNetwork();
net.train([
{input:[1, 1.2, 0.8], output:{correct:1}},
{input:[1.0, 1.1, 0.9], output:{correct:1}},
{ input:[1.4, 1.5, 1.45], output:{defect:1}},
{ input:[0.5, 0.6, 0.7], output:{defect:1}},]);
// What is the expected output of defect?
let result = net.run([1.3, 1.4, 0.7]);
```

The classification result is as follows:

```
defect: 0.9824727773666382,
correct: 0.017612572759389877.
```

If the input data for the result is within the valid interval, the result will be the opposite:

```
result = net.run([1.1, 1.15, 0.85]);
correct: 0.9825793504714966,
defect: 0.017426729202270508.
```

In conclusion, this simple experiment confirms that circuit features such as resistance and tolerance, along with Machine Learning tools, are the basis for the classification of PCB images into defective and correct classes without involving the user in the process.

4.4 Defects of Misaligned Traces

The former approach defines each PCB circuit as one of four possible types: correct and the same as in the template PCB; chains of the wrong shape or with the wrong metal filling; circuits with connection defects.

The last stage of the defect detection algorithm is to measure the distances of possible component displacements on the manufactured printed circuit board compared to the template. A working model for the algorithm is the same as for the resistance analysis that is without the chains with the connection defects. Each chain is characterized by the coordinates of the center of mass with pixel precision:

$$\overline{x}(s) = 1/k_s \sum X(s), \overline{y}(s) = 1/k_s \sum Y(s),$$
$$x_i \in X(s), y_i \in Y(s)$$

or with a rectangle precision:

i=, j=.

$$C_n = (i-1) * L_x + \text{j}$$

The size of printed circuit boards for experiments is 56 × 65 mm, and their image has a size of 450×580 pixels, which means that one square millimeter corresponds to 8×9 pixels. One pixel along the OX and OY axes represents a size of approximately 0.1×0.1 mm on the board surface. Thus, the pixel accuracy in practical cases is not satisfied by the manufacturing process. The accuracy of the rectangle is controlled and depends on its dimensions. For example, for a size of 5×5, one rectangle represents a size of approximately 0.5×0.5 mm on the surface of the board.

Two fragments of a large image of a printed circuit board from [1] are shown in Figure 22 (a - correct, b - with a defect). Only two small chains are marked with red and blue fill. The defect consists in the incorrect relative location of these two chains.

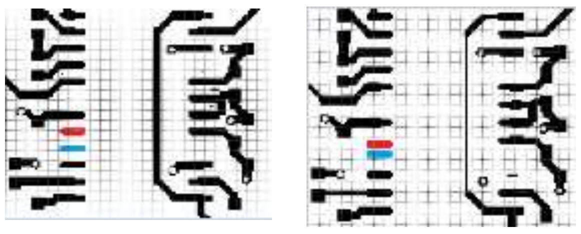

(a) (b)

Fig. 22: PCB fragments: correct (a) and misaligned (b)

Two images are covered by a grid with a size 150×150 which means that one cell has an approximate size of 3×4 pixels or 0.3×0.4 mm. This size is comparable to the width of traces and is acceptable for measuring distances between board components.

Calculation of centers of mass for marked two chains gives the following results:

correct: c_r={35, 115). c_b={35, 119),
defective: c_r={35,116). C_b={35, 118).

where c_r. c_b are centers of mass of red and blue chains, and the coordinates are given in the numbers of columns and rows in which the cell is located.

Thus, in the correct image distance between two chains is four cells of a grid, and in the defective case two cells or twice less. On the board, the correct distance is 1.6 mm and the incorrect is 0.8 mm. The decision about the accessibility of a smaller distance is taken due to comparison with the tolerance value.

To measure distances between the closest pairs of chains the following procedure is developed. At the beginning, all centers of mass for all chains in the template PCB image are calculated:

$$C_t=(C_{t1}, C_{t2}, C_{t3},\dots,C_{tn}),$$

where $C_{ti}=(x_{ti}, y_{ti})$ are the coordinates of the *ti*-th center of mass.

Then in the sample PCB image for all chains with the same numbers as in the template the centers of mass, are also calculated:

$$C_s=(C_{s1}, C_{s2}, C_{s3},\dots,C_{sn}),$$

where $C_{si}=(x_{si}, y_{si})$ are the coordinates of the *si*-th center of mass for the chains in the sample image. These sets may contain a different number of components since in the second case circuits with open and short defects are excluded from consideration as having incomparable properties. But the serial numbers in both sets match.

The elements of the set C_t correspond to the vertices of a complete graph that can be constructed by segments connecting them. The lengths of the segments correspond to the distance between the centers of mass of the chain. Their number is n^2 (n is the number of chains).

To reduce the checking calculation the output sequence of segments is determined as n-1 edges of the Minimum Spanning Tree (MST) in the fully connected graph. To find this tree, Prim's algorithm, [24] is applied to the full graph. The examples of such trees for the correct and defective fragments are shown in Figure 23. Their vertices are located in the centers of mass of chains.

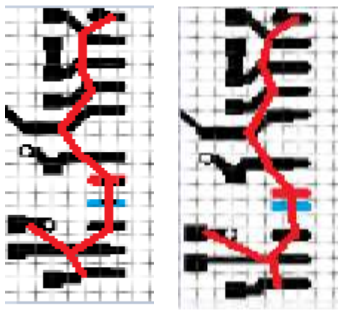

(a) (b)

Fig. 23: Two fragments with MST: correct (a) and with misaligned tracks (b)

Figure 26 shows that if one edge has a decreased length, the other edge has an increased length. The same applies to the shifted positions of individual tracks.

For further operations, the tree itself is not needed in the sample image, because it is used as an array of edge weights L_{ij} with incident vertices n_i, n_j indicating chain numbers:

$$n_1(\mathrm{t}), n_2(\mathrm{t}), L_{12}(\mathrm{t}), \ldots, n_i(t), n_j(t), L_{ij}(t), \ldots$$

A sample image does not require an MST search. The array found earlier for the standard indicates for which chains the distance between their centers of mass is calculated:

$$n_1(s), n_2(s), L_{12}(\mathrm{s}), \ldots, n_i(s), n_j(s), L_{ij}(s), \ldots$$

The last step of the algorithm is the comparison of distances and the selection of pairs of schemes, the distances between which differ by more than the permissible values for the corresponding distances between the components of the printed circuit board:

$$|L_{ij}(t)\text{-}L_{ij}(s)|\leq tol. \quad n_i, n_j \in\{1, 2. 3, \ldots, n\}.$$

The user sets the tolerance value in a relative form since the distances have different values. The difference sign indicates a decrease or increase in distance values compared to the standard image.

5 Experiments

Two developed algorithms were tested on fragments of the printed circuit board from [1] with six defects: two of connection, three of lack of metal, and one of incorrect placement. Corresponding defects are shown in Figure 24(a) by red, blue, and green lines. The original circuit is shown in Figure 24(b) for comparison.

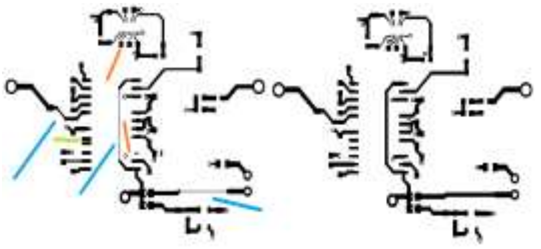

(a) (b)

Fig. 24: PCB fragments: with defects (a), etalon (b)

Increased places with connection defects are shown in Figure 25(a) and their presentation by skeletons with specific points in Figure 25(b).

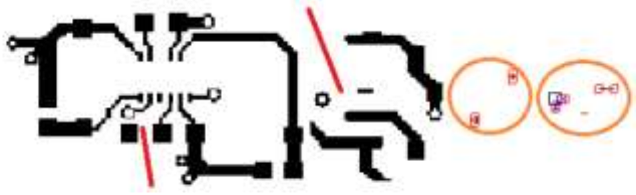

(a) (b)

Fig. 25: Two defects of open defects in traces (a) and specific points on the skeleton (b)

Thus, after numbering and comparing the corresponding corteges, missing or redundant pads are detected and displayed, and thinner traces are detected and marked in Figure 26.

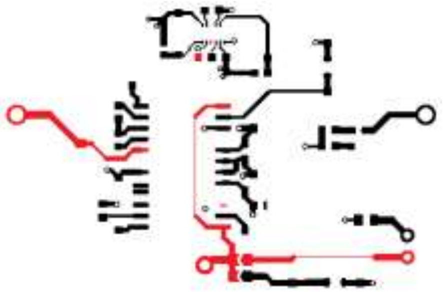


Fig. 26: Marked defects in PCB image

In conclusion, three types of printed circuit board defects are considered: connections and missing (redundant) elements, non-standard sizes of traces, and incorrect arrangement of elements. According to different types, three approaches are used for their detection.

6 Conclusion

The work is devoted to the determination of types of defects in PCB images, to detect operational and defective circuits and locations of the defective chains. The algorithms for the determination of features of the considered images such as the cumulative histograms of correct and defective chains, and the tolerance values on resistances of PCB chains are developed. Tolerances include the proposed from the user values and calculated for the manufactured PCB sample. Found features are input data for Machine Learning algorithms which may be applicable for large sets of manufactured PCBs. The multi-layer ANN application to learn and recognize defective PCB images is based on the powerful instrument Brain.js which has many control input parameters for choosing the architecture, accuracy, losses, and time for processing. This ANN can successfully classify the PCB images, where the aspect is placed on the range of colors and tolerances.

The approach includes such algorithms and methods as the numbering of chains, the filling algorithm for selecting and separating chains, formulas for special subtraction of corteges, and calculating the resistance characteristics of circuit components.

The proposed approach contains operations for the selection of two classes of defects: connection and incorrect resistance. Defective chains and resistances are used to train the multi-layer neural network for further automatic division of the PCB images into two classes: having defects and without defects.

So, the approach combines two mechanisms: processing by developed software and Machine Learning by data transmitted from the program. It reduces the time consumed by the developed software and excludes the user from the monotonous process of quality inspection.

References:

[1] Elena Jasiuniene, Renaldas Raišutis, Vykintas Samaitis and Audrius Jankauskas. Comparison of Different NDT Techniques for Evaluation of the Quality of PCBs Produced Using Traditional vs. Additive Manufacturing Technologies. A survey. *Sensors,* 2024, 24, 1719, DOI: 10.3390/s24061719.

[2] Ebayyeh A.A.R.M.A., Mousavi, A., A Review and Analysis of Automatic Optical Inspection and Quality Monitoring Methods in Electronics Industry. *IEEE Access,* 2020, Vol.*8*, pp.183192–183271, DOI: 10.1109/ACCESS.2020.3029127.

[3] K. P. Anoop, N.S. Sarath and V. V. Sasi Kumar, Review of PCB Defect Detection Using Image Processing, *International Journal of Engineering and Innovative technology (IJEIT),* 2015, Vol. 4, Is. 11, pp. 188-192, ISSN: 2277-3754.

[4] D.B. Anitha, and M. Rao, A survey on Defect Detection in Bare PCB and Assembled PCB using Image Processing Techniques, *International Conference on Wireless Communications, Signal Processing and Networking*, Chennai, India, 2017, pp. 39-43, DOI: 10.1109/WiSPNET.2017.8299715.

[5] Cai L., Li J., PCB defect detection system based on image processing, *Journal of Physics: Conference Series*, 2022, Vol. 2383, No. 1, pp. 012077,IOP Publishing, DOI: 10.1088/1742-6596/2383/1/012077.

[6] Zhu A. Wu and X. Liu, Printed circuit board defect visual detection based on wavelet denoising, *IOP Conference Series: Materials Science and Engineering*, 2018, Vol. 392, Iss. 6, pp. 062055, DOI: 10.1088/1757-899X/392/6/062055.

[7] Y. Hanlin and W. Jun, Automatic Detection Method of Circuit Boards Defect Based on Partition Enhanced Matching, *Information Technology Journal*, 2013, Vol.12, Iss.11,, pp. 2256-2260, DOI: 10.3923/itj.2013.2256.2260.

[8] Vikas Chaudhary, Ishan R. Dave and Kishor P. Upla, S. V., Visual Inspection of Printed Circuit Board for Defect Detection and Classification. *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET),* Chennai, India, 22-24 March, 2017, pp. 732-737, DOI**:** 10.1109/WiSPNET.2017.8299858.

[9] F. B. Nadaf and V. S. Kolkure, Detection of Bare PCB Defects by using Morphology Technique, *Morphology Technique International Journal of Electronics and Communication Engineering,* 2016, Vol. 9, No. 1, pp. 63-76, DOI: 10.23919/ChiCC.2017.8029117.

[10] J.Nayaka, K. Anitha, B.D. Parameshachari, R. Banud and P. Rashmi, PCB Fault Detection Using Image Processing, *IOP Conference Series: Materials Science and Engineering*, 2017, Vol. 225, pp. 1-5, DOI: 10.1088/1757-899X/225/1/012244.

[11] Ling Q., Isa N.A.M., Printed Circuit Board Defect Detection Methods Based on Image Processing, Machine Learning and Deep Learning: A Survey. *IEEE Access,* 2023, Vol. 11, pp. 15921–15944, DOI: 10.1109/ACCESS.2023.3245093.

[12] Medium. Building an End-to-End Defect Classifier Application for Printed Circuit Boards, [Online], https://medium.com/towards-data-science/building-an-end-to-end-deep-learning-defect-classifier-application-for-printed-circuit-board-pcb-6361b3a76232 (Accessed Date: November 10, 2024).

[13] Sanli Tang, Fan He, Xiaolin Huang, Jie Yang, Online PCB Defect Detector On A New PCB Defect Dataset. arXiv: Computer Vision and Pattern Recognition. https://arxiv.org/abs/1902.06197.

[14] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks, *Electronics*, 2020, Vol. 9, No. 9, pp. 1547, DOI: 10.3390/electronics9091547.

[15] Abhiroop Bhattacharya, Sylvain G. Cloutier, End-to-end deep learning framework for printed circuit board manufacturing defect classification, *Scientific Reports*, 2022, [Online]. https://www.nature.com/articles/s41598-022-16302-3 (Accessed Date: November 10, 2024).

[16] Jungsuk Kim, Jungbeom Ko, Hojong Choi and Hyunchul Kim, Printed Circuit Board Defect Detection Using Deep Learning via A Skip-Connected Convolutional Autoencoder, *Sensors,* No.21(15), 2021, 4968, DOI: 10.3390/s21154968.

[17] Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., & Gupta, S. K. (2021), Image-based surface defect detection using deep learning: A review. *Journal of Computing and Information Science in Engineering*, 21(4), 040801. DOI: 10.1115/1.4049535.

[18] C. L. Perera, S. C. Premaratne, An Ensemble Machine Learning Approach for Forecasting Credit risk of Loan Applications, *WSEAS Transactions on Systems, vol. 23, pp. 31-46, 2024*. DOI: 10.37394/23202.2024.23.4.

[19] Atul Kachare, Mukesh Kalla, Ashutosh Gupta, Visual Question Generation Answering (VQG-VQA) using Machine Learning, *WSEAS Transactions on Systems*, vol. 22, pp. 663-670, 2023, https://doi.org/10.37394/23202.2023.22.67.

[20] Hilditch's Algorithm for Skeletonization, [Online]. https://cgm.cs.mcgill.ca/~godfried/teaching/projects97/azar/skeleton.html (Accessed Date: November 10, 2024).

[21] Machine Learning in JavaScript, [Online]. https://www.w3schools.com/ai/ai_whatis.asp (Accessed Date: November 10, 2024).

[22] An end-to-end platform for machine learning, [Online]. https://www.tensorflow.org/ (Accessed Date: November 10, 2024).

[23] Melnyk Roman, Vorobii Vitalii, PCB Image Defects Detection by Artificial Neural Networks and Resistance Analysis. *WSEAS Transactions on Circuit and Theory*, 2023, 22(1), pp. 35–42, https://doi.org/10.37394/23201.2024.23.7.

[24] Prim's Algorithm, [Online]. https://www.programiz.com//dsa/prim-algorithm (Accessed Date: November 10, 2024).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)
The authors contributed in the ratio of 70 (Melnyk) to 30 (Viazovskyy) in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself
No funding was received for conducting this study.

Conflict of Interest
The authors have no conflicts of interest to declare