Non-intrusive Embedded Systems Anomaly Detection using Thermography and Machine Learning

José Paulo G. de Oliveira
Universidade Federal de Pernambuco
Universidade de Pernambuco
Recife, Brazil
ORCID 0000-0001-9438-6829

Carmelo J. A. Bastos Filho
Universidade de Pernambuco
Recife, Brazil
ORCID 0000-0002-0924-5341

Sérgio Campello Oliveira
Universidade de Pernambuco
Recife, Brazil
ORCID 0000-0003-1058-1139

***Abstract*—Quality control in electronic system manufacturing is achieved mainly through system testing. Device miniaturization and multilayer Printed Circuit Boards have increased the electronic circuit test complexity considerably and processes based on manual inspections have become outdated and inefficient. The concept of Industry 4.0 has enabled the manufacturing of customized products based on customers' demands, which demands a high degree of flexibility in production processes, with low cost and without placing numerous test points. In this paper, we propose two automated test solutions based on machine learning and thermographic analysis. We propose deploying autoencoders and random forest in two different manners to detect firmware or hardware anomalies based on the circuit board's temperature signature. We validate our proposal using two firmware versions running independently on the test board. We obtained an anomaly detection rate above 98%. In the random forest approach, we require all data classes for training, whereas the autoencoder only requires the reference class, which is expected in real scenarios.**

***Keywords*— Anomaly detection; embedded systems test; thermography; autoencoders; deep learning; random forest.**

I. Introduction

Testing embedded systems is an essential task during the design and manufacturing phases [1], which ensures system correct operation and avoids rework and economic losses [2]. Anomalies in embedded systems have several causes, such as defective components, improper assembly, welding defects and software-related errors. Testing techniques commonly used for anomaly detection are automated, where computers process images or signals of the system to be tested.

Meanwhile, the concept of industry 4.0 [3] poses an additional challenge for system testing: a high level of product customization [4][5]. The use of intrusive test methods that require test points on the circuit or flying probe analyzers shows many disadvantages [6], such as high cost and complexity. On the other hand, a flexible and non-intrusive method, capable of detecting defects in both hardware and firmware, shall provide more agility and reliability to the design flow of embedded systems [7].

In this work, we present an automated and non-invasive test approach based on the thermographic signature of the embedded system. The core of the system is a machine learning model called autoencoder [8]. Machine learning has become a frequently applied tool for solving complex problems, especially where there is a large data availability. In our work, we obtained the required data using commercially available thermographic cameras. The proposed method detects anomalies without the need of a physical connection to the tested system. Autoencoders are neural networks trained to reconstruct in the output a pattern similar to that presented at the input. Therefore, anomalies can be detected by reconstructing and comparing images.

This study was financed in part by the Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco (FACEPE) – APQ-0532-3.04/1.

The simplified architecture of the proposed test system is illustrated in Fig. 1. The system to be tested is called DUT (Device Under Test). The information collected from the DUT that is processed by the test system detects anomaly is obtained wirelessly. Hence, changing the DUT model during the manufacturing process can be done seamlessly. An identification tag can be used to inform the test system which DUT model is currently being tested. This is accomplished in an automated fashion.

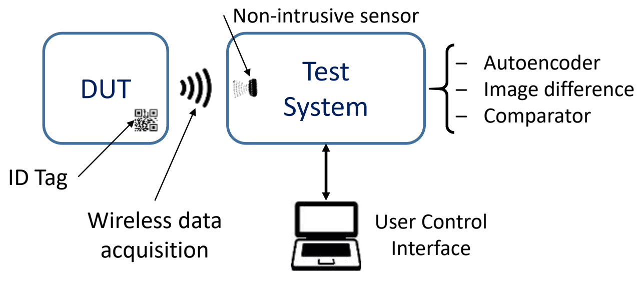


Fig. 1. Anomaly detection system architecture.

The test system consists of a previously trained autoencoder, which calculates the difference between the autoencoder's input and output images, and a comparator. The comparator determines whether the DUT passes or fails the test according to the image difference magnitude. Further details are outlined in section III and Fig. 2.

The images used for training and anomaly detection, however, are not the thermal images themselves. Thermographic information is obtained from the captured images using a specific computer application. The application generates a graph indicating the temperature variation in a determined DUT's region as a function of time. From that graph, a spectrogram is constructed. The spectrogram is the image used by the autoencoder.

In order to properly detect anomalies, the autoencoder only must be trained with images from the non-anomalous class. Thus, the proposed solution has the following characteristics: simplicity of implementation and flexibility.

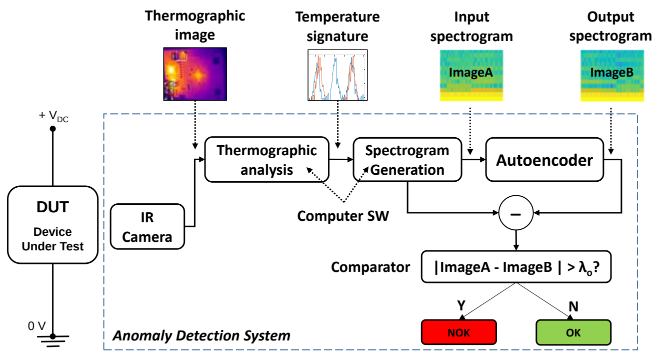


Fig. 2. Summary of the operation of the anomaly detection system with autoencoder: the infrared camera captures the DUT's thermal image; the temporal temperature signature is plotted; the spectrogram is obtained from this signature and converted to imageA; imageA is used as input of a trained autoencoder, which reconstructs it as imageB; the difference between imageA and imageB is compared to a threshold λ_o, if $|\text{imageA-imageB}| > \lambda_o \Rightarrow$ Fail.

To validate the proposed approach, we built an experimental setup using an embedded system development board. We have implemented two classes of firmware that can be run independently on the test board. One of these versions is free of anomalies (FW_{OK}) and the other emulates a system with anomalies (FW_{NOK}).

Given the promising results and the attractive characteristics of the proposed system, namely flexibility and effectiveness, we believe that using the proposed solution application in a real manufacturing environment shall improve the embedded system design and testing process, especially in highly customizable electronic products.

The rest of the article is structured as follows. We discuss related works in Section II. We describe the proposed approach in Section III, where we address the theory of thermography and the machine learning techniques explored in this work: autoencoders and random forest. Section IV discusses the implementation aspects. We present the experimental results in Section V. Finally, we present our conclusions outline related future work in Section VI.

II. Theorical Background

Embedded systems are computational systems typically manufactured and mounted on Printed Circuit Boards (PCBs). They consist of (at least): a processor, memory, input/output interfaces, and an interconnection system [9]. These four hardware subsystems work collaboratively to ensure a proper execution of stored programs, which we call firmware. Embedded system complexity, size, and computational power vary enormously and depend on the final application and objective.

An embedded system defect occurs if any parameter or function does not attend to the manufacturer's functional criteria. Such defects can be categorized according to the extent of performance degradation they may cause. A usual metric is counting the percentage of defect PCBs related to the total amount of tested boards for quantitative analysis. In testing, we denote the system under design as Device Under Test (DUT). A set of selected input patterns – so-called test patterns – is applied to DUT's input for testing purposes. Then, the output behavior is compared with the expected behavior. Currently, even modest PCBs feature a high component density. Therefore, automated techniques are required to overcome human limitations applied in manual inspection.

Since embedded systems consist of hardware and firmware, manufacturers should expect both elements to present anomalies. Testing hardware and firmware separately is a common approach. Nevertheless, it is a two-step work. Furthermore, test pattern generation is usually based on fault models [10][11]. Unfortunately, while good fault models exist for hardware testing, the same type of model for embedded software testing is unavailable [12].

Regarding hardware testing, the most common types of defect embedded systems present are PCB mounting, faulty component, and final product malfunction [1]. A myriad of hardware test methods can be employed depending on financial constraints and technical barriers. One of the most reliable PCB test methods is using a fixture or bed of nails [13], where fixed probes power up and actuate the board's circuitry. A less expensive similar approach is the use of flying probes [14]. In this case, needles attached to a probe on an x-y grid match the circuit board. Alternatively, digital image processing has also been efficiently employed in circuit testing. Computer vision techniques apply a camera to capture PCB images that are algorithmically analyzed [15][16]. For PCB's internal anatomy inspection, x-ray images are frequently employed [17]. Automated Optical Inspection (AOI) refers to test techniques where PCB images are compared to a reference image, and dissimilarities denote possible defects [18][19]. Thermography is a similar approach to AOI, where infrared images are used to detect defects [20].

Non-invasive test approaches have been published for hardware testing. However, the proposed solution is restricted due to specific device/circuit characteristics [21] or data availability [22].

Regarding software test, automation techniques have been developed and systematically used in the industry [23]. The solutions proposed in [23], [24] and [25] use either machine learning models or artificial neural networks. However, as in high-level software, the objective is to predict the occurrence of failures and not detect anomaly in an embedded system. Moreover, the analysis is based on software metrics [26].

III. Proposed Approach for Anomaly Detection

Our proposed method is an adaptation of [27], in which the electric current consumed by the DUT is used to generate the spectrogram images instead of the thermal signature [28] (a more conventional current signature approach can be found in [29]). In this section, we provide a description on how the anomaly detection is performed. The block diagram of the proposed system is illustrated in Fig. 2. First, we acquire a region of interest (ROI) of the thermal image that belongs to the circuit to be tested. The maximum temperature in the ROI is plotted using a computer software during a specific time interval ΔT (acquisition window). Then, we determine the temporal behavior of temperature in that region, and the temperature as a function of time is plotted. This graph is called a sample. Then, we generated the spectrogram from that sample. The spectrogram image serves as input data for a previously trained autoencoder, which attempts to reproduce a copy of the input. Next, we calculate the difference between the autoencoder's output and input images using mean square error. The anomaly detection algorithm compares this difference to a decision threshold λo. Therefore, the test result (pass or fail) depends on the similarity between the autoencoder input and output. Fig. 3 illustrates the ROI definition and the thermographic processing. The thermographic sample generation process is further explained in section IV.B.

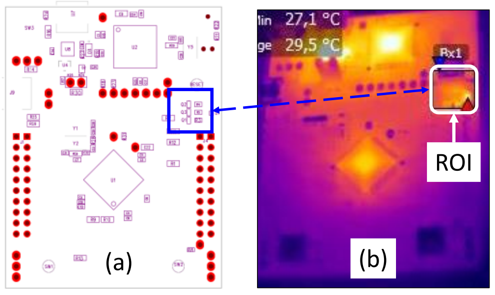


Fig. 3. PCB region of interest. The maximum temperature in the marked square (a and b) is acquired and plotted as a function of time.

In order to ensure that the built system has the required flexibility, several of the parameters shown in Fig. 2 are adjustable: ΔT - acquisition window; Construction of the spectrogram - image size (height x width); frequency range on the y axis; Autoencoder - number of layers; training and construction parameters of the neural network; Comparator - decision threshold λo. Additionally, multiple acquisitions can be carried out in the same test. This allows the test to be repeated for the same DUT for a certain number of times. Despite increasing the test time, this increases the anomaly detection reliability.

A. *Thermography*

Physically, the heat can be treated as an infrared ray. The infrared wavelength range is generally considered to be from 0.7 to 100 μm. Infrared rays are spontaneously irradiated by all objects with a temperature above absolute zero. The emitted energy indicates the temperature of the object.

The equipment used in thermography consists of a special infrared camera that scans the object and a computer application. The camera unit contains an optical system that scans the field of view at a very high speed and focuses the infrared radiation on a detector that converts the radiation signal into an electrical signal.

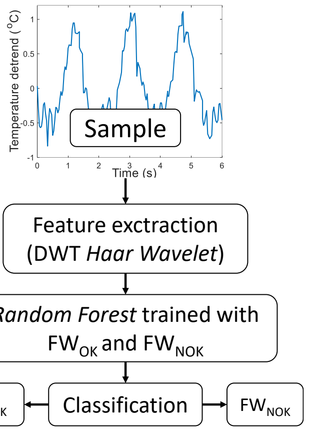
The computer application processes the captured signal. In our experiment, we used a FLIR T530 infrared camera and the FLIRTools+ software [30] for data processing.

B. *Autoencoders*

Autoencoders can be seen as artificial neural networks trained to learn a representation of the input data and, based on this representation, reconstruct the input data on its output [8]. The representation of information, called coding, can be used for various purposes, such as dimensional data reduction or anomalies detection [31]. An autoencoder consists of an input layer, one or more hidden intermediate layers, and an output layer. Traditionally, these networks have a symmetrical structure, in which the input and output layers have the same number of neurons. Autoencoder implementation involves three steps: definition of the structure (size, number of neurons, number of layers, etc.); training; and validation.

C. *Random Forest*

Random Forests are a widely-used machine learning model, which can be trained to perform multiclass classification. In this work, to obtain a basis of comparison with the results of the proposed system – based on autoencoders – we analyzed experimental samples with a Random Forest model [32]. For the sake of simplicity, we built a classifier and trained only with two types of firmware: FW_{OK} and FW_{NOK}. The characteristics of the samples were previously extracted through wavelet analysis (DWT – Discrete Wavelet Transform) to improve the quality of the classification process [33]. Fig. 4 shows the Random Forest and DWT based classifier. Briefly, the classifier reads the samples, extracts their features and sends them to the trained model for classification. The extracted features are wavelet coefficients. In this experiment, the data were split in a 75% and 25% ratio for Random Forest training and test, respectively.



Temperature detrend (°C)
Time (s)
Sample
Feature exctraction (DWT *Haar Wavelet*)
Random Forest trained with FW_{OK} and FW_{NOK}
Classification
FW_{OK}
FW_{NOK}

Fig. 4. A temperature signal sample is applied to the feature extraction block; the coefficients generated by the DWT are processed by the Random Forest, which classifies the sample in one of the two classes: FW_{OK} or FW_{NOK}.

IV. Experimental Setup

In this section, we describe the design and building of an experimental validation system certify the anomaly detection efficiency. First, we detail the implementation of a DUT, which provides the data used for the validation tests. Then, we show how the information is acquired, pre-processed and used to train the models for anomaly detection, based on Random Forest and Autoencoder.

A. *Device Under Test - Design and Construction*

The DUT consists of two parts: hardware (Fig. 5-a) and software (Fig. 5-b).

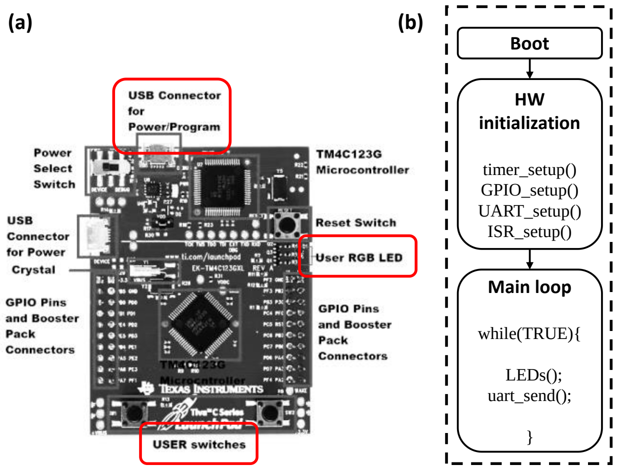


Fig. 5. Implementation of the DUT. Development. Test software composed of infinite loop and LED activation functions and sending information via serial interface (UART).

The hardware was implemented in an ARM Cortex M4F microcontroller evaluation platform from Texas Instruments [34]. The development board provides multiple general-purpose interfaces (GPIO – General-Purpose Input/Output), which can emulate typical embedded systems applications. The firmware has the typical structure of an embedded application. It has an initial hardware bootstrap followed by an infinite loop. The system periodically activates an RGB (Red Green Blue) LED and performs data transmission via serial interface. In experimental tests, two firmware versions were implemented. One version represents a simulated operational form of the system. A summary of these versions can be seen in TABLE I.

TABLE I. TEST FIRMWARE VERSIONS AND THEIR BEHAVIOR.

Designation	**FW_{OK}**	**FW_{NOK}**
Expected Tasks	Periodic RGB LED blinking	Periodic RGB LED blinking
Modification (simulated anomaly)	None (anomaly free)	LED activation delay + UART data transmission

The implemented firmware behavior, although quite simple, is typical for reactive embedded systems [9]. Such systems execute cyclically, reading and triggering GPIO pins and transmitting/receiving information over one or more communication interfaces, such as UART. Two board control keys were used to switch the execution mode between the two firmware versions. In Fig. 6, two examples (excerpts) of the test firmware used in the experiment and the respective thermographic signatures, given as a function of time, are illustrated. A slightly change in the LED blinking cadence – caused by the added delay on FW_{NOK} – and the unexpected UART transmission produces a noticeable impact on the thermographic signature. The generation of thermographic signatures is explained next in section IV.

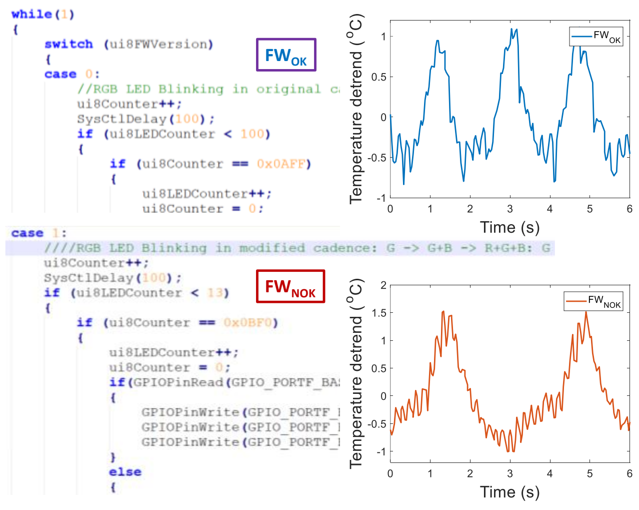

```
while(1)
{
    switch (ui8FWVersion)
    {
    case 0:
        //RGB LED Blinking in original c
        ui8Counter++;
        SysCtlDelay(100);
        if (ui8LEDCounter < 100)
        {
            if (ui8Counter == 0x0AFF)
            {
                ui8LEDCounter++;
                ui8Counter = 0;
case 1:
    ////RGB LED Blinking in modified cadence: G -> G+B -> R+G+B: G
    ui8Counter++;
    SysCtlDelay(100);
    if (ui8LEDCounter < 13)
    {
        if (ui8Counter == 0x0BF0)
        {
            ui8LEDCounter++;
            ui8Counter = 0;
            if(GPIOPinRead(GPIO_PORTF_BA
            {
                GPIOPinWrite(GPIO_PORTF_
                GPIOPinWrite(GPIO_PORTF_
                GPIOPinWrite(GPIO_PORTF_
            }
            else
            {
```

Fig. 6. Sections of the embedded test firmware and the respective thermographic signature.

B. Data Acquisition

i) Infrared images acquisition

The data used to train and validate the autoencoder are obtained from the DUT's thermographic behavior. Thus, the infrared image acquisition is the first stage of data acquisition. A picture of this setup is shown in Fig. 7. In our experiment, the total acquisition time was approximately two hours for both firmware versions FW_{OK} and FW_{NOK}.

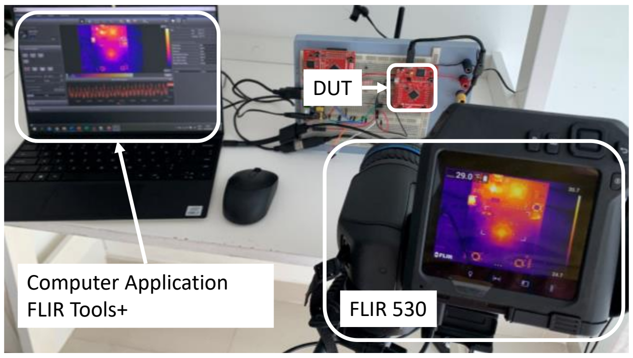


Fig. 7. Infrared data acquisition system with a FLIR T530 camera.

ii) Thermographic samples generation

After the acquisition, the two hours long signal is split into multiple ΔT long sections, called thermographic samples. The total number of samples generated from the acquired signal depends on ΔT, i. e., a higher number of samples requires a lower ΔT. However, there is a tradeoff between the number of generated samples and the anomaly detection accuracy: the longer the acquisition length, the more detail it contains; hence, the higher the accuracy. In our experiments, ΔT = 4, 5 and 6 s were evaluated. In all cases, we used the same the number of samples, to ensure a fair comparison. We obtained the best result with ΔT = 6 s. Then, from the two hours long collected data, a series of 6 seconds long samples were obtained (see details in TABLE II). From the operational perspective, ΔT is a parameter of the anomaly detection system that can be adjusted.

TABLE II. TOTAL AMOUNT OF GENERATED SAMPLES AND IMAGES FOR TRAINING, VALIDATION AND TEST PURPOSES.

Thermographic Samples and Spectrogram Images	**FW_{OK}**	**FW_{NOK}**
Total	1204	1204
Autoencoder training	643	*Not Applicable*
Autoencoder validation	160	*Not Applicable*
Autoencoder testing	401	401
Random forest training	903	903
Random forest testing	301	301

iii) Spectrograms generation

We used the Matlab function "*spectrogram (X, S_len, N_overlap, Nf, Fs)*" to generate the spectrogram. This function generates the spectrogram of *X* (discrete signal acquired with sampling rate Fs) as follows: The signal is divided into sections of length "*S_len*"; A Hamming window is applied to the section; An overlapping "*N_overlap*" points is applied between adjunct sections; The spectrum (Fourier) is calculated at the frequencies *⌊Nf / 2 + 1⌋*, where "*⌊ ⌋*" is the floor operator. The following values were used in our experiment: *S_len = 64*; *N_overlap = 50*; *Nf = 60*; *Fs = 30 Hz*. Then, the generated spectrogram is converted into color images of 128 x 96 pixels. This process of converting the spectrogram to an image makes it possible to filter the frequencies of the samples by adjusting the limits of the y-axis. Thus, it is possible to filter out unwanted sample noise and acquisition artifacts. In our examples, the spectral range of interest of the measured temperature signal is restricted to low-frequency values. Therefore, frequency values above 5 Hz were filtered out by the image generation process. This filtering is adjustable and can be configured for a new DUT. Fig. 8 shows examples of two samples and their spectrograms. The final image generated from the spectrogram does not have the axes, only the spectral information. From Fig. 8, it is possible to notice that the differences between the samples cause minimal differences in the spectrograms. Nevertheless, this minimal difference can be detected by the autoencoder quite effectively. This is the fundamental aspect of the method proposed in this work.

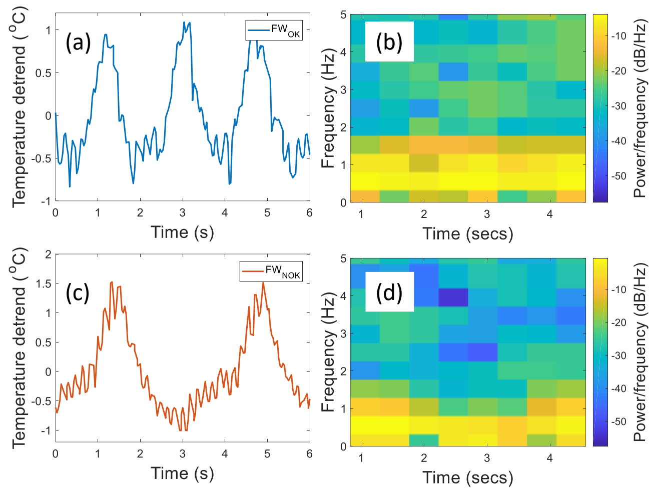


Fig. 8. Samples with width (ΔT = 6 s). (a) - FW_{OK}; (b) - FW_{NOK}. (c) and (d) - Spectrograms of the samples displayed in (a) and (b), respectively.

Finally, for each thermographic sample, one spectrogram image is generated. TABLE II summarizes sample and image generation and application within the anomaly detection process. Random Forest uses thermographic samples, while Autoencoder uses spectrogram images.

C. Data Preprocessing

The temperature signal as a function of time varies very slowly – observed frequencies are of the order of 5 to 10 Hz. Therefore, the spectrograms generated directly from the obtained signal show a slight variation. Thus, the autoencoder cannot correctly distinguish images that belong to the OK class from those that belong to the NOK class. To get around this restriction, we change the temperature signal utilizing a simple non-linear transformation: the DC level of the signal is removed before obtaining the spectrogram. This transformation represents a form of preprocessing applied to the data. In Matlab, this is simply accomplished through the *detrend()* function. It is important to note that this approach is only necessary in the case where images are used, e. g., the autoencoder of our anomaly detection system. The curves depicted in Fig. 9 illustrates this transformation.

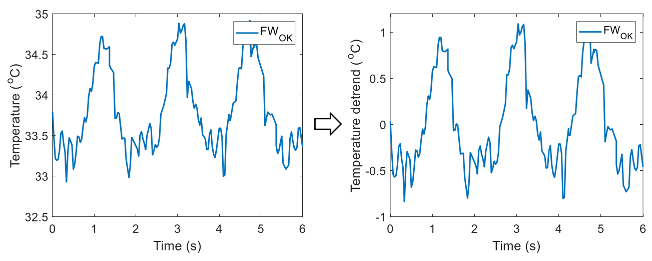


Fig. 9. Preprocessing example using Matlab *detrend()* function.

D. Random Forest Training

Here, we detail the process of building and training the Random Forest used to classify the firmware executed by the DUT into two classes: OK and NOK. The model was built using the R language [35], a well-established library for this purpose. The random forest model's performance can be optimized through parameter adjustment. Thus, we observed the model's performance as a function of parameters *Ntree* and *Mtry*, the number of trees to grow, and the number of variables randomly sampled as candidates at each split. For the experiment, for anomaly detection, we chose the model with *Ntree = 80 and Mtry = 20* for it has been shown the best results, with a test error of approximately 0.16 % (Fig. 10).

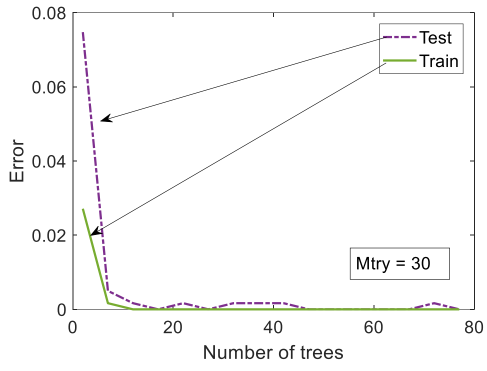


Fig. 10. Train and test errors obtained for a Random Forest model as a function of the number of trees (Ntree) and number of variables randomly sampled as candidates at each split (Mtry).

E. Autoencoder Training

We implemented an autoencoder based on a famous structure [36], where the layers consist of convolutional and filtering networks. We used Python and the open-source artificial neural network library *Keras* to implement the autoencoder [37]. The autoencoder summary is shown in Fig. 11.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                      Output Shape           Param #
=================================================================
input_1 (InputLayer)              (None, 96, 128, 3)     0
_________________________________________________________________
conv2d_1 (Conv2D)                 (None, 94, 126, 32)    896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2      (None, 47, 63, 32)     0
_________________________________________________________________
conv2d_2 (Conv2D)                 (None, 45, 61, 64)     18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2      (None, 22, 30, 64)     0
_________________________________________________________________
conv2d_3 (Conv2D)                 (None, 20, 28, 64)     36928
_________________________________________________________________
flatten_1 (Flatten)               (None, 35840)          0
_________________________________________________________________
dense_1 (Dense)                   (None, 2304)           82577664
_________________________________________________________________
reshape_1 (Reshape)               (None, 24, 32, 3)      0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr      (None, 48, 64, 64)     1792
_________________________________________________________________
batch_normalization_1 (Batch      (None, 48, 64, 64)     256
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr      (None, 96, 128, 64)    36928
_________________________________________________________________
batch_normalization_2 (Batch      (None, 96, 128, 64)    256
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr      (None, 96, 128, 32)    18464
_________________________________________________________________
conv2d_4 (Conv2D)                 (None, 96, 128, 3)     867
```

Fig. 11. Autoencoder summary.

Before using the autoencoder to detect anomalies, it is mandatory to train it. In our case, the training process must be carried out using anomaly free class images solely. The quality of the training process is measured by two parameters: loss and accuracy.

We performed cross-validation during the training process to avoid autoencoder overfitting. The, training is performed on one subset (called the training set), and validation is performed on the other subset (called the validation). We used data subsets to perform training and validation, as listed in TABLE II.

Training and validation outcomes are shown in Fig. 12. An epoch denotes the number of times all the training data have passed through the neural network in the training/validation process. Both *loss* and *accuracy* converge to reasonable values after a few dozen iterations. Hence, the training process is successful.

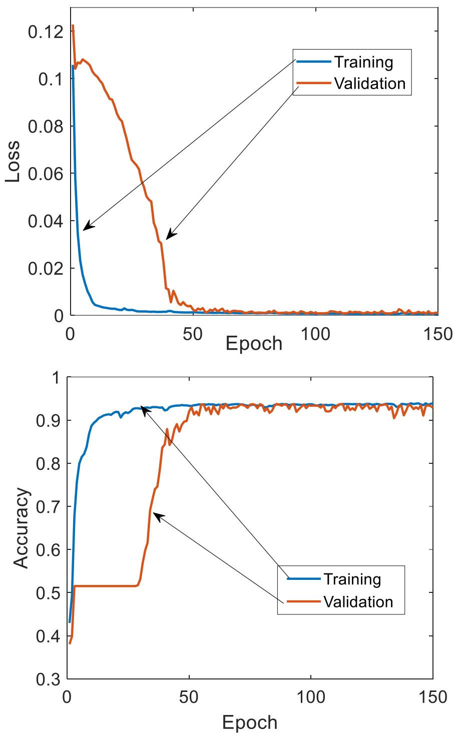


Fig. 12. Loss and Accuracy for Training and Validation processes.

V. Experimental Results

A. Anomaly detection with Random Forest

Using the structure shown in Fig. 4 and the data listed in TABLE II, we performed an anomaly detection test for two configurations, *Mtry = 4* with *Ntree = 20* and *Mtry = 20* with *Ntree = 80*. The model's classification error shows that the model can perform a near-optimal classification for the second situation. This result is confirmed by ROC curves, as shown in Fig. 13.

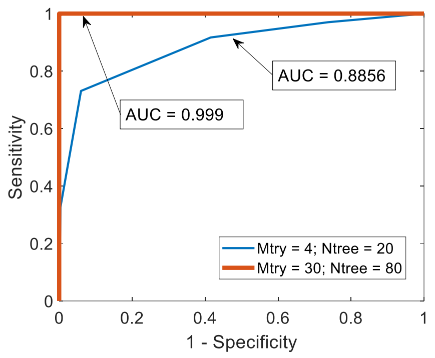


Fig. 13. ROC curves regarding firmware classification for two random forest implementations.

The confusion matrix constructed for this test is depicted in TABLE III, from which an overall anomaly detection accuracy rate of 99.67 % is obtained.

TABLE III. Confusion matrix for the random forest anomaly detection.

		Prediction	
		FW$_{OK}$	**FW$_{NOK}$**
Actual class	**FW$_{OK}$**	**300**	0
	FW$_{NOK}$	1	**301**

B. Anomaly detection with Autoencoder

The detection accuracy strongly depends on the decision threshold. In practical situations, optimal threshold selection is a gradual process. In this case, values are refined with empirical data from the system in operation [22].

For the anomaly detection system to work correctly, the reconstruction error's variance for the anomaly-free cases must be limited. Thus, there must be a gap between the anomaly-free distribution and any other distribution in order to define a threshold that leads to maximum detection accuracy.

The autoencoder reconstruction error distributions obtained in our tests are shown in Fig. 14. There is a gap between FW$_{OK}$ and FW$_{NOK}$ distributions, where the optimal decision threshold $\lambda_o = 5.375 \times 10^{-3}$ can be placed.

Fig. 14 presents curves with an interesting shape: the FW$_{OK}$ curve is narrow and has a low average. The FW$_{NOK}$ curve, on the other hand, is broader and has a high average. Such a trend indicates that, in practice, the reconstruction errors for NOK signals should be higher, which facilitates anomaly detection.

The optimal decision threshold can be found from the reconstruction errors obtained during the tests as described next. First, we perform multiple tests with incremental values for the decision threshold λ.

The error or accuracy of anomaly detection is obtained for each case and plotted as a function of λ, as illustrated in Fig. 15.

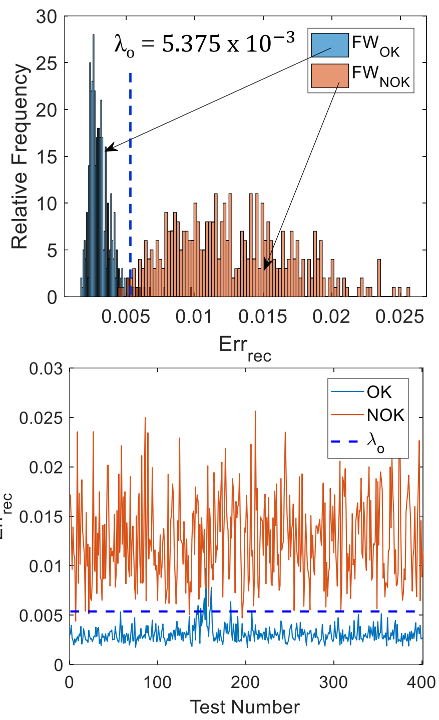


Fig. 14. Autoencoder reconstruction error obtained from 401 tests.

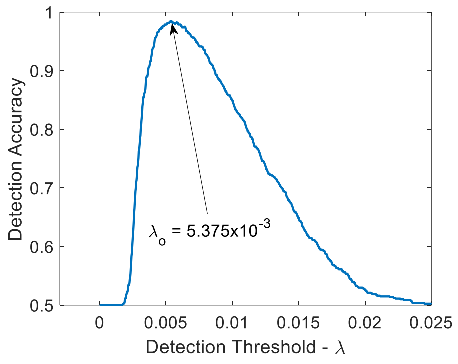


Fig. 15. Anomaly detection accuracy versus decision threshold values. The best case defines the optimum decision threshold.

For the optimal threshold, anomaly detection is accomplished with nearly 100% accuracy. This is confirmed by the ROC curve obtained from this experiment shown in Fig. 16 and by the confusion matrix constructed for the Random Forest test, depicted in TABLE IV. From the confusion matrix in TABLE IV, we find the overall correct anomaly detection rate of 98,50%.

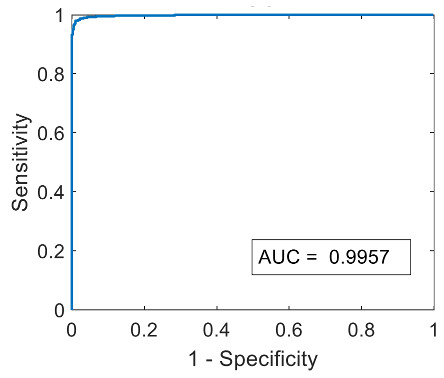


Fig. 16. ROC curves for the Autoencoder anomaly detection using threshold λ_o = 5.375 x 10^{-3}.

TABLE IV. CONFUSION MATRIX FOR THE AUTOENCODER ANOMALY DETECTION.

		Prediction	
		FW$_{OK}$	**FW$_{NOK}$**
Actual class	**FW$_{OK}$**	**393**	8
	FW$_{NOK}$	4	**397**

C. *Analysis*

The ROC curves and confusion matrix demonstrate that both autoencoder and random forest models can distinguish data from different firmware classes. Autoencoders have the advantage that they can be trained with anomaly-free data only.

For the Autoencoder approach, since the reconstruction errors extent varies with the spectrogram class (FW$_{OK}$ or FW$_{NOK}$), the anomaly detection accuracy depends on the decision threshold. After performing multiple detection tests, we can use the curve in Fig. 15 to obtain an informed estimation for the decision threshold. We can confirm this by comparing the obtained ROC, AUC and detection accuracy to similar [38] or superior [22], [39] to that published in previous works, with the advantage of being completely non-intrusive. The mentioned works are also limited regarding the class of anomaly they can detect. The work presented in [20] used thermography to detect hardware anomalies. However, in that case, only the soldered joint quality of PCB mounted LEDs was analyzed. Similarly, the approach described in [28] is focused on electrical component's defects, instead of system anomalies.

To summarize, the results demonstrate the effectiveness of the proposed embedded system anomaly detection method; for two test firmware versions, the detection accuracy is significantly high; and pre-processing applied on the data was essential to anomaly detection accuracy. The proposed method can be easily adapted to other embedded systems by retraining the autoencoder with appropriate anomaly-free thermographic sample and spectrograms.

VI. CONCLUSION AND FUTURE WORKS

In this work, we investigated the application of machine learning and thermography to detect anomalies in embedded systems. We designed and built a validation DUT, which consists of two parts: hardware and firmware. For comparison purposes, a machine learning model based on Random Forest was used to detect anomalies on the same board we used to validate the Autoencoder approach. The test results showed that both approaches are effective. However, Random Forest training requires data from all classes. Autoencoders, on the other hand, only require the reference class (anomaly free). This is fundamental because, in a practical scenario, data from anomalous classes are challenging to obtain.

Regarding this research work, our next steps are:

- Investigate autoencoder's optimization;
- Test the system with more samples (i. e., increase the data acquisition time);
- Investigate the effects of other non-linear transformation on the data before training the Autoencoder;

- Design a machine learning model based on a Convolutional Neural Network with multiple inputs, each one receiving data from a different ROI.

References

[1] Khandpur, Raghbir Singh. “Printed circuit boards: design, fabrication, assembly and testing,” Tata McGraw-Hill Education, 2006.

[2] Justyna Zander, Ina Schieferdecker, Pieter J. Mosterman. „Model-Based Testing for Embedded Systems,” CRC Press, 2017.

[3] Gilchrist, Alasdair. “Industry 4.0: the industrial internet of things,” Apress, 2016.

[4] Jiage Huo, Felix T. S. Chan, Carman K. M. Lee, Jan Ola Strandhagen, Ben Niu. “Smart control of the assembly process with a fuzzy control system in the context of Industry 4.0,” Advanced Engineering Informatics. Volume 43, January 2020, 101031.

[5] Bartodziej, Christoph Jan. "The Concept Industry 4.0," The Concept Industry 4.0. Springer Fachmedien Wiesbaden, 2017. 27-50.

[6] M. R. Johnson, "The Increasing Importance of Utilizing Non-intrusive Board Test Technologies for Printed Circuit Board Defect Coverage," 2018 IEEE AUTOTESTCON, National Harbor, MD, 2018, pp. 1-5, doi: 10.1109/AUTEST.2018.8532499.

[7] Ming-Chuan Chiu, Chi-Hsuan Tsai. “Design a personalized product service system utilizing a multi-agent system.,”Advanced Engineering Informatics. Volume 43, January 2020, 101036. https://doi.org/10.1016/j.aei.2020.101036.

[8] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. “Deep learning,” MIT press, 2016.

[9] Edward A. Lee and Sanjit A. Seshia, “Introduction to Embedded Systems, A Cyber-Physical Systems Approach,” Second Edition, MIT Press, ISBN 978-0-262-53381-2, 2017.

[10] M. Esser, P.Struss. “Fault-model-based Test Generation for Embedded Software,” IJCAI'07: Proceedings of the 20th international joint conference on Artificial intelligence. 2007. Pages 342–347.

[11] “Test Pattern Generation and Fault Simulation. In: Integrated Circuit Test Engineering,”. Springer, London. https://doi.org/10.1007/1-84628-173-3_10.

[12] Peter Marwedel. “Embedded System Design - Embedded Systems Foundations of Cyber-Physical Systems,”. 321- 326. DOI 10.1007/978-94-007-0257-8.

[13] Lu, S., Chu, J. & Jang, H. “Development of a novel coordinate transposing fixture system,” Int J Adv Manuf Technol 13, 350–358 (1997). https://doi.org/10.1007/BF01178255.

[14] Gómez, J., et al. "A robotic system for PCBS inspection based on computer vision and mobile probes," IFAC Proceedings Volumes 40.3 (2007): 171-176.

[15] M. Baygin, M. Karakose, A. Sarimaden And E. Akin, "Machine vision based defect detection approach using image processing," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, 2017, pp. 1-5, doi: 10.1109/IDAP.2017.8090292.

[16] F. Guo and S. Guan, "Research of the Machine Vision Based PCB Defect Inspection System," 2011 International Conference on Intelligence Science and Information Engineering, Wuhan, 2011, pp. 472-475, doi: 10.1109/ISIE.2011.47.

[17] Chuang, S., Chang, W., Lin, C. et al. “Misalignment inspection of multilayer PCBs with an automated X-ray machine vision system,” Int J Adv Manuf Technol 51, 995–1008 (2010). https://doi.org/10.1007/s00170-010-2664-9.

[18] Richter, Johannes, Detlef Streitferdt, and Elena Rozova. "On the development of intelligent optical inspections," Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual. IEEE, 2017.

[19] Wenting Dai, Abdul Mujeeb, Marius Erdt, Alexei Sourin. “Soldering defect detection in automatic optical inspection,” Advanced Engineering Informatics. Volume 43, January 2020, 101004. https://doi.org/10.1016/j.aei.2019.101004.

[20] Y. Mamchur, V. Ivanova, G. Monastyrsky, T. Melnychenko, G. Zheng and S. Voronov, "Thermography investigation of soldered joints for LED mounting," 2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO), Kyiv, Ukraine, 2020, pp. 143-147, doi: 10.1109/ELNANO50318.2020.9088886.

[21] Nabil El-Belghiti, Patrick Tounsi, Alexandre Boyer, Arnaud Viard . “Upgrading In-Circuit Test of High Density PCBAs Using Electromagnetic Measurement and Principal Component Analysis,” Journal of Electronic Testing (2018) 34:749–762. https://doi.org/10.1007/s10836-018-5763-4.

[22] Abdul Mujeeb, Wenting Dai, Marius Erdt, Alexei Sourin. “One class based feature learning approach for defect detection using deep autoencoders,”. Advanced Engineering Informatics. Volume 42, October 2019, 100933. https://doi.org/10.1016/j.aei.2019.100933.

[23] Pengyang Zong ; Yichen Wang ; Feng Xie. “Embedded Software Fault Prediction Based on Back Propagation Neural Network,”. 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). DOI: 10.1109/QRS-C.2018.00098.

[24] P. Deep Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, Noida, 2017, pp. 775-781, doi: 10.1109/CONFLUENCE.2017.7943255.

[25] Manjula, C., Florence, L. “Deep neural network based hybrid approach for software defect prediction using software metrics,” Cluster Comput 22, 9847–9863 (2019). https://doi.org/10.1007/s10586-018-1696-z.

[26] R. Moser, W. Pedrycz, and G. Succi. “A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction,” In Proc. Int’l Conf. on Softw. Eng. (ICSE’08), pages 181–190, 2008.

[27] José Paulo G. de Oliveira, Carmelo Bastos Filho, Sérgio Campello Oliveira, “Non-invasive embedded system hardware/firmware anomaly detection based on the electric current signature”. Advanced Engineering Informatics, unpublished.

[28] N. El Belghiti Alaoui, P. Tounsi, A. Boyer and A. Viard, "Detecting PCB Assembly Defects Using Infrared Thermal Signatures," 2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems", 2019, pp. 345-349, doi: 10.23919/MIXDES.2019.8787089.

[29] J. Liang, S. K. K. Ng, G. Kendall and J. W. M. Cheng, "Load Signature Study—Part I: Basic Concept, Structure, and Methodology," in IEEE Transactions on Power Delivery, vol. 25, no. 2, pp. 551-560, April 2010, doi: 10.1109/TPWRD.2009.2033799.

[30] FLIR Tools User Guide. http://support.flir.com/answers/A1568/FLIR%20Tools%20User%20 Guide%20v2.1.1.pdf (last access 05 August 2021).

[31] Mayu Sakurada and Takehisa Yairi. “Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction,” In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis. Association for Computing Machinery, New York, NY, USA, 4–11. DOI: https://doi.org/10.1145/2689746.2689747.

[32] Robin Genuer, Jean-Michel Poggi, Christine Tuleau-Malot. “Variable selection using random forests,” Pattern Recognition Letters. Volume 31, Issue 14, 15 October 2010, Pages 2225-2236.

[33] N. Saravanan, K. I. Ramachandran. “Incipient gear box fault diagnosis using discrete wavelet transform (DWT) for feature extraction and classification using artificial neural network (ANN),” Expert Systems with Applications. Volume 37, Issue 6, June 2010, Pages 4168-4181.

[34] MAZIDI, Muhammad Ali et al. TI Tiva ARM Programming For Embedded Systems: Programming ARM Cortex-M4 TM4C123G with C (Volume 2). 2017.

[35] Y. Rimal, "Machine Learning Random Forest Cluster Analysis for Large Overfitting Data: using R Programming," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019, pp. 1265-1271.

[36] Chollet, Francois. “Building Autoencoders in Keras,” Online 2016. https://blog.keras.io/building-autoencoders-in-keras.html. Last access: 05-28-2020.

[37] Chollet, Francois. “Deep Learning with Python,” Manning Publications, (2017).

[38] G. Acciani, G. Brunetti and G. Fornarelli. “A Multiple Neural Network System to Classify Solder Joints on Integrated Circuits,” International Journal of Computational Intelligence Research. ISSN 0973-1873 Vol.2, No.4 (2006), pp. 337-348.

[39] L. H. d. S. Silva, G. O. d. A. Azevedo, B. J. T. Fernandes, B. L. D. Bezerra, E. B. Lima and S. C. Oliveira, "Automatic Optical Inspection for Defective PCB Detection Using Transfer Learning," 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guayaquil, Ecuador, 2019, pp. 1-6, doi: 10.1109/LA-CCI47412.2019.9037036.