Image-Based Detection of Modifications in Gas Pump PCBs with Deep Convolutional Autoencoders

Diulhio Candido de Oliveira*, Bogdan Tomoyuki Nassu, Marco Aurelio Wehrmeister

*Department of Informatics, Federal University of Technology - Parana (UTFPR), 80230-901 Curitiba, Brazil
bComputer Science Department, University of Münster, 48149 Münster, Germany

Abstract

In this paper, we introduce an approach for detecting modifications in assembled printed circuit boards based on photographs taken without tight control over perspective and illumination conditions. One instance of this problem is the visual inspection of gas pumps PCBs, which can be modified by fraudsters wishing to deceive customers or evade taxes. Given the uncontrolled environment and the huge number of possible modifications, we address the problem as a case of anomaly detection, proposing an approach that is directed towards the characteristics of that scenario, while being well-suited for other similar applications. The proposed approach employs a deep convolutional autoencoder trained to reconstruct images of an unmodified board, but which remains unable to do the same for images showing modifications. By comparing the input image with its reconstruction, it is possible to segment anomalies and modifications in a pixel-wise manner. Experiments performed on a dataset built to represent real-world situations (and which we will make publicly available) show that our approach outperforms other state-of-the-art approaches for anomaly segmentation in the considered scenario, while producing comparable results on the popular MVTec-AD dataset for a more general object anomaly detection task.

Keywords: Deep Learning, Anomaly detection, Autoencoder, Visual inspection, Manufacture

2010 MSC: 00-01, 99-00

1. Introduction

Detecting anomalies in assembled printed circuit boards (PCBs) is an important problem for fields such as quality control in manufacturing [1, 2] and fraud detection [3]. One instance of the latter is the detection of frauds in gas pumps, a common problem in countries such as Brazil and India [4, 5]. For example, modifying the gas pump PCB, by replacing, adding, or removing components, allows offenders to force the pump to display a fuel volume different from the one actually put into the tank. It may be difficult for law enforcers to detect this kind of fraud simply by testing the pump, since the offender can use a remote control to deactivate the fraud during inspections. Thus, inspectors have to remove the PCB from the gas pump and visually compare the suspicious board to a reference design or sample — for example, in Brazil gas pump PCB designs are approved and controlled by a regulatory body, and cannot be changed without authorization. To avoid worries such as legal action from gas station owners who lose profits while the pump cannot be operated, inspections should be quick, but this is frequently not possible, given the complexity of these PCBs. Figure 1 shows an example of a PCB containing modifications — the amount of small components makes it hard even for a specialist to notice these modifications. The task is further complicated if inspectors are not specialists, which leads them to rely solely on visual comparisons. For these reasons, a system that assists inspectors by automatically detecting modifications or suspicious regions can be an interesting proposition.

While we have the fraud detection scenario as our main motivation, this problem shares most charac-
teristics with image-based inspection of PCBs in general — a task for which several methods have been proposed in recent years. Some methods are used to detect defects in unassembled PCBs [6, 7], where common anomalies are missing holes and open circuits, while other methods deal with assembled PCBs [3, 8, 9]. These methods are usually based on supervised machine learning, where a decision model is trained by observing samples both with and without defects or anomalies. One of the foremost challenges when working with this kind of data-driven technique is providing a representative dataset, containing a wide range of situations that reflect the actual variety of possibilities faced in practice well enough to allow generalization. For an unmodified board, that means having samples with varied lighting conditions and camera angles, but anomaly samples are harder to obtain, because they are rare, expensive to reproduce, or may manifest in unpredictable ways.

We adopt a different approach, and address the task as an anomaly detection problem. In this formulation, models are trained only on normal samples, learning to describe their distribution, based on the premise that it is possible to detect anomalies based on how well the learned model is able to describe a given sample — i.e. samples containing anomalies are not well described by the model, and will appear as outliers. Many recent studies aimed at industrial inspection on various settings explore this idea [1, 2, 10, 11, 12, 13].

In this paper, we address the problem of detecting modifications in a PCB using a deep neural network. More specifically, we propose using a convolutional autoencoder architecture for reconstruction-based anomaly detection. This kind of architecture compresses the input image to a feature vector, called “latent space”, and then reconstructs the same image based only on these features. The rationale behind the proposed method is that, if the model is trained only with anomaly-free samples, it will be able to reconstruct only this kind of sample. Thus, when it receives as input an image containing anomalies, it will be unable to properly reconstruct the output, or even reconstruct the image without its anomalies. This idea is illustrated in Figure 2.

We performed experiments comparing our proposed method to other state-of-the-art reconstruction-based anomaly detection methods [10, 12, 2, 13], which achieved good performance on the MVTec-AD dataset [1], a general anomaly detection image dataset. In experiments performed on a dataset containing PCB images under varied illumination conditions and camera angles, our method outperformed these state-of-the-art techniques, producing a more precise segmentation of the modifications, and obtaining better scores on the measured metrics — pixel-wise intersection over union (IoU) and detection and segmentation area under the receiver operating characteristic curve (AUROC). Additionally, on the more general MVTec-AD dataset, our method performed similarly to the other methods, achieving better results for addition or removal of objects.

The main contributions of this work are:

- We introduce a robust method for modification detection on PCBs that can be applied to images containing perspective distortion, noise and lighting variations. The method was designed to work on the circumstances commonly found in practice during the inspection of gas pump PCBs in gas stations, where images must be captured by mobile devices, without relying on controlled lighting or positioning.

- We propose a convolutional autoencoder architecture for reconstruction-based anomaly detection, which is trained combining the content loss and mean squared error functions. The proposed model can be trained only with anomaly-free images, making it more suitable for real-world applications where this kind of
Figure 2: The reconstruction-based inference process using a convolutional autoencoder. The autoencoder is trained to recon-
struct only anomaly-free samples, so when it receives as input an image containing anomalies, the reconstructed output does
not show the modification. Thus, it is possible to segment the anomaly by comparing the input and the output.

sample is much more common or easier to ob-
tain than a representative set of samples con-
taining anomalies.

• We provide a labeled PCB image dataset for
training and evaluating anomaly detection and
segmentation methods. The dataset is publicly
available and contains 1742 4096×2816-pixel
images from an unmodified gas pump PCB, as
well as 55 images containing modifications,
along with corresponding segmentation masks.

The remainder of this paper is organized as fol-
lows. Section 2 discusses related work on defect and
anomaly detection on PCBs, as well as anomaly de-
tection for industrial inspection in general. Section
3 details the proposed approach. Section 4 presents
the experimental setup and the obtained results.
Finally, Section 5 draws some conclusions and indi-
cates directions for future work.

2. Related work

Several algorithms have been proposed for image-
based anomaly detection in PCBs. For instance,
deep learning techniques have been used to detect
anomalies such as missing holes and defective cir-
cuits in unassembled boards [6, 7]. Although these
approaches were successful, they rely on controlled
capture conditions, and only work for limited types
of anomaly, found in unassembled boards. For
assembled boards, a common strategy is using su-
pervised training to produce a component detector
[8, 9]. The layout of the detected components
can be compared to a reference, providing a way
of detecting anomalies, but this strategy demands
considerable effort to obtain labeled training data
(for example, [8] generates artificial samples from
3D models). Moreover, this strategy is limited to
detecting known components, possibly failing when
the modification involves adding some unknown
component.

Of special relevance is the system proposed in [3],
which addresses the same problem as we do. We
employed the same method used by that work to
deal with variations in camera angle, and the same
idea of partitioning the board to analyze each re-
gion independently. Our main test dataset includes
some of the images used by that work. However,
our anomaly detection strategy differs significantly
— they employ SIFT features and Support Vec-
tor Machines to classify each region as normal or
anomalous, while we segment anomalies using a re-
construction deep network. Moreover, that work
uses supervised learning, with anomalies being ar-
tificially created by placing small patches extracted
from other samples, while our model is trained only
on normal samples.

Several methods that rely only on normal sam-
ples for anomaly detection for industrial visual
inspection (not limited to PCBs) have been re-
cently proposed. The most successful methods
are based on reconstructions or embedding simi-
larity. Reconstruction-based methods compute a
compressed representation of the input image, and
attempt to reconstruct the original image based on
it. Our method falls into this category. Models
that can be employed for the reconstruction in-
clude autoencoders (AE) [1, 12], variational autoen-
coders (VAE) [14, 15] and generative adversarial
networks (GAN) [16]. The main advantage of these
approaches is that it is easy for a human to un-
derstand and interpret their results. However, if a
method still reconstructs an anomaly [17], it may
remain undetected, as there is no noticeable differ-

https://github.com/Diulhio/pcb_anomaly/tree/main/dataset
ence between the input and the reconstruction.

Embedding similarity methods \cite{2, 10, 13} use deep convolutional networks pretrained on large generic datasets (e.g. ImageNet) as feature extractors. The distribution of the features extracted from anomaly-free samples is then modeled as a probability density function \cite{2}. Given a distance metric, the feature vectors from images with anomalies tend to be more distant to the center of the distribution (e.g. the mean vector), compared to normal samples. These methods are applicable to new problem domains without requiring additional training of the basic feature extractor, but their results are hard to interpret. Moreover, computation of the density function can have high memory requirements, and be complicated when the dataset has high variability.

A popular benchmark for anomaly visual inspection is the MVTec-AD \cite{1, 11} dataset, which contains 5354 images, with 70 types of anomalies for 15 kinds of objects. Most of the anomaly detection methods cited above were evaluated on this dataset, so our method will also be tested on it.

3. Proposed method

Many existing approaches for anomaly detection produce a binary classification that refers to the entire sample, telling whether it contains a modification or not. However, this may be insufficient in a real world scenario, since the specific structures or components which characterize the modification are not identified. Methods that produce bounding boxes or that segment anomalies may be more suitable for PCB inspection. Thus, the approach we propose in this work performs anomaly segmentation. It employs a deep convolutional neural network for image reconstruction, trained only on normal samples, i.e. images without any modifications/defects/anomalies.

3.1. Image registration and partitioning

Similarly to the work from \cite{3}, our approach assumes the PCB is shown from an overhead. However, different from several other studies on visual inspection \cite{11, 13, 6, 8}, where positioning is strict to avoid variations, we suppose the input image may be the product of an image registration step. In other words, the PCB may be photographed from an angled view, being aligned to a reference image after capture (see Figure 3), using an algorithm for image registration based on SIFT features and the RANSAC algorithm \cite{18}. In the resulting image, the components on the PCB may have some degree of perspective distortion and variations in position, since image registration can be slightly imprecise, and the algorithm treats only planar distortions, without taking into account the 3D aspect of the components, as shown in Figure 4. Moreover, our approach does not demand controlled lighting, so there can be reflections, shadows and other variations, which can be hard to distinguish from actual modifications or anomalies. These assumptions make our approach suitable for real-world applications where the inspection may occur in an open and uncontrolled environment.

Anomaly detection methods frequently work on fixed-size inputs, reducing the captured image to a smaller size, which also reduces computation and memory requirements. However, for PCB inspection, doing this for the entire board can result in certain components and modifications becoming too small. To avoid this, we partition the input image into 1024×1024-pixel patches, which are then resized to 256×256 pixels and processed independently. Figure 5 illustrates this procedure.
3.2. Convolutional autoencoder architecture

After the original image is partitioned, each 256×256 patch is given as an input to a convolutional autoencoder (CAE) [19]. Using a series of convolutional layers, CAEs encode the high-dimensional input image to a compressed low-dimensional vector called “latent space”, and expands (decodes) this vector to the original dimensionality. The encoder function \( z = g_\phi(x) \) receives the input an maps it to the latent space \( z \). The decoder function \( x' = f_\theta(z) \) computes the reconstruction from the latent space. Thus, the entire network is expressed as \( f_\theta(g_\phi(x)) = x' \), and in a perfect CAE \( x = x' \).

In our approach, one CAE is trained for each patch region (i.e. for the board shown in Figure 5, we have 12 CAEs). These networks are trained using only anomaly-free samples, ideally becoming able to reconstruct only this type of image — when receiving images showing anomalies, the CAE will produce visible artifacts or reconstruct them without the anomalies, as illustrated in Fig. 2.

The CAE architecture we use in our approach is shown in Table 1. The network was built using convolutional layers in the encoder, and transposed convolutional layers in the decoder, with 5×5 kernels in both cases. Each convolutional layer is followed by Batch Normalization (BN) and a Leaky ReLU activation, with slope of 0.2. The last layer of the encoder and the first layer of the decoder are fully connected layers of 1024 nodes, followed by BN and Leaky ReLU. The latent space is the output of a fully connected layer with 500 values.

Table 1: The architecture of our convolutional autoencoder. All convolutional and transposed convolutional layers use 5×5 kernels and stride 2.

| Layer Type | Filters | BN | LeakyReLU |
|------------|---------|----|-----------|
| Conv       | 32      |    |           |
| Conv       | 64      |    |           |
| Conv       | 128     |    |           |
| Conv       | 128     |    |           |
| Conv       | 256     |    |           |
| Conv       | 256     |    |           |
| Conv       | 256     |    |           |
| Fully conn. | 1024    |    |           |
| Fully conn. | 500     |    |           |
| Fully conn. | 1024    |    |           |
| TranspConv | 256     |    |           |
| TranspConv | 256     |    |           |
| TranspConv | 256     |    |           |
| TranspConv | 128     |    |           |
| TranspConv | 128     |    |           |
| TranspConv | 64      |    |           |
| TranspConv | 32      |    |           |
| TranspConv | 3      |    | Sigmoid   |

3.3. Content loss function for training

The loss functions most commonly used for training autoencoders are pixel-wise functions, such as the Mean Square Error (MSE). However, these functions assume the pixels are not correlated, which is often not true — in general, images have structures formed by the relations between pixel neighborhoods. Pixel-wise functions also frequently result in blurred outputs when used for reconstruction. For these reasons, we used the content loss function when training the autoencoder.

Content loss, introduced by [20], identifies differences between two images (in our case, the input and the reconstruction) based on high-level features. It was used for applications such as style transfer [20][21], super resolution [21][22] and image restoration [23]. Features are extracted from an image classification network (VGG19 [24], in our work) pretrained on general-purpose datasets (ImageNet [25], in our work). This function encourages the network to reconstruct images with feature representations similar to those of input, rather than considering just differences between pixels.

Let \( \phi_j(x) \) be the activation of the \( j \)th layer of a pretrained network \( \phi \) when image \( x \) is processed. Since \( j \) is a convolutional layer then \( \phi_j(x) \) will present an output of shape \( C_j \times H_j \times W_j \), where \( C_j \) is the number of filter outputs, and \( H_j \times W_j \) is
the size of each filter output at layer \( j \). The content loss is the squared and normalized distance of the feature representations of the reconstruction \( \hat{y} \) and the reference \( y \), as expressed in Eq. (1).

\[
L^{\phi,j}_{feat}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \| \phi_j(\hat{y}) - \phi_j(y) \|_2^2 \tag{1}
\]

This function tries to find an image \( \hat{y} \) that minimizes the reconstruction loss using the initial layers of the pretrained network \( \phi \). A CAE trained with this function tends to produce images similar to target \( y \) in image content and overall spatial structure [21]. In this work, we sum the differences in the 5th, 8th, 13th and 15th layers from VGG19, based on empirical experiments.

The content loss function controls the reconstruction of larger structures in the image but fails to reconstruct details and textures. For this reason, we combine content loss with the MSE, as expressed by Eq. (2), where \( \lambda_1 \) and \( \lambda_2 \) are the weights of each loss function. We empirically defined the parameters \( \lambda_1 = 0.01 \) and \( \lambda_2 = 1 \). Figure 6 illustrates the entire loss calculation.

\[
L_{rec} = \lambda_1 L_{MSE} + \lambda_2 L_{feat} \tag{2}
\]

Figure 6: The loss calculation flow during training.

3.4. Anomaly segmentation

After training, the network can be used to segment anomalies by comparing the reconstructed image to the input. If the CAE was “perfect”, a simple pixel-wise absolute difference would be enough to segment the anomalies. However, images in a real situation have perspective distortion, noise, and lighting variations that may make the reconstruction hard. These variations may cause small differences along edges, or in regions containing shadows or reflections. In these cases, pixel-wise metrics may result in many false positives. Figure 7 shows an image of a PCB without modifications, its reconstruction, and the absolute difference between them. Even though the differences are hard to notice, the pixel-wise absolute difference has high values at some positions.

To address these challenges, we propose a comparison function based on the content loss concept, i.e. instead of isolated pixels, we focus on structures and higher level features. Once again, we used the VGG19 network trained on the ImageNet dataset to extract high-level features, from the input \( y \) and the reconstruction \( \hat{y} \). The features are compared by summing the absolute differences between the activations of layer \( \phi_j \), as expressed in Eq. (3).

\[
A(\hat{y}, y) = \sum_i |\phi_j, i(\hat{y}) - \phi_j, i(y)| \tag{3}
\]

where \( C_j \) is the number of filter outputs in layer \( j \). \( A \) is a matrix that represents the anomaly map, and has the same size \((H_j \times W_j)\) as the outputs from layer \( \phi_j \). In initial tests performed on a small dataset, the 12nd layer from VGG19 showed the best results, with 512 output of size 28×28.

To get the final segmentation, the input map is resized using bilinear interpolation to the same size as the input, normalized, and binarized with a threshold \( T \). Normalization is based on the min-max range from the entire test set, which must contain images showing modifications, so that we have a measure of the magnitudes of the values produced by these anomalies. The \( T \) parameter gives a measure of how rigorous the detection is, and will be varied during the experiments.

4. Experiments and results

In this section, we present the experiments performed to test the proposed approach for anomaly...
detection and compare it with other state-of-art methods, on our MPI-PCB dataset and the MVTec-AD dataset. Everything was implemented in the Python language, using the TensorFlow\(^2\) and OpenCV\(^3\) libraries. Experiments were performed on the Google Colab\(^4\) platform. The source code is publicly available at [https://github.com/Diulhio/pcb_anomaly/](https://github.com/Diulhio/pcb_anomaly/)

### 4.1. MPI-PCB Dataset

The main dataset used in this work is the Multi Perspective and Illumination PCB (MPI-PCB) dataset, which we built based on many of the same images originally collected for the work in [3]. The dataset contains 1742 images showing an unmodified PCB from a gas pump, with 4096×2816 pixels. The images were captured using a Canon EOS 1100D camera, with 18-55mm lenses. The set also contains 55 images showing the board with modifications, which were manually added by the authors, and are meant to be representative of situations encountered in actual frauds. These samples must not be used in the training step, only for testing. One of the contributions of our paper is making this dataset available, including labeled semantic segmentation masks.

Images were captured from a generally overhead view, but without strict demands on position or illumination, as expected in a real-world situation. To reduce variations that may occur in the image registration step and focus on the anomaly detection problem, the dataset contains the images after the registration procedure described in Section 3.1.

### 4.2. Baseline Methods

To the best of the authors’ knowledge, there are no previous work addressing specifically image-based anomaly segmentation in assembled PCBs — as previously discussed in Section 2, existing approaches focus on unassembled PCBs, or in determining if anomalies are present in a given region, without per-pixel segmentation. For that reason, we compared our approach to other general anomaly segmentation methods, which achieved promising results on the popular MVTec-AD dataset. Our work can be compared to these methods more directly, since they have similar training procedures, and produce segmentation masks as outputs. We chose baseline methods that provide the source code and can run in the infrastructure used for our work. We also selected at least one reconstruction-based method and one embedding similarity method.

Up to the time our experiments were performed, the PaDiM approach [10] had the state-of-art results for anomaly segmentation on the MVTec-AD dataset. It is an embedding similarity method which obtained the best results when using the Wide ResNet-50-2 network to extract features, but due to the very high memory requirements, we used the smaller ResNet18 as a feature extractor in our comparison. Other embedding similarity methods we used as baseline were SPTM [13] and SPADE [2]. For the latter, we reduced the input resolution from the default 224 × 224 to 192 × 192, also due to the high memory requirements. As a reconstruction-based baseline, we took the DFR method [12], which uses regional features extracted from a pretrained VGG19 as inputs for CAEs.

### 4.3. Evaluation metrics

We considered two per-pixel metrics to evaluate the segmentation performance of the techniques: the intersection over the union (IoU) and the area under the receiver operating characteristic curve (ROC-AUC). We also evaluated the ROC-AUC for anomaly detection: while segmentation considers the per-pixel classification, detection expresses if an anomaly exists or not in the image. To avoid detecting noise, we consider an anomaly exists in an image if it contains at least 10 anomalous pixels. The metrics are computed over the (per pixel or per image) count of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) classifications.

ROC-AUC is a widely used metric for evaluating anomaly segmentation methods, and has been usually reported for approaches tested on the MVTec-AD dataset [1, 2, 11, 13] [12]. It expresses the normalized area under the ROC curve, which is the curve obtained by plotting the true versus the false positive rates (TPR and FPR, respectively) at different classification thresholds. TPR and FPR are computed by Equations 4 and 5.

\[
TPR = \frac{TP}{TP + FN} \tag{4}
\]
\[ FPR = \frac{FP}{FP + TN} \] (5)

IoU, also referred to as the Jaccard index, is also reported for several semantic segmentation tasks and challenges such as COCO\(^5\). For anomaly segmentation, the IoU quantifies the overlapping between the ground truth mask and the binarized anomaly map, as given by Equation 6.

\[ IoU = \frac{TP}{TP + FN + FP} \] (6)

In our experiments, we report the best IoU score obtained by each method when varying the classification threshold. Compared to the ROC-AUC, the IoU is more sensitive to variations in the shape of the segmented regions.

### 4.4. Training details

We took the 1742 images from the MPI-PCB dataset showing the unmodified board, and randomly split them as follows: 1518 images for training, 169 for validation, and 55 for testing (the same amount we have with the modified board, to a total of 110 test images). As for the MVTec-AD dataset, the split is: 3266 for training, 363 for validation, and 1725 for test [1].

Due to the large variety of perspective distortions, as well as the limited number of training samples, we used data augmentation on the training sets from both datasets. For the MPI-PCB dataset, we apply a random position offset between 0 and 80 pixels when extracting patches, simulating variations that may occur in the image registration step. We also apply a random cutout, which randomly masks out rectangular regions of the inputs during training, with an effect similar to dropout, but in input space — generating images with simulated occlusions forces the model to take more of the image context into consideration when extracting features, improving network generalization [25].

As for the MVTec-AD dataset, we apply random variations on rotation, shear, saturation, contrast, brightness and scale.

For training the proposed architecture, we used a batch size of 128 on our MPI-PCB dataset, and 16 on the smaller MVTec-AD dataset. We used the Adam optimizer with cosine learning rate decay with a warm-up phase.

5. Results on the MPI-PCB dataset

We evaluated the performance of our method and of the baseline methods on the test set from the MPI-PCB dataset, considering six board regions. All these regions contain inserted modifications, like integrated circuits and jumper wires. The tested methods depend on at least part of test samples from each region containing modifications, to define the range for normalization. A total of 110 samples were tested, 55 with and 55 without modifications in the observed region.

Table 2 shows the results obtained by the tested techniques for each region. Bold text indicates the best results for each metric. The results show that the proposed method outperforms or has similar results compared to approaches that attain state-of-art results in the MVTec-AD dataset.

For simple anomaly detection (measured by the detection ROC-AUC), our method, PaDiM and SPTM present similar performance. The proposed method shows detection ROC-AUC 1.0 in 5 out of 6 regions, which means it identified all modifications in these regions. SPADE and DFR presented significantly worse results, this is explained by the difficulty of finding a threshold that attains a good trade-off between TPR and FPR.

All the methods achieved segmentation ROC-AUC higher than 0.9 for almost every region. This shows these methods can segment most of the anomalies correctly. The proposed method and PaDiM showed the best average performance. The difference between detection and segmentation ROC-AUC results for SPADE and DFR is explained by the class imbalance in each problem. For detection, the test set is balanced, since it contains the same number of positive and negative samples. However, for per-pixel segmentation, the classes are very imbalanced, with less than 2% being positive pixels. This allows the model to generate small segmentation errors in several images without impacting segmentation ROC-AUC, but with high impact in detection ROC-AUC.

Despite the similar ROC-AUC results obtained by our approach and PaDiM, we observed that the segmentation in several samples was visibly different. We noticed that this happened because of the imbalance between positive and negative pixels, which leads to high ROC-AUC values even when the model produces false positive classifications. The IoU can express the segmentation precision better than the ROC-AUC, being more sensitive to incor-
directly classified pixels and, consequently, to deviations in the shape and size of the segmented objects. This can be seen in Figure 8, which shows some segmentation samples produced by our technique and by the baseline methods. We note that most baseline methods had several false positives, i.e. these methods successfully localize the modifications in a general manner, but several additional pixels are detected, so the segmented shape does not match the anomaly. Generally, the models identify large regions around modifications or smaller shapes which do not cover an entire component. That might be interpreted as a false detection by a human inspector without specialized knowledge, because it does not cover just a component but a region that includes parts of other components.

Regarding the IoU, the proposed method outperformed the baseline methods for all evaluated regions, achieving an IoU higher than 0.5 for all regions — this is a relevant mark, since challenges such as Pascal VOC and COCO use IoU > 0.5 as one possible criterion for a successful detection. Note that the IoU is sensitive to the size of the modification, as the weight of an incorrectly classified pixel is higher for smaller objects. Our method was able to segment small modifications, like the jumper wire in the “grid3_2” region (the first row in Fig. 8). PaDiM presented IoU close to 0.5 for all regions, except for the “grid3_2” region, which contains the smallest modification — there was a high number of false negatives, which led to a partial segmentation. As for SPADE, SPTM, and DFR, performance was lower in several cases. These techniques displayed a higher number of false positives, segmenting large regions around the modifications, and in many cases detecting modifications where none exist. That can be explained by the lighting and perspective variations in this dataset.

In conclusion, the obtained results show that, while all techniques are able to detect and segment modifications (as indicated by the detection and segmentation ROC-AUC metrics), the proposed method can better approximate the shape of objects (as indicated by the IoU). In a practical scenario, this advantage can help a human inspector identify the specific components that characterize a modification.

4.6. Results on the MVTec-AD dataset

To evaluate the performance of our method for other anomaly localization contexts, apart from the PCB modifications it was designed for, we tested it, along with the baseline methods, on the MVTec-AD dataset. Results are shown in Table 3. Following the categorization defined by [1], anomalies are grouped by type: “objects” and “textures”. The
former shows certain types of objects, with most anomalies involving the addition, removal or modification of parts or components; while the latter shows close-ups of surfaces, with anomalies consisting in alterations to a common texture pattern.

Table 3: Results of the proposed method and the baseline methods for the MVTec-AD dataset. We show the results for two main categories defined by: textures and objects.

| Metric          | Method | Tex. | Obj. |
|-----------------|--------|------|------|
| **Detection**   |        |      |      |
| ROC-AUC         | Ours   | 0.870| 0.890|
|                 | PaDiM  | 0.960| 0.880|
|                 | SPADE  | 0.860| 0.850|
|                 | DFR    | 0.930| 0.910|
|                 | SPTM   | 0.980| 0.930|
| **Segmentation**|        |      |      |
| ROC-AUC         | Ours   | 0.880| 0.960|
|                 | PaDiM  | 0.950| 0.970|
|                 | SPADE  | 0.970| 0.960|
|                 | DFR    | 0.910| 0.940|
|                 | SPTM   | 0.960| 0.870|
| **IoU**         | Ours   | 0.290| 0.430|
|                 | PaDiM  | 0.330| 0.410|
|                 | SPADE  | 0.380| 0.420|
|                 | DFR    | 0.310| 0.310|
|                 | SPTM   | 0.320| 0.380|

For the “texture” category, the method did not perform as well as the baseline methods. This behavior can be explained by the way we combined the content loss function with the pixel-wise mean squared error. In other tasks, content loss is usually employed in conjunction with the “style loss” function, which tries to keep feature distributions in each layer the same in both the image and its reconstruction. Content loss only captures the aspect of image structures, while MSE compares individual pixels in the image and its reconstruction. That means our model is less capable of representing general texture patterns, being directed towards representing structures and pixel organizations observed during training (on the other hand, that is what allows our approach to detect even small anomalies). The problem is exacerbated by the small number of training samples in the MVTec-AD dataset, which has only around 50 training images per class.

As for the “objects” category, the proposed method performed similarly to the baseline methods, particularly SPADE and PaDiM. This indicates that our method may present better results for problems where most anomalies or modifications are the addition or removal of objects in the inspected area.

5. Conclusions

In this paper, we addressed the problem of detecting modifications in PCBs based on photographs. For that purpose, we proposed a reconstruction-based anomaly detection method using a CAE architecture, trained using just anomaly-free samples with a combination of the content loss and the mean squared error functions. We also introduced MPI-PCB, a labeled PCB image dataset for training and evaluating anomaly de-
tection and segmentation methods. Experiments on that dataset showed that our method has superior results for modification segmentation when compared to other state-of-art methods. We also performed experiments in the popular MvTec-AD dataset, with our method attaining results close to other methods when detecting anomalies such as adding or removing objects, showing that it can be employed in other problem domains.

In future research, we plan to create a more varied dataset, with a greater number of modifications to evaluate the performance in other situations, such as very small modifications. Other possible improvement is designing a loss function capable of learning better texture information, based on techniques such as adversarial learning.

References

[1] P. Bergmann, M. Fauser, D. Sattlegger, C. Steger, Mvtec ad—a comprehensive real-world dataset for unsupervised anomaly detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9592–9600.

[2] N. Cohen, Y. Hoshen, Sub-image anomaly detection with deep pyramid correspondences, arXiv preprint arXiv:2005.02357.

[3] T. J. M. De Oliveira, M. A. Wehrmeister, B. T. Nassu, Detecting modifications in printed circuit boards from fuel pump controllers, in: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE, 2017, pp. 87–94.

[4] P. Chakraborty, 8 Lucknow pumps caught using ‘cheating’ chip (2017).

[5] G. Slattery, Special Report: In Brazil, organized crime siphons billions from gas stations (2021).

[6] V. A. Adibahla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Ahmed, J.-S. Shieh, Defect detection in printed circuit boards using you-only-look-once convolutional neural networks, Electronics 9 (9) (2020) 1547.

[7] W. Shi, L. Zhang, Y. Li, H. Liu, Adversarial semi-supervised learning method for printed circuit board unknown defect detection, The Journal of Engineering 2020 (13) (2020) 505–510.

[8] D. Li, C. Li, C. Chen, Z. Zhao, Semantic segmentation of a printed circuit board for component recognition based on depth images, Sensors 20 (18) (2020) 5318.

[9] M. A. Malliyan Sathiaseelan, O. P. Paradis, S. Taheri, N. Asadianjani, Why is deep learning challenging for printed circuit board (pcb) component recognition and how can we address it?, Cryptography 5 (1) (2021) 9.

[10] T. Defard, A. Setkov, A. Loesch, R. Audigier, Padim: a patch distribution modeling framework for anomaly detection and localization, in: International Conference on Pattern Recognition, Springer, 2021, pp. 475–489.

[11] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, C. Steger, The mvtec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection, International Journal of Computer Vision 129 (4) (2021) 1038–1059.

[12] Y. Shi, J. Yang, Z. Qi, Unsupervised anomaly segmentation via deep feature reconstruction, Neurocomputing 424 (2021) 9–22.

[13] G. Wang, S. Han, E. Ding, D. Huang, Student-teacher feature pyramid matching for unsupervised anomaly detection, arXiv preprint arXiv:2103.04257.

[14] S. Venkataramanan, K.-C. Peng, R. V. Singh, A. Mahalanobis, Attention guided anomaly localization in images, in: European Conference on Computer Vision, Springer, 2020, pp. 485–503.

[15] K. Sato, K. Hama, T. Matsubara, K. Uehara, Predictable uncertainty-aware unsupervised deep anomaly segmentation, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–7.

[16] S. Akcay, A. Atapour-Abarghouei, T. P. Breckon, Ganomaly: Semi-supervised anomaly detection via adversarial training, in: Asian conference on computer vision, Springer, 2018, pp. 622–637.

[17] P. Perera, R. Nallapati, B. Xiang, Ocgan: One-class novelty detection using gans with constrained latent representations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2898–2906.

[18] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[19] I. G. Courville, Y. Bengio, Aaron, Deep Learning, MIT Press, 2016.

[20] L. A. Gatys, A. S. Ecker, M. Bethge, Image style transfer using convolutional neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2414–2423.

[21] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: European conference on computer vision, Springer, 2016, pp. 694–711.

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4681–4690.

[23] H. Zhao, O. Gallo, I. Frosio, J. Kautz, Loss functions for image restoration with neural networks, IEEE Transactions on computational imaging 3 (1) (2016) 47–57.

[24] S. Liu, W. Deng, Very deep convolutional neural network-based image classification using small training sample size, in: 2015 3rd IAPR Asian conference on pattern recognition (ACPR), IEEE, 2015, pp. 730–734.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[26] T. DeVries, G. W. Taylor, Improved regularization of convolutional neural networks with cutout, arXiv preprint arXiv:1708.04552.