computer vision
& data science

Anomaly Detection for the Identification of Building Components Defects

NHL Stenden Lectoraat in Computer Vision & Data Science

Hamidreza Amiri

Supervisor: Lucas Alexandre Ramos

Abstract—Building inspection is essential for assessing the technical performance of buildings and planning maintenance actions to improve their condition and functionality. Traditional manual inspection methods are time-consuming, costly, and hazardous. Utilizing anomaly detection as a computer vision technique can automate the building inspection process. However, automated building inspection faces challenges due to the diverse conditions, materials, and maintenance histories of buildings, as well as limitations in imaging quality and component identification. This paper focuses on the automatic identification of defective construction components, specifically roofs and window casings, using anomaly detection. A dataset of drone images from buildings in the Netherlands was generated, providing diverse perspectives and a large amount of training data. The research aims to evaluate the effectiveness of anomaly detection in identifying defects in roofs and window casings, explore the factors influencing accuracy and efficiency, and identify limitations and challenges in real-world scenarios. The PatchCore anomaly detection model, which uses a patch-based approach, is employed as the primary model in this research. Different backbone networks, such as WideResNet50, DeepLabv3+, and ResNet50, and different feature layers of them were investigated for feature extraction. The best performance on the Casings dataset was achieved by using the WideResNet50 backbone with layers 2 and 3, and despite the limited number of data and the diversity in casing types, increasing the number of properly labeled images could further improve the model's performance. On the other hand, the model's performance on the Roof dataset was generally unsatisfactory. The tiled roof dataset consistently showed improved results, demonstrating the model's ability to identify nominal and defective roof tiles with more data and uniformity. The retrained ResNet50 backbone architecture outperformed the baseline performance in the PatchCore anomaly detection model.

Index Terms—Anomaly Detection, Defect Detection, Automatic Building Inspection, PatchCore Anomaly Detection, Computer Vision, Deep Learning

1 Introduction

Building inspection plays a crucial role in technical performance assessment of a building for planning maintenance actions to enhance its condition and functionality [1]. One of the key tasks in building inspection is the identification of building defects, which are deviations from the normal or expected condition of a building that can potentially impact its structural integrity, comfort, and energy efficiency. The detection of building defects is traditionally performed through manual inspection, which is time-consuming, costly, and dangerous, particularly for extensive or tall structures [2].

By employing anomaly detection as a computer vision technique, the building inspection process can be automated, reducing the need for manual inspection. It could enable building owners and managers to plan maintenance actions more effectively, enhancing the condition and functionality of their buildings. Anomaly detection is the process of identifying events or data points that deviate from the majority of regular patterns in a dataset [3]. In the context of building defects, anomaly detection can be used to identify and classify deviations from the normal or expected condition of a building.

Automated building inspection using anomaly detection presents several challenges. Buildings can vary greatly in their condition, construction materials, and maintenance history, which makes it challenging to develop a universal approach to defect identification and classification. Building defects can take many forms, and some may be more challenging to detect and classify than others, requiring sophisticated techniques for automated inspection. Additionally, imaging can be limited by factors such as image quality, angle, and the challenge of identifying distinct building components within a single image, which also adds complexity to anomaly detection and could require specific techniques and models tailored to each component.

This study aims to develop an approach that can be used in industry for the automatic identification of defective construction components, based on a dataset generated by drone imaging from many buildings in different locations in the Netherlands. The focus of this research is developing an anomaly detection approach for improving the metrics of identifying defects in roofs and window casings. We selected roofs and windows for our dataset due to their abundance and the availability of a substantial number of annotated defects. The annotations for both the components and defects were meticulously carried out by domain specialists.

To achieve this, we have explored the effectiveness of anomaly detection in a real-world setting and evaluated its metrics in identifying defects in roofs and window casings. This study addresses the challenges of dataset diversity and annotation accuracy in building inspection and defect detection. It adapts the PatchCore model, a state-of-the-art anomaly detection model, and explores leveraging alternative backbone networks, retraining them for classification and segmentation, and utilizing deeper layers of features. The aim is to improve performance in detecting building defects and enhance reliability.

- What is the performance of anomaly detection techniques in identifying defects in roofs and window casings?
- What are the key factors influencing accuracy and efficiency?
- What are the limitations and challenges associated with using anomaly detection techniques for identifying defects in roofs and window casings in real-world scenarios?

- *Hamidreza Amiri is a Computer Vision and Data Science master student at the NHL Stenden University of Applied Sciences, E-mail: hamidreza.amiri@student.nhlstenden.com*
- *Lucas Alexandre Ramos is a researcher at the NHL Stenden Professorship in Computer Vision & Data Science, E-mail: lucas.ramos@nhlstenden.com*

2 Related Work

In recent years, deep learning and computer vision have been extensively employed to develop new methods for detecting and localizing surface defects in building components and industrial products.

Some approaches are based on different types of Convolutional Neural Networks (CNNs). Pathak et al. [4] utilized 2D and 3D data to train models for detecting and localizing damages. Their proposed method employed various Faster-RCNN configurations, such as Resnet-101, Inception V2, Inception-Resnet, Resnet-FPN, and Nas, for damage detection in 3D models of a UNESCO World Heritage site in India. The results demonstrated the robustness of the proposed approach over image data pertaining to architectural styles of heritage monuments. Nasrollahi et al. [5] proposed a method for detecting concrete surface defects using a Deep Neural Network (DNN) based on LiDAR scanning. They employed PointNet, a CNN, to analyze 3D point sets of bridge surfaces and successfully detect surface defects. Staar et al. [6] proposed the use of CNNs for automated optical quality inspection in industrial applications. Their approach involved employing deep metric learning with triplet networks to detect anomalies, even for new surface/defect classes not included in the training data. The average area under the curve (AUC) metric for different classes and experiments was around 0.8, indicating promising results.

Furthermore, Generative Adversarial Networks (GANs) have been widely used in anomaly detection problems. Hong et al. [7] proposed a novel method for upsampling low-density point clouds, which has applications in crack detection, depth calculation, and segmentation tasks. By combining point cloud data with corresponding 2D images of the object, their GAN architecture generated new, high-density point clouds, leading to improved accuracy in crack detection, depth calculation, and segmentation compared to previous methods.

Transformer-based networks have also been employed for anomaly detection. Researchers have proposed defect-aware Transformer-based networks for the surface of products in manufacturing defect detection for industrial quality control. These networks utilize a Transformer-based encoder architecture, comprising multiple Transformer layers, to address the limitations of CNNs in capturing long-range dependencies necessary for detecting tiny and irregular defects. The integration of Defect-aware modules equips the models with the ability to perceive and capture geometric and characteristic features of defects. Additionally, Global Positional Encoding (GPE) provides crucial positional information, leading to improved model performance and increased adaptability to varying defect patterns [8]. Another proposed approach includes a two-stage end-to-end Transformer-based encoder-decoder for surface defect detection in industrial quality control. This approach demonstrates effectiveness and can be trained with a relatively small number of samples, making it suitable for industrial applications with limited sample sizes [9].

Siamese networks have been utilized to learn similarity metrics between input samples. Takimoto et al. [10] proposed an anomaly detection method using a Siamese network with an attention mechanism and Attention Branch Loss (ABL) to address the lack of sufficient abnormal data for training deep-learning models in practical applications. Experimental results showed the effectiveness of the proposed method, even with limited abnormal data.

Supervised segmentation models have also been employed for detecting defects on surfaces. Tabernik et al. [11] presented a deep learning approach with two sets of CNN networks for detecting and segmenting surface cracks with a small number of training samples. Their proposed method outperformed other state-of-the-art segmentation networks, such as U-net and DeepLab v3, and required only around 25-30 defective samples for training. Additionally, they created a new dataset based on a real-world quality control case, making it publicly available for further research and evaluation.

Unsupervised learning frameworks have been used to detect cracks in 3D laser-scanned point clouds of building components in post-disaster scenarios. The CrackEmbed method [12], a point-based deep neural network, extracted discriminative point features using a feature embedding network trained with the triplet loss function. An unsupervised anomaly detection algorithm was then applied to separate damaged points from non-damaged points based on their distribution in feature space, allowing for the identification of cracked regions. The proposed method achieved high accuracy and was evaluated in real-world disaster scenarios.

In addition, some models adopt a memory bank of nominal features obtained from a pre-trained backbone network to detect anomalies at both the image and pixel levels. These models, such as PatchCore, Padim, and SPADE [13], offer competitive inference times and achieve state-of-the-art performance for both detection and localization.

In the realm of deep learning and computer vision, various approaches have been developed for the detection and localization of surface defects in building components and industrial products. These include techniques based on CNNs, GANs, Transformers, Siamese networks, supervised segmentation networks, and more. Among these methods, the PatchCore anomaly detection model has garnered attention due to its unique features and advantages. One of its main advantages is that it only requires normal images for training, making it attractive for many use cases. By leveraging a memory bank of nominal features derived from a pre-trained backbone network, the model effectively detects anomalies at both the image and pixel levels. The model offers competitive inference times while achieving state-of-the-art performance for both detection and localization [13]. Unlike supervised models, anomaly detection models like PatchCore can detect previously unseen anomalies, as they are trained solely on nominal data without relying on known defects encountered during the training process. This advantage makes them suitable for scenarios with limited or unavailable training data for specific defects. Additionally, the structure of the model provides the flexibility to explore different configurations beyond the original paper, such as leveraging various backbones and different output layers tailored to specific requirements. Through rigorous experimentation and evaluation, the effectiveness of the PatchCore model can be assessed in specific domains, contributing to advancements in anomaly detection for real-world datasets with diverse characteristics.

3 Materials and Methods

This section explains important details about networks, datasets, annotations of data, and basic calculations that have been used in this research. The main flow of this research starts with the anomaly detection model and related dataset and annotation.

3.1 PatchCore Anomaly Detection

In this paper, we use a state-of-the-art method for anomaly detection in images called PatchCore. PatchCore is an anomaly detection model that uses a patch-based approach to detect anomalies in images. It works by dividing an image into patches and extracting features from each patch. PatchCore has been shown to achieve state-of-the-art performance on several benchmark datasets, including the MVTec AD and STC datasets [13].

Figure 1 provides a visual representation of the PatchCore model and its functioning. The model follows a two-step process for training and testing. During the training phase, the PatchCore model takes normal images as input and processes them using a backbone network. The backbone network extracts patch-level features from the input images. The backbone network used in the PatchCore model was pre-trained on the ImageNet dataset. However, it is important to note that the original research primarily focused on leveraging medium-level layers to mitigate potential biases introduced by the pre-trained network towards the features present in ImageNet classes. In deeper layers, there is a possibility of increased bias towards ImageNet features, which we aim to address and investigate in this study. To reduce storage requirements and inference time, a coreset subsampling technique is applied to select representative patches. The features of these patches are then stored in a memory bank. The memory bank serves as a repository of

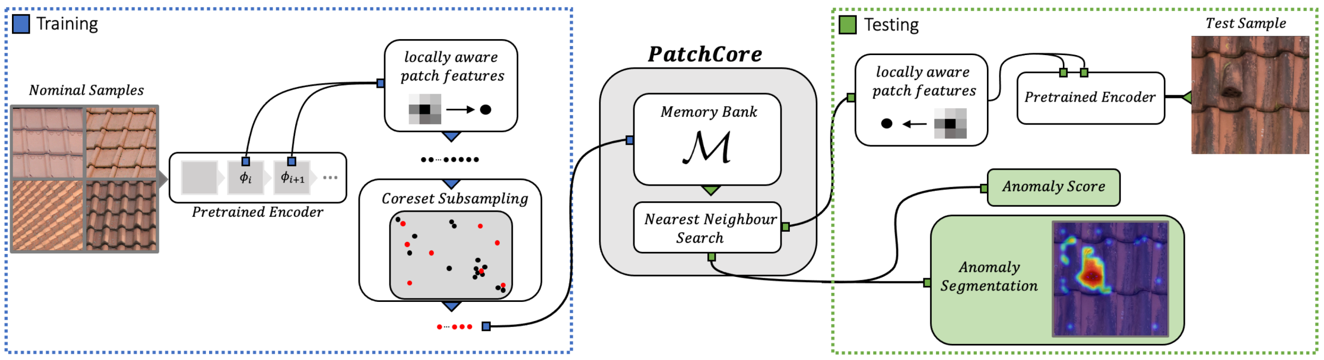


Fig. 1: An Overview of the PatchCore Model [13]. The PatchCore model is trained by passing normal images through a backbone network to extract patch-level features. These features are then stored in a memory bank, representing the pattern of normal images. During testing, the patch-level features of test set images are compared to the normal patches stored in the memory bank using a nearest neighbor search. Based on the differences or similarities of the patch-level features, an anomaly score is calculated for each patch in the image.

normal patch-level features, capturing the patterns and characteristics of the normal images used for training. These features act as a reference for comparison during the testing phase.

In the testing phase, the PatchCore model takes test images as input. Similar to the training phase, the backbone network extracts patch-level features from these test images. These features are compared to the stored normal patch-level features in the memory bank using a nearest neighbor search. By measuring the difference or similarity between the patch-level features of the test images and the stored normal patches, an anomaly score is calculated for each patch in the test images. This anomaly score indicates the likelihood of a patch being an anomaly or deviating from the normal patterns learned during training. The PatchCore model leverages the collective anomaly scores of the patches to identify and localize anomalies within an image. A higher anomaly score suggests a higher likelihood of an anomaly being present in the corresponding patch.

It is important to note that the visual representation provided in Figure 1 offers a clearer understanding of the model's architecture and the flow of data during training and testing. By referring to the figure, one can easily grasp the key steps involved in the PatchCore model and how it operates to detect anomalies in images.

3.2 Backbone Networks

3.2.1 WideResNet50

The original PatchCore model utilizes a pre-trained ResNet network based on the ImageNet dataset, specifically the WideResNet50 or WideResNet101 architectures. However, due to potential bias in feature extraction caused by the ImageNet classes, the model only uses medium-level layers. We hypothesized that utilizing high-level feature layers could improve feature accuracy. To test this hypothesis, we conducted experiments with other networks trained on the data of the building component images.

3.2.2 Deeplabv3+

Deeplabv3 is a model that is originally built upon the ResNet-101 network and adapted to other models later on. Deeplab makes use of the ResNet blocks and applies atrous convolutions to compute feature responses in fully connected networks [14]. In PyTorch, the model was trained on the ImageNet dataset and additionally, we retrained the DeepLabv3+ network on similar images from our dataset to perform roof and casing segmentations. The objective of retraining these networks was to examine whether training backbones with a similar dataset, but for a different task, improves the performance of the model and enables the utilization of deeper layers for more detailed features and/or faster computations.

3.2.3 ResNet50

ResNet50 is a deep neural network architecture commonly used for image classification tasks. It utilizes the concept of residual learning, enabling the network to learn residual mappings instead of direct mappings. This characteristic allows for training deeper networks without encountering vanishing gradient issues [15]. In our research, we employed the pre-trained ResNet50 model in PyTorch, which was originally trained on the ImageNet dataset. Additionally, we retrained this model specifically for the classification task of distinguishing between nominal and defective images using the same dataset and train-validation split. This ensures that the images used for testing and validation in the anomaly detection model have not been previously seen by the ResNet50 model.

3.3 Layer Selection for Feature Extraction

In the PatchCore anomaly detection model, we explored the use of different layers of backbone networks to extract features in various scenarios. The choice of feature layers plays a crucial role in balancing detail and generalization. When extracting features from deeper layers, we obtain more detailed and specific information about the input data. However, it is important to consider the potential bias towards the dataset on which the backbone networks were originally trained.

Alternatively, utilizing shallower layers of the backbone networks provides more general features that capture lower-level patterns and concepts. However, the computation time and usage of the memory increase due to the larger data size involved.

To address these considerations, the original paper proposed leveraging the medium-level layers (2 and 3) of the WideResnet50 architecture as the baseline approach. This choice was made to strike a balance between capturing sufficient detail in the features extracted by the model and avoiding excessive bias towards the pretrained dataset.

Furthermore, we investigated the impact of retraining the backbone networks, which were already trained on ImageNet class, based on the specific dataset in our research. In our research, we performed retraining on the ResNet50 network for classification purposes, using the same dataset and split to classify nominal and defective building components. Additionally, we retrained a ResNet101 backbone using Deeplabv3 for roof and windows segmentation on similar images from our dataset to perform roof and casing segmentation. By exploring different feature layers and considering the implications of retraining backbone networks, we aimed to optimize the balance between detail and generalization, and ultimately improve the effectiveness of the anomaly detection model in our specific context.

3.4 Dataset

The data used in this study consisted of images captured by a drone from various buildings and locations in the Netherlands, obtained from an aerial scan company. The images were taken from different angles and categorized into different units or buildings. with each unit representing one or a set of buildings imaged in a single session. The aerial images obtained from the aerial scan company were used to generate a 3D model of the buildings. The company then employed this 3D model to annotate each unit separately, with the annotations projected onto the corresponding 2D images.

For anomaly detection, it is crucial to have a large number of good or nominal images that have regular patterns. In our dataset, the Casing and Roof components had the highest number of images, and the images of these components showed a higher degree of similarity. Additionally, the quality of annotation needs to be considered. Therefore, these components were chosen as the focus of our research, as they provided the best opportunity to work on a well-defined and uniform set of images for anomaly detection. As the first step of our investigation, we initially focused on obtaining masks for these components and their corresponding defects. An example of an original image and its corresponding annotated image with building components and defects is shown in Figure 2 and 3. For example in Figure 2, we specifically focused on casings and their associated defects. These defects encompassed various issues such as dirtiness, rot, damage, moss, and more. However, during the preparation process, we encountered challenges with the projection of annotations from the 3D model to the 2D images. As a result, there were instances where the highlighted yellowish mask in the middle of the image appeared unrelated to the casings. This occurred when the annotation represented components or defects that were not visible in the 2D image, as they were located behind or on the other side of the building.

3.4.1 Data Preparation

To prepare the data, it was necessary to create a curated list of the most useful images from a pool of approximately 8000 images, as described later. This approach aimed to minimize the extensive manual cleaning effort during the final data processing stage. The objective was to extract and crop the target components from the images, resulting in a larger number of components compared to the initial number of images, as each image may contain multiple components. We chose images that had well-annotated roofs, casings, or glass, along with all components and defects, captured from an angle as close to perpendicular as possible. This was done to ensure that the images were taken with consistent and straight angles, which facilitates training the model on a uniform dataset.

3.4.2 Casing Dataset

To generate cropped images of the casings, we first used the casing masks to crop the images. Next, based on the overlap of the casing annotation and defect annotation in the region of the cropped image, we recognized whether the cropped images contained defects or not. To ensure that the cropped images contained only the casing, we set the pixel values outside of the casing mask to zero. Similarly, to isolate the glass region of the casing, we set the pixel values inside the glass mask and outside the casing mask to zero. We hypothesize that this can help the model only focus on the relevant features of the casing. The resulting cropped images were saved in their respective categories, with defective cropped images having a new mask generated as ground-truth and all defects categorized as one class. Figure 4 shows an example of a cropped defective image with its overlayed defect and mask file. After the generation of cropped images, we conducted another manual inspection to remove images that did not include the whole casing component with an almost 90-degree angle and with no obstacle present in front of it in the cropped image. For non-defective cropped images, it was crucial that the entire casing component was fully intact with no defects or obstructions present. This process ensured the selection of the most appropriate images for further analysis, resulting in a high-quality

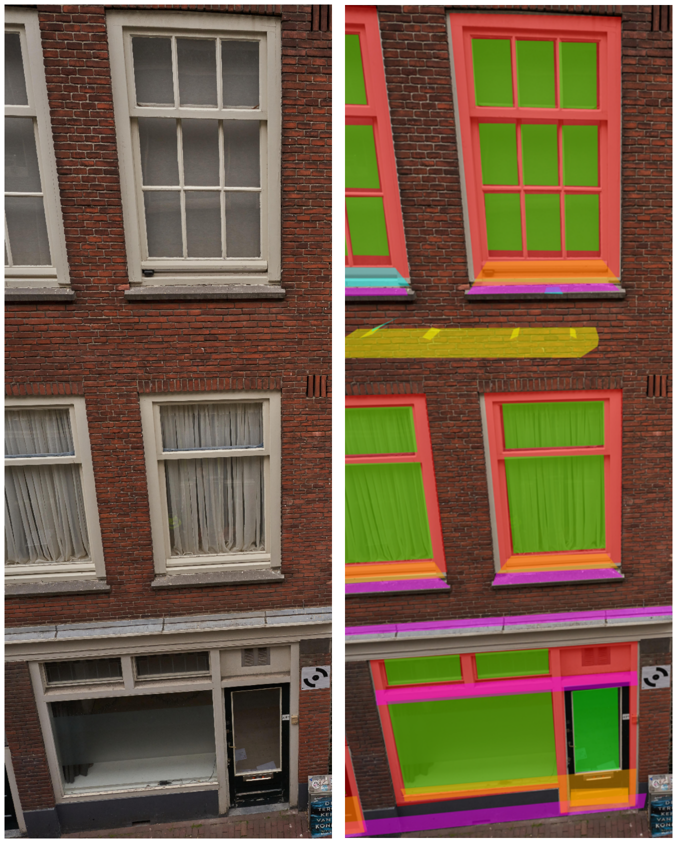


Fig. 2: An image and masks of the casing components and defects on it. Masks for casings, glasses and different defects overlaid onto the original images

Color	Description
🟥	Casing
🟩	Glass
🟨	Growth: moss, algae
🟦	Damage: end-stage intensity
🟪	Dirt, deposits, discoloration
🟧	Peeling: end-stage intensity

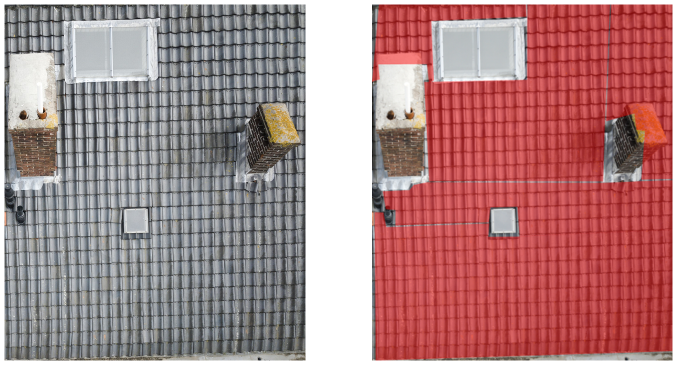


Fig. 3: An image and masks of the roof components.

Color	Description
🟥	Roof

dataset for the subsequent training of the anomaly detection models. After performing manual cleaning, the initial set of 1100 cropped images was reduced to 455 final images with 102 cleaned nominal images. However, we encountered a challenge with the dataset that during testing, we observed a significant class imbalance between the defective and nominal images. To address these issues, we applied data augmentation techniques exclusively to the nominal images, including vertical flipping and 180-degree rotation, effectively tripling their quantity in the dataset. Moreover, augmenting the quantity of nominal images could help the training process of the model.

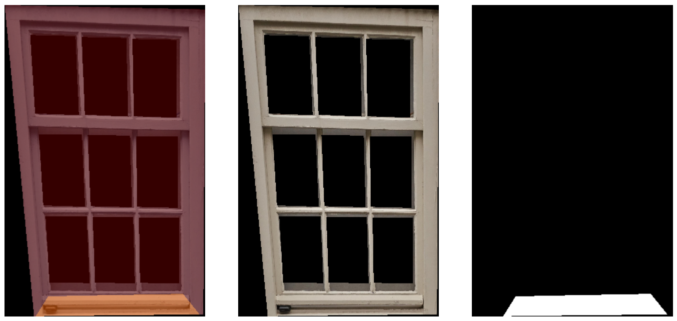

(a) Overlayed casing and defect mask on a sample (b) A final sample as defective (c) The mask of the defect(s)

Fig. 4: Final processed images for casings.

3.4.3 Roof Dataset

To prepare the roof dataset, we first removed other components from the images, leaving only the roof area. We then automatically divided the images into two groups, Nominal and Defective, based on their defect mask. We manually checked and cleaned the data to ensure the accuracy of the classification between nominal and defective data. Additionally, we checked the angle of the images and inspected for any annotation issues such as the presence of other components in the images. After undergoing manual cleaning, the original collection of 955 cropped images was refined to a final set of 440 images for the roof dataset. Figure 5 illustrates an example of the data cleaning process for the roof dataset. Similar to the casing dataset, we encountered a challenge with the limited number of images, having only 71 nominal images available. To address this issue and improve the balance between the nominal and defective classes during testing, we applied vertical flipping and 90-degree rotation to augment the existing nominal images. This augmentation technique increased the number of nominal images and helped create a more balanced dataset.

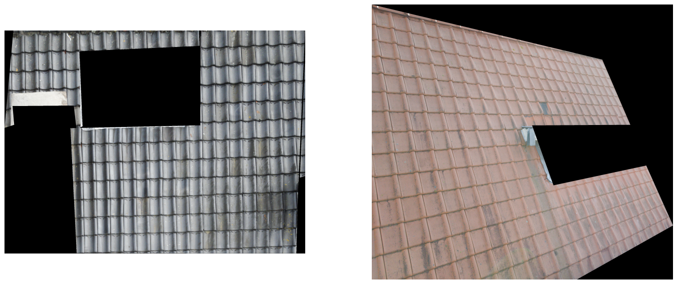
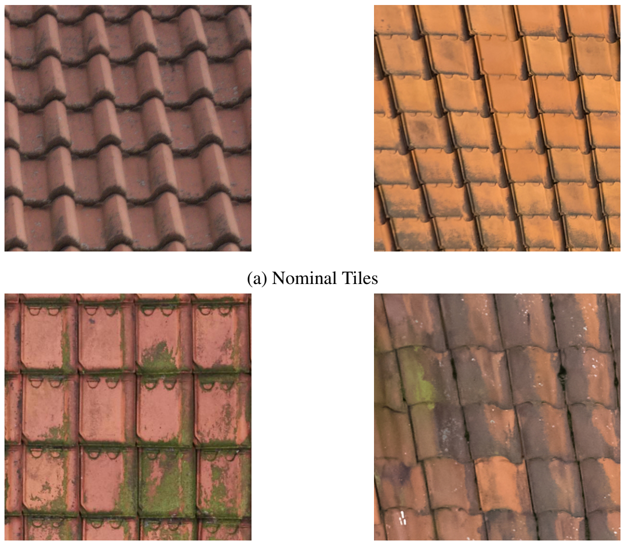


Fig. 5: Examples of data preparations for roofs

3.4.4 Tiled Roof Dataset

The roof dataset comprises a relatively small number of images, and these images exhibit significant variations, particularly in terms of roof shapes and the presence of black parts resulting from removed components. This diversity poses challenges for an anomaly detection model, as it is desirable for the training data to exhibit similarity to effectively represent features of a nominal image. To increase the number of normal or nominal images and to reduce the noise of the roof dataset, we generated a new dataset by dividing each roof image into 512×512 pixel tiles. As roofs typically have a repetitive pattern, we were able to identify and keep only the tiles that contained complete roof images. We then manually inspected 3003 tiles to classify them as either nominal or defective, resulting in a total of 795 nominal and 890 defective tile images. As only some parts of a roof were usually defective, many tiles generated from defective images were classified as nominal or normal images after tiling. However, it is important to acknowledge that during the tiling process, tiles from the same original image could appear in both the training and test sets. This can potentially introduce a bias in the evaluation process. Figure 6 provides examples of both nominal and defective tiles.


(a) Nominal Tiles

(b) Defective Tiles

Fig. 6: Examples of nominal and defective samples of tiled dataset

Table 1 provides an overview of the dataset information for the three datasets.

3.5 Metrics

When the model makes a prediction based on the calculated threshold value, there are four possible outcomes:

- True Positive (TP): The model correctly detects a defective image as defective.
- False Positive (FP): The model incorrectly detects a nominal image as defective.
- True Negative (TN): The model correctly detects a nominal image as nominal.
- False Negative (FN): The model incorrectly detects a defective image as nominal.

Ideally, the model should predict as many true positives and true negatives as possible and as few false positives and false negatives as possible. The confusion matrix representing these outcomes is shown in Table 2. Precision is the ratio of correct true positive predictions (ground truth) to the positive predictions made by the model (true positives and false positives). The precision formula is shown in Equation 1.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{1}$$

Recall is the ratio of correct true positive predictions (ground truth) to all Defective samples (true positives and false negatives). The recall

Dataset	Initial Images	Cleaned Images	Train/Val/Test Sets for Nominal class	Val/Test Sets for Defective class
Casing	1100	455	182 / 60 / 62	70 / 71
Roof	905	440	127 / 43 / 42	74 / 75
Tiled Roof	3003	1685	477 / 158 / 160	163 / 165

Table 1: Dataset Information

	Predicted Defective	Predicted Nominal
Actual Defective	True Positive (TP)	False Negative (FN)
Actual Nominal	False Positive (FP)	True Negative (TN)

Table 2: Confusion matrix for model predictions.

formula is shown in Equation 2.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2}$$

F1-score is the harmonic mean of precision and recall metrics. The F1-score formula is shown in Equation 3, which links the previously defined metrics.

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

The Area Under the Curve (AUC) is a common metric used in evaluating the performance of anomaly detection models. It measures the model's ability to rank anomalies higher than normal samples. A higher AUC value indicates better discrimination between anomalies and normal samples.

The AUC is calculated by integrating the Receiver Operating Characteristic (ROC) curve. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold values. The formula for AUC is shown in Equation 4.

$$\text{AUC} = \int_0^1 \text{TPR}(FPR^{-1}(t))\,dt \quad [16] \tag{4}$$

In addition to these main metrics, it can be useful to calculate the accuracy of each class, which provides insights into the model's performance for individual classes. The accuracy of the "Nominal" class is given by the ratio of true negatives to the sum of true negatives and false positives. Similarly, the accuracy of the "Defective" class is given by the ratio of true positives to the sum of true positives and false negatives.

3.6 Thresholds and Inference

In this research, we used thresholds to determine whether patches and entire images were normal or anomalous. These thresholds were crucial for distinguishing between normal and abnormal patches within an image.

To establish the threshold values, we formulated an optimization problem with the objective of maximizing the F1-Score. This optimization was performed during the validation phase using a dedicated dataset. The range of each patch's anomaly score was defined, with the lower level representing the minimum intensity level and the upper level set to prevent high-intensity noise from being classified as anomalous. By considering these threshold ranges, we could identify and classify normal and abnormal patches in an image.

The optimization problem involved some key constraints. First, the lower and upper threshold values were bounded by the minimum and maximum anomaly scores across all patches of all images in the validation set. This ensured that the threshold values remained within a reasonable range. Then, the threshold value for flagging the whole image was constrained to be between 0.5% and 3%.

The optimization problem can be expressed as follows:

$$\begin{aligned} \text{Maximize:}\quad & \text{F1-Score} \\ \text{Constraints:}\quad & \text{min(scores)} \le \text{Lower threshold} \le \text{max(scores)} \\ & \text{min(scores)} \le \text{Upper threshold} \le \text{max(scores)} \\ & \text{Lower threshold} < \text{Upper threshold} \\ & 0.05\% \le \text{Threshold for flagging the whole image} \le 0.3\% \\ & \text{Accuracy of "Nominal" class} \ge 0.5 \\ & \text{Accuracy of "Defective" class} \ge 0.5 \end{aligned}$$

The last constraints ensure that the threshold values are within a reasonable range and that the accuracy of both the "Nominal" and "Defective" classes is above 0.5, which helps prevent bias in cases where the number of images in the two classes is highly unbalanced.

To solve the optimization problem, we aimed to find the optimal threshold values that would maximize the F1-Score. This allowed us to effectively flag an entire image as either normal or anomalous based on a specific threshold percentage.

Once the thresholds were calculated in the validation phase, they were applied in the test loop to label the images. To generate informative output images, we utilized the anomaly scores of patches as an anomaly map. This map provided a visual representation of the areas within an image that exhibited anomalies. By overlaying the anomaly map as a heatmap on the original image, we created a comprehensive visualization that highlighted the anomalous regions.

To ensure the clarity and focus of the inference, we further refined the heatmap by setting the values of normal patches to zero. This adjustment eliminated the contribution of normal patches from the heatmap, allowing anomalies to be more easily identified and distinguished.

By following this process of threshold determination and utilizing anomaly maps and heatmaps, we could accurately classify and visualize the anomalous parts of the images.

4 Experiments and Results

4.1 Experiments

In this section, we present a series of experiments aimed at training an anomaly detection model using different backbone networks, generated datasets, and layers of feature extraction. Throughout the training process, the upsampling of extracted features for storage in the memory bank exhibits potential variations. This arises from the utilization of a greedy search algorithm and the introduction of variety through shuffling the input data. Consequently, these factors introduce variability in the results obtained from multiple runs. To address this inherent randomness, each experiment was conducted 10 times, and the average and standard deviation of the outcomes were calculated to provide a robust representation of the experimental results. We evaluated the performance of the PatchCore anomaly detection model on different datasets, which are outlined as follows:

4.1.1 Experiment 1:Casing Dataset (focused on identifying defects on windows)

In this experiment, we aimed to evaluate the performance of the PatchCore anomaly detection model on the Casing dataset.

4.1.2 Experiment 2: Roof Dataset (focused on identifying defects on roofs using the whole image)

To assess the effectiveness of the PatchCore anomaly detection model on the Roof dataset, we trained the model with the same three backbone networks.

4.1.3 Experiment 3: Tiled Roof (focused on identifying defects on roofs using images tiles)

In these experiments, we employed three different backbone networks and varied the layers used for feature extraction. Additionally, we explored two approaches: using the default pre-trained weights and re-training the networks on our data, either for different tasks or similar tasks. Due to the availability of a more extensive and curated dataset for the Tiled Roof dataset, we conducted additional experiments to further investigate the performance of different layers in pre-trained backbones. Additionally, we introduced the ResNet18 network into our experiments in this specific dataset.

4.2 Results

In this section of our paper, we present both quantitative and qualitative assessments of our model's performance. The quantitative results are summarized in Tables 3, 4, and 5 for experiments 1, 2, and 3 respectively, providing numerical evaluations of the defined metrics for different experiments.

4.2.1 Quantitative Results

Casing Dataset: Table 3 presents the anomaly detection results for the Casing Dataset. We evaluated the performance of different backbone networks and layers of feature extraction. The WideResNet50 Pretrained network achieved the best F1-score of 60.4% with layers 2 and 3, and the highest AUC of 61.0%. It also had the highest Nominal Class Accuracy of 65.6% and the highest Defective Class Accuracy of 56.3%. networks, with various combinations of layers, achieved F1-scores ranging from 44.5% to 57.7% and AUC scores ranging from 45.4% to 58.0%. The accuracy in the Nominal class and Defective class varied between 31.6% and 55.9% and between 42.6% and 67.3%, respectively. The DeepLab Pretrained network and the DeepLab Retrained network showed lower performance compared to WideResNet50, while the ResNet50 Retrained networks performed slightly better. It indicates that retraining the ResNet50 classification network yielded superior results compared to retraining the DeepLab segmentation network.

Roof Dataset: Table 4 presents the anomaly detection results for the Roof Dataset. In the best scenario, the ResNet50 Retrained network with layers 3 and 4 achieved the highest F1-Score of 55.4% and also obtained the highest AUC of 56.3% and showed promising performance in both the Nominal Class Accuracy (50.8%) and Defective Class Accuracy (61.8%), which mean re-training the backbones for the classification task and utilizing higher-level feature layers, improved the performance. In the second best scenario, the ResNet50 Retrained network with layers 2, 3, and 4 achieved an F1-Score of 55.0% and an AUC of 57.6%. The other configurations showed varying levels of performance.

Tiled Roof Dataset: The ResNet50 Retrained network with layers 2, 3, and 4 achieved the highest F1-Score of 82.2%and also yielded the highest AUC of 82.2% and demonstrated Nominal Class Accuracy (80.9%) and Defective Class Accuracy (83.6%). Once again, the re-training of backbones for the classification task and the utilization of higher-level feature layers proved to enhance the performance. In terms of the next best scenarios, the DeepLab Pretrained network with layers 2 and 3, as well as the ResNet50 Pretrained network with layers 2 and 3, both achieved an F1-Score of 81.4%(0.03) and 81.1%(0.01), respectively. These scenarios also showcased high AUC values of 81.4% and 82.2%, along with competitive Nominal Class Accuracy and Defective Class Accuracy metrics. The ResNet18 network with layers 2 and 3 achieved an F1-Score of 75.9%, with a corresponding AUC of 76.0%. These results suggest that the Retrained ResNet50 network with layers 2, 3, and 4 performs the best on the Tiled Roof dataset. However, the next best scenarios, which include the DeepLab Pretrained network with layers 2 and 3 and the ResNet50 Pretrained network with layers 2 and 3, exhibit comparable performance.

4.2.2 Qualitative Results

In addition to these quantitative measures, we also analyze the qualitative results, which are illustrated through the output images generated by our model. These images showcase different scenarios and offer visual insights into the detection and localization of anomalies in the examined data. Each qualitative result image consists of two parts: the right-side image displays the original image with the heatmap overlay, highlighting regions of potential anomalies, while the left-side image presents the original image with segmented contours of the heatmap drawn to provide a clearer understanding of the detected anomalies. These images presented in Figures 7 to 17 in Appendix A. For certain images, additional necessary explanations are provided in the sub-captions, offering concise descriptions of the observed heatmaps.

Casing Dataset: Although the number of results for the casing dataset was relatively low, the output images still provide valuable information. Figures 7 showcase examples of correct predictions for the defective dataset, where the heatmaps accurately outline the location of defects. However, there are cases like Figures 8 where the predictions are correct, but the heatmaps and segmentation do not provide useful information about the location of the defects. For instance, in Figures 8a and 8c, the defective annotations include irrelevant objects such as glasses and parts of bricks, resulting in incorrect heatmaps and segmentation. Additionally, in Figure 8b, the type of casing is significantly different from the other types in our dataset, making it challenging for the heatmap to indicate relevant areas. Moreover, the heatmap of Figure 8d does not reveal any specific region due to the nature of the casing. Figure 9 shows examples of incorrect prediction of the Defective class. In Figure 9b, the defect appears small, and as a result, the entire image was not detected as defective. Similarly, in Figure 9a and Figure 9c, there seem to be no issues with the casing, which could be due to small defects or errors in the annotation process. Figures 10 and 11 show examples of Correct and Incorrect predictions for the Nominal class of the Casing dataset. These qualitative results provide valuable insights into the performance of the model and highlight the challenges faced in accurately detecting and localizing anomalies in the casing dataset.

Roof Dataset: Despite having similar range values for the metrics, the output images for the roof dataset, as shown in Figure 12, do not provide significant assistance in determining the location of defects and lack specific meaning, unlike the casing dataset. This can be attributed to the extensive diversity present in the roof images within the dataset. As a result, the model's performance for the roof dataset is not satisfactory both quantitatively and qualitatively. The variability in roof types, textures, images and conditions adds complexity to the task, making it difficult for the model to provide informative heatmaps and segmented contours.

Tiled Roof Dataset: We obtained better quantitative results for the tiled roof dataset, and the corresponding output images support this. Figure 13 showcases examples of correct predictions for defective tiled roofs. The model performs well in detecting some sort of dirtiness, as shown in Figure 13a, and it is also effective in identifying fractures and changes in pattern, as demonstrated in Figure 13d. Additionally, Figure 13b illustrates the detection of moss on the roof, while Figure 13c presents an interesting case where a piece of rope on the roof is detected, although it is not considered a defect. Similar to the casing dataset, the heatmaps generated for the defects in tiled roofs do not necessarily correspond closely to the actual location of the defect, as depicted in Figure 14. In some instances, the model mistakenly identifies shadows as anomalies or defects, as shown in Figure 14c.

Figure 15 demonstrates incorrect predictions for defective tiled roofs. In Figure 15b, although a defect exists, it is relatively small, and based on the calculated threshold, the tile is labeled as normal.

Casing Dataset					
Backbone Network	**Layer**	**F1-Score**	**AUC**	**Nominal Class Accuracy**	**Defective Class Accuracy**
WideResNet50 Pretrained	**2,3**	**60.4% (±5.9%)**	**61.0% (±6.1%)**	**65.6% (±12.4%)**	**56.3% (±7.3%)**
DeepLab Pretrained	2,3	44.5% (±2.5%)	45.4% (±2.3%)	31.6% (±6.6%)	59.2% (±6.6%)
DeepLab Retrained (Segmentation)	2,3	47.6% (±5.5%)	48.3% (±5.7%)	54.0% (±12.4%)	42.6% (±6.3%)
	3,4	51.3% (±4.2%)	51.5% (±4.2%)	46.6% (±7.4%)	56.5% (±9.6%)
	2,3,4	53.3% (±3.3%)	54.1% (±3.3%)	46.1% (±14.3%)	62.0% (±10.2%)
ResNet50 Pretrained	2,3	50.1% (±3.6%)	50.4% (±3.4%)	45.6% (±10.7%)	55.3% (±5.9%)
ResNet50 Retrained (Classification)	2,3	52.7% (±5%)	53.0% (±5.2%)	51.6% (±8.5%)	54.5% (±10.3%)
	3,4	56.3% (±2.9%)	56.6% (±2.8%)	46.0% (±6.3%)	67.3% (±3.9%)
	2,3,4	57.7% (±1.4%)	58.0% (±1.4%)	55.9% (±8.8%)	60.1% (±6.3%)

Table 3: Anomaly Detection Results – Casing (Average of 10 times running and STD)

Roof Dataset					
Backbone Network	**Layer**	**F1-Score**	**AUC**	**Nominal Class Accuracy**	**Defective Class Accuracy**
WideResNet50 Pretrained	2,3	42.3% (±1.7%)	42.3% (±1.7%)	32.5% (±4.5%)	52.1% (±5.8%)
DeepLab Pretrained	2,3	42.4% (±2.8%)	42.6% (±2.6%)	27.1% (±7.2%)	58.0% (±4.2%)
DeepLab Retrained (Segmentation)	2,3	50.2% (±7.7%)	50.8% (±8.3%)	39.3% (±16.6%)	62.3% (±7.1%)
	3,4	50.3% (±4.5%)	50.7% (±4.1%)	32.0% (±9.3%)	69.5% (±7.2%)
	2,3,4	45.2% (±2.7%)	47.2% (±1.5%)	22.9% (±11.7%)	71.5% (±12.8%)
ResNet50 Pretrained	2,3	43.5% (±2.1%)	43.8% (±1.9%)	37.6% (±4.9%)	50.0% (±5.6%)
ResNet50 Retrained (Classification)	2,3	42.4% (±1.7%)	42.8% (±1.7%)	38.9% (±3%)	46.7% (±3.9%)
	3,4	**55.4% (±1%)**	**56.3% (±1.8%)**	**50.8% (±11.7%)**	**61.8% (±8.5%)**
	2,3,4	**55.0% (±2.2%)**	**57.6% (±2.4%)**	**64.3% (±9.5%)**	**50.9% (±7.3%)**

Table 4: Anomaly Detection Results – Roof (Average of 10 times running and STD)

Tiled Roof Dataset					
Backbone Network	**Layer**	**F1-Score**	**AUC**	**Nominal Class Accuracy**	**Defective Class Accuracy**
WideResNet50	2,3	77.4% (±14.8%)	79.1% (±9.9%)	71.9% (±24.4%)	86.2% (±6.5%)
	3,4	65.5% (±1.5%)	65.7% (±1.3%)	65.3% (±8.5%)	66.2% (±7.5%)
	2,3,4	69.2% (±1.4%)	69.4% (±1.4%)	64.1% (±6.6%)	74.7% (±7%)
ResNet50 Pretrained	**2,3**	**81.1% (±2.8%)**	**81.2% (±2.7%)**	**82.5% (±2.1%)**	**79.9% (±5.7%)**
	3,4	61.5% (±1.7%)	61.9% (±1.7%)	70.3% (±3%)	53.5% (±2.6%)
	2,3,4	69.6% (±1%)	69.7% (±1.1%)	68.3% (±4.5%)	71.1% (±6%)
ResNet50 Retrained (Classification)	2,3	70.0% (±4.1%)	70.4% (±3.8%)	76.3% (±6.3%)	64.5% (±11.7%)
	3,4	74.3% (±0.8%)	74.4% (±0.8%)	77.5% (±3.8%)	71.3% (±3%)
	2,3,4	**82.2% (±1.1%)**	**82.2% (±1.1%)**	**80.9% (±2.8%)**	**83.6% (±3.6%)**
ResNet18 Pretrained	2,3	75.9% (±1.8%)	76.0% (±1.8%)	79.1% (±4%)	73.0% (±4.9%)
DeepLab Pretrained	**2,3**	**81.4% (±1.2%)**	**81.4% (±1.2%)**	**75.8% (±4.6%)**	**87.0% (±4.3%)**
	3,4	73.0% (±2.2%)	73.1% (±2.2%)	73.6% (±4.2%)	72.5% (±6%)
	2,3,4	**81.1% (±0.8%)**	**81.1% (±0.8%)**	**75.3% (±3.1%)**	**86.9% (±3.4%)**
DeepLab Retrained (Segmentation)	2,3	52.0% (±8%)	54.2% (±4.9%)	54.5% (±20.6%)	53.8% (±17.2%)
	3,4	57.1% (±2.7%)	57.3% (±2.7%)	54.7% (±7%)	59.9% (±7.5%)
	2,3,4	56.0% (±2.5%)	56.2% (±2.5%)	51.0% (±5.2%)	61.4% (±7.2%)

Table 5: Anomaly Detection Results – Tiled Roof (Average of 10 times running and STD)

The model encounters difficulties in detecting certain types of dirtiness, as seen in Figure 15d. When it comes to detecting Nominal tiles, the model performs well, as illustrated in Figure 16. However, there are still some challenges, such as misclassifying tiles with shadows (Figure 17a) or non-common types of roofs in our dataset (Figures 17b and 17c), as well as instances of incorrect annotation (Figure 17d).

5 Conclusion and Discussion

Our study focused on the performance evaluation of the PatchCore anomaly detection technique in identifying defects in roofs and window casings. For the Casing Dataset, the WideResNet50 backbone with layers 2 and 3 demonstrated superior performance in terms of F1-score, AUC, and class accuracy metrics. This configuration proved to be the most effective in detecting anomalies in the casing images.

In the Roof Dataset, our model's performance was unsatisfactory both quantitatively and qualitatively. The extensive diversity in roof types, textures, images, and conditions posed challenges for accurate anomaly detection.

The Tiled Roof Dataset significantly improved the model's performance compared to the Roof Dataset. The ResNet50 Retrained network with layers 2, 3 and 4 achieved the highest F1-Score and AUC.

In general, our results show that using the ResNet50 backbone, which was retrained based on the classification task using the same dataset, and utilizing a higher-level feature layer of this backbone yielded the best configuration in our experiments. This configuration demonstrated superior performance in detecting anomalies in our specific domain. These findings highlight the importance of leveraging alternative backbone networks and retraining them specifically for the classification task, as well as utilizing deeper layers of extracted features. Such approaches can significantly enhance the performance of the model and contribute to the advancement of anomaly detection in real-world datasets with diverse characteristics.

For the Casing Dataset, the WideResNet50 backbone with layers 2 and 3 demonstrated superior performance in terms of F1-score, AUC, and class accuracy metrics. This configuration proved to be the most effective in detecting anomalies in the casing images. In the Roof Dataset, our model's performance was unsatisfactory both quantitatively and qualitatively. The extensive diversity in roof types, textures, images, and conditions posed challenges for accurate anomaly detection. The Tiled Roof Dataset significantly improved the model's performance compared to the Roof Dataset and the ResNet50 Retrained network with layers 2 and 3, and 4, and the DeepLab Pretrained network with layers 2 and 3 achieved the highest F1-Score and AUC. The model exhibited good capabilities in detecting dirtiness, damage, and changes in the pattern on tiled roofs. However, it faced challenges in accurately predicting images with shadows and certain types of dirtiness.

This study utilizes the PatchCore anomaly detection model as a state-of-the-art approach. While the original purpose of this model was to detect defects in industrial products with similar design, shape, size, and color, we have adapted it for the detection of building components and a dataset that is derived from a building inspection company, which is a novel application of this model. Additionally, the baseline research exclusively utilized WideResNet50 networks as the backbone, with a specific emphasis on medium-level layers. This choice was made to mitigate any potential bias towards objects present in the ImageNet dataset, on which the model is pre-trained [13]. In contrast, we have expanded our approach by incorporating various networks and retraining them using our dataset. Furthermore, we have leveraged the extracted features from deeper layers. Employing various backbone architectures and layer configurations, allowing us to assess their impact on the model's performance. This methodology strengthens our findings by demonstrating the importance of selecting appropriate backbone architectures and leveraging specific layers for optimal anomaly detection results.

The diversity within the Roof Dataset posed challenges for accurate anomaly detection, affecting both quantitative and qualitative evaluations. Unlike standard datasets of industrial products, in this study, we had to extract and label building components from images based on their defects. The accuracy of these annotations was crucial, and we aimed to automate and expedite this process. However, due to the lack of segmentation annotations, we had to perform extensive manual cleaning, which was time-consuming and had implications for the overall dataset quality. This limitation could have an impact on our ability to effectively differentiate between good and bad results, further complicating the anomaly detection task.

In addition to the challenges related to accurate annotations, another significant hurdle in this study was the variation in the angle and distance at which the images of building components were captured by drones. Furthermore, there were instances where objects obstructed the view of the components, making it even more difficult to obtain clean and labeled component images for experimentation. These factors introduced additional complexities and further compounded the difficulties in conducting accurate anomaly detection on the Roof Dataset.

Taking into account all the aforementioned challenges that had an impact on the model's performance, significant efforts were made to address them and prepare the dataset using the mentioned ideas and techniques, particularly for the tiled roof dataset. Despite the difficulties, the results obtained from the casing and tiled roof dataset, along with the information provided by the heatmap outputs regarding the location of anomalies, were quite promising. The quantitative and qualitative results for the Tiled Roof dataset exhibited significant improvement compared to the Roof dataset. This indicates the effectiveness of the solutions we implemented on a real-world dataset, including data cleaning and curation, as well as increasing the quantity of nominal images. These measures played a crucial role in surpassing the performance of the main model.

The PatchCore anomaly detection model imposes significant hardware memory requirements due to the process of generating the memory bank. In order to construct the memory bank, all the embedding features from different layers need to be obtained and concatenated. This operation not only incurs high computational costs but may also become infeasible, particularly when using multiple layers of features or features from initial network layers. To address this challenge, we explored an alternative solution by employing a simpler network architecture, such as ResNet18. This choice resulted in significantly reduced feature sizes and computational costs compared to the baseline model. Despite this limitation, ResNet18 still achieved commendable results, comparable to the best-performing configurations. This adaptation enhances the model's practical applicability, as it strikes a balance between computational efficiency and effective anomaly detection.

5.1 Future Works

Considering the challenges associated with creating a more uniform dataset or making it similar to industrial product datasets, a potential avenue for future work involves implementing a model that can detect and analyze each ceramic tile of the roof individually. By examining the tiles separately, it may be possible to achieve more precise defect detection, potentially leading to improved results.

To reduce the computational cost and memory requirements during feature extraction from images, an alternative solution is to incorporate Squeeze-and-Excitation Networks (SENet). By integrating SENet into the model architecture, it is possible to improve the model's performance while also reducing memory usage during training.

Our model, as demonstrated in the results, has the capability to segment the locations of defects within the components. With well-annotated defect locations, we can further enhance the analysis and accuracy of our model. Instead of solely classifying images as nominal or defective, we can delve deeper into understanding the specific nature and extent of defects present. By leveraging the segmentation information, we can gather more detailed insights about the size, shape, and characteristics of the defects. This additional level of analysis could enable a more precise and comprehensive understanding of the anomalies present within the components. Therefore, future work could focus on leveraging the segmentation output to perform detailed defect analysis and provide more nuanced information about the nature of the defects detected.

Acknowledgements

This project is a collaboration between the Computer Vision & Data Science professorship and Aeroscan B.V.

References

[1] Rafaela Bortolini and Núria Forcada. Building inspection system for evaluating the technical performance of existing buildings. *Journal of Performance of Constructed Facilities*, 32(5):04018073, 2018.

[2] Siwei Chang and Ming-Fung Francis Siu. Computer vision-based techniques for quality inspection of concrete building structures. 2022.

[3] Akanksha Toshniwal, Kavi Mahesh, and R Jayashree. Overview of anomaly detection techniques in machine learning. In *2020 fourth international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC)*, pages 808–815. IEEE, 2020.

[4] Rachna Pathak, Anil Saini, Arnav Wadhwa, Himanshu Sharma, and Dhiraj Sangwan. An object detection approach for detecting damages in heritage sites using 3-d point clouds and 2-d visual data. *Journal of Cultural Heritage*, 48:74–82, 2021.

[5] Majid Nasrollahi, Neshat Bolourian, and Amin Hammad. Concrete surface defect detection using deep neural network based on lidar scanning. In *Proceedings of the CSCE Annual Conference, Laval, Greater Montreal, QC, Canada*, pages 12–15, 2019.

[6] Benjamin Staar, Michael Lütjen, and Michael Freitag. Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, 79:484–489, 2019.

[7] Nhung Hong Thi Nguyen, Stuart Perry, Don Bone, Ha Le Thanh, Min Xu, and Thuy Thi Nguyen. Combination of images and point clouds in a generative adversarial network for upsampling crack point clouds. *IEEE Access*, 10:67198–67209, 2022.

[8] Hongbing Shang, Chuang Sun, Jinxin Liu, Xuefeng Chen, and Ruqiang Yan. Defect-aware transformer network for intelligent visual surface defect detection. *Advanced Engineering Informatics*, 55:101882, 2023.

[9] Xiaofeng Lu and Wentao Fan. Transformer-based encoder-decoder model for surface defect detection. In *2022 the 6th International Conference on Innovation in Artificial Intelligence (ICIAI)*, pages 125–130, 2022.

[10] Hironori Takimoto, Junya Seki, Sulfayanti F. Situju, and Akihiro Kanagawa. Anomaly detection using siamese network with attention mechanism for few-shot learning. *Applied Artificial Intelligence*, 36(1):2094885, 2022.

[11] Domen Tabernik, Samo Šela, Jure Skvarč, and Danijel Skočaj. Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing*, 31(3):759–776, 2020.

[12] Jingdao Chen and Yong Kwon Cho. Crackembed: Point feature embedding for crack segmentation from disaster site point clouds with anomaly detection. *Advanced Engineering Informatics*, 52:101550, 2022.

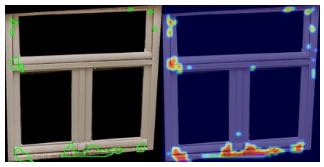
[13] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2022.

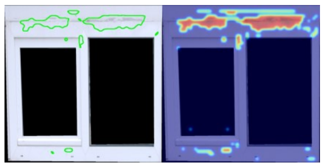
[14] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

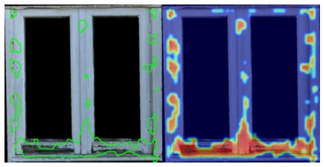
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Vít Škvára, Tomáš Pevnỳ, and Václav Šmídl. Is auc the best measure for practical comparison of anomaly detectors? *arXiv preprint arXiv:2305.04754*, 2023.

A Appendix A - Examples of output images for different datasets


(a)


(b)


(c)

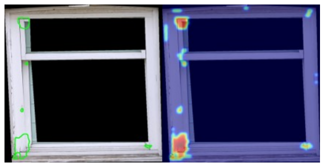
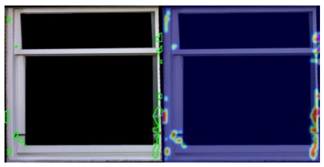
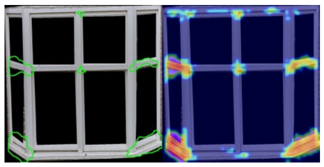
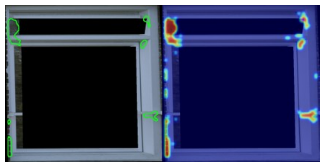

(d)

Fig. 7: Correct Prediction for Defective Casing - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a)


(b)


(c)

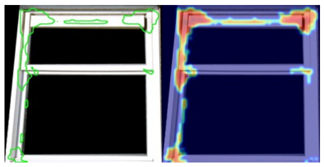
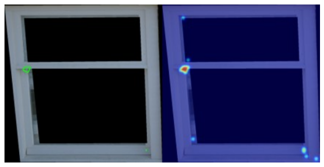
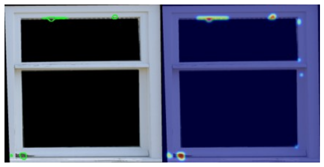
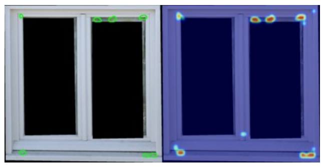

(d)

Fig. 8: Correct Prediction for Defective Casing with Irrelevant Heatmap - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a)


(b) Small defect


(c)

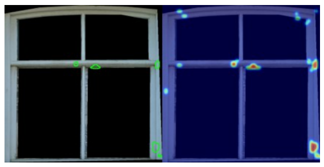
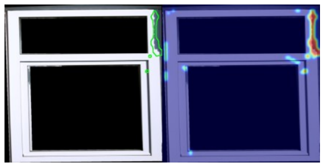
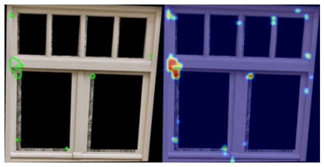
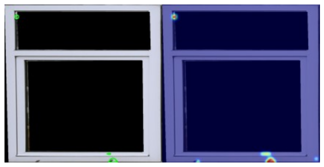

(d)

Fig. 9: Incorrect Prediction for Defective Casing - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a)


(b)


(c)

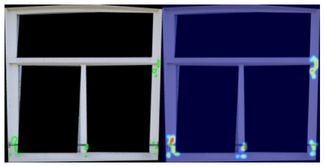
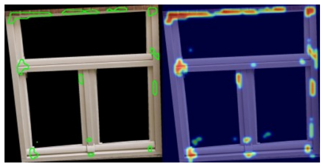
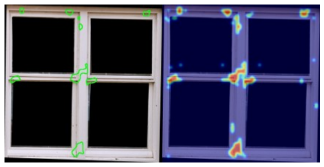
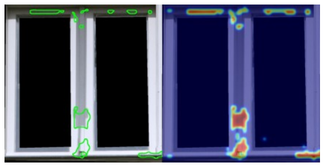

(d)

Fig. 10: Correct Prediction for Nominal Casing - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of defects - The heatmap and segmentation outputs highlight small potential defective areas and the images were not recognized as defective.


(a) Inaccurate annotation


(b)


(c) Shadow

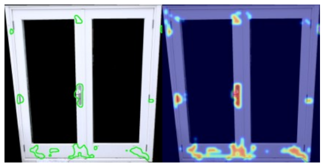
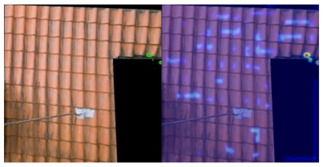
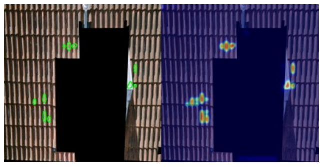
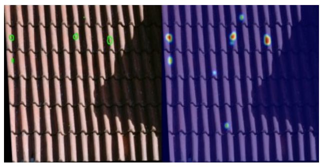

(d) Hinges

Fig. 11: Incorrect Prediction for Nominal Casing - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a) Defective - Defective


(b) Defective - Negative


(c) Negative - Negative

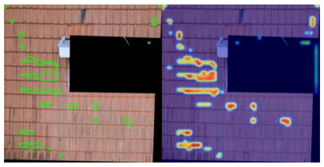
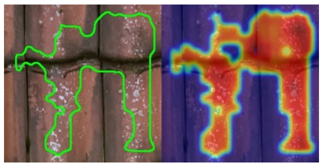
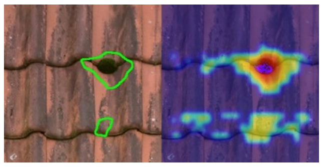
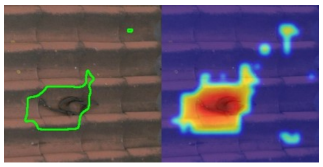

(d) Negative - Defective

Fig. 12: Predictions for the whole roof dataset - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a) Dirt


(b) Moss


(c) An object on the roof

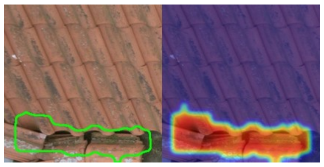
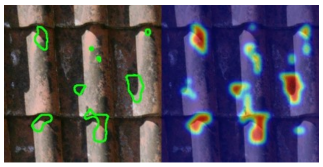
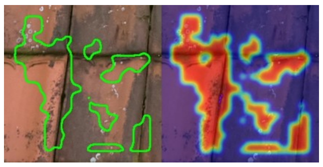
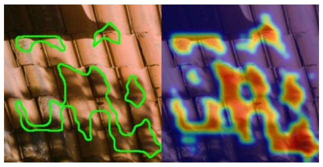

(d) Damage

Fig. 13: Correct Prediction for Defective Tiled Roof - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a)


(b)


(c) Shadow

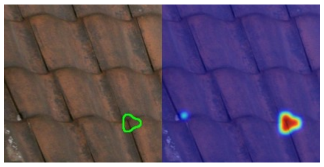
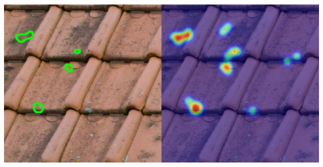
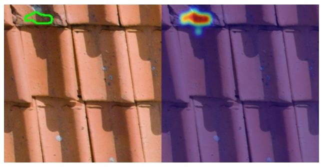
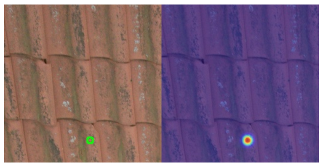

(d)

Fig. 14: Correct Prediction for Defective Tiled Roof with Irrelevant Heatmap - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a)


(b)


(c)

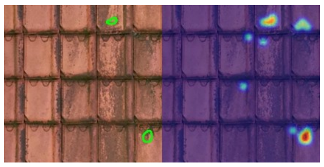
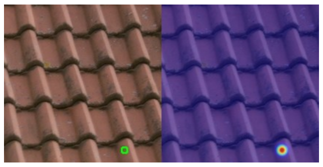
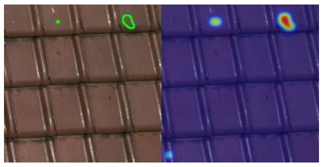
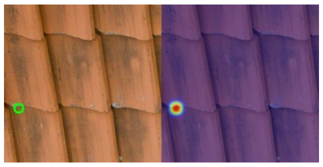

(d)

Fig. 15: Incorrect Prediction for Defective Tiled Roof - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of defects - The model could not detect defects in a, c and d. In b the defect is too small.


(a)


(b)


(c)

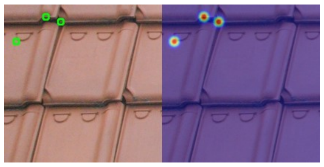
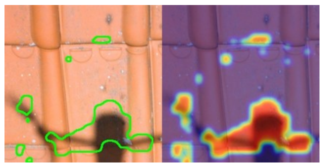
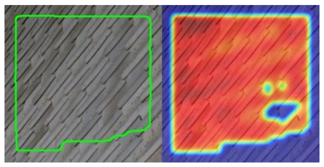
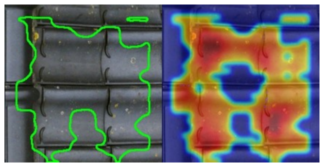

(d)

Fig. 16: Correct Prediction for Nominal Tiled Roof - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.


(a) Shadow


(b) Non-common types of roof ceramics


(c) Non-common types of roof ceramics

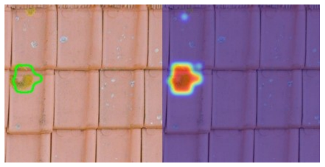

(d) Wrong annotation

Fig. 17: incorrect Prediction for Nominal Tiled Roof - Right image depicts heatmap of potential defects on the resized original image, while the left image shows the contour overlay of the heatmap for precise localization of potential defects.