Article

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

Diulhio Candido de Oliveira [1,*] **, Bogdan Tomoyuki Nassu** [1] **and Marco Aurelio Wehrmeister** [1,2]

1 Departament of Informatics, Federal University of Technology—Parana (UTFPR), Curitiba 80230-901, Brazil
2 Computer Science Department, University of Münster, 48149 Münster, Germany
* Correspondence: diulhio@alunos.utfpr.edu.br

Abstract: In this paper, we introduce a one-class learning approach for detecting modifications in assembled printed circuit boards (PCBs) based on photographs taken without tight control over perspective and illumination conditions. Anomaly detection and segmentation are essential for several applications, where collecting anomalous samples for supervised training is infeasible. 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. We propose a loss function that can be used to train a deep convolutional autoencoder based only on images of the unmodified board—which allows overcoming the challenge of producing a representative set of samples containing anomalies for supervised learning. We also propose a function that explores higher-level features for comparing the input image and the reconstruction produced by the autoencoder, allowing the segmentation of structures and components that differ between them. Experiments performed on a dataset built to represent real-world situations (which we made 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 a more general object anomaly detection task.

Keywords: anomaly detection; autoencoder; deep learning; manufacture; visual inspection

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 fraud 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 put into the tank. It may be difficult for law enforcers to detect this 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 mitigate concerns over 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 interesting. Such a system must be flexible enough to work on-site, without requiring large capture structures, controlled lighting, or fixed camera positioning.
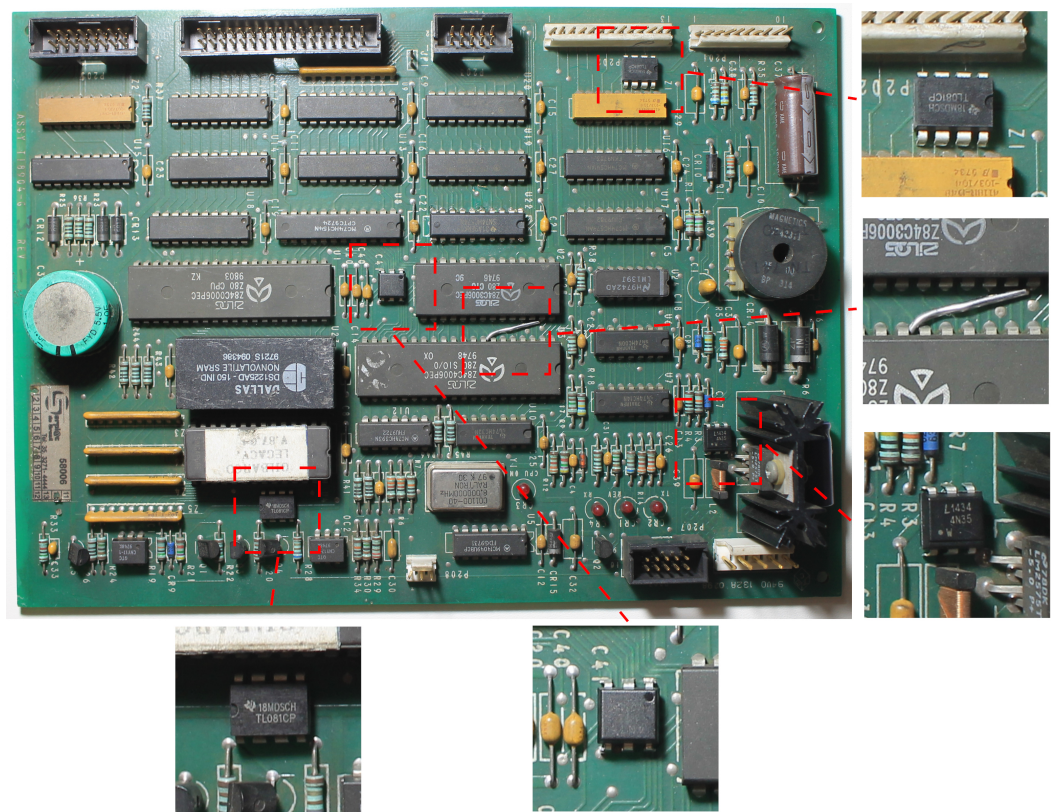
Figure 1. An example of PCB containing modifications. Some of them are easier to identify, while others require more attention. The modifications consist of adding small IC chips, and a jump wire in one case.

While we have the fraud detection scenario as our main motivation, this problem shares most characteristics with the 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 [6–10], where common anomalies are missing holes and open circuits, while other methods deal with assembled PCBs [3,11,12]. These methods are usually based on supervised machine learning, where a decision model is trained by observing samples 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 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 a representative set of anomaly samples is harder to obtain, because anomalies are rare, expensive to reproduce, or may manifest in unpredictable ways.

We adopt a one-class learning approach and address the task as an anomaly detection problem. In this formulation, models are only trained on normal samples, learning to describe their distribution, using the premise that it is possible to detect anomalies based on how well the learned model can 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 in various settings explore this idea [1,2,13–20].

In this paper, we address the problem of detecting modifications in an assembled 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 only based on these features. The rationale behind the proposed method is that, if the model is trained only with anomaly-free samples, it can only reconstruct this kind of sample. Thus, when it receives an image containing anomalies as input, it will

be unable to properly reconstruct the output, or even reconstruct the image without its anomalies. This idea is illustrated in Figure 2.
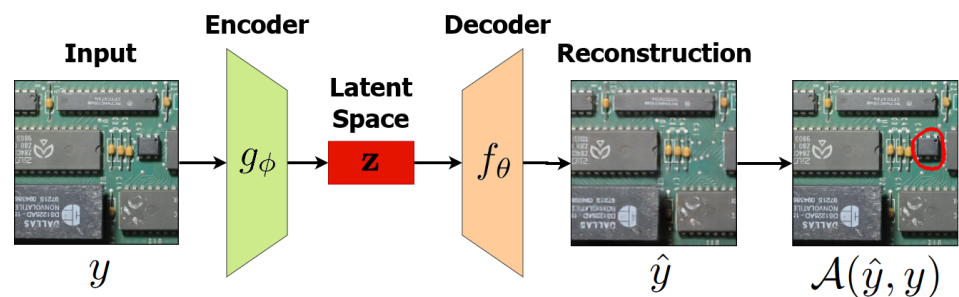
Figure 2. The reconstruction-based inference process using a convolutional autoencoder. The autoencoder is trained to reconstruct only anomaly-free samples, so the reconstructed output does not show the modification when it receives an image containing anomalies as input. Thus, it is possible to segment the anomaly by comparing the input and the output.

We performed experiments comparing our proposed method to other state-of-the-art one-class anomaly detection methods [2,13,15,16] that 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), precision, recall, F-score, 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 adding or removing objects. We also performed an ablation study about the loss function to identify each loss component's contribution and verify the proposed method's effectiveness.

The main contributions of this work are:

- We propose a loss function that combines the content loss concept, and the mean squared error function for training a denoising convolutional autoencoder architecture for reconstruction-based anomaly detection. The proposed model can be trained using only anomaly-free images, making it suitable for real-world applications where this kind of sample is much more common and easier to obtain than a representative set of samples containing anomalies.
- We propose a comparison function that can be used to locate and segment regions that differ between a given input image and the reconstructed image produced by a convolutional autoencoder. The comparison is based on higher-level features instead of individual pixels, leading to the detection of structures and components instead of sparse noise.
- We employ the proposed loss and comparison functions to design a robust method to detect modifications on PCBs that can be applied to images containing perspective distortion, noise, and lighting variations. Thus, the method aims to work under the circumstances commonly found in practice, e.g., during the on-site inspection of gas pump PCBs [3], where mobile devices are used to capture images without relying on controlled lighting or positioning. Nonetheless, it is important to highlight that the proposed method may also be applied to other monitoring tasks with similar characteristics, such as quality assurance in an industrial setting.
- We provide a labeled PCB image dataset for training and evaluating anomaly detection and segmentation methods. The dataset is publicly available (https://github.com/Diulhio/pcb_anomaly/tree/main/dataset (accessed on 10 January 2023)) and contains 1742 4096 × 2816-pixel images from one unmodified gas pump PCB, as well as 55 images containing modifications, along with the corresponding segmentation masks.

The remainder of this paper is organized as follows. Section 2 discusses the related work on defect and anomaly detection on PCBs, as well as anomaly detection 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 indicates 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 circuits in unassembled boards [6–10]. Although these approaches were successful, they rely on controlled capture conditions, and only work for limited types of anomalies, found in unassembled boards. For assembled boards, a common strategy is using supervised training to produce a component detector [11,12]. The layout of the detected components can be compared to a reference, providing a way of detecting anomalies. However, this strategy demands considerable effort to obtain labeled training data (for example, ref. [11] 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 particular 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 region 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 vector machines to classify each region as normal or anomalous, while we segment anomalies using a deep reconstruction network. Moreover, that work uses supervised learning, with anomalies being artificially created by placing small patches extracted from other samples, while our model is single-class, being trained only on normal samples.

Several one-class learning methods were recently proposed, which rely only on normal samples for anomaly detection for industrial visual inspection (not limited to PCBs). The most successful methods are based on reconstructions or embedding similarity. 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 include autoencoders (AEs) [1,15], variational autoencoders (VAEs) [18,21,22], and generative adversarial networks (GANs) [23]. The main advantage of these approaches is that it is easy for humans to understand and interpret their results. However, if a method still reconstructs an anomaly [24], it may remain undetected, as there is no noticeable difference between the input and the reconstruction.

Embedding similarity methods [2,13,16,17,19,20] use deep convolutional networks pre-trained 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 [2]. Given a distance metric, the feature vectors from images with anomalies tend to be more distant from 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 in the basic feature extractor, but their results are difficult to interpret. Moreover, the computation of the density function can have high memory requirements and be complicated when the dataset has high variability.

A popular benchmark for visual anomaly inspection is the MVTec-AD [1,14] dataset, which contains 5354 images, with 70 types of anomalies for 15 kinds of objects. Most 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, indicating whether it contains a modification. 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 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 on samples from a single class, i.e., images without modifications/defects/anomalies.

3.1. Image Registration and Partitioning

Similarly to the work in [3], our approach assumes that the PCB is shown from an overhead view. However, in contrast to several other studies on visual inspection [6,11,14,16], where positioning is strict to avoid variations, we suppose that 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). It is a procedure similar to that employed by applications that involve face images, where the faces are aligned to reduce variability and improve the model performance. We employed a widely used and mature algorithm for image registration based on SIFT features and the RANSAC algorithm [25], but note that any algorithm with good performance could be used. More relevant for our discussion are the implications of relying on an image registration step: 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 only treats planar distortions, without taking into account the 3D aspect of the components, as shown in Figure 4. Moreover, our approach does not require 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.

(**a**)   (**b**)
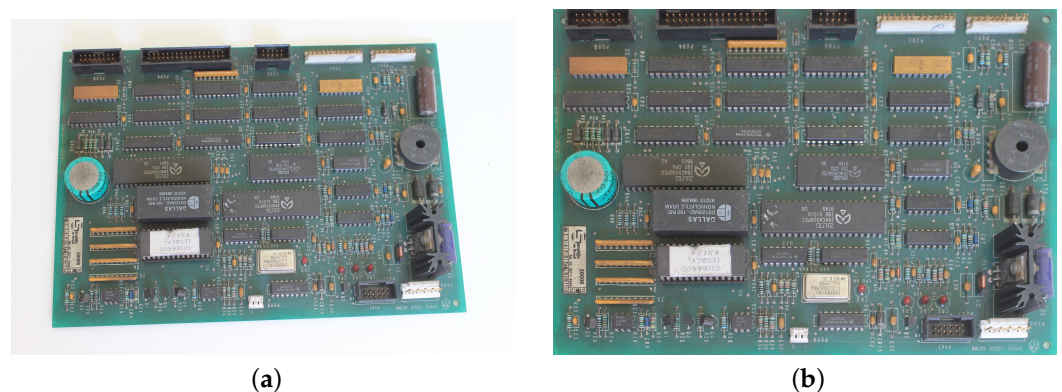
Figure 3. Comparison before and after the image registration process using SIFT and RANSAC: (**a**) Original image; (**b**) The image after registration using SIFT and RANSAC.

Anomaly detection methods frequently work on fixed-size inputs, reducing the captured image to a smaller size and reducing the computation and memory requirements. However, for PCB inspection in the proposed dataset, resizing the entire image to a manageable size can result in certain components and modifications becoming too small. To avoid this, we partition the input image into 1024×1024-pixel patches (to avoid having an overly large number of patches per image), which are then resized to 256×256 pixels (to reduce the computational costs) and processed independently. Figure 5 illustrates this procedure.
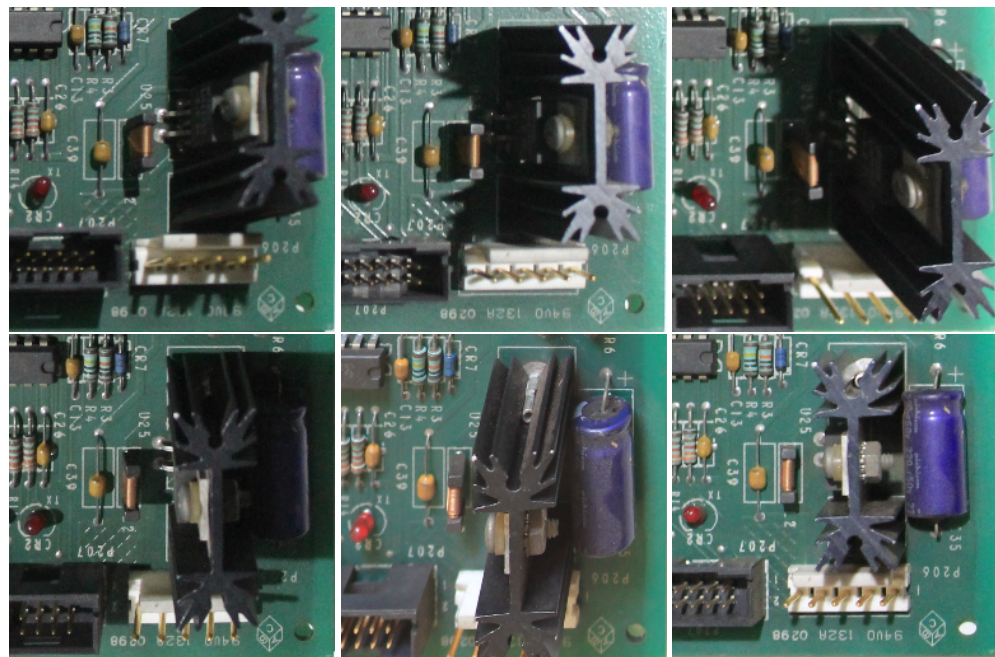
Figure 4. Taller components can have considerable aspect variation even after the image registration procedure.
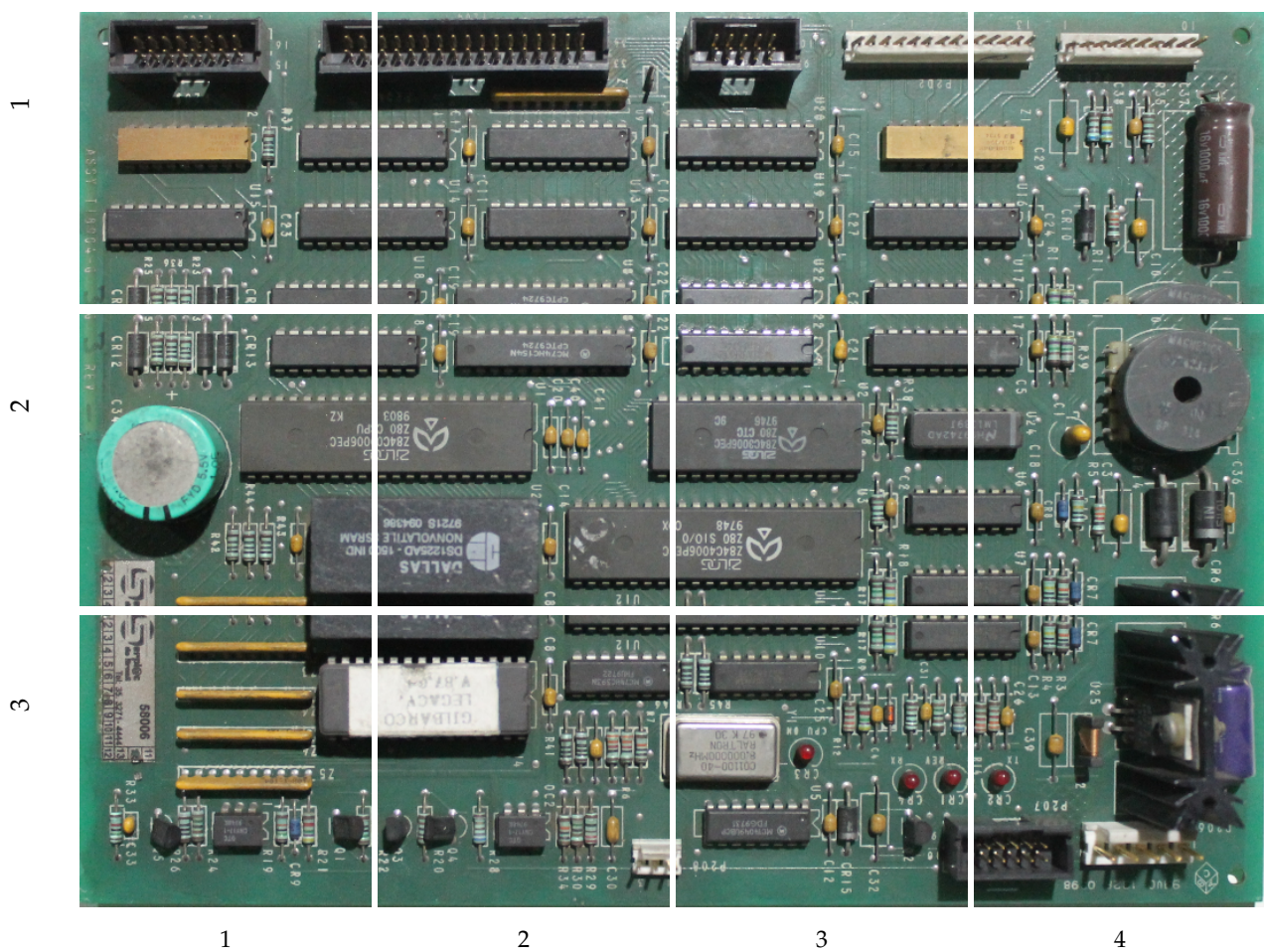

Figure 5. A 4096 × 2816 image split into 1024 × 1024 patches. Some regions are present in more than one patch—the regions overlap because 2816 is not divisible by 1024.

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) [26]. Using a series of convolutional layers, CAEs encode the high-dimensional input image to a compressed low-dimensional vector called "latent space" and expand (decode) this vector to the original dimensionality. The encoder function $z = g(y)$ receives the input y and maps it to the latent space z. The decoder function $\hat{y} = f(z)$ computes the reconstruction $\hat{y}$ from the latent space z. Thus, the entire network is expressed as $f(g(y)) = \hat{y}$, and in a perfect CAE $y = \hat{y}$.

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 Figure 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 a slope of 0.2. The encoder's last layer and the decoder's first layer 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.

Input: $x_{256 \times 256 \times 3}$	Feature Maps
Conv(filters = 32); BN; LeakyReLU;	$128 \times 128 \times 32$
Conv(filters = 64); BN; LeakyReLU;	$64 \times 64 \times 64$
Conv(filters = 128); BN; LeakyReLU;	$32 \times 32 \times 128$
Conv(filters = 128); BN; LeakyReLU;	$16 \times 16 \times 128$
Conv(filters = 256); BN; LeakyReLU;	$8 \times 8 \times 256$
Conv(filters = 256); BN; LeakyReLU;	$4 \times 4 \times 256$
Conv(filters = 256); BN; LeakyReLU;	$2 \times 2 \times 256$
Fully connected (1024); BN; Leaky ReLU;	1024
Fully connected (500); Leaky ReLU;	500
Fully connected (1024); BN; Leaky ReLU;	1024
TranspConv(filters = 256); BN; LeakyReLU;	$4 \times 4 \times 256$
TranspConv(filters = 256); BN; LeakyReLU;	$8 \times 8 \times 256$
TranspConv(filters = 128); BN; LeakyReLU;	$16 \times 16 \times 128$
TranspConv(filters = 128); BN; LeakyReLU;	$32 \times 32 \times 128$
TranspConv(filters = 64); BN; LeakyReLU;	$64 \times 64 \times 64$
TranspConv(filters = 32); BN; LeakyReLU	$128 \times 128 \times 32$
TranspConv(filters = 3); Sigmoid;	$256 \times 256 \times 3$

During training, each input image is corrupted by randomly masking out rectangular regions; denoising autoencoders use this data corruption strategy to prevent the network from simply memorizing the training data. The effect is similar to dropout, but in input space; generating images with simulated occlusions forces the model to consider more of the image context when extracting features, improving network generalization [27]. Note that the loss is still computed by comparing the produced output with the original, non-corrupted input.

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 [28], identifies the 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 [28,29], super-resolution [29,30] and image restoration [31]. Features are extracted from an image classification network (VGG19 [32], in our work) pretrained on general-purpose datasets (Imagenet [33], in our work). This function encourages the network to reconstruct images with feature representations similar to those of the input, rather than considering just differences between pixels.

Let $\phi_j(x)$ be the activation of the jth layer of a pre-trained network ϕ when image x is processed. Since j is a convolutional layer, $\phi_j(x)$ 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 reconstruction $\hat{x}$ and reference x, as expressed in Equation (1).

$$l_{feat}^{\phi,j}(\hat{x}, x) = \frac{1}{C_j H_j W_j} \left\| \phi_j(\hat{x}) - \phi_j(x) \right\|_2^2 \qquad (1)$$

The training procedure based on the content loss function tries to minimize the reconstruction loss between images x and $\hat{x}$ using the initial layers of the pre-trained network ϕ. A CAE trained with this function tends to produce images similar to target x in image content, and an overall spatial structure [29]. 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 the content loss with the MSE, as expressed by Equation (2), where λ_1 and λ_2 are the weights of each loss function. This approach has been applied in several works and methods in the literature [28–30,34]. We empirically defined the parameters $\lambda_1 = 0.01$ and $\lambda_2 = 1$. Figure 6 illustrates the entire loss calculation.

$$\mathcal{L}_{rec} = \lambda_1 \mathcal{L}_{MSE} + \lambda_2 \mathcal{L}_{feat} \qquad (2)$$
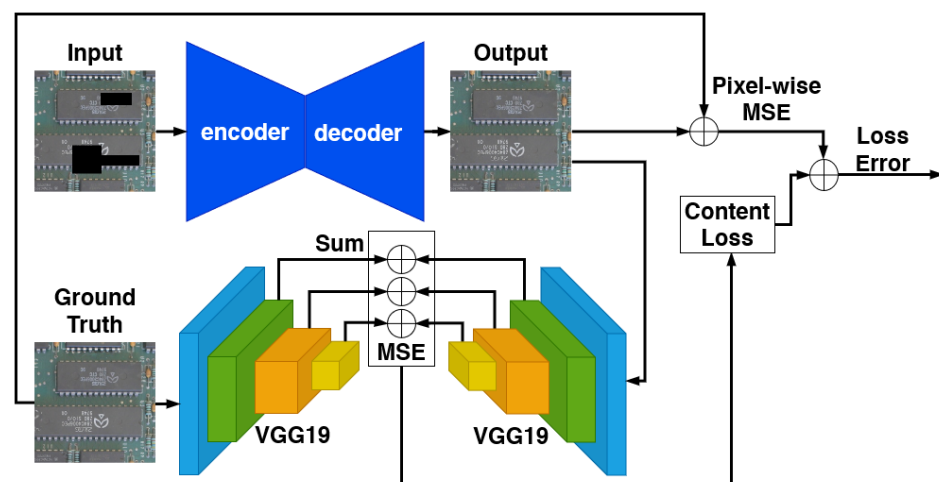

Figure 6. Loss calculation flow during training. The proposed loss function combines the pixel-wise MSE between the autoencoder input and its reconstructed output, and the content loss between the reference (ground truth) image and the reconstruction.

3.4. Anomaly Segmentation

After training, the network can segment anomalies by comparing the reconstructed image to the input. If the CAE were "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 7c shows an example of the absolute difference between the original images of a PCB (with and without modifications) and their reconstructions. The pixel-wise absolute difference has high values at several positions, even in places where the differences are very difficult to notice.

| (a) | (b) | (c) | (d) |
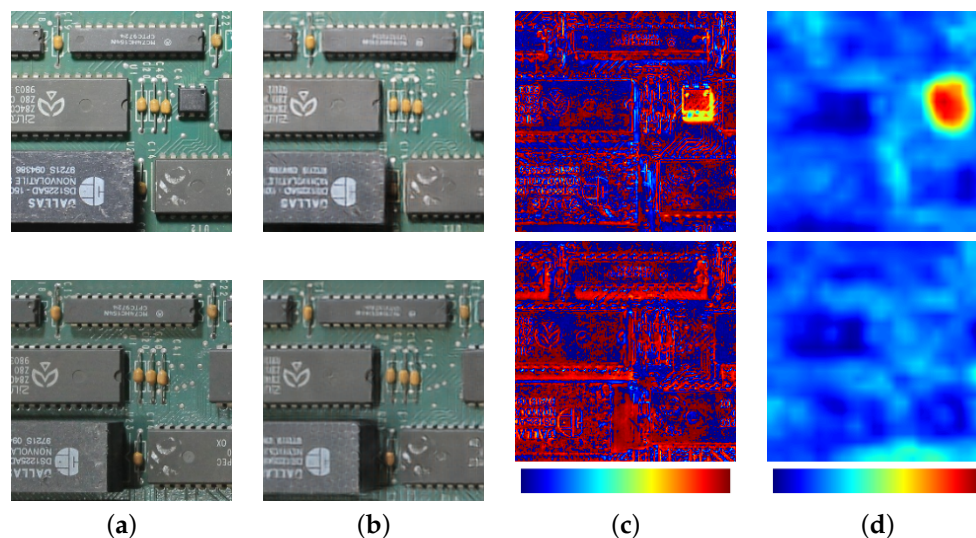
Figure 7. (**a**) Input image showing a PCB with (**top**) and without (**bottom**) modifications, (**b**) reconstruction produced by the autoencoder network, (**c**) the pixel-wise absolute difference between the input and its reconstruction, and (**d**) the proposed anomaly segmentation method using perceptual difference. The absolute difference shows high values spread around many areas, even in regions without modifications. With the proposed method, the region containing the anomaly has markedly higher values than regions without anomalies.

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. Tiny modifications that manifest in isolated pixels may pass undetected, but the overall robustness is increased, since actual modifications to PCBs appear as clusters of pixels, as long as the board is photographed with a good enough resolution. 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 ϕ_j, as expressed in Equation (3).

$$\mathcal{A}(\hat{y}, y) = \sum_{i=0}^{C_j} \left| \phi_{j,i}(\hat{y}) - \phi_{j,i}(y) \right| \tag{3}$$

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

We obtained the final segmentation, resizing the anomaly map 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 it is possible to measure the magnitudes of the values produced by these anomalies. The T parameter measures how rigorous the detection is and will be varied during the experiments, to show how it affects the detection performance (and for computing ROC curves). Figure 7d shows an example of the proposed

segmentation method. Note how differences in regions without anomalies are much less noticeable than when using the pixel-wise absolute difference. On the other hand, the region containing the anomaly has much higher values in the anomaly map than other regions.

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-the-art one-class methods, on our MPI-PCB dataset and the MVTec-AD dataset. The code was implemented in the Python language, using the TensorFlow (www.tensorflow.org, accessed on 15 December 2022) and OpenCV (opencv.org, accessed on 15 December 2022) libraries. Experiments were performed on the Google Colab (colab.research.google.com, accessed on 15 December 2022) platform. The GPUs available in Google Colab may vary due their avalability, but during the experiments we used the P100 GPUs with 16 GB of memory. The source code is publicly available at https://github.com/Diulhio/pcb_anomaly/ (accessed on 15 December 2022).

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 4096 × 2816-pixel images showing an unmodified PCB from a gas pump. The images were captured using a Canon EOS 1100D camera with 18–55 mm lenses. The set also contains 55 images showing the board with modifications manually added by the authors, which are meant to be representative of situations encountered in actual frauds. Since our aim is performing one-class learning, 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, no previous work addresses specifically image-based anomaly segmentation in assembled PCBs—as previously discussed in Section 2, existing approaches focus on unassembled PCBs or use supervised training to determine whether anomalies are present in a given region, without per-pixel segmentation. This makes it hard to directly compare this approach with the proposed method. Therefore, our comparisons are focused on other general anomaly segmentation methods, which achieved promising results on the popular MVTec-AD dataset. Our work can be more directly compared with these methods, since they have similar one-class 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.

When our experiments were performed, the PaDiM approach [13] had state-of-the-art results for anomaly segmentation on the MVTec-AD dataset. It is an embedding similarity method that obtained the best results when using the Wide ResNet-50-2 network to extract features. However, 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 baselines were SPTM [16] 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 [15], which uses regional features extracted from a pre-trained VGG19 as inputs for CAEs.

4.3. Evaluation Metrics

We considered 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), as well as the usual precision, recall, and F-score in the best case. We also evaluated ROC-AUC for anomaly detection: while segmentation considers per-pixel classification, detection expresses whether an anomaly exists in the image. To avoid detecting noise, we consider that 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.

Precision indicates the proportion of detected pixels that were correct, i.e., values close to 1 indicate that there were few false detections. In contrast, recall indicates the proportion of expected pixels that were detected, i.e., values close to 1 indicate that most of the anomalies were detected. More formally, precision (Equation (4)) expresses the ratio of correctly predicted positive samples to the total predicted positive samples; and recall (Equation (5)), also known as true positive rate (TPR), expresses the ratio of correctly predicted positive samples to all the samples in the positive class. The *F*-score (Equation (6)) is the harmonic mean of precision and recall.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall \text{ or } TPR = \frac{TP}{TP + FN} \tag{5}$$

$$F\text{-}score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)} \tag{6}$$

ROC-AUC is a widely used metric for evaluating anomaly segmentation methods, and is usually reported for approaches tested on the MVTec-AD dataset [1,2,13–16]. It shows how well a technique balances true and false positive rates (i.e., its ability to cover the expected detections while avoiding false detections) as a certain threshold parameter varies. ROC-AUC is the normalized area under the ROC curve, which is 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 (5) and (7).

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

IoU, also referred to as the Jaccard index, is also reported for several semantic segmentation tasks and challenges such as COCO (Common Objects in Context—http://cocodataset.org (accessed on 15 December 2022)). For anomaly segmentation, the IoU expresses how similar two shapes are, quantifying the overlap between the ground truth mask and the binarized anomaly map, as given by Equation (8).

$$IoU = \frac{TP}{(TP + FN + FP)} \tag{8}$$

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.

We defined the optimal threshold T in our experiments by varying this parameter and selecting the value which resulted in the maximum geometric mean (G-mean) of recall

and specificity, as given by Equation (9). This method is widely used in machine learning applications, especially in imbalanced classification problems.

$$\text{G-mean} = \sqrt{TPR * (1 - FPR)} \tag{9}$$

4.4. Training Details

We took the 1742 images from the MPI-PCB dataset showing the unmodified board. We randomly split them: 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 the testing [1].

Due to the large variety of perspective distortions, and 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. As for the MVTec-AD dataset, we apply random variations on rotation, shear, saturation, contrast, brightness, and scale.

The proposed architecture was trained with a batch size of 128 for 1000 epochs. As an optimizer, we used Adam with cosine learning rate decay and a warm-up phase. The learning rate starts at 1×10^{-5}, and after three epochs ramps up to 0.0072, and decays to 1×10^{-5} using a cosine function.

4.5. Results on the MPI-PCB Dataset

We evaluated the performance of our method and the baseline methods on the test set from the MPI-PCB dataset, considering six board regions. All these regions contain inserted modifications, such as integrated circuits and jumper wires. We selected these regions because they contain anomalies in the test set; these are needed to compare our approach and other works properly. The tested methods depend on at least part of the test samples from each region containing modifications, to define the range for normalization. A total of 110 samples were tested, namely 55 with and 55 without modifications in the observed region.

Figures 8 and 9 as well as Table 2 show the results obtained with the tested techniques for each region. The bold text in Table 2 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-the-art results in the MVTec-AD dataset.

For simple anomaly detection (measured by the detection ROC-AUC, see Figure 8), our method, PaDiM, and SPTM present similar performances in most cases. The proposed method shows the detection ROC-AUC 1.0 in four out of six 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.

As for anomaly segmentation (Figure 9), all the methods achieved an ROC-AUC higher than 0.9 for almost every region. This shows that these methods can segment most of the anomalies correctly. The proposed method and PaDiM showed the best average performance. The class imbalance explains the difference between the detection and segmentation ROC-AUC results for SPADE and DFR in each problem. The test set is balanced for detection since it contains the same number of positive and negative samples. However, the classes are very imbalanced for per-pixel segmentation, with less than 2% being positive pixels. This allows the model to generate small segmentation errors in several images without impacting the segmentation ROC-AUC, but with a high impact in detection ROC-AUC.

Table 2. Results of the proposed method and the baseline methods for the MPI-PCB dataset. We show the results for image regions containing at least one anomaly in the test set. The grid numbers indicate the column/row of each region in the partitioned image (see Figure 5). Higher values indicate a better performance, where the segmentation precision, recall, and F-score are shown at the best IoU threshold.

Metric	Method	grid2_2	grid2_3	grid3_1	grid3_2	grid4_1	grid4_3	Avg.
IoU	Ours	**0.755**	**0.664**	**0.608**	**0.525**	**0.778**	**0.732**	**0.677**
	PaDiM	0.603	0.624	0.489	0.145	0.656	0.524	0.507
	SPADE	0.319	0.272	0.419	0.353	0.457	0.474	0.382
	DFR	0.297	0.098	0.117	0.386	0.196	0.190	0.214
	SPTM	0.505	0.428	0.447	0.314	0.240	0.502	0.406
Segmen. Precision	Ours	**0.858**	**0.752**	**0.767**	**0.643**	**0.849**	**0.840**	**0.785**
	PaDiM	0.732	0.742	0.594	0.240	0.765	0.687	0.627
	SPADE	0.364	0.301	0.621	0.417	0.526	0.533	0.460
	DFR	0.246	0.078	0.117	0.419	0.141	0.221	0.204
	SPTM	0.601	0.577	0.457	0.410	0.300	0.627	0.495
Segmen. Recall	Ours	**0.876**	**0.858**	0.758	**0.747**	**0.915**	**0.856**	**0.835**
	PaDiM	0.856	0.851	**0.826**	0.413	0.896	0.744	0.764
	SPADE	0.754	0.754	0.572	0.715	0.793	0.833	0.737
	DFR	0.687	0.491	0.310	0.691	0.347	0.407	0.489
	SPTM	0.760	0.853	0.668	0.643	0.395	0.760	0.680
Segmen. F-Score	Ours	**0.863**	**0.805**	**0.769**	**0.688**	**0.889**	**0.851**	**0.811**
	PaDiM	0.785	0.791	0.691	0.307	0.829	0.714	0.686
	SPADE	0.489	0.436	0.597	0.521	0.632	0.645	0.553
	DFR	0.357	0.126	0.163	0.511	0.209	0.283	0.275
	SPTM	0.676	0.687	0.533	0.492	0.347	0.680	0.569
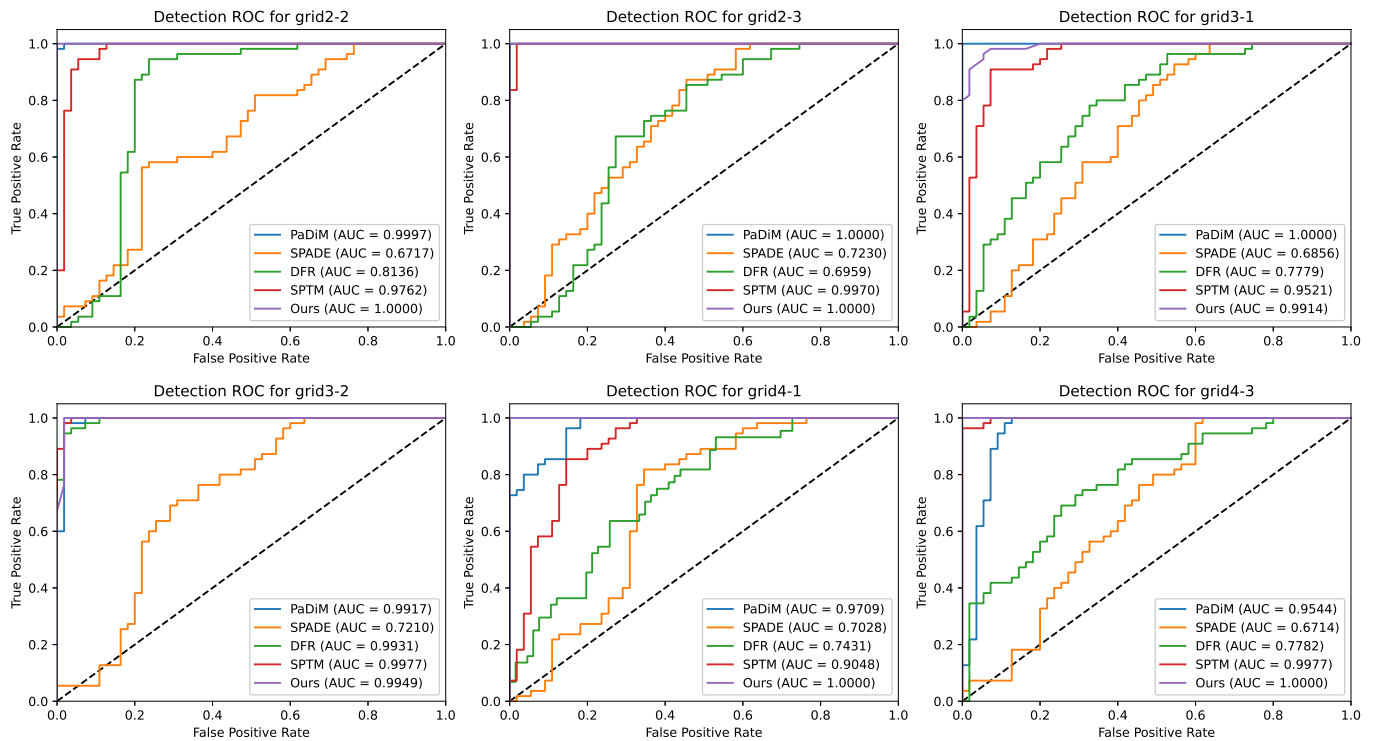
Figure 8. Detection ROC curves and AUC for each tested region. The grid numbers indicate the column/row of each region in the partitioned image (see Figure 5).
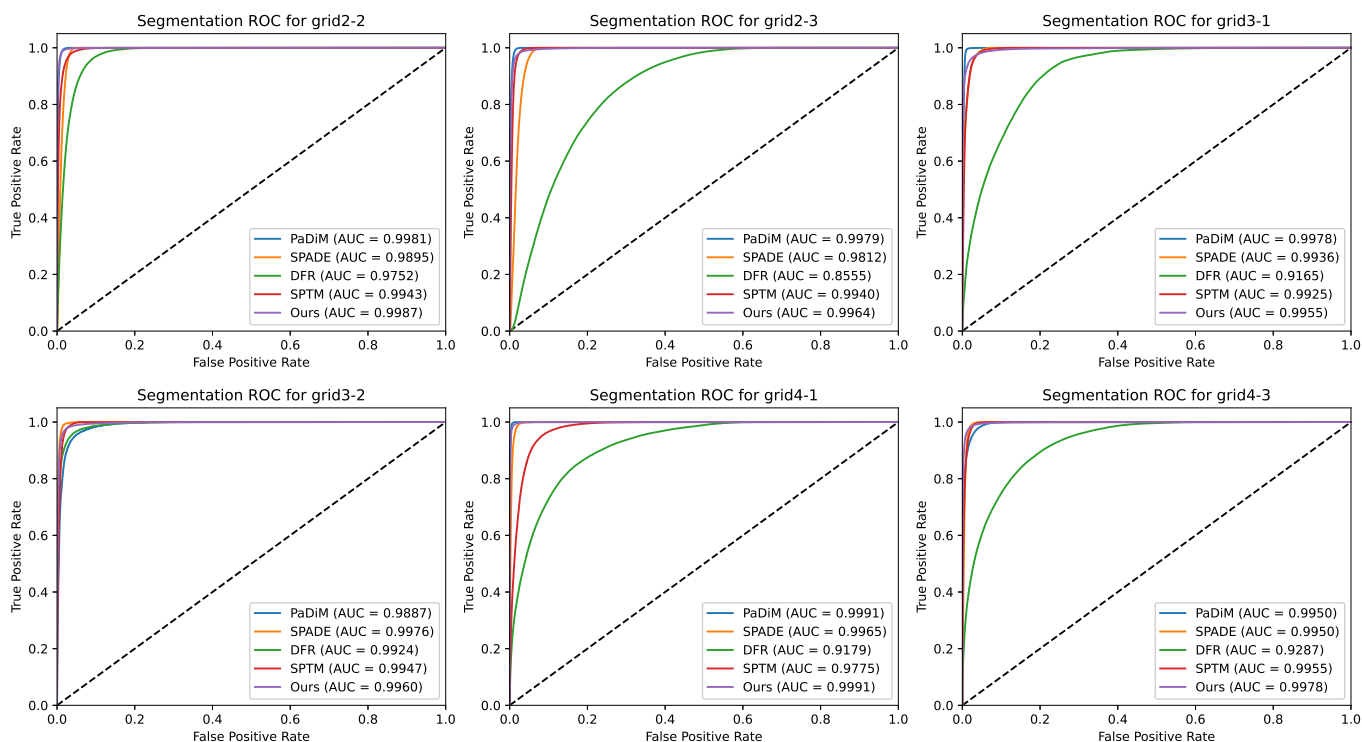
Figure 9. Segmentation ROC curves and AUC for each tested region. The grid numbers indicate the column/row of each region in the partitioned image (see Figure 5).

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 this happened because of the imbalance between the positive and negative pixels, leading to high ROC-AUC values even when the model produces false positive classifications. IoU can express the segmentation precision better than the ROC-AUC, since it is more sensitive to incorrectly classified pixels and, consequently, to deviations in the shape and size of the segmented objects. This can be seen in Figure 10, which shows some segmentation samples produced by our technique and 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. However, 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. This might be interpreted as a false detection by a human inspector without specialized knowledge, because it covers not just a component but a region that includes parts of other components. Additionally, Figure 10 shows that some baseline methods can produce more false positive detections when there are no anomalies in the board.

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 (http://host.robots.ox.ac.uk/pascal/VOC/index.html, accessed on 15 December 2022) and COCO use IoU > 0.5 as one possible criterion for 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, such as the jumper wire in the "grid3_2" region (the first row in Figure 10). PaDiM presented an 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, the performance was worse in several cases. As discussed above, these techniques displayed a higher number of false positives, segmenting large regions around the modifications and

detecting modifications where none exist. The lighting and perspective variations in this dataset can explain that.

The difference between the segmentation quality of the techniques is reinforced if we observe the precision, recall, and F-score metrics. Our method presented the best segmentation precision for all regions, meaning that it could better detect pixels that represent anomalies with fewer false positives. At the same time, regarding segmentation recall, our method outperforms the baseline methods for almost all regions by a significant margin, showing that the proposed method presents less false negatives. Our method's advantages are reflected in the average F-score, which is significantly higher than the one achieved by the baseline methods.
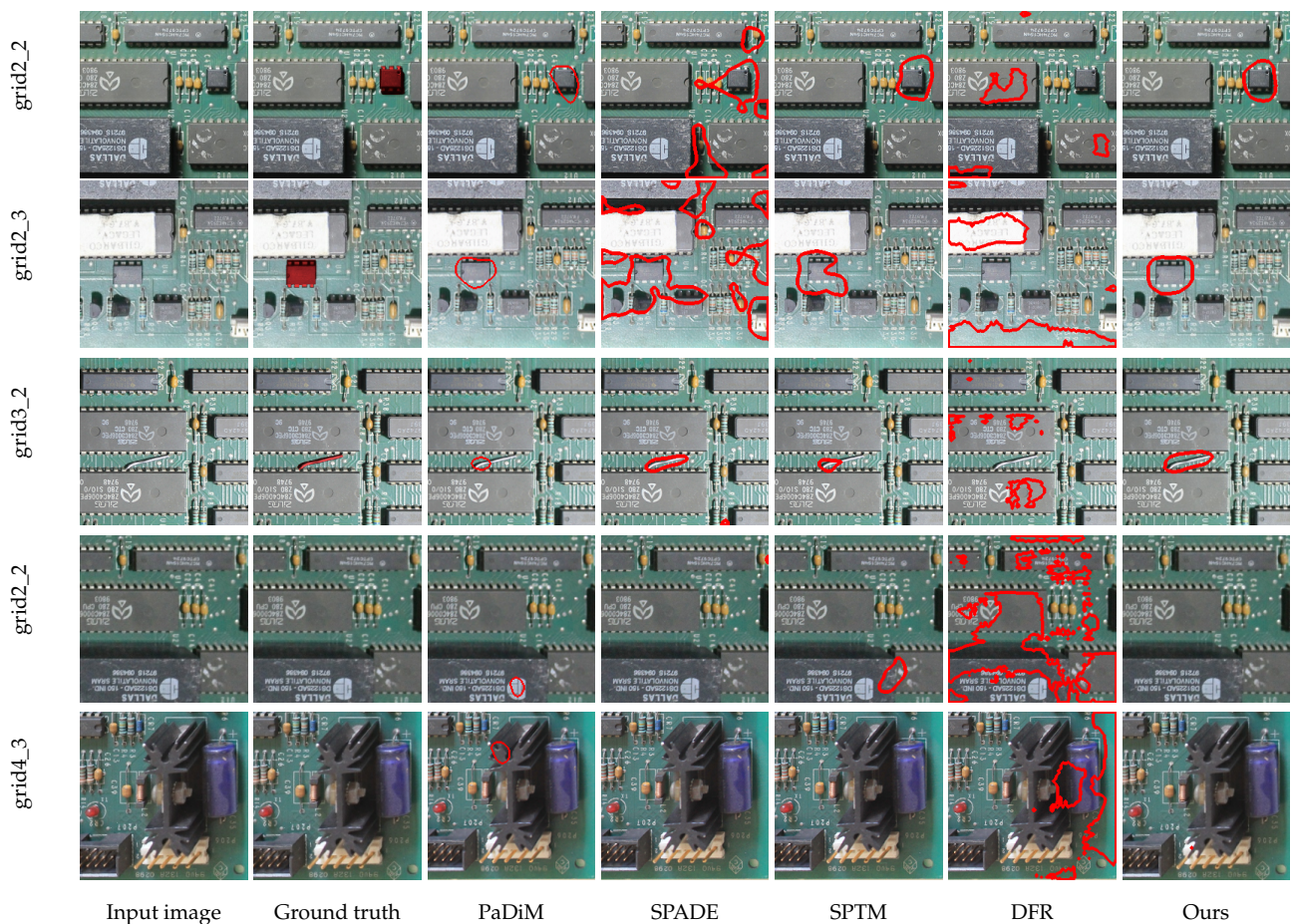
Figure 10. Segmentation comparison between our method and the baseline methods in image regions grid3_2, grid2_3 and grid2_2 with modifications, and in regions grid2_2 and grid4_3 without modifications. The red contours represents the anomalies detected by the evaluated algorithms.

In conclusion, the obtained results show that while all techniques can 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, precision, recall, and F-score). This advantage can help a human inspector identify the specific components that characterize a modification in a practical scenario.

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. Figure 11 shows the dataset's detection and segmentation ROC for all objects and textures. Table 3 shows the evaluated metrics following the categorization defined by [1], with anomalies 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 the close-ups of surfaces, with anomalies consisting of 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 the two main categories defined by textures and objects.

Metric	Method	Texture	Object
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
Segmentation Precision	Ours	0.440	**0.562**
	PaDiM	0.408	0.485
	SPADE	**0.460**	0.518
	DFR	0.364	0.481
	SPTM	0.364	0.389
Segmentation Recall	Ours	0.458	0.625
	PaDiM	0.628	**0.677**
	SPADE	**0.682**	0.640
	DFR	0.614	0.515
	SPTM	0.598	0.575
Segmentation F-score	Ours	0.446	**0.591**
	PaDiM	0.490	0.560
	SPADE	**0.546**	0.571
	DFR	0.452	0.478
	SPTM	0.448	0.382

According to Table 3, our method did not perform as well as the baseline methods for the "texture" category. This behavior can be explained by how the content loss function with the pixel-wise mean squared error was combined. In other tasks, the 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. This means that 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, this 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 only has approximately 50 training images per class.
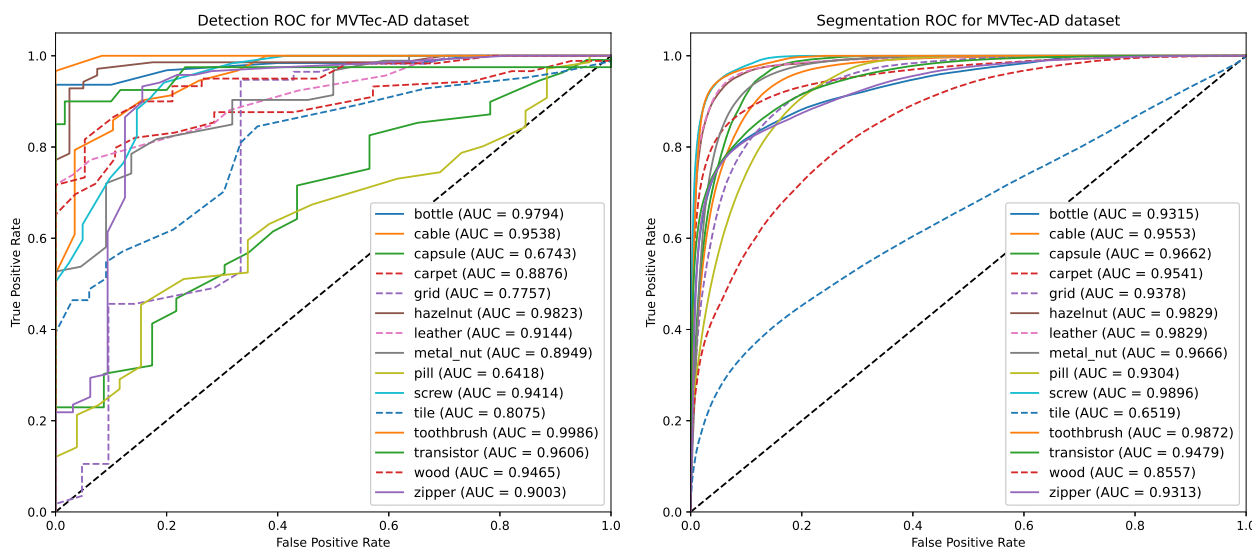
Figure 11. Detection and segmentation ROC and AUC of our method for all textures and objects in the MVTec-AD dataset. Solid lines are used for the "object" category and dashed for the "textures" category.

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.
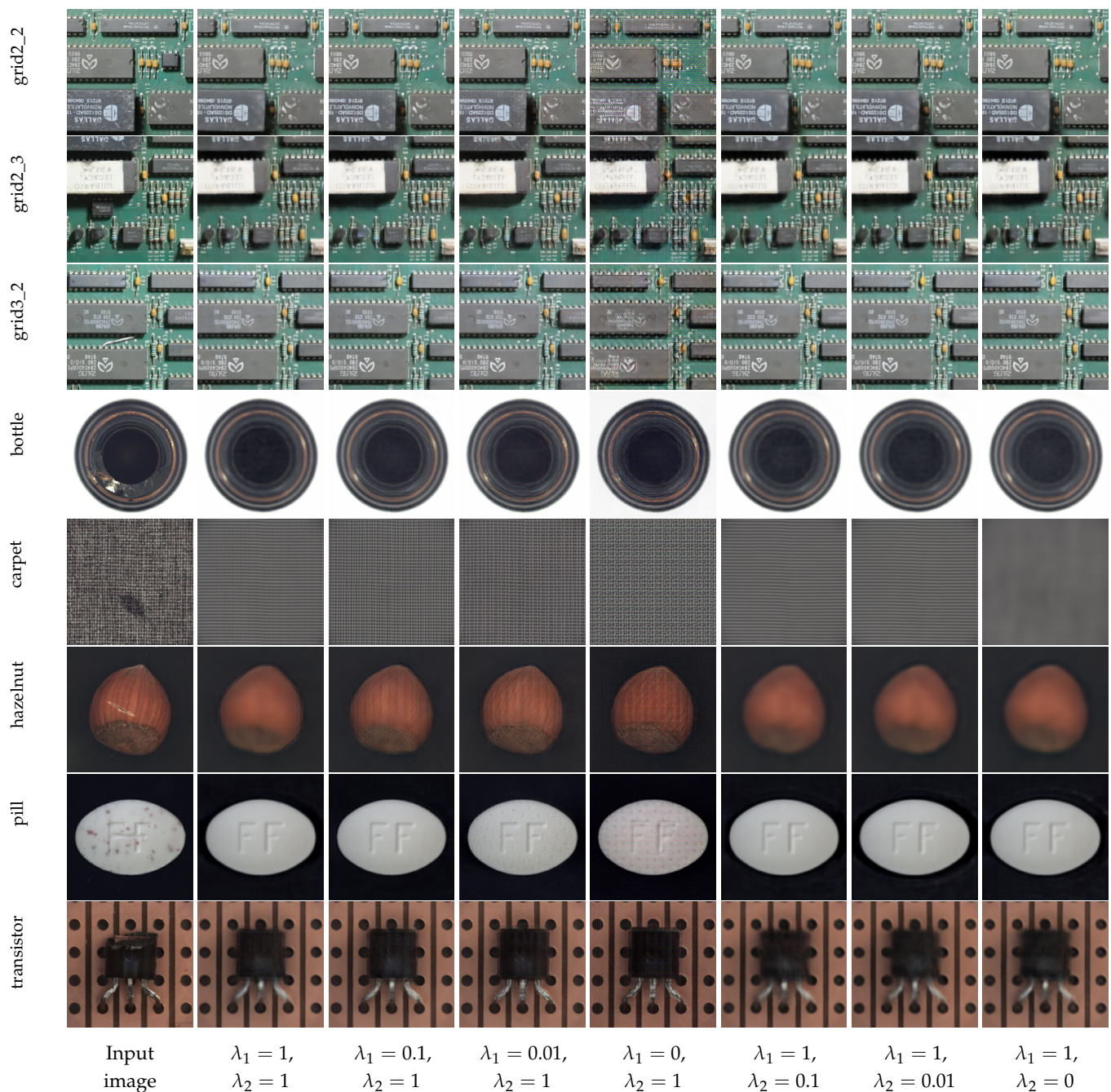
4.7. Loss Function Ablation Study

To investigate the contribution of each component of the loss function, we performed an ablation study about the MSE and perceptual loss weights. As the loss function is the core component of our method, these experiments are essential to identify their advantages and disadvantages in different applications. We conducted these experiments on the MPI-PCB and MVTec-AD datasets, evaluating the qualitative results of the reconstruction. The main objective is to identify the combination of loss component weights that generate reconstruction images that most visually similar to the input image. We used seven different combinations of loss weights, with values of 0, 1, 0.1, or 0.01.

Figure 12 shows the reconstruction results of a few regions from the MPI-PCB dataset, as well as objects and textures from the MVTec-AD dataset. These results show the importance of perceptual loss for reconstruction quality. For the images from the MPI-PCB dataset, perceptual loss plays an essential role in reconstructing fine details. We can observe that when the λ_2 is less than λ_1, the CAE is unable to reconstruct the PCB tracks. Additionally, with lower values of λ_2, known issues from relying solely on the MSE become more evident, such as blurred images and irregular edges. On the other hand, the model trained with only perceptual loss ($\lambda_1 = 0$ and $\lambda_2 = 1$) generates images with irregular textures, especially in regions containing smalls components, such as resistors or integrated circuit legs.

The contribution of perceptual loss is more evident in the reconstructions of the MVTec-AD images. As this dataset has less data available for training, with more weight on the perceptual loss, the model can better reconstruct images with fine details and consistent edges, since content loss relies on another model, which was pre-trained on a large dataset. Moreover, in all combinations, the major limitation of our loss function is the difficulty of reliably reconstructing texture patterns. We can observe this behavior on the carpet and hazelnut reconstructions, where all evaluated models presented problems reconstructing textures. This behavior is explained by the nature of the convolutional filters present in the pre-trained models used by perceptual loss. Recent works [35] proved that convolutional filters tend to behave similarly to high-pass filters used to detect edges,

corners, and other abrupt intensity changes. This explains why perceptual loss significantly improves the reconstruction of edges and fine details, but also fails to reconstruct texture patterns. Our experiments demonstrate the importance of perceptual loss and reinforce the conclusions obtained during the experiments with MPI-PCB and MVTec-AD datasets, that our approach is more suited to detect changes in structures and objects than in texture patterns. Furthermore, these experiments demonstrate the effectiveness of perceptual loss on image reconstruction with less training data.

| | Input image | $\lambda_1 = 1,$ $\lambda_2 = 1$ | $\lambda_1 = 0.1,$ $\lambda_2 = 1$ | $\lambda_1 = 0.01,$ $\lambda_2 = 1$ | $\lambda_1 = 0,$ $\lambda_2 = 1$ | $\lambda_1 = 1,$ $\lambda_2 = 0.1$ | $\lambda_1 = 1,$ $\lambda_2 = 0.01$ | $\lambda_1 = 1,$ $\lambda_2 = 0$ |

Figure 12. Reconstruction results of different grids from the MPI-PCB dataset, and objects and textures from the MVTec-AD dataset. In these experiments, we vary the weights λ_1 and λ_2 from Equation (2) in ranges between 0 and 1.

4.8. Discussion

The results show that our method can successfully segment anomalies in the images of assembled PCBs taken without tight control over perspective and illumination conditions.

In the MPI-PCB dataset, our method outperformed the state-of-the-art baseline methods, showing superior performance on segmentation and detection. For anomaly segmentation, our method presented the approximated shape of the anomalies in all evaluated regions, showing less false positive and false negative pixels. A better segmentation may be advantageous for a human inspector identifying the specific components as modifications. The experiments performed on the MVTec-AD dataset demonstrated that our method could be used for anomaly detection in other contexts, when the analyzed object or surface does not contain textures with random patterns.

One limitation of the proposed approach is that it is only capable of detecting visible modifications in the images that form structures occupying groups of pixels—which means it may fail if the images have very poor quality or low resolution. Although this can be avoided simply by using good cameras and taking some care when capturing the images, invisible modifications are still undetectable—e.g., some modifications are hidden below a chip, which is removed and resoldered; and others involve replacing memory units or cloning components. These modifications cannot be detected by any vision-based approach, requiring radically different approaches, such as electrical tests or completely disassembling the board. However, we note that our approach was mainly designed to support the work of human inspectors, which, in the considered scenario, perform their work solely based on visual cues, so detecting this kind of invisible modification is outside the scope of our work.

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 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 detection and segmentation methods. Experiments on that dataset showed that our method has superior results for modification segmentation compared to other state-of-the-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, or evaluate the possibility of using techniques such as transfer learning and fine-tuning to adapt models trained for one PCB to another quickly. Another possible improvement is designing a loss function capable of better learning texture information, based on techniques such as adversarial learning.

References

1. Bergmann, P.; Fauser, M.; Sattlegger, D.; Steger, C. MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9592–9600. [CrossRef]
2. Cohen, N.; Hoshen, Y. Sub-image anomaly detection with deep pyramid correspondences. *arXiv* **2020**, arXiv:2005.02357. https://doi.org/10.48550/arXiv.2005.02357.
3. De Oliveira, T.J.M.; Wehrmeister, M.A.; Nassu, B.T. Detecting modifications in printed circuit boards from fuel pump controllers. In Proceedings of the 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Rio de Janeiro, Brazil, 17–20 October 2017; IEEE: Niteroi, RJ, Brazil, 2017; pp. 87–94. [CrossRef]
4. Chakraborty, P. The Times of India—8 Lucknow Pumps Caught Using 'Cheating' Chip. Available online: https://timesofindia.indiatimes.com/city/lucknow/8-city-pumps-caught-using-cheating-chip/articleshow/58407561.cms (accessed on 15 December 2021).
5. Slattery, G. Reuters—Special Report: In Brazil, Organized Crime Siphons Billions from Gas Stations. Available online: https://www.reuters.com/article/us-brazil-fuel-crime-special-report-idUSKBN2B418U (accessed on 15 December 2021).
6. Adibhatla, V.A.; Chih, H.C.; Hsu, C.C.; Cheng, J.; Abbod, M.F.; Shieh, J.S. Defect detection in printed circuit boards using you-only-look-once convolutional neural networks. *Electronics* **2020**, *9*, 1547. [CrossRef]
7. Shi, W.; Zhang, L.; Li, Y.; Liu, H. Adversarial semi-supervised learning method for printed circuit board unknown defect detection. *J. Eng.* **2020**, *2020*, 505–510. [CrossRef]
8. Kim, J.; Ko, J.; Choi, H.; Kim, H. Printed Circuit Board Defect Detection Using Deep Learning via A Skip-Connected Convolutional Autoencoder. *Sensors* **2021**, *21*, 4968. [CrossRef] [PubMed]
9. Adibhatla, V.A.; Huang, Y.C.; Chang, M.C.; Kuo, H.C.; Utekar, A.; Chih, H.C.; Abbod, M.F.; Shieh, J.S. Unsupervised Anomaly Detection in Printed Circuit Boards through Student-Teacher Feature Pyramid Matching. *Electronics* **2021**, *10*, 3177. [CrossRef]
10. Volkau, I.; Mujeeb, A.; Dai, W.; Erdt, M.; Sourin, A. The Impact of a Number of Samples on Unsupervised Feature Extraction, Based on Deep Learning for Detection Defects in Printed Circuit Boards. *Future Internet* **2022**, *14*, 8. [CrossRef]
11. Li, D.; Li, C.; Chen, C.; Zhao, Z. Semantic Segmentation of a Printed Circuit Board for Component Recognition Based on Depth Images. *Sensors* **2020**, *20*, 5318. [CrossRef] [PubMed]
12. Mallaiyan Sathiaseelan, M.A.; Paradis, O.P.; Taheri, S.; Asadizanjani, N. Why Is Deep Learning Challenging for Printed Circuit Board (PCB) Component Recognition and How Can We Address It? *Cryptography* **2021**, *5*, 9. . [CrossRef]
13. Defard, T.; Setkov, A.; Loesch, A.; Audigier, R. Padim: A patch distribution modeling framework for anomaly detection and localization. In Proceedings of the International Conference on Pattern Recognition, Shanghai, China, 15–17 October 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 475–489. ._35. [CrossRef]
14. Bergmann, P.; Batzner, K.; Fauser, M.; Sattlegger, D.; Steger, C. The MVTec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *Int. J. Comput. Vis.* **2021**, *129*, 1038–1059. . [CrossRef]
15. Shi, Y.; Yang, J.; Qi, Z. Unsupervised anomaly segmentation via deep feature reconstruction. *Neurocomputing* **2021**, *424*, 9–22. [CrossRef]
16. Wang, G.; Han, S.; Ding, E.; Huang, D. Student-teacher feature pyramid matching for unsupervised anomaly detection. *arXiv* **2021**, arXiv:2103.04257. https://doi.org/10.48550/arXiv.2103.04257.
17. Roth, K.; Pemula, L.; Zepeda, J.; Schölkopf, B.; Brox, T.; Gehler, P. Towards total recall in industrial anomaly detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 14318–14328. . [CrossRef]
18. Wang, L.; Zhang, D.; Guo, J.; Han, Y. Image Anomaly Detection Using Normal Data Only by Latent Space Resampling. *Appl. Sci.* **2020**, *10*, 8660. [CrossRef]
19. Hermann, M.; Umlauf, G.; Goldlücke, B.; Franz, M.O. Fast and Efficient Image Novelty Detection Based on Mean-Shifts. *Sensors* **2022**, *22*, 7674. [CrossRef] [PubMed]
20. Tang, T.W.; Hsu, H.; Huang, W.R.; Li, K.M. Industrial Anomaly Detection with Skip Autoencoder and Deep Feature Extractor. *Sensors* **2022**, *22*, 9237. [CrossRef] [PubMed]
21. Venkataramanan, S.; Peng, K.C.; Singh, R.V.; Mahalanobis, A. Attention guided anomaly localization in images. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 485–503. ._29. [CrossRef]
22. Sato, K.; Hama, K.; Matsubara, T.; Uehara, K. Predictable uncertainty-aware unsupervised deep anomaly segmentation. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–7. [CrossRef]
23. Akcay, S.; Atapour-Abarghouei, A.; Breckon, T.P. Ganomaly: Semi-supervised anomaly detection via adversarial training. In Proceedings of the Asian Conference on Computer Vision, 2–6 Perth, Australia, December 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 622–637. ._39. [CrossRef]
24. Perera, P.; Nallapati, R.; Xiang, B. Ocgan: One-class novelty detection using gans with constrained latent representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2898–2906. [CrossRef]

25. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. ./B:VISI.0000029664.99615.94. [CrossRef]
26. Courville, I.G.; Bengio, Y.; Aaron. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
27. DeVries, T.; Taylor, G.W. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv* **2017**, arXiv:1708.04552. https://doi.org/10.48550/arXiv.1708.04552.
28. Gatys, L.A.; Ecker, A.S.; Bethge, M. Image style transfer using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2414–2423. [CrossRef]
29. Johnson, J.; Alahi, A.; Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 694–711. ._43. [CrossRef]
30. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; 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, Honolulu, HI, USA, 21–26 July 2017; pp. 4681–4690. [CrossRef]
31. Zhao, H.; Gallo, O.; Frosio, I.; Kautz, J. Loss functions for image restoration with neural networks. *IEEE Trans. Comput. Imaging* **2016**, *3*, 47–57. [CrossRef]
32. Liu, S.; Deng, W. Very deep convolutional neural network based image classification using small training sample size. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3–6 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 730–734. [CrossRef]
33. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE conference on computer vision and pattern recognition, Miami, FL, USA, 20–25 June 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 248–255. [CrossRef]
34. Tang, Y.; Tan, S.; Zhou, D. An Improved Failure Mode and Effects Analysis Method Using Belief Jensen–Shannon Divergence and Entropy Measure in the Evidence Theory. *Arab. J. Sci. Eng.* **2022**, 1–14. [CrossRef]
35. Park, N.; Kim, S. How Do Vision Transformers Work? In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022. [CrossRef]