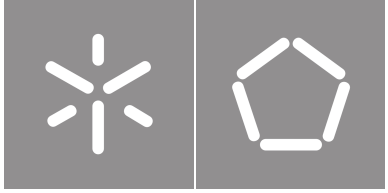
Universidade do Minho
Escola de Engenharia

João Pedro Matos Ribeiro Soares

Automatic defect detection in leather

October, 2022



Universidade do Minho
Escola de Engenharia

João Pedro Matos Ribeiro Soares

Automatic defect detection in leather

Master Thesis
Master in Informatics Engineering

Work developed under the supervision of:
Professor Doctor Luís Gonzaga Mendes Magalhães

October, 2022

Acknowledgements

The present master's dissertation is the end of my academic journey. During this school year, I spent many hours on this dissertation. However, it was just possible with the support of the most related, whom I want to thank.

In first place, I want to thank my parents, Leandro and Paula, for giving me the possibility to get here. Without the values they transmit to me and the ambition they instill in me, this dissertation was not possible. Also, I want to thank my brother, Eduardo, for sharing soo many fun moments with me.

In second place, I want to thank professor Luís Magalhães for guiding me during this year, accomplishing this project, and introducing me to a new perspective on the problem.

Also, I can not forget my friends, Bruno, Carlos, Catalão, Diogo, and Paulo, for following me during the entire academic journey, and giving me good moments that I will not forget. Especially, I want to thank Diogo for the uncountable hours discussing the work of both, sharing knowledge, raising doubts, and confirming certainties.

In last, I can not forget my girlfriend, Diana. Thanks for the support during the entire process and for believing in me.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Resumo

Deteção automática de defeitos em couro

Esta dissertação desenvolve-se em torno do problema da deteção de defeitos em couro. A deteção de defeitos em couro é um problema tradicionalmente resolvido manualmente, usando avaliadores experientes na inspeção do couro. No entanto, como esta tarefa é lenta e suscetível ao erro humano, ao longo dos últimos 20 anos tem-se procurado soluções que automatizem a tarefa. Assim, surgiram várias soluções capazes de resolver o problema eficazmente utilizando técnicas de Machine Learning e Visão por Computador. No entanto, todas elas requerem um conjunto de dados de grande dimensão anotado e balanceado entre as várias categorias. Assim, esta dissertação pretende automatizar o processo tradicional, usando técnicas de Machine Learning, mas sem recorrer a datasets anotados de grandes dimensões. Para tal, são exploradas técnicas de Novelty Detection, as quais permitem resolver a tarefa de inspeção de defeitos utilizando um conjunto de dados não supervsionado, pequeno e não balanceado. Nesta dissertação foram analisadas e testadas as seguintes técnicas de novelty detection: MSE Autoencoder, SSIM Autoencoder, CFLOW, STFPM, Reverse, and DRAEM. Estas técnicas foram treinadas e testadas com dois conjuntos de dados diferentes: MVTEC e Neadvance. As técnicas analisadas detectam e localizam a maioria dos defeitos das imagens do MVTEC. Contudo, têm dificuldades em detetar os defeitos das imagens do dataset da Neadvance. Com base nos resultados obtidos, é proposta a melhor metodologia a usar para três diferentes cenários. No caso do poder computacional ser baixo, SSIM Autoencoder deve ser a técnica usada. No caso onde há poder computational suficiente e os exemplos a analisar são de uma só cor, DRAEM deve ser a técnica escolhida. Em qualquer outro caso, o STFPM deve ser a opção escolhida.

Palavras-chave: Machine Learning, Visão por Computador, Couro, Deteção, Defeitos

Abstract

Automatic defect detection in leather

This dissertation develops around the leather defects detection problem. The leather defects detection problem is traditionally manually solved, using experient assorters in the leather inspection. However, as this task is slow and prone to human error, over the last 20 years the searching for solutions that automatize this task has continued. In this way, several solutions capable to solve the problem efficiently emerged using Machine Learning and Computer Vision techniques. Nonetheless, they all require a high-dimension dataset labeled and balanced between all categories. Thus, this dissertation pretends to automatize the traditional process, using the Machine Learning techniques without requiring a large dimensions labelled dataset. To this end, there will be explored Novelty Detection techniques, that intend to solve the leather inspection task using an unsupervised small and non-balanced dataset. This dissertation analyzed and tested the following Novelty Detection techniques: MSE Autoencoder, SSIM Autoencoder, CFLOW, STFPM, Reverse, and DRAEM. These techniques are trained and tested in two distinct datasets: MVTEC and Neadvance. The analyzed techniques detect and localize most MVTEC defects. However, they have difficulties in defect detection on Neadvance samples. Based on the obtained results, it is proposed the best methodology to use for three distinct scenarios. In the case where the computational power available is low, SSIM Autoencoder should be the technique to use. In the case where there is enough computational power and the samples to inspect have the same color, DRAEM should be the chosen technique. In any other case, the STFPM should be the chosen option.

Keywords: Machine Learning, Computer Vision, Leather, Detection, Defects

Contents

List of Figures

List of Tables

Acronyms

AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
CAE	Convolution Autoencoder
CNN	Convolution Neural Network
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
FPS	Frames per Second
GAN	Generative Adversarial Networks
GLCM	Gray Level Co-Occurrence Matrix
HSV	Hue Saturation Value
KNN	K Nearest Neighbors
LBP	Local Binary Patterns
ML	Machine Learning
MSE	Mean Squared Error
OCBE	One-Class Bottleneck Embedding
PCA	Principal Component Analysis
RGB	Red Green Blue
SSIM	Structural Similarity Index Measure
STFPM	Student-Teacher Feature Pyramid Matching
SVM	Support Vector Machine

Chapter 1

Introduction

This chapter begins by providing a brief explanation of the background and motivation of the present dissertation. After that, it also describes the objectives that this dissertation claims to attain. Lastly, the whole document's structure is presented, expressing how the content of this dissertation is divided into chapters.

1.1 Context and motivation

Leather is a natural product derived from animal skins such as cows, sheep, lizards, and goats. Since long ago, leather has had many applications. In the past, it was used to protect the human body from cold and wind, but these days, with industrial and technological evolution, leather is used in several industries, such as the fashion industry, to produce clothes, shoes, bags, and wallets. Additionally, it is also used in the automobile and furniture industries to create leather-based products.

In many developing countries economies, cattle raising is very relevant, being the meat industry the principal economic growth factor. In addition, as the value of the hides can represent 3% to 10% of the animal's market value [1], bovine producers are motivated to sell them to maximize profit. Due to this monetary significance, the product's quality is crucial to ensure producers' financial thriving. However, the following question arises "How is the leather quality defined?". One could state that less anomalous areas characterize higher quality leather, thus yielding higher profits. And this is what leather companies

desire.

Traditional leather inspection uses experienced assorters in the inspection process that will help define the price of leather. However, this manual inspection is a tedious procedure, being a highly challenging and time-consuming task. Because defects are not easy to find, usually, it is necessary to use several points of view and distances to find a single anomaly. Workers use their vision in this inspection task. In this way, the process is slow and prone to error because humans tend to miss some defects.

Figure 1: Manual leather inspection. Source: [2].

The increased demands for objectivity, confidence, and efficiency have required the development of automatic inspection systems in the traditional leather industry. Such systems are crucial to adopt an automated system to ensure the quality assessment process, improving time-cost and defect detection accuracy.

1.2 Main goals

The main goal of this work is leather inspection automatization. Hereupon, the proposed solution should solve the problem using Unsupervised Learning techniques and achieve state-of-the-art performance.

In the literature, there are solutions for this problem with good results. However, a high dimension labeled dataset is required to train the ML predictors. However, in this work a labeled dataset is not guaranteed. As a consequence, one of the goals of this work is to experiment efficient techniques that could be trained without a labeled dataset. Ideally, these unsupervised techniques, known as novelty detection, should reach high performance using the minimum computational power possible.

1.3 Dissertation structure

This document is divided into six chapters. This current chapter provides an overview of the problem, describing its context and the motivation. The Chapter 2 presents a review about Machine Learning and Computer Vision and details automatic leather defects detection existing solutions. In the Chapter 3, the causes of this problem are detailed, and the challenges to solving this problem are identified. At the end of this chapter a solution is proposed. The Chapter 4 details five distinct techniques based on the solution presented in Chapter 3. After that, the Chapter 5 describes a set of experiments to evaluate the performance of the Novelty Detection techniques on leather defects. The Chapter 6 presents and discusses each technique's performance results. The Chapter 7 summarises this dissertation's research work and provides future work directions.

Chapter 2

Machine Learning and Computer Vision

This chapter presents an overview of ML, DL, and Computer Vision (CV) methods. In closing, works that employ such techniques for the leather inspection problem are described.

2.1 Machine Learning

In 1959, Arthur Samuel introduced ML with the following claim: "A computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program" [3]. In a more general sense, a good definition applied to a chess game, but a global ML definition can be: a sub-field of Artificial Intelligence (AI) that aims to develop computational models to simulate human intelligence. ML models train to capture relationships between independent and dependent features to produce new predictions. The model's knowledge starts from scratch and learns how to solve a problem with the experience like humans.

ML is not a new sub-field. However, in the last two decades it has gained attraction because most of the problems that can be solved using ML techniques require high computational power. This factor was not warranted at the beginning of the computational era. Nowadays, with the exponential growth in computing power ML is used [4]. ML started to be used to solve a lot of problems and applied in every context of our society. Some examples of ML applications are present in the economic system to detect fraudulent transitions and simulate client loans. More examples are health applications to diagnose diseases [5] [6],

e-commerce to recommend products based on client preferences [7], and autonomous car driving [8].

There are a lot of ML based architectures, which can be divided into three main groups: Supervised Learning, Unsupervised Learning, and Semi-Supervised Learning.

Supervised Learning

Supervised Learning builds prediction models using labeled datasets composed of features and respective labels [9]. Supervised Learning algorithms are divide into two stages, the training stage, and the predicting stage. The training stage uses attributes and labels to create a model that learns from the training data, iteratively adjusting the model's parameters until the loss function has been minimised. After the training phase, the model is ready for the predicting phase. In this phase, new data elements, only composed of attributes, are used to test the model and/or realize new predictions.

Unsupervised Learning

Unsupervised Learning creates prediction models using training datasets, just composed of attributes. The model tries to understand the relations between data elements and how attributes depend on each other, creating a data structure. In Unsupervised algorithms, evaluation is not easy to perform because there is no labeled data. So usually, Unsupervised Learning estimates the training data elements' similarity. The most common way to estimate the similarity is using distance functions like Euclidean, Minkowski, or Cosine.

Semi-Supervised Learning

Semi-Supervised Learning is a learning approach that uses labeled data and unlabeled data to create a ML predictor [10]. Since the annotation task is time-consuming, usually, a small set of labeled data is used together with a large pool of unlabeled data to train a ML algorithm. This approach requires less human annotation effort and achieves the same performance. The two most known approaches are Active Learning and Reinforcement Learning.

2.1.1 Architectures

ML architectures learn dataset patterns from the training dataset to find the optimized parameters and make new predictions. There are a lot of ML models of different types. So, this section shows an overview of the most known models developed.

2.1.1.1 Supervised

Naive Bayes is a probabilistic model used for classification problems based on Bayes Theorem. This algorithm assumes that features are independent and not related between them. Naive Bayes uses the training dataset to estimate the conditional probabilities, representing the frequency of each feature value for a given class value divided by the frequency of instances with that class value. This ML model, it is easy to implement, and has a low computational time cost. Another advantage is the inherent interpretability in the model, which explains the predictions based on probabilistic rules.

Another popular model is the decision tree, which expresses a recursive partition of the feature space to sub-spaces that constitute a basis for prediction. The decision tree is composed of nodes, branches, and leaves. In the inference, the data passes through a sequence of nodes until reaches the leaf. In each node, it performs a conditional test on an attribute element. The result of the conditional test will select the following branch. There are some popular algorithms to create decision trees like ID3 [11] and C4.5 [12] that build the decision tree based on the information gain ratio.

In 1992 Boser et al. [13] proposed a algorithm for classification, known as Support Vector Machine (SVM). SVM is a discriminative classifier used to solve a binary classification problem. It uses labelled data to create a hyper-plane to separate data elements from different classes, as presented in Figure 2. The main goal of SVM is to optimize the classification function to split the classes from the training data. There are a lot of solutions to discriminate the classes. However, the optimized function should be at the same distance from both classes. So, it is necessary to find a function that increases the margin distance between both labels. This optimization process is good because it fits in the training data and in the unknown data. Many applications use SVM to solve their problems. One of them is image classification because it makes it possible to use non-linear functions to create discriminators to solve non-linear problems.

Another class of supervised models is the one based on Artificial Neural Network (ANN), used to solve

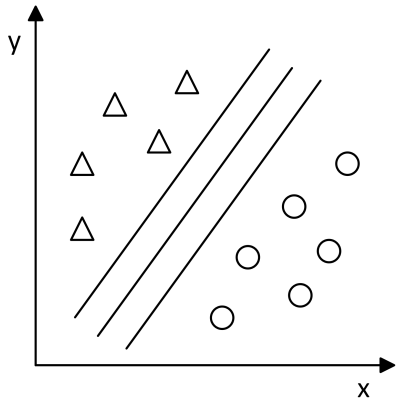


Figure 2: Support Vector Machine.

non-linear classification and regression problems. It has this name because its structure is inspired by the human neural system. The ANN is composed of interconnected artificial neurons like the human system. The ANN structure is simple, an ANN is composed of layers, and each layer is composed of neurons. There are three types of layers: input, hidden, and output, as presented in the Figure 3. The input layer receives the input data, the output layer presents the result of ANN prediction, and between these two layers, there are the hidden layers. The hidden layers estimate a weighted sum from the input and transform this weighted sum using a non-linear function. The result flows to the next layer neurons in one direction, from the input layer to the hidden layers and then to the output layer [14]. When the weighted sums reach the final layers, they are converted to predictions. The Neural Networks optimize the neuron's weights using the gradient to reduce the loss and obtain an optimized predictor during the training phase. In this way, it is possible to use ANN to make predictions.

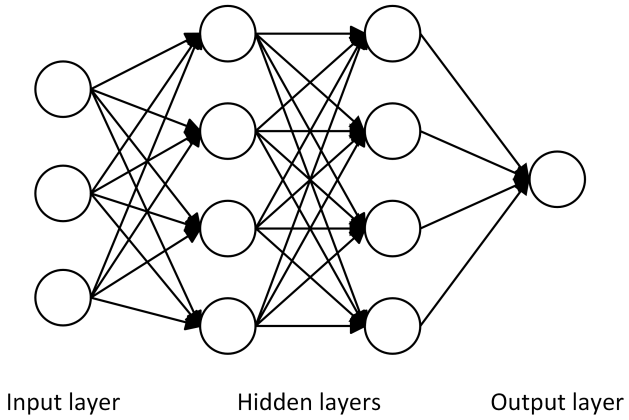


Figure 3: Neural Network.

2.1.1.2 Unsupervised

In Unsupervised Learning, the prediction models are used to understand how the data are connected. The algorithms train using just the unlabelled data elements. So, the prediction models only can be built with the data features. The unsupervised architectures can be divided into two classes: reduction-based and clustering-based.

The dimensional reduction approach techniques reduce redundant features, noisy and irrelevant data to obtain a clean dataset. The dimension reduction techniques can work as a preprocessing technique that will improve the feature accuracy and reduce the training time [15]. The most known unsupervised architecture regarding dimensional reduction is Principal Component Analysis (PCA). The main idea of PCA starts with the data features normalization followed by the covariance matrix computation. The covariance matrix tries to understand how the input features are different from the mean, estimating the eigenvectors and the eigenvalues used to identify the principal components. The PCA obtained good results in a faces recognition problem, producing eigenvectors to reduce the dimension of big dimension images [16]. In this way, using a distance measure, it is possible to associate a new eigenvector to the closest labeled eigenvector that should represent the same person.

Another unsupervised learning approach is clustering. Clustering is a division of data into groups of similar elements. Each group, called a cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups [17]. The K-Means algorithm, described in the Figure 4, assigns each data element to the most similar centroid [18]. Usually, the similarity is calculated by distance measures, in this case, the data element assigns to the closest cluster centroid. K-Means splits into two phases: the data elements assigned to the closest centroid, and the re-estimation of centroids. These two steps work in a loop until the stop criterion is reached. The main goal is to reduce the distance of centroids to the data elements associated with their cluster.

2.1.1.3 Semi-Supervised

Active Learning does not use any particular Semi-Supervised learning architecture. Active Learning is an algorithm that trains Supervised Learning architectures using labeled data and unlabeled data. The labeled data is used during the model training initial phase. After that, Active Learning starts a loop and an Oracle queries the unlabeled pool dataset, selecting a set of data elements that should be labeled to train the model, in such a way that the ML model learns faster. In this way, it is possible to reduce the

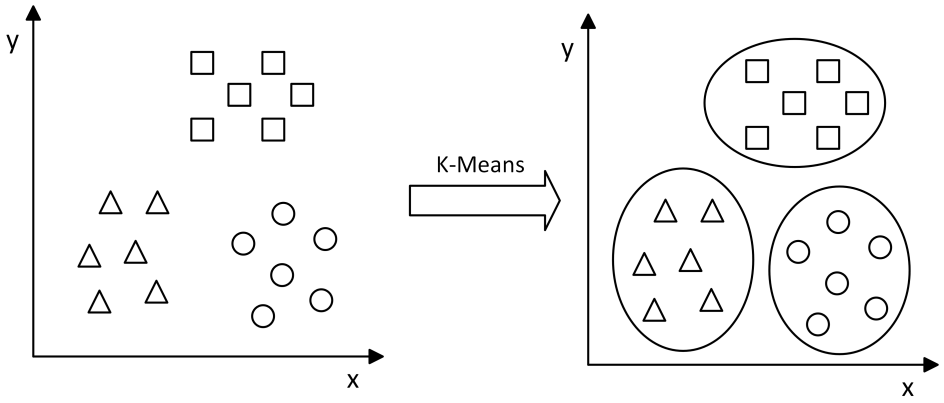


Figure 4: Clustering data - K-Means.

time cost of the labeling procedure.

An example of a Semi-Supervised architecture is the Semi-Supervised Generative Adversarial Network (SSGAN) [19]. This architecture is composed of a segmentation network and a fully convolutional discriminator network. The segmentation network uses labeled and unlabeled data to learn to produce probability maps similar to the ground truth mask. The probability maps produced are used as input of the fully convolutional discriminator network to generate the confidential density maps of unlabeled images. This work achieves results close to the state-of-the-art results only using 25% of the labeled dataset, proving that Semi-Supervised learning can reduce the number of labeled data elements required.

2.1.2 Workflow

The ML workflow can be split in two phases: training phase and predicting phase. The training phase looks for the optimal parameters for the ML model, following a specific sequence as described in the Figure 5. Firstly, the raw data works as input in a pre-processing step, where the data suffers transformations. Such as normalization, discretization, attribute selection, and attribute reduction. It intends to convert raw data into more quality data.

In the next step, the feature extraction step uses the pre-processed data. The feature extraction is responsible for combining features from the pre-processing step. It is useful when it is crucial to reduce the number of attributes without losing representative information. It will be helpful in the training stage because with a reduced number of features will be easier to find an optimized ML model and reduce the workload.

The next step is the data split. The most basic approach to split data is to use 70% of the data in the training step, 15% for validation, and 15% for testing. Although, there are other approaches like Cross-Validation. Cross-Validation is a re-sampling strategy that uses different sets to train and test on distinct iterations, splitting data in folds, usually 10. The number of iterations is equal to the number of folds. In each iteration, the testing phase uses one folder. And the remaining are used to train the model. At each iteration, the folder used in the testing step changes. The goal is to estimate the performance of an ML model by calculating the mean performance of all iterations. This stage is crucial to evaluating the ML model. It gives more confident results because it uses the entire dataset.

The following step is the training step. The training step optimizes an ML model and finds the optimized parameters for the ML model to solve a specific problem. At this point, details of training will not be presented, because there are a lot of ML models that work differently from each one. Although, they all work the same way: estimating the loss function and optimizing the ML model parameters. The last step is the evaluation. In this phase, the model uses the test data to evaluate the performance and check if the model is ready to be predicting phase.

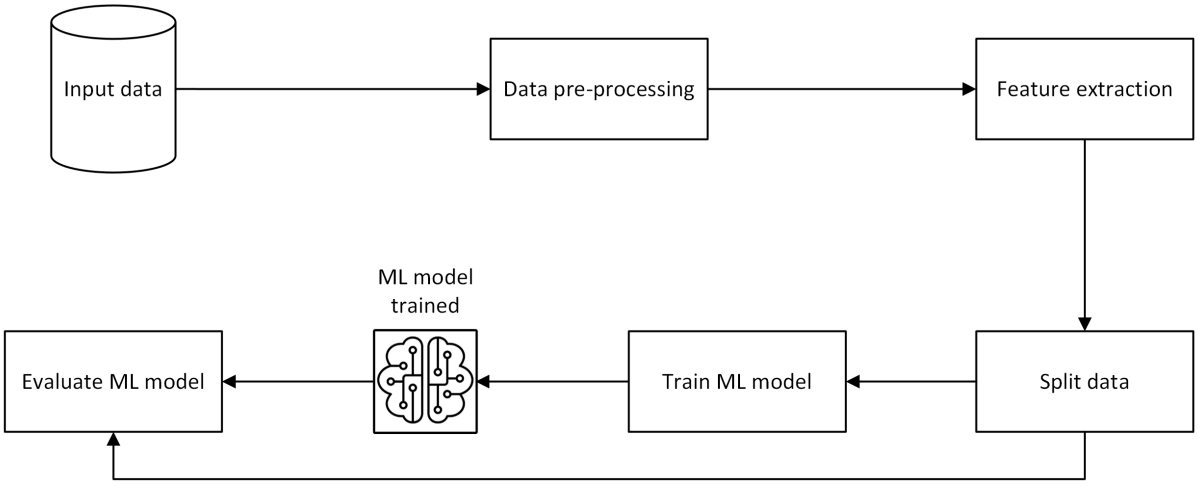


Figure 5: ML workflow - training phase.

During the predicting phase, the ML model is ready to infer new data elements. Nevertheless, before the new predictions, it is necessary to follow the same data-processing and feature extraction techniques as used during training phase. The predicting phase workflow is illustrated in the Figure 6.

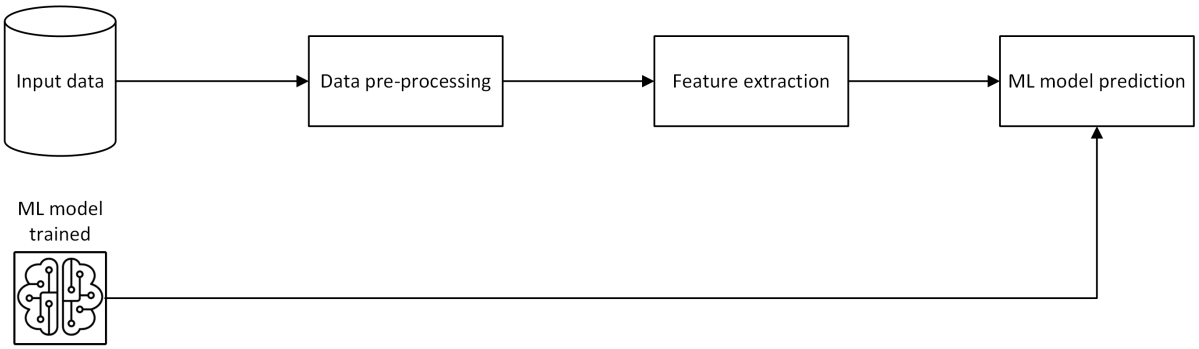


Figure 6: ML workflow - predicting phase.

2.2 Deep Learning

DL is a sub-field of ML that, like ML, intends to give computational devices the human intelligence capacity. DL allows computational models composed of multiple processing layers to learn representations of data with a high abstraction level [20].

One advantage of DL is that overcomes ML techniques by the capacity to process raw data to solve problems like image recognition, speech recognition, self-driving, and so many others, better than the previous architecture from ML. The first difference is the number of layers used in ANN, DL models can have thousands of ANN layers. The second difference is that the DL architectures perform the feature extraction and the model prediction inside the model. Instead, ML architectures perform the feature extraction, and just after this step, the model makes predictions, as described in the Figure 7.

The Deep Neural Networks (DNN) have a high number of layers, and thus, DNN has a high number of neurons to optimize. This complex optimization brings a new problem. DL requires high computing power to train a model. In the last decades, high computational power became available in hardware solutions like GPUs. Another significant characteristic of DL is the big data requirement. As DL is a complex model with a lot of parameters to optimize a high dimension labeled dataset must represent the problem in the real world.

At this moment, an overview about DL was presented and can be conclude that to build a DL predictor the requirements are:

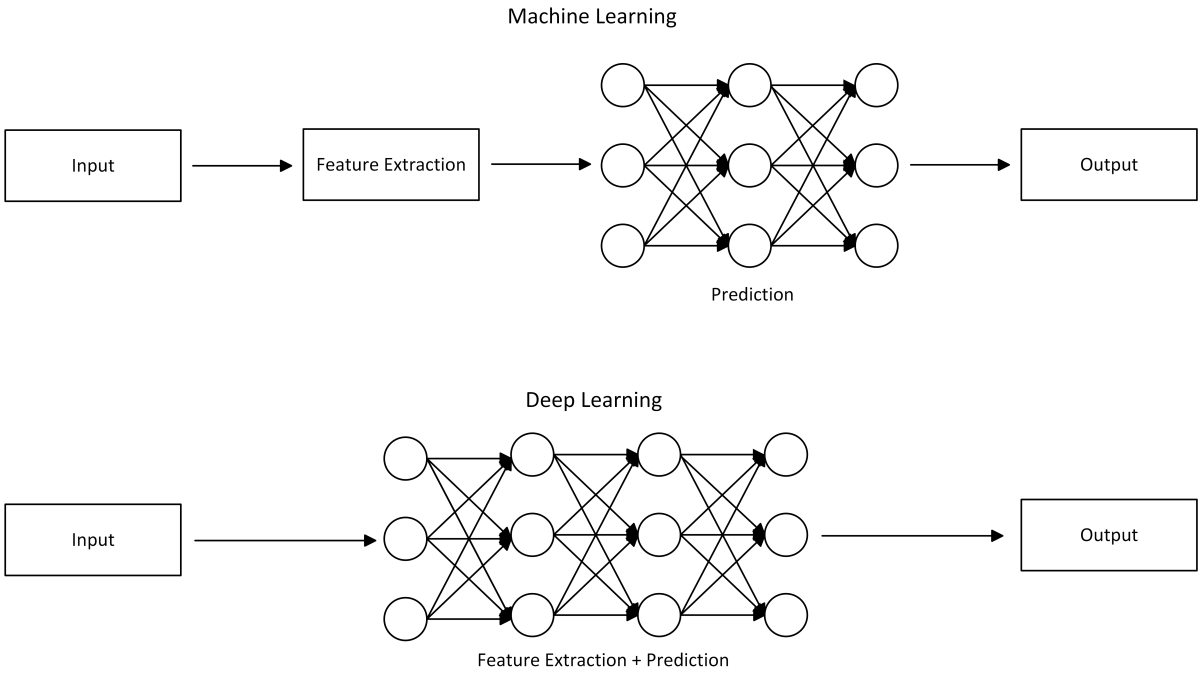


Figure 7: Machine Learning and Deep Learning.

- Hardware with high computational power;
- Large dataset;
- ML or DL Architecture.

2.2.1 Architectures

In the last decade, a lot of DL architectures have emerged. As there are a lot of DL architectures, this section only covers two of the most recognized architectures. One solves image classification problems and the other image segmentation problems.

The image field is one of the most evolved with DL, being the Convolution Neural Network (CNN), presented in the Figure 8, the most popular algorithm in image classification. CNN was firstly introduced by Kunihiko Fukushima [21]. Later, Yann LeCun [22] [23] proposes to solve handwritten digits and document recognition problems using CNN. CNN brings to image predictors the ability to emulate the behavior of arbitrary and complex non-linear functions [14].

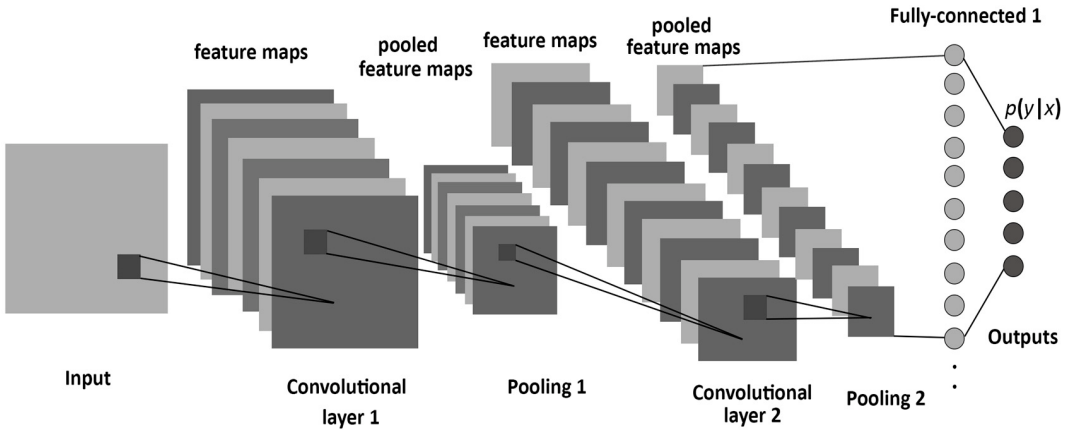


Figure 8: CNN architecture. Source: [24].

The CNN architecture has a sequence of convolution and pooling layers. As mentioned before, CNN is responsible for image feature extraction. So, convolution layers play this role. In the first layers, the features extracted identify small patterns that will be combined in the following convolution layers to build more complex patterns. After each convolution layer, there is a pooling layer. The pooling layer is responsible for a down-sampling operation, applying the operation to optimize the representation size and reduce the number of parameters. After a sequence of convolution and pooling layers, a fully-connected layer combines the features extracted to classify the input image.

The CNN was an overcome in the image classification problem. However, when the problem is image segmentation, other architectures appear. The U-Net [25] is one of the segmentation techniques that emerged in the last years, presenting great results in the biomedical field. It works like the convolutional network architecture, but it is modified to handle small training datasets and keep a high-performance segmentation. The U-Net is encoder-decoder structured, and it is composed of two parts, the contracting and the expansive part. The contracting part uses a CNN architecture, where two 3x3 convolutional operations are applied to an image, followed by a ReLU activation function and a max-pooling operation for downsampling. In the expansive part, the features upsample before a 2x2 convolution, followed by a ReLU function. In the end, a 1x1 convolution maps the features vector to a segmented image.

2.3 Computer Vision

CV is a Computer Graphics field that tries to simulate human visual ability in computers. CV tries to perform visual tasks like objects detection, classification, and localization using digital content like images

and videos. In last years, CV started to combine images processing and feature extraction techniques with ML and DL to perform these tasks.

Pre-processing

In the data pre-processing step, image thresholding, morphological and smoothing operations are usually applied. The morphological operations in image pre-processing are transformations applied to grey-scale and binary images. In a morphological operation, each pixel adjusts its value based on the value of its neighbours, who are obtained by a kernel. There are several operators, but, the most known are erosion, dilation, opening, and closing. The erosion is responsible to reduce the object boundaries and the dilation does the opposite. The opening operator applies the erosion followed by the dilation, the erosion removes the white noise and reduces the object, so it is necessary to apply the dilation to return the object to the original size. The closing operation applies the dilation followed by the erosion and it closes the small holes. In the Figure 9, there are presented the morphological operations mentioned before.

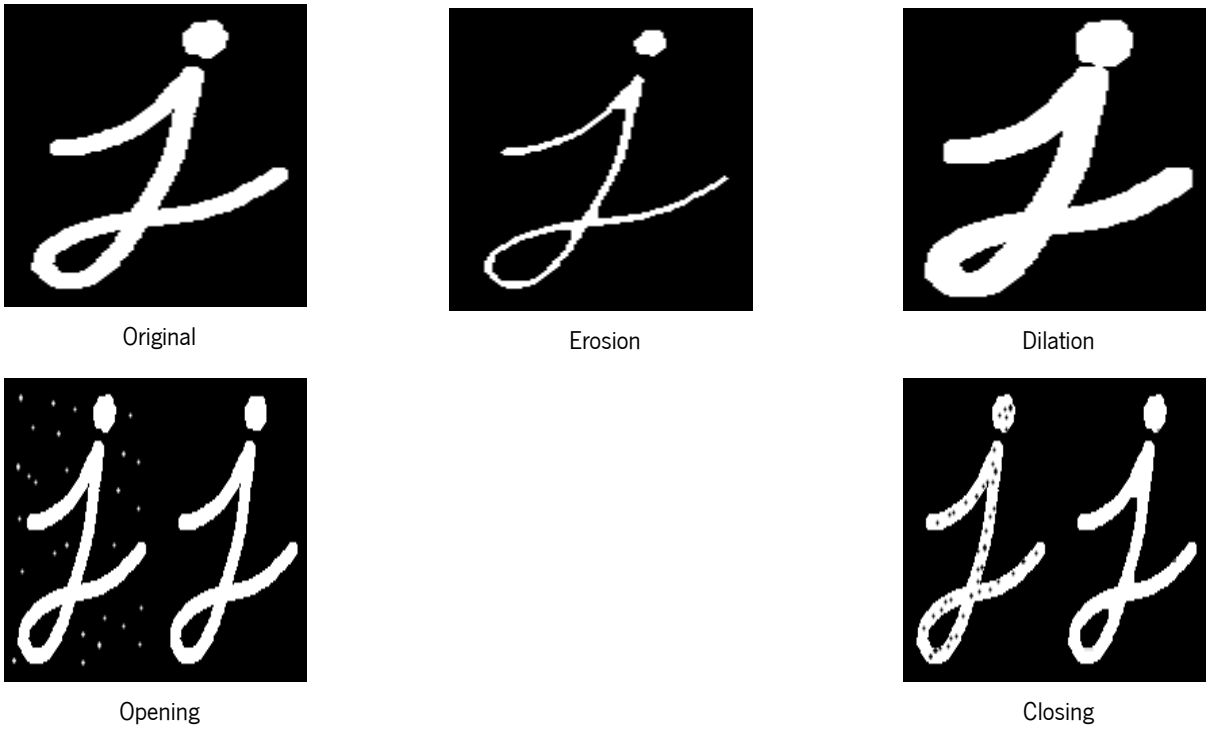
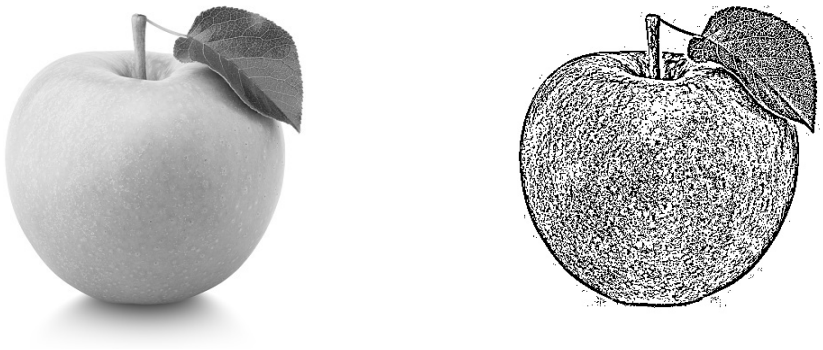


Figure 9: Morphological operations. Source [26].

There are other techniques like smoothing operations that are very used to blur images, by applying 2D convolutional low pass filters. Techniques like averaging, median and gaussian blur are applied to smooth the image and thus remove noise. The image morphological and smoothing operations are very

important in image pre-processing. However, to achieve a good performance in these tasks there are usually combined different pre-processing techniques, like image thresholding. Image thresholding is a naive way to segment objects from a background, which binarizes an image by pixel intensity. In the simple techniques, if the pixel value is smaller than the threshold, the pixel is set to zero. On the other hand, if the pixel value is bigger than the threshold, the pixel value is set to one. The expected result is a binary map, thresholding the original image, as presented in the Figure 10.



Original Binary

Figure 10: Binary image thresholding.

There are other thresholding techniques more advanced than the previous, techniques like adaptative thresholding [27] and Otsu's Binarization [28] that produce better results. In 2001, Kwak C. et al. [29] applied an image thresholding technique to segment leather defects. In this approach, two different thresholds were used, because defects normally are brighter or darker than the background. So, it was necessary one threshold to identify the brighter defects and another to identify the darker defects.

Feature extraction

In the feature extraction pipeline, the most common approach is to adopt an edge detection technique and use the result as input to a feature extraction technique.

Edge detection is a computer vision technique applied in many problems like pattern recognition, image morphology and feature extraction. Also, it is usually used as one of the first steps to solve them. The main goal of edge detection is to locate and identify sharp discontinuities in an image [30]. The image's discontinuities represent the boundaries of the objects and are defined by the region changes of pixel intensity, like colour and brightness variations.

There are several edge detection techniques, some are gradient-based, and others are gaussian-based. The techniques based on the gradient like Robert [31], Sobel [32], and Prewitt operators [33], estimate the gradient in the direction of maximum pixels intensity change. They calculate the first derivate of an image Δf in vertical $\frac{\partial f}{\partial x}$ and horizontal $\frac{\partial f}{\partial y}$ directions and combine the result with the Euclidean norm 2.1.

$$\Delta f = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2} \tag{2.1}$$

The gaussian-based techniques estimates the edges of the images by looking for the second derivate zero crossings. They are called gaussian-based, because they use the gaussian filter to smooth the images, reducing the noise. This step is very important, because makes it possible to detect zero-crossings over different scales. There are techniques as Laplacian of Gaussian and Canny that follow this approach. These edge detectors mentioned before are represented in the Figure 11.

Figure 11: Edge detectors.

After the edge has been detected, a feature extraction technique is applied. The features obtained with the second technique will be used to create a ML predictor. There are many techniques used to extract

features from edge images. However, the most used belong to two types: colour features and texture features techniques.

The colour features can be extracted from the colour distribution of the image. The image can be represented in different colour systems like grey-scale, RGB, HSV or others, but the goal, independently of the colour system, is to extract statistical features from their colour histogram. In 2016, Kavitha et al. [34] used colour feature extraction to solve the melanoma detection problem. In this case, colour features are a good option because the colour is a very representative descriptor in melanoma diagnoses. On the other hand, there are some problems where the texture descriptors are a crucial feature in the detection process. In these cases, other techniques should be adopted. The most popular texture feature extraction techniques are Gray Level Co-Occurrence Matrix (GLCM), Gabor Filter, Fourier Power Spectrum, Wavelet Transform and Local Binary Patterns (LPB).

The GLCM is an approach presented by Robert Haralick [35] to obtain 14 statistical features. GLCM calculates the spatial correlation between two pixels. It measures the occurrence of two pixels at a certain distance in four different orientations (0°, 45°, 90°, 135°). GLCM result is not used directly as a feature, instead of that, second-order statistical features are obtained from the GLCM output grey tones. This technique presents good results, however, it has significant computational and time complexity, which could be a good reason to choose other extraction feature techniques.

Gabor Filters is a feature extraction technique developed by Dennis Gabor [36] in 1946. Gabor created this technique to be appropriated for texture representation and discrimination. The Gabor Filter uses a linear filter to detect textures frequency and orientation. The linear filter used is a Gaussian Kernel function with different orientations to extract and localize textures. The output of Gabor Filters obtains statistical features to be used to create a ML model. With Gabor Filter, Dennis Gabor represents object textures' frequency and orientation, as the human visual system does. These filters have been shown to possess optimal localization properties in both spatial and frequency domains. To sum up, the Gabor Filter is a good approach to texture features representation.

Wavelet Transform is a texture feature extraction technique that applies multi-resolution analysis to extract texture features [37]. In the Wavelet Transform, the image is divided into four sub-bands (Low-Low, Low-High, High-Low and High-High). The Low-Low sub-band represents the coefficients that are most proximal from the original image, the rest sub-bands represent the finest scales wavelet coefficients, that

looked like detailed images [38]. This decomposing process is repeated using the Low-Low sub-band to obtain a new level of wavelet coefficients. This process is repeated until the decomposition level has been achieved. The selection of the decomposition levels number can be done by the wavelet band selection procedure. This is done using the energy function and reducing the resolution until the energy ratio is minimum. The Wavelet Transform coefficients resultant represents approximation and detail images used in the texture analysis. In the end, the Wavelet Transform output is applied to obtain wavelet statistical features and wavelet co-occurrence features. One advantage of Wavelet transforms when compared with Fourier transforms is the capacity to extract local and temporal information simultaneously and choose the wavelets that will be used. This technique became well-known, not just in terms of the feature extraction results, but also to be applied to real-time problems.

Local Binary Patterns (LBP) is a feature extraction technique created to detect objects' local features [39] and texture descriptors. Instead of estimating the global representation of the texture like GLCM, LBP estimates the local features representation. LBP represents the correlation between pixels in a local area, which is very useful to reveal local information. The result of the LBP technique is a feature vector resultant from histogram values. The main idea is to compare each pixel with its neighbours' pixels. In the first step, the image is converted to grey scale. After that, for each pixel, the LPB estimates the radius neighbour's region of interest. Thereafter in each neighbourhood, the number of neighbours bigger than a threshold is calculated, and that value will represent the central pixel. This process produces a new image, that will be used to construct an intensities histogram. As a result of the intensities histogram, frequency values will be used as features resultant. LBP can characterise and distinguish textures with a good performance. This technique has a reduced computational and time cost. Nonetheless, if the size of the neighbourhood area increases exponentially the computational cost will also increase.

2.4 Leather defects detection

The automatization of leather inspection is not a new problem, and researchers have been trying to solve it for a long time. In 2000, Kwak et al. [40] proposed an automated system to detect and classify defects in leather. The imaging system captured the leather images and split them into 8 cm x 12 cm patches to find local anomalies. The system located defects by applying a segmentation threshold system and morphological operations. The threshold system was composed of two fixed thresholds, one to find the defects that look brighter than the background and the other to find the defects that look darker. After

that, extract geometric features (first and second-order statistical measures) to train two multi-layered perceptron models. If multiple image patches classify as a defect, a line combination test verifies if the defective tests belong to the same defect.

In 2003, Georgieva et al. [41] proved that X^2 criteria allow distinguishing a standard image from a defective image. X^2 criteria compute the difference between the grey-level histogram of an image using a standard histogram. The standard histogram calculation was an average value from a set of histograms from non-defect images. This approach worked because anomalous images had different colors and shapes when compared with non-anomalous. These differences generate histograms that allow distinguishing defective from non-defective ones.

In Brazil, the leather economy is very important, and a fair remuneration system makes the leather profits increase, so it was necessary to create an automated inspection system [42]. They train three supervised learning classifiers: an SVM, a normalized Gaussian radial basis function network, and a KNN algorithm to solve the problem. They extract features using GLCM, obtaining 63 textures features, and extract from HBS color space. In this way, they build a dataset to perform the model training. In this work, they reach their best performance with an SVM model obtaining results above 94% correct classification.

A few years later, with the evolution of computational power, the use of ANN became a good option because the ANN training time was not a problem anymore. However, the ANN is a black-box model that has no interpretability. So, Jian et al. [43] combine ANN and decision trees to overcome this problem. They propose to use a ANN to create a supervised model trained with distinct features groups to discover what set of features was the best to train a model. In that way, they get advantages from ANN and use the best features to train a decision tree, overcoming in terms of interpretability.

In 2014, Jawahar et al. [37] presented a new approach for leather texture classification using a wavelet method for feature extraction. The images are captured and transformed in the frequency domain by wavelet transform. The next step decomposes the wavelet into statistical and co-occurrence features to build an SVM classifier with a Gaussian kernel to discriminate between defective and non-defective leather samples. The classifier trains with three different sets of features: the first set was composed of statistical features, the second by co-occurrence features, and the third with a combination of both sets. The results achieve a higher accuracy (98.56%) combining both statistical and co-occurrence feature sets.

As seen before, GLCM is a good feature extraction technique. However, it is hard to implement GLMC in real-time problems because it has a considerable time cost complexity. So, the authors compare an SVM classifier trained with GLCM statistical features and color moments with a CNN that uses features obtained from a pre-trained AlexNet [44]. Applying features results from these two different techniques, the authors express that the pre-trained approach had better performance than hand-crafted features, with pre-trained CNN achieving 99% accuracy with a lower time cost.

Other research automatically classifies the tick bite defects on calf leather [45]. The authors thought into different ways to extract features from leather samples. They adopted hand-crafted and data-driven feature descriptors to extract the local information of the leather patches. To extract hand-crafted features, they used different approaches: edge detectors (Prewitt, Roberts, Sobel, Laplacian of gaussian, Canny and ApproxCanny) and statistical approaches (histogram of pixel intensity values, histogram of oriented gradient, local binary pattern). To extract data-driven features was used an ANN. After feature extraction, the adopted classifiers were: decision tree, discriminant analysis, SVM, Nearest Neighbor, and some ensemble classifiers. The authors get good results when combining the feature extraction methods with these classifiers, applying ApproxCanny as the pre-processing method. This work shows that combining edge detectors and a data-driven approach could be a step up.

In 2021, Gan et al. [46] developed a statistical approach based on automatizing leather defect detection tasks. Firstly, each image was pre-processed and converted to grayscale. Consequently, they split an image into 100 small patches and calculate patch histograms to obtain statistical features such as median, variance, skewness, kurtosis, lower quartile value, and upper quartile value, thus allowing the discovery of local features from each image. After that, the K-S test selects the three most representative statistical features from images patches. In this way, each image has 100 patches, and the entire image has 300 features. In the next step, they use percentile thresholding to reduce the number of features. In this way, it obtains three representative values (one per statistical feature) from all the 100 patches to train the model. To validate this approach, the authors used two different datasets, the first composed of dark-line defective images and non-defective images and the second composed of bite-like anomalous and non-anomalous images. For dataset 1, the features selected were mean, variance, and lower quartile range. After the train, the Medium KNN and Cubic KNN show better performance with 99.16% accuracy. On the other hand, for dataset 2 the features selected were variance, mean, and skew using the Gaussian Naive Bayes classifier. It achieves the highest accuracy of 77.13% and the lowest error rate of 22.87%.

In 2020, Deng et al. [47] developed a new framework for classifying leather surface defects based on a parameter optimized residual network. In ResNet, parameter optimization has a critical role in the efficiency, so, they optimize two different parameters: sliding patch window size and the number of samples in the dataset. The sliding patch window obtains more patches from the original image and it is necessary to optimize the number of patches. Because, if the patch window is too small cannot fully contain the defect area. Alternatively, if the patch window is too large, the features will become global. In this way, they try to find the optimal patch window size. They determine the optimal windows size using six distinct datasets with distinct window sizes. These datasets train and test the ResNet, and with the accuracy results obtained by each dataset, was applied the least-squares method to discover the optimal window dimension. After that, they optimize the parameters, and the ResNet trained and tested. It achieves better performance when compared with other networks like Lenet5, Caffenet, Faster R-CRESNET, CornerNet, and ResNet without optimized parameters, and it is time-efficient and easy to prepare while compared with the SSD.

In 2020, Aslam et al. [48] develop a new approach where ensembled models were used to automatize the visual inspection. They combine knowledge from different domains to help in the leather domain. Firstly, EfficientNet-B3 and DenseNet-201 were trained from scratch using the skin, concrete, and ImageNet datasets. After that, the models fine-tune with leather image data. They verified that using transfer learning with the ImageNet dataset had better results than using the skin and concrete dataset. Thus, different models trained by transfer learning with ImageNet and fine-tunned by leather image data were ensemble because ensemble models obtain a higher accuracy than a single classifier. In the end, they conclude that EfficientNet-B3+ResNext-101 are the best models to ensemble. This model provides better AUC and F1-score than the other ensemble combinations.

Chapter 3

Problem

The next chapter contextualizes leather's role in society and details the leather production process. Also, it expresses leather defects and their causes. The next section describes the leather inspection problem. In the end, it proposes a new solution to automatize leather defects detection.

3.1 Leather

Leather is a natural material derivated from the skins or hides of animals like cows, sheep, lizards, and goats. Since ancient times, it has been used to create clothing to protect humans from weather conditions. It keeps their bodies dry and their temperature constant. Nowadays, leather still plays a crucial role in our society, because, leather has characteristics like comfort, flexibility, and resistance making leather a product used until now. Also, it is a big industry that plays a critical role in a lot of countries' economies.

Before obtaining leather, the raw hides go through a process split into three stages: pre-tanning stage, tanning stage, and post-tanning stage, as illustrated in Figure 12. As presented in [49], the pre-tanning stage prepares the raw hide for the tanning process, converting raw leather into pelt. In this stage, the hides are trimmed and soaked to restore the moisture loss and clean the piece from salt and other solids. After that, they remove the excess tissues, muscles, and fat using a fleshed process. Also, it applies the unhairing by the liming process. The leather must be soft and flexible, so the raw is bated and limed. At the end of the pre-tanning stage, the pickling process adjusts the acidity.

The following phase is called tanning. In this phase, the pelt converts to wet-blue leather. This process dries pelt to a flexible form without putrefying. There are two similar methods used in the tanning process: vegetable tanning and chrome tanning. Depending on the final application, one of these processes is applied. The vegetable tanning process lasts three weeks until the dye penetrates the pelt. After, the pelt dips in sodium bicarbonate or sulphuric acid drums for bleaching and removes tannins bound to the surface. Another hand, the chrome tanning applies the trivalent chromium salt to the pelt.

The third phase is post-tanning which converts wet-blue leather into crust leather. This process involves steps such as wetting back, sammying, splitting, shaving, re-chroming, neutralization, re-tanning, dyeing, fat liquoring, filling, stuffing, stripping, whitening, fixating, setting, drying, conditioning, milling, staking, and buffing.

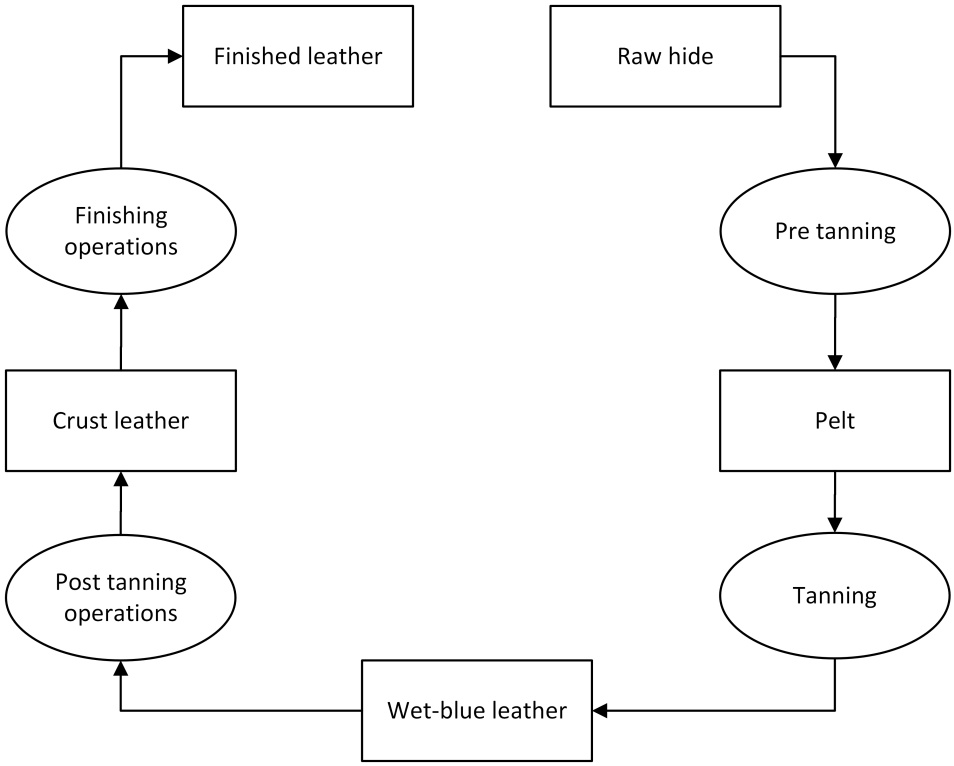


Figure 12: Leather process.

Also, there is an optional phase where they coat the crust leather and apply the following processes: oiling, brushing, padding, impregnation, buffing, spraying, roller coating, curtain coating, polishing, plating, embossing, ironing, combing, glazing, and tumbling.

After describing the leather producing process, it is time to explore the defects, starting by clarifying

the defect concept. Defects are variations in the regular pattern of textures. So, in this problem, defects are damages like scars, spots, wrinkles, ticks, iron brands, cuts, scabies, and holes caused by several conditions, as presented in the Figure 13. Defects can be very irregular, even if both belong to the same defect category. In this way, it is unlikely to find two defects with the same shape and size.

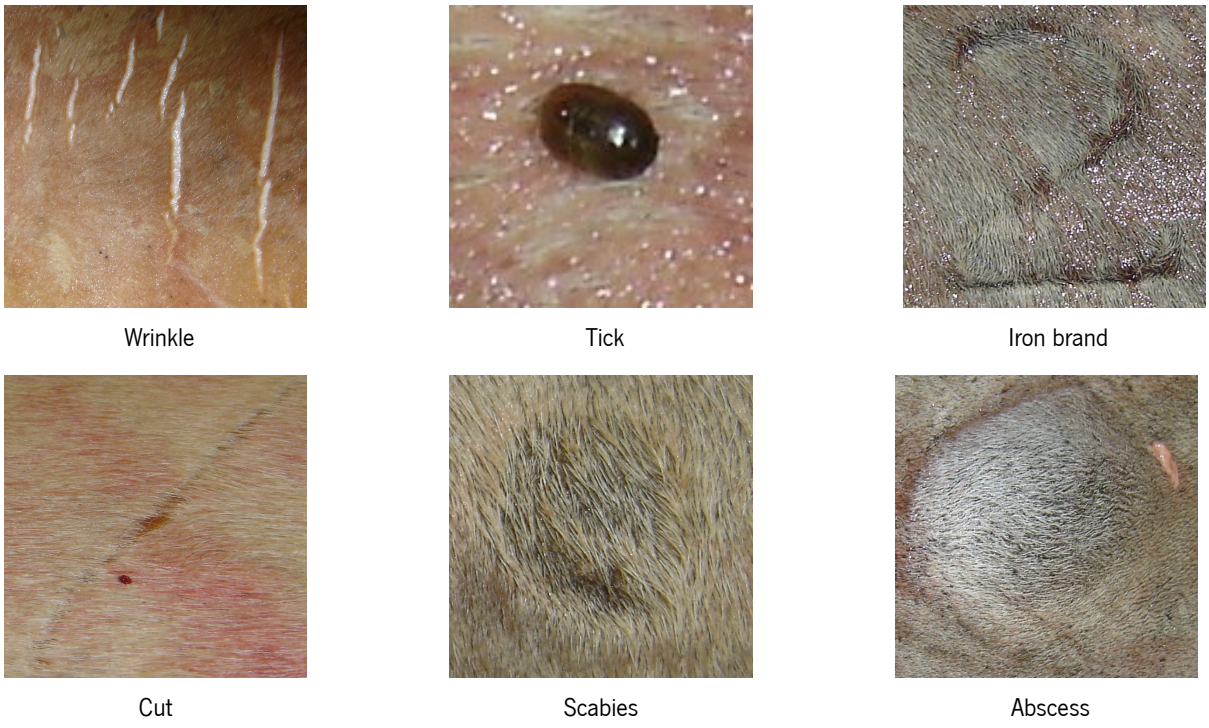


Figure 13: Defects examples.

To produce high-quality leather, it is vital to analyze the defects to understand their cause and implement measures to prevent future anomalies. For example, if in leather inspection are found a high quantity of defects caused by wired fences, the fences should be repaired or changed with a fence material that does not damage the animal's hide. With measures like that, the ratio of high-quality leather will improve.

Nowadays, the industry knows most of the defect's causes and the different stages where they can be produced. Firstly, the stages can be separated into two: during animal lifetime and post animal lifetime. During an animal's lifetime, the animal can suffer cuts produced by barbed iron and horns. Also, they can suffer iron brand burns, tick bites, scabies, whiplashes, or skin diseases such as ring worm and streptothricosis. In the post animal lifetime, the defects can result from the entire process to obtain leather from raw hide. It can be in the transportation phase if the hides are not well stored in the trucks or can happen in the flaying process if the skinning techniques are not well applied. As well can result from the curing process if the quantity of salt used exceeds.

In the leather inspection process, there are different defects. In this way, it is important to classify them into different types. Using the leather defects classification system presented in [1], defects are classified into five types based on the stage that they happen: Natural, Mechanical, Flaying, Curing and Storing, Tanning. The Table 1 presents the different defect types and their causes.

Table 1: Leather defects and causes.

Defects	**Causes**
Natural Defects	Skin diseases, scabies, mosquito bites, scars and birth signs
Mechanical Defects	Scratches from wired fences, brand marks, bruises, and wound in living animals
Flaying Defects	Scores, cuts, corduroy, pulling machine damage, grain, and cracks
Curing and Storing Defects	Putrefaction caused by excess salt or low-quality salt used to preserve the leather
Tanning Defects	Fleshing cuts, incorrect physical/chemical in the tanning process.

3.2 Problem identification

In this work, automatization of leather defects detection is the problem proposed to solve. The automatization of these process is essential because the manual process has a high cost and does not present a good performance when it is repeated during long shifts. The cost of the traditional method is high because it requires a human expert to inspect the leather.

The standard process is very slow because the defect detection task is complex. Detecting leather defects is not easy for the naked eye. To improve these tasks the workers manipulate the leather with distinct movements and points of view with light support to localize the defects. Usually, the difference between a non defective piece and a defective one is very small, keeping the same clean pattern in most of the piece regions. These issues make human inspection harder and induce bad performance. Also, the inspection process produces distinct results with distinct workers. So, the results from the manual inspection system are subjective.

For manual inspection to achieve good results, humans must be very concentrated and handle eye fatigue caused by illumination. Beyond, leather inspection results vary from human inspector to human inspector because workers can have different experience levels or be emotional/physically changed. Problems like that slow down leather inspection and reduce the detection accuracy. In this way, it is clear that manual inspection is not the optimized solution.

To conclude the identification of this problem, it's clear that the solution to be developed involves an automatic system that avoids human workers. In this way, it will be possible to reduce the time cost with a real-time defects detector and reduce the errors rate, providing a system with better performance than humans. So, the solution will be composed of an objective CV technique that integrates AI techniques to detect leather defects accurately.

3.3 Challenges

As presented in the Chapter 2, the solutions for this problem are based on ML and DL approaches integrated into a CV system. The majority of them follows the same workflow: image capture, image processing, feature extraction and supervised detection.

As known, the main requirement to build a supervised model is a high dimension dataset. It is impossible to train a supervised model without labeled data. The labeled dataset building process is tedious, requiring human experts to label the data. In this problem, building a high dimension dataset is even harder because manual defect detection is very difficult and time-consuming. So, creating a dataset for this problem has a huge cost.

Another challenge is getting a balanced dataset. It is difficult because most of the leather samples are non-defective, just a small percentage are from defective classes. In this way, without a balanced dataset is impossible to teach a supervised model to detect the defects.

3.4 Solution

In the previous section were presented lacks and challenges for this problem. In order to overcome them, Bergmann et al. [50] adopt unsupervised solutions techniques to detect and localize defects in distinct textures and objects, using the MVTec Anomaly Detection dataset. Choosing an unsupervised strategy for solving this problem is a wise option because the defects produced can be from unlimited patterns, and the dataset used can not represent all the possible patterns.

In their work, they adopted distinct techniques to detect anomalies such as convolutional autoencoders, generative adversarial networks, and feature descriptors using pre-trained CNNs. The strategies adopted allow to identify if a new image follows the same pattern as the images used in the train (defects-free

images). In 2014, Pimentel et al. [51] conducted a survey to review the novelty detection strategies. This survey is a great overview of the novelty strategy and presents different methods that can be applied to this problem. The survey explores distinct approaches like probabilistic, distance-based, reconstruction-based, and domain-based. All these approaches are distinct and the algorithms used in each one could be different, but, the workflow is the same.

The novelty strategies classify the data as inlier or outlier elements. If the sample follows the pattern presented is an inlier, and if not it is an outlier. In novelty detection, firstly, an unlabeled dataset is acquired, composed of anomalous samples and non-anomalous samples. After that, the ML architecture trains using the dataset, learning the inliers pattern because outliers samples are rare. In this way, the goal of the ML algorithm is to perform good predictions with the inliners samples and fails when trying to predict outliers samples. Thus, with the result obtained from predictions is possible to distinguish between inliners and outliers.

In [50] the leather samples splits into two datasets during the training phase. One dataset trains an ML algorithm to learn the leather features. After that, the other dataset combined with the trained model estimates a threshold to segment the defective pixels. This procedure is illustrated in the Figure 14.

In the testing phase, labeled defective and non-defective images are used to test the model. The architecture makes predictions for each data element, and the anomalies map is estimated. The anomalies map represents the probability of each pixel being defective. It is helpful in the localization of the defect. However, a threshold can be used to segment the entire image into defective and non-defective pixels. In the Figure 15 is illustrated the expected anomaly map and binary map using the procedure illustrated in the Figure 16.

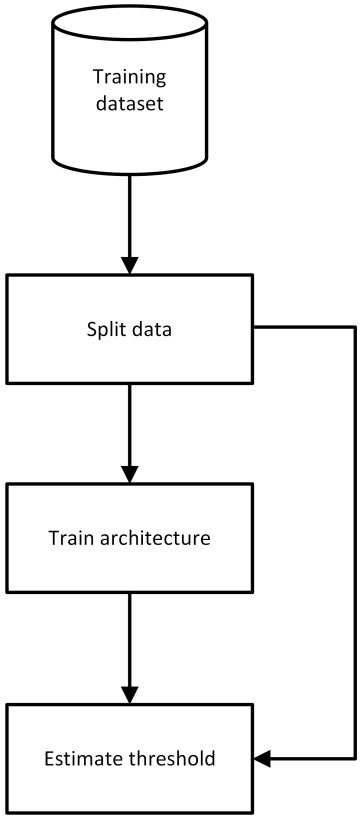
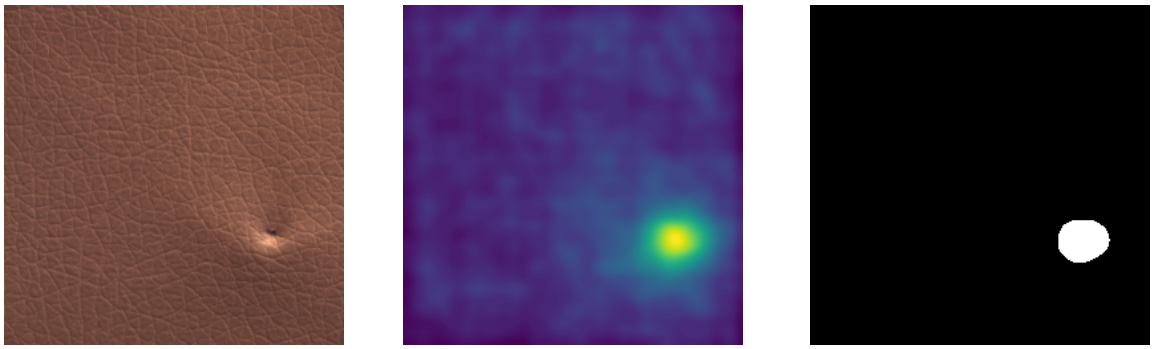


Figure 14: Novelty strategy - training phase.



Defective image | Anomalies score map | Anomalies binary map

Figure 15: Defects detection example.

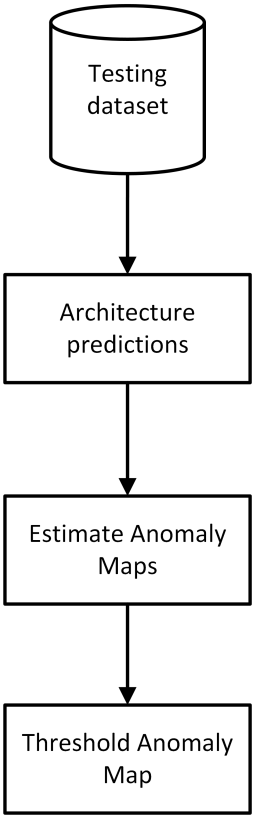


Figure 16: Novelty strategy - inference phase.

After careful research on novelty detection state-of-the-art, it seems that novelty detection can be the solution to this problem. Novelty detection overcomes the lack of the leather defects detection state-of-the-art solutions and the challenges referred before. So, the techniques chosen to solve the problem are:

- MSE and SSIM Convolution Autoencoder [50];
- CFLOW-AD [52];
- Student-Teacher Feature Pyramid Matching [53];
- Reverse Distillation from One-Class Embedding [54];
- Discriminatively Trained Reconstruction Anomaly Embedding Model [55].

These techniques were chosen because they are based on distinct approaches: Reconstruction-based, Embedding Similarity-based, and an approach that converts the unsupervised problem into a supervised problem.

Chapter 4

Novelty Detection

As seen in the last chapter, this dissertation uses novelty detection techniques to solve the leather detection problem. This chapter presents the novelty detection concept and how its techniques can solve the defects detection problem. Also, it presents all the novelty techniques used in this dissertation: MSE and SSIM Convolution Autoencoder, CFLOW, Student-Teacher Feature Pyramid Matching, Reverse Distillation from One-Class Embedding, Discriminatively Trained Reconstruction Embedding.

In ML, the most critical requirement is data. Without a large dimension and representative dataset, it is impossible to create an optimized model. In this problem, although been crucial a high dimension dataset, the real challenge is to acquire a representative dataset. It is impossible to obtain samples representing all the defect classes. Even if we train it to cover all known defect types, there is always a chance of appearing a defective class never seen before. Beyond this, the anomalous sample occurrence is rare, difficulting the dataset creation task.

A solution to the leather defects detection problem is novelty detection, in which an ML model learns a pattern using an unlabeled dataset. This problem fits into novelty detection because novelty detection learns the normal patterns, allowing the detection of elements that differ from the pattern. As leather anomalies occurrence are rare, novelty detection will learn the features from the clean samples and will be able to detect the anomalous regions in the defective samples.

The novelty detection techniques are divided into two types: Reconstruction-based and Embedding Similarity-based. Reconstruction-based are techniques that, as the name says, try to reconstruct input data. The reconstruction technique learns to reconstruct good samples and fails when finding anomalous regions. The architectures used in reconstruction are Autoencoders, Variational Autoencoder, and Generative Adversarial Networks (GAN). The reconstruction methods can localize the anomalies using the pixel error or can use a structural similarity function.

On other hand, the Embedding Similarity-based techniques use pre-trained DL networks to extract vectors describing an entire image for anomaly detection. The extracted vectors are combined to perform an anomaly score map. One advantage of the embedding methods is to obtain features from different layers. In this way, if the output extracted is from the first layers, the features obtained will represent local defects. If the extracted output is from the last layers, the obtained features will represent global defects.

To help other people to understand and develop new novelty detection techniques, Markou et al. [56] detailed the principles that a novelty detection technique should follow. The most important are summarized as follows:

- **Principle of robustness and trade-off**: A novelty detection method must maximize the exclusion of novel/anomalous samples and minimize the normal samples exclusion in the test dataset.

- **Principle of uniform data scaling**: Training and testing data should have the same normalization range.

- **Principle of independence**: The novelty detection methods should be independent of the number of features and classes. Also, should perform well when trained by a non-balanced dataset.

- **Principle of adaptability**: A system that recognizes novel samples during the test should be able to use this information for retraining.

- **Principle of computational complexity**: The computational complexity should be as minimum as possible because the novelty application has to work as a real-time application.

4.1 MSE and SSIM Convolution Autoencoder

In 2016, Goodfellow et al. [57] developed a new image reconstruction architecture, the Convolution Autoencoder (CAE). The CAE improves from the original version of the Autoencoder (AE), changing the fully-connected layers to convolution layers. As in the original AE, the input shape is the same as the output shape because the main goal of an AE is the reconstruction of a data element. In the case of images, the reconstruction can be very useful in problems such as image comprehension and image denoising [58] [59].

An AE is composed of encoder-decoder network architecture as described in the Figure 17. The encoder extracts the most important features from the input over the layers producing a representative and small dimension feature vector from the original input [60]. The representative features extracted are known as latent space, and are used as decoder input. The role of the decoder network is to reconstruct the input from the latent space. The CAE follows the same encoder-decoder architecture, the only difference is the use of convolution layers. The convolution layers learn to extract features independently of the positions.

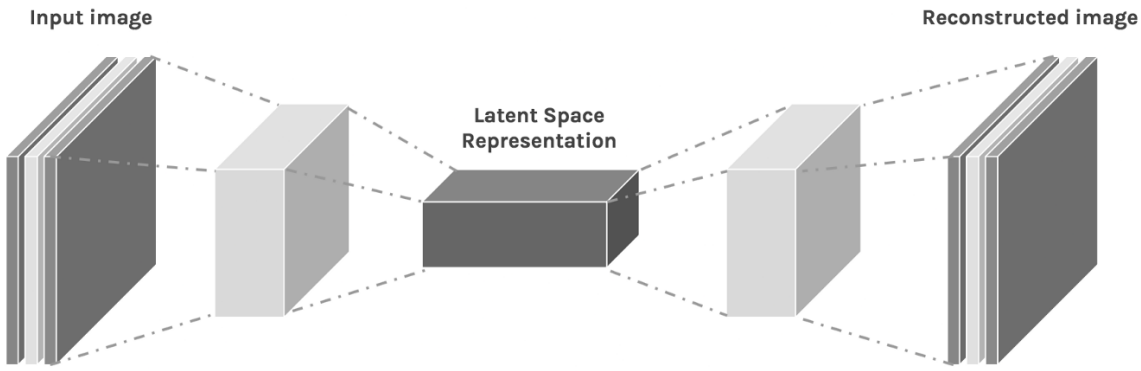
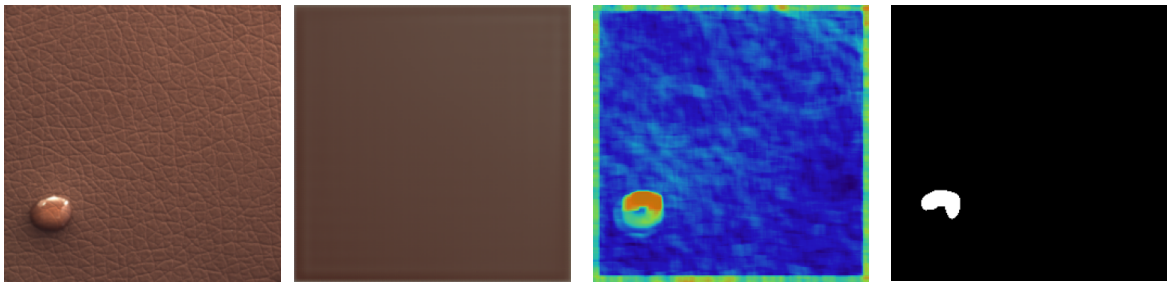


Figure 17: Convolution Autoencoder Architecture.

Beyond image comprehension and image denoising, CAE can be very useful in other problems such as novelty detection. The CAE can learn to extract features from a non-defective sample and learn to reconstruct a non-defective image. One example of the application of CAE in unsupervised problems is the work of Bergmanm et al. [61]. They describe a technique that combines a CAE with Structural Similarity Index Measure (SSIM) and MSE. They pretend that the architecture will be able to reconstruct non-defective images. However, in a defective image the CAE's reconstruction should fail.

In this method, the reconstructed image obtains the anomaly score map using a pixel-per-pixel comparison between the reconstructed image and the original. Thus, it is possible to identify the reconstructed areas that differentiate from the original image. The authors choose SSIM and MSE to obtain the anomaly map. In this way, the anomaly map should indicate the image reconstruction errors, which means the defective areas. After that, it binarizes the image in defective and non-defective areas.

As seen in the Figure 18, the original image is reconstructed by the CAE, obtaining the reconstructed image. The reconstructed image and the original image compare pixel errors using SSIM or MSE to obtain the anomalies score map. The anomaly score map is thresholded to obtain the binary map.



Defective image | Reconstructed image | Anomalies score map | Anomalies binary map

Figure 18: Defects detection with CAE.

4.1.1 Architecture

This work adopted the same architecture as the one used in Bergmann et al. [62]'s paper. It is composed of two distinct networks: the encoder and the decoder. The encoder is composed of a CNN as described in the Table 2, and the decoder has an inverse architecture.

Table 2: Encoder Architecture.

Layer	Output size	Kernel	Stride	Padding
Input	height*width*depth	-	-	-
Conv1	64*64*32	4*4	2	1
Conv2	32*32*32	4*4	2	1
Conv3	32*32*32	3*3	1	1
Conv4	16*16*64	4*4	2	1
Conv5	16*16*64	3*3	1	1
Conv6	8*8*128	4*4	2	1
Conv7	8*8*64	3*3	1	1
Conv8	8*8*latent dimension	3*3	1	1

4.1.2 SSIM and MSE metrics

In the image processing field, image quality metrics can play distinct functions in image applications [63]. It can evaluate a video stream quality or the quality of an image reconstruction like in the autoencoders. In this method, the SSIM and MSE metrics estimate the anomaly score maps, calculating the error between the original and the reconstructed image.

4.1.2.1 MSE

The naive way to evaluate an image quality is using the Mean Squared Error (MSE) because the MSE is easy to calculate (Equation 4.1). MSE is useful to estimate the error between the pixel values of two images but does not allow us to understand the image degradation. It is a mistake in image processing because the pixel error can not quantify the image quality. Two distinct images can have the same error signal, but they can have distinct error levels to the human vision. The MSE can be used as a loss function to quantify the error between two images. MSE loss sums the squared difference between the original image X and the reconstructed image Y pixels. However, the squared error is also used to estimate the anomaly maps.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(X_i - Y_i)^2 \tag{4.1}$$

Natural image signals are highly structured: their pixels exhibit strong dependencies, especially when they are spatially proximate [64], and these dependencies carry significant information about the structure of the objects in the visual scene. So, it was necessary to develop an evaluation metric used to extract image structural information based on the human visual system.

4.1.2.2 SSIM

In this way, Wang et al. [65] developed SSIM to explore the image structural information. The SSIM is an image quality measure system that separates the similarity estimation into three comparisons: luminance, contrast, and structure.

Figure 19 presents the SSIM measurement diagram. It receives two distinct image signals (in this problem, one is the original image and the second is the reconstructed image), and as the image luminance and contrast vary across a scene, the local luminance and contrast are estimated.

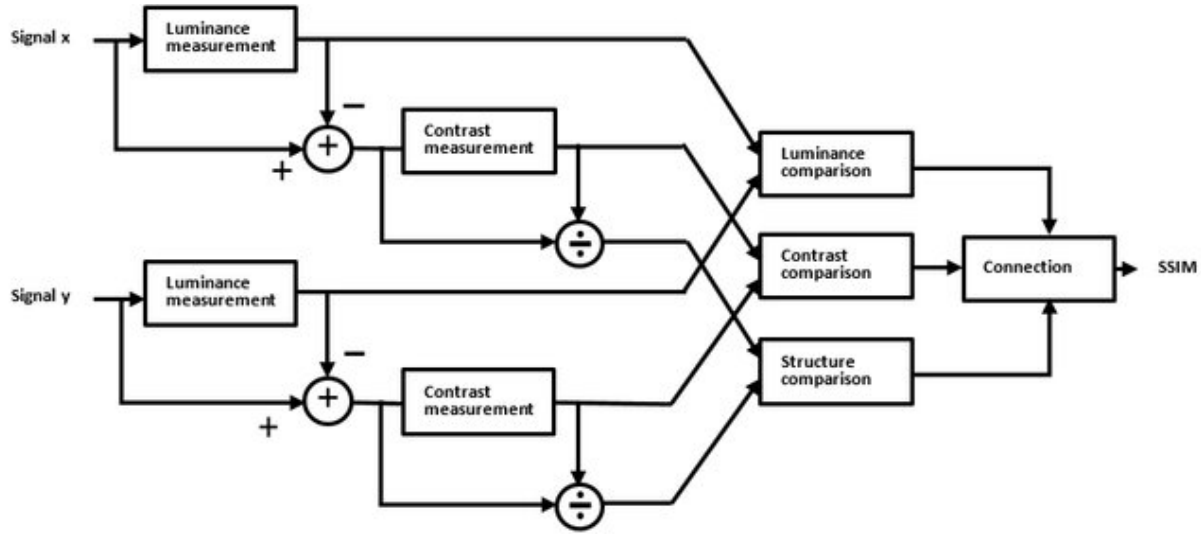


Figure 19: Diagram of the SSIM. Source: [65].

First, the luminance of each image signal μ_x is estimated Equation 4.2, using the mean value of the entire pixels N. After that, the μ_x and μ_y estimates the luminance comparison $l(x, y)$ using the Equation 4.3, where the constant C_1 is included to avoid instability when is very close to zero.

$$\mu_x = \frac{1}{N}\sum_{i=1}^{n} x_i \qquad (4.2) \qquad\qquad l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2\mu_y^2 + C_1} \qquad (4.3)$$

To estimate the image contrast measurement, it is used the square root of variance (Equation 4.4). After the σ_x and σ_y estimation, the contrast comparison is done using the Equation 4.5. Finally, to obtain the structure comparison (Equation 4.7) it is necessary to estimate the σ_{xy} (Equation 4.6). In these two comparisons two constants are used C_2 and C_3 that perform the same function as C_1.

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{n}(x_i - \mu_x)\right)^{\frac{1}{2}} \qquad (4.4) \qquad\qquad c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2\sigma_y^2 + C_2}. \qquad (4.5)$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y) \qquad (4.6) \qquad\qquad s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \qquad (4.7)$$

In the end, the three comparisons estimate the final result of the SSIM using the Equation 4.8 where the constants $\alpha > 0$, $\beta > 0$ and $\gamma > 0$.

$$SSIM(x, y) = [l(x, y)^{\alpha}] * [c(x, y)^{\beta}] * [s(x, y)^{\gamma}] \tag{4.8}$$

In practice, SSIM is not used. Instead, the mean SSIM (MSSIM) is performed. It estimates the mean SSIM value for patches obtained using a sliding window approach. Wang et al. [65] used MSSIM to prove why the MSE metric should not be used in image problems. Figure 20 presents a boat image with distinct distortions. However, they all have the same MSE value and distinct MSSIM. The distortions with bigger MSSIM are more adapted for the human vision system, allowing humans to extract structural information from the viewing field.

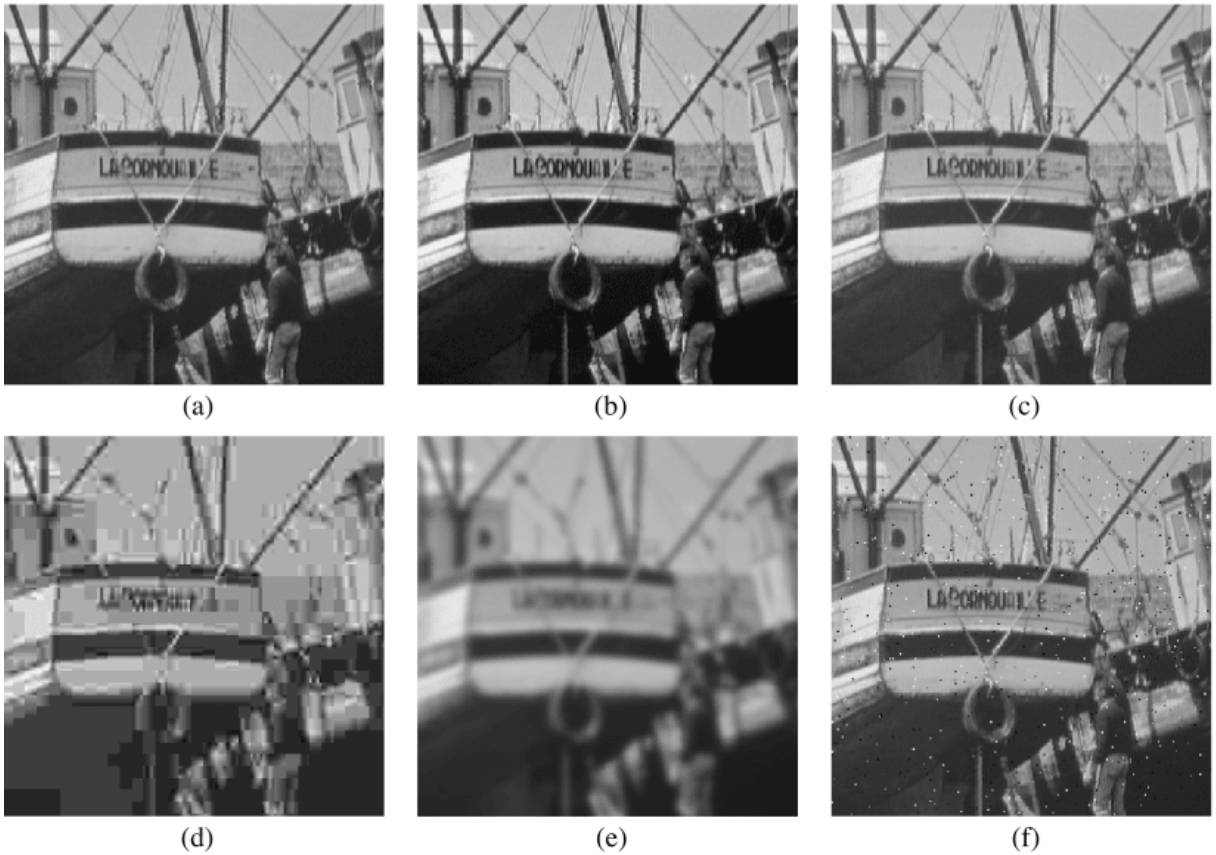


Figure 20: The image (a) is the original image; the image (b) is a contrast stretched image with MSSIM = 0.9168; (c) Mean-shifted image, MSSIM = 0.9900; (d) JPEG compressed image, MSSIM = 0.6949; (e) Blurred image, MSSIM = 0.7052; (f) Salt-pepper impulsive noise contaminated image, MSSIM = 0.7748. Source: [65].

This dissertation recreates this technique, implementing the same autoencoder architecture. The autoencoder is trained using Adam optimizer and can estimates the loss using SSIM or MSE function. If the loss function is SSIM, the anomaly score map is obtained using SSIM. If MSE, vice versa.

4.2 CFLOW-AD

In the last years, pre-trained DNNs had become very useful in the feature extraction task. For this problem, instead of training a new DNN for a specific problem, usually, it is used a pre-trained network trained using a large database like ImageNet. In this way, it is possible to extract features that should be different in the presence of defects [66], allowing anomaly detection.

Based on the Embebbiding Similarity techniques, Gudovski et al. [52] developed CFLOW to solve image anomaly detection problems. CFLOW is a feature extraction scheme with multiscale feature pyramid pooling that extracts feature maps at different spatial dimensions. It is composed by a discriminative pre-trained CNN and a conditional normalizing flow network, as described in the Figure 21. In this way, they pretend to reduce the anomaly detection time, reducing the computational complexity.

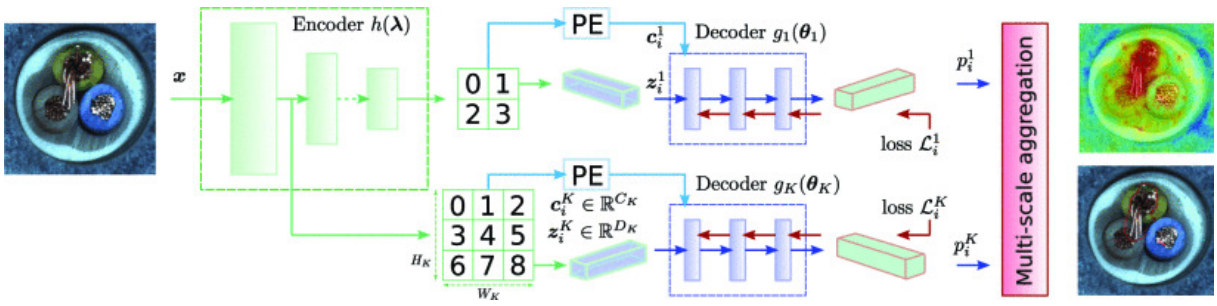


Figure 21: CFLOW-AD architecture. Source: [52].

4.2.1 Discriminative network

The discriminative pre-trained CNN has an encoder architecture $h(\lambda)$ where are extracted features from k pooling layers. In this way, the CNN encoder maps image patches into a feature vectors $z_i^K, i \in \{H_k * W_k\}$, that contain relevant semantic information about their content. As seen before, the size of the leather defects can vary. So, it is important to extract vectors to capture all types of defects. In this way, to extract vectors from distinct pooling layers, it was adopted a multi-scale feature pyramid pooling

approach. The early pooling layers extract local features that can represent smaller defects, and the latter's pooling layers extract global features that can represent bigger defects.

4.2.2 Conditional normalizing flow network

After extracting the feature z_i^K vectors from the encoder, in order to give a sequence to the z_i^K vectors, each vector goes through a specific positional encoder to obtain the c_i^K vectors. This step is necessary to inject information about the relative or absolute position of the vectors, adding information to the vectors of which layer extract it [67]. This way, the c_i^K produced vectors will have information about the local features or global features. Each c_i^K has sine and cosine harmonics that are unique to its spatial location. To estimate the sine and cosine harmonics Equations 4.9 and 4.10 were used where pos is the position and i is the dimension. The results c_i^K of the sine and cosine harmonics is a vector with the same size as the z_i^K because these two vectors will be combined in the decoder.

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{2i/size}}\right) \tag{4.9}$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{2i/size}}\right) \tag{4.10}$$

For each positional encoder, a decoder architecture $g_K(\theta)$ is created. It will use c_i^K and z_i^K as input to reconstruct the original image. Each CFLOW decoder is composed of a sequence of conventional coupling layers, and each coupling layer comprises the fully-connected layer with the kernel, softplus activation, and output vector permutations. The CFLOW decoders trains using a maximum likelihood, minimizing Equation 4.11, where the random variable $u_i = g^{-1}(z_i, c_i, \theta)$ and the the Jacobian $J_i = \nabla_z g^{-1}(z_i, c_i, \theta)$.

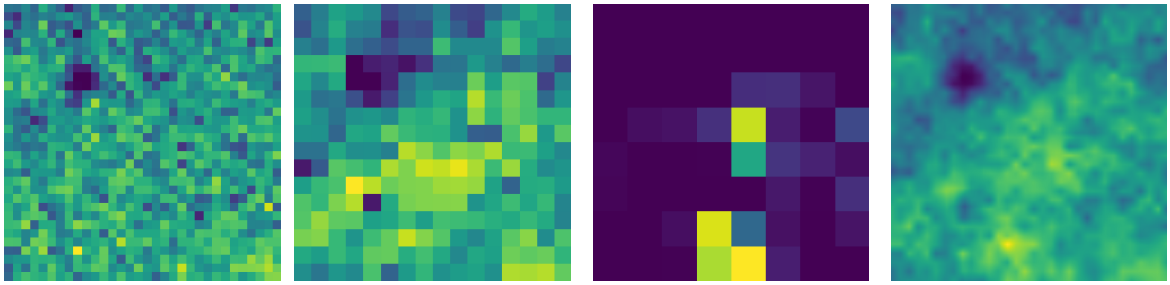
$$L(\theta) = \frac{1}{N}\sum_{i=1}^{n}\left[\frac{||u_i||_2^2}{2} - log|detJ_i|\right] + const \tag{4.11}$$

After training all decoders $g_K(\theta_k)$ the log-likelihoods for the test dataset are estimated using Equation 4.11. In the next step, the log-likelihoods are converted to probabilities (anomaly score map) $p_i^k = e^{log\hat{p}z(z_i,c_i,\theta)}$ for each kth scale using Equation 4.12 and normalize them to be in [0 : 1] range.

$$log\hat{p}_z(z_i, c_i, \theta) = -\frac{||u_i||_2^2 + log(2\pi)}{2} + log|detJ_i| \tag{4.12}$$

In the end, the reconstructed images are upsampled to the original image dimension using bilinear interpolation, followed by an aggregation of the kth anomaly score map.

Figure 22 depicts this technique's expected result. Using $k = 3$ pooling layers, there will be used 3 $g_K(\theta)$ decoders to reconstruct the 3 z_i^K extracted vectors. In this way, the image (a), (b), and (c) are the anomaly scores map p_i obtained, and image (d) is the result of the aggregation of the 3 score maps.



(a) 1th map (b) 2th map (c) 3th map (d) Aggregated map

Figure 22: Kth anomaly score maps and aggregated map.

In their research, the authors use distinct pre-trained DNN like: ResNet18 [68], WideResNet50 [69] and MobileNetV3L [70] with distinct number k of pooling and coupling layers. The results obtained show better performance when using the WideResnet50 as an encoder, with the number of pooling layers $k = 3$ and coupling $= 8$.

4.3 Student-Teacher Feature Pyramid Matching

In the last year, there have been developed many Embedding Similarity-based approaches, and the Student-Teacher Feature Pyramid Matching (STFPM) [53] is one of them. STFPM follows a student-teacher framework to extend both efficiency and accuracy. A pre-trained network working as a teacher distills the knowledge to teach the student network anomaly-free image representations. Like [52], the STFPM uses a feature extraction pyramid scheme to produce a multi-level knowledge to detect anomalies of various dimensions.

4.3.1 Knowledge Distillation

In ML, there are high complexity problems, such as speech and object recognition which need to work in real-time solutions. However, these solutions are complex, requiring high computation power. In this way, Hinton et al. present Knowledge Distillation to solve the problem [71].

Knowledge distillation plays a fundamental role in a teacher-student learning architecture, where the student-teacher approach distills knowledge from a pre-trained model (teacher) to a new model (student). Usually, the student is a smaller model with less complexity. The final goal is to obtain a small network that imitates the teacher. In this way, the student could reach the same performance with a less complex solution, reducing the inference time.

In [72] a teacher-student architecture is applied to solve the anomaly detection problem. However, the smaller network working as a student model looses significant information in the distilling knowledge process. In this way, the STFPM pretends to overcome the previous solutions. In STFPM, the author chose a pre-trained ResNet18, trained with the ImageNet dataset, to work as a teacher architecture. One improvement relatively the previous techniques is the student architecture chosen, in this technique, the authors use a student network with the same architecture as the teacher. In this way, the student network learns the non-defective distribution by combining their features with the pre-trained network features. Using this transfer of knowledge process, more important information is transferred. STFPM aims to obtain teacher and student feature pyramids to estimate the anomaly map scores, indicating the probability of an anomaly occurring. The STFPM architecture is presented in Figure 23.

In the training phase, the goal is to obtain a student network that performs like the teacher network. Using a dataset $D = I_1, I_2, ..., I_n$, for each input image $I_k \in \mathbb{R}^{w*h*c}$, where h is the height, w is the width and c is the number of the depth channels, there are extracted lth teacher and student bottom layers outputs. $F_t^l(I_k) \in \mathbb{R}^{w_l*h_l*d_l}$ and $F_s^l(I_k) \in \mathbb{R}^{w_l*h_l*d_l}$ are respectively the teachers and students features maps obtained, where w_l , h_l and d_l denote the width, height and channel number of the feature map. After the $F_t^l(I_k)$ and $F_s^l(I_k)$ been calculated, the loss is estimated. Firstly, for each layer, the l_2-normalized vectors, $\hat{F}_t^l(I_k)$ and $\hat{F}_s^l(I_k)$, are estimated using Equations 4.13 and 4.14, respectively.

$$\hat{F}_t^l(I_k)_{ij} = \frac{F_t^l(I_k)_{ij}}{\left\|F_t^l(I_k)_{ij}\right\|_{l_2}} \qquad (4.13)$$

$$\hat{F}_s^l(I_k)_{ij} = \frac{F_s^l(I_k)_{ij}}{\left\|F_s^l(I_k)_{ij}\right\|_{l_2}} \qquad (4.14)$$

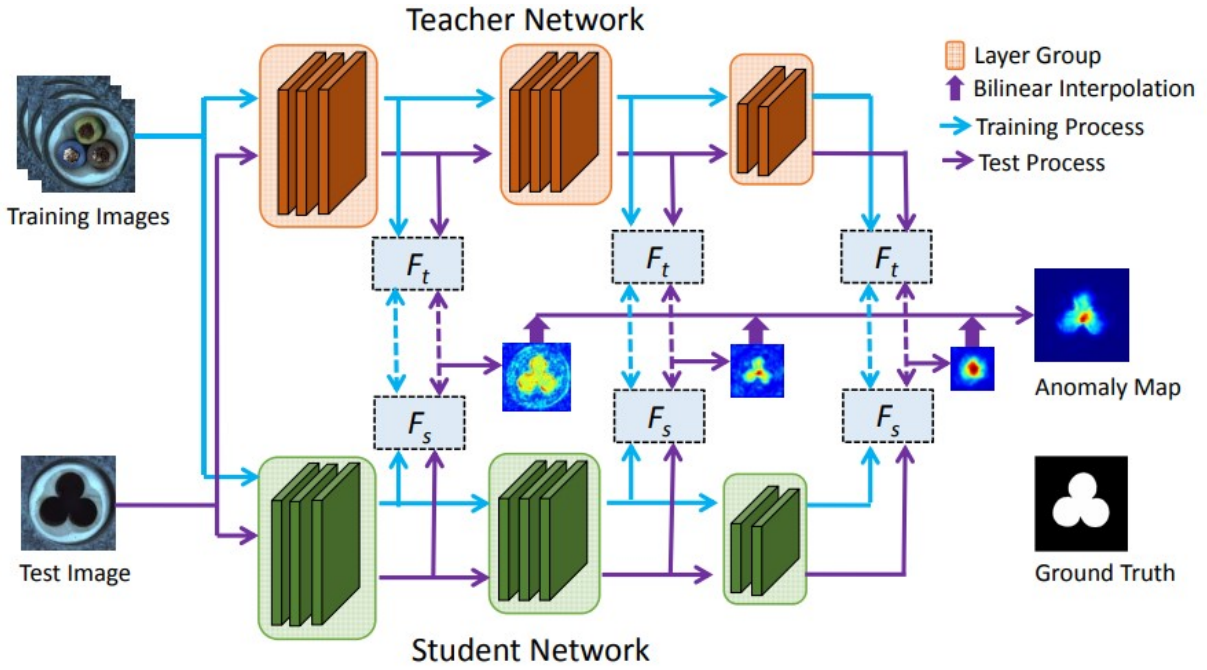


Figure 23: STFPM architecture. Source: [53].

After that, the l_2 distance is estimated between $\hat{F}_t^l(I_k)$ and $\hat{F}_s^l(I_k)$ using Equation 4.15.

$$l_2^l(I_k) = \frac{1}{2} * \left\| \hat{F}_t^l(I_k)_{ij} - \hat{F}_s^l(I_k)_{ij} \right\|_{l_2}^2 \tag{4.15}$$

Consequently, the average loss $l_2^l(I_k)$ at each position is estimated using Equation 4.16. In the end, each layer losses are combined using a weighted average of the loss at different pyramid scales using Equation 4.17.

$$l_2^l(I_k) = \frac{1}{w_l h_l} * \sum_{i=1}^{w_l} \sum_{j=1}^{h_l} l_2^l(I_k)_{ij} \tag{4.16}$$

$$l_2(I_k) = \sum_{l=1}^{L} \alpha_l * loss^l(I_k), \quad \alpha_l \geq 1 \tag{4.17}$$

During the training phase, the student model parameters update using the stochastic gradient descent. The optimized parameters from the pre-trained teacher did not change.

In the testing phase, an image $J \in \mathbb{R}^{w*h*c}$, where h is the height, w is the width and c is the number of the depth channels, is forwarded on teacher and student models, generating $F_t^l(J)$ and $F_s^l(J)$ patches. For each lth layer, the student and teacher patches combine to obtain an anomaly map $\Omega^l(J)$ using Equation 4.15. As all $\Omega^l(J)$ have different dimensions, the anomaly maps upsample to the input image dimension applying bilinear interpolation. The final anomaly map $\Omega^l(J)$ is obtained by a product of upsamples anomaly maps as in Equation 4.18.

$$\Omega(J) = \prod_{l=1}^{L} \text{Upsample}\Omega^l(J) \tag{4.18}$$

In this work, the STFPM will use the same network architecture as the authors, the ResNet-18. The layers chosen to produce the pyramid feature extraction scheme are the conv2_x, conv3_x, and conv4_x for teacher and student networks. Using the same training parameters as the authors, the STFPM should obtain better results than SSIM with the MVTec dataset, as shown in the paper.

Another advantage of this technique shown in the author's work is that STFPM presents good results even when trained with only 5% or 10% of the data. It is a great advantage because it works well when there is not a large dataset. It overcomes the Spade technique [73] when training with few images. Also, STFPM outperforms the student-teacher technique presented in Bergmann work [50].

4.4 Reverse Distillation From One-Class Embedding

The knowledge distillation approach achieves good results in anomaly detection problems [53]. Even though STFPM overcame the previous solutions, Deng and Li [54] argue that using similar architecture for teacher and student models hinders the diversity of anomaly representations. From the previous knowledge distillation solutions, using only non-defective samples to train the student model, it is expected that the student produces distinct anomaly maps from the teacher when the input images are defective. However, the authors argue that previous solutions do not produce distinct anomaly maps because the teacher and student models have the same architecture and data flow, producing similar outputs. The use of smaller student architectures can solve this problem. However, as seen previously, using smaller networks result in crucial information loss to detect anomalies. So, the authors propose a new knowledge distillation approach using a teacher encoder and student decoder, creating a new knowledge transfer method called

reverse distillation. They denominated as Reverse Distillation from One-Class Embedding, in this work, it will be abbreviated as Reverse. Also, they present a one-class bottleneck embedding (OCBE) to produce a latent space that represents the entire image information.

4.4.1 Knowledge Distillation versus Reverse Knowledge Distillation

Knowledge distillation and Reverse knowledge distillation are two distinct novelty detection approaches. Figure 27 presents a visual comparison between both approaches.

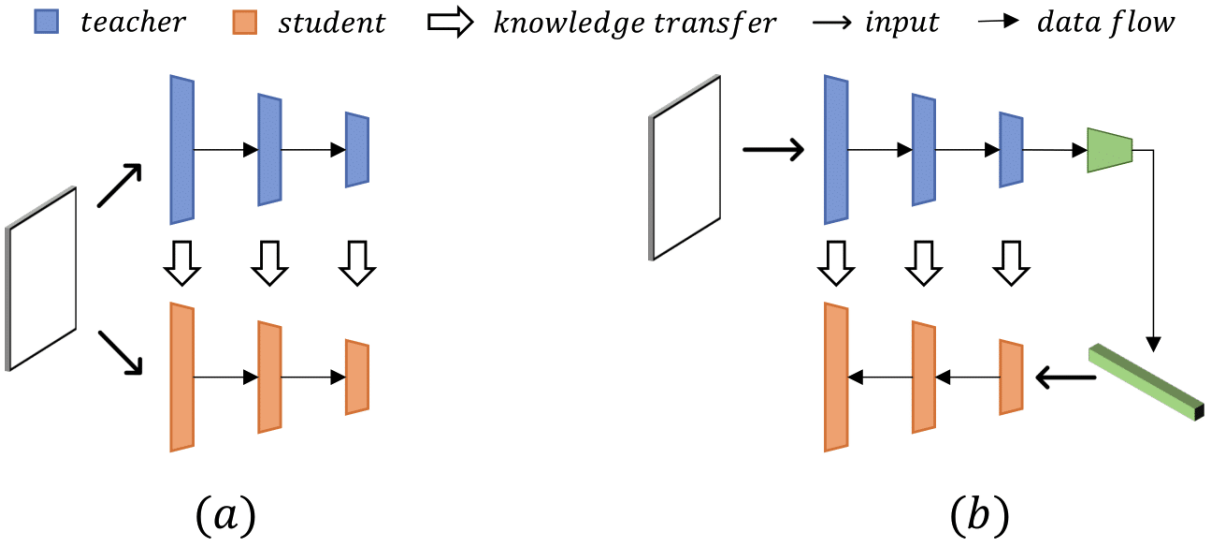


Figure 24: Knowledge distillation (a) and reverse knowledge distillation (b). Source: [54]

As seen in the previous figure, the author's approach did not use the input image as student input. Instead, this approach uses a latent representation produced by a trainable OCBE to reduce the teacher features and preserve normal pattern information, excluding anomalies. Also, in this knowledge distillation system, the encoder and decoder have an inverse architecture. In this way, the reverse distillation architecture has a pre-trained encoder E, an OCBE, and a decoder D. The encoder follows the previous teacher-student architectures, using pre-optimized parameters from a pre-trained network trained with ImageNet dataset. The features are extracted from encoder and go through the OCBE module, composed of a multi-scale feature fusion (MFF) block and one-class embedding (OCE) block. The OCBE produces a latent representation ϕ of the extracted features maps, combining the information, ensuring that the crucial info is not lost.

As presented in the Figure 25, the MFF combines the multi-scale features, downsampling the features

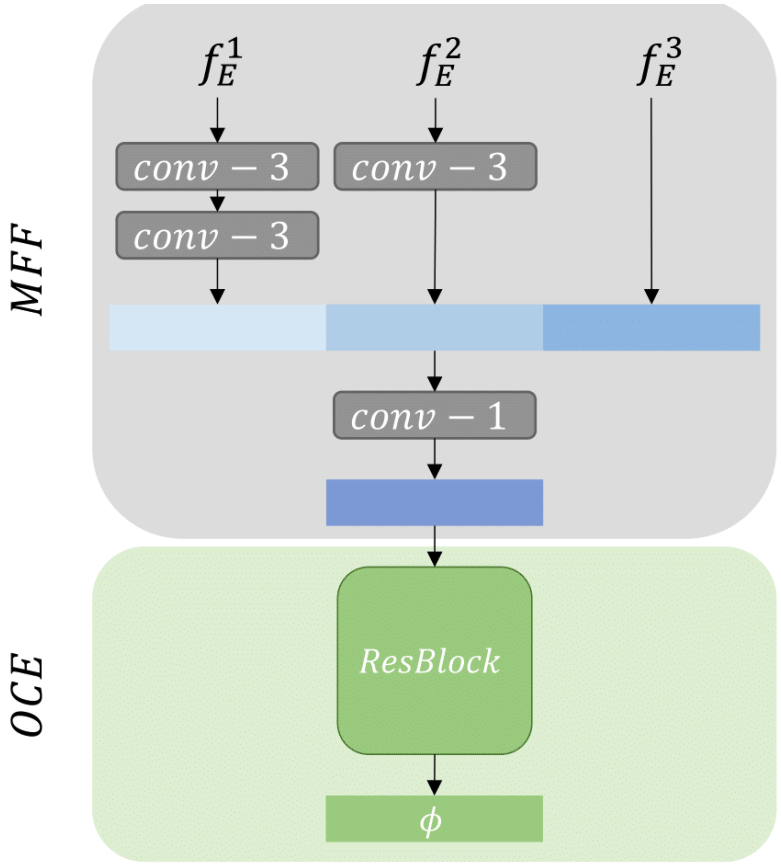


Figure 25: One-Class Bottleneck Embedding Architecture. Source: [54].

using 3 × 3 convolutional layers, followed by batch normalization and ReLU activation function. Thereafter, a 1×1 convolutional layer and a batch normalization with relu activation function are applied. The MFF result goes to OCE where ResBlock trains to collect discriminative information, reducing the result to a compact bootleneck.

After that, the D model tries to mimic the teacher's behavior by restoring the E model's features in different scales to obtain f_D^k features. In the end, the resulting features from both networks combine to estimate the loss and allow the OCBE and D models to learn. In the Figure 26, the architecture workflow is presented.

This technique uses L^t anomaly-free images, $L^t = \{L_1^t, ..., L_n^t\}$, to find the elements that do not follow the normal distribution. During the training, given an input element $I \in L^t$, the teacher extract multi-scale features $f_E^k \in \mathbb{R}^{C_k * H_k * W_k}$ where C_k, H_k and W_k denote the number of channels, height and width of the k^{th} layer. These features follow to the OCBE, producing the latent space that is reconstructed with the decoder D model, originating $f_D^k \in \mathbb{R}^{C_k * H_k * W_k}$. In the end, when f_E^k and f_D^k are estimated, they

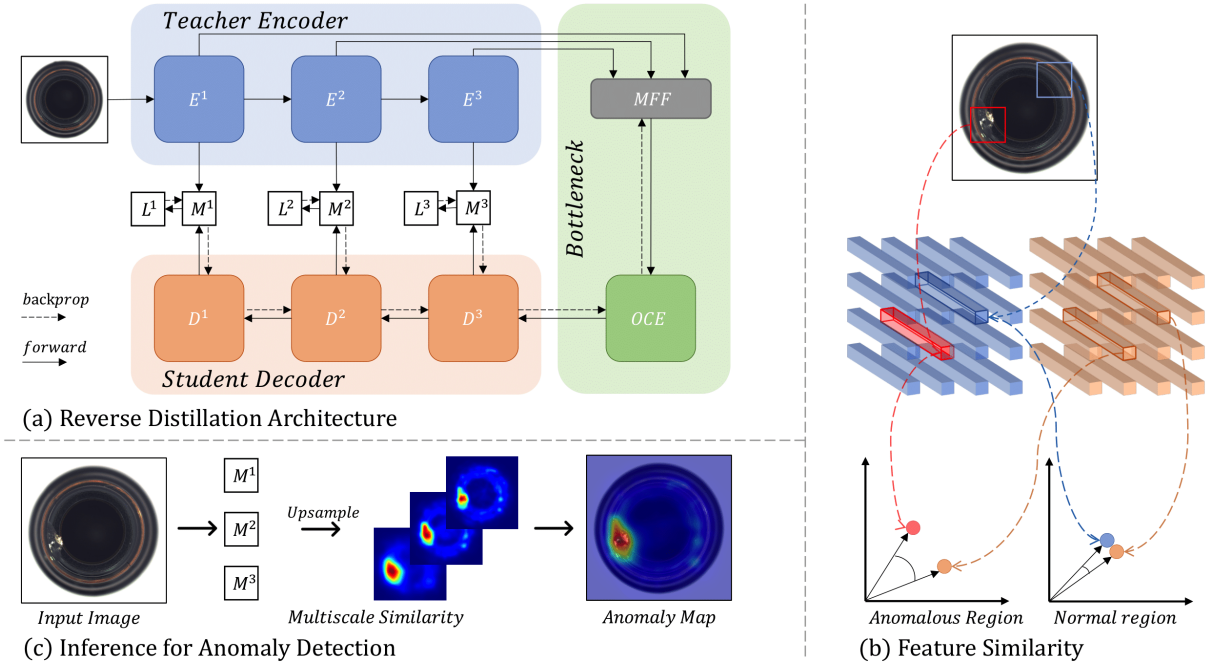


Figure 26: Reverse Distillation from One-Class Embedding Architecture. Source: [54].

are combined to obtain M^k anomaly maps using cosine similarity (Equation 4.19). After that, the M^k anomaly maps upsample using bilinear interpolation. Then, the final score map is created as the pixel accumulation of all M^k.

$$M^k(h,w) = 1 - \frac{(f_E^k(h,w))^T * f_D^k(h,w)}{\left\| f_E^k(h,w) f_D^k(h,w) \right\|} \tag{4.19}$$

To train OCBE and decoder D model, the cosine similarity is used as loss function because it allows to capture the relation between low-dimensional information along the channel axis and obtain a 2D anomaly map $M^k \in \mathbb{R}^{C_k * H_k * W_k}$. Taking into account the distillation of multi-scale knowledge, the function of scalar loss for the optimization of the student model is obtained by the accumulation of multi-scale anomalies maps, using Equation4.20, where a large pixel value indicates the presence of a anomalous pixel.

$$L_{KD} = \sum_{k=1}^{K} \left\{ \frac{1}{H_k W_k} \sum_{h=1}^{H_k} \sum_{w=1}^{W_k} M^k(h,w) \right\} \tag{4.20}$$

Using this approach, in the inference phase, the teacher E should capture the anomalous feature in the defective samples, and the student D should fail in the extraction of these anomalous features.

This happens because the student has a reverse architecture and OCBE did not learn to compress the anomalous features. This will result on student features reconstruction failures. In other words, D model should generate distinct maps from the E when the sample is defective.

The reverse distillation has two main advantages: non-similarity structure and compactness embedding. The use of reverse networks in the teacher-student systems gives non-similarity to the architecture because the teacher uses downsampling filters, and the student uses upsampling filters, avoiding the production of similar teacher and student outputs. The use of OCBE gives compactness embedding because it reduces the teacher features, avoiding the propagation of perturbations to the student model. This fact helps the production of distinct features between the teacher and student model, which improve anomaly detection.

In the authors experiments, they explore distinct backbones architectures: ResNet18, ResNet50, and Wide-ResNet50 and obtain the best results with ResNet18. Also, they prove that their OCBE approach outperforms the results obtained using f_E^k features directly and using the f_E^k features combined with OCE model. The final results from Reverse knowledge distillation using OCBE research shows that is more efficient than the previous solution, consuming less memory and consecutively improving the inference time. This is due to the fact that it depends solely on a supplementary CNN model.

4.5 Discriminatively Trained Reconstruction Anomaly Embedding Model

Until this moment, the leather detection problem is treated as an anomaly detection problem. However, Discriminatively Trained Reconstruction Anomaly Embedding Model (DRAEM) [55] authors fits this problem as a surface anomaly detection problem. The authors argue that anomaly detection problems consider anomalies when the entire images differ significantly from the anomaly-free images used during the training. So, they describe it as a surface anomaly detection problem when anomalies occupy a small image area, and most of the regions follow the normal pattern.

As seen before, novelty detection solutions can classify into two types: Reconstruction-based and Embedding Similarity-based. The Reconstruction-based techniques have been explored Autoencoders as in Section 4.1 and using GAN [74]. The use of reconstruction techniques allows for finding anomalies

by comparing the original image with the reconstructed image. However, the reconstruction techniques have a problem. Even if they only learn to reconstruct normal samples, they perform well in anomalous reconstructions. Difficulting the localization of the anomalies when comparing pixel per pixel the original and the reconstructed image.

Usually, anomaly detection problems are solved using segmentation techniques. However, the state-of-the-art in the segmentation field are the supervised techniques that require an annotated dataset. As in this problem there is no available supervised dataset, the only solution is to create an artificial supervised dataset, producing anomalous samples and the respective masks. In this way, segmentation supervised networks can discriminate if a pixel is anomalous or is not. However, this leads to overfitting synthetic defects, creating a decision boundary that performs badly in the real anomalies.

The authors propose that the discriminative model overfitting can reduce by training the model, combining the synthetic image and reconstructed samples obtained from a reconstruction-based technique. In this way, the discriminative model does not overfit because it learns the normal patterns of a good sample (reconstructed image) and the anomalous patterns (synthetic image). Following this idea, the authors propose DRAEM, a discriminative technique trained using artificial generated anomalous samples, that do not represent the leather defects domain. DRAEM architecture has a reconstructive and a discriminative sub-networks, presented in the Figure 27. The reconstructive sub-network detects anomalies and reconstructs the anomalies as non-defective regions and preserves the non-defective areas. The discriminative sub-network uses the reconstructed image concatenated with the input image and learns to detect the defective area, producing an anomaly map.

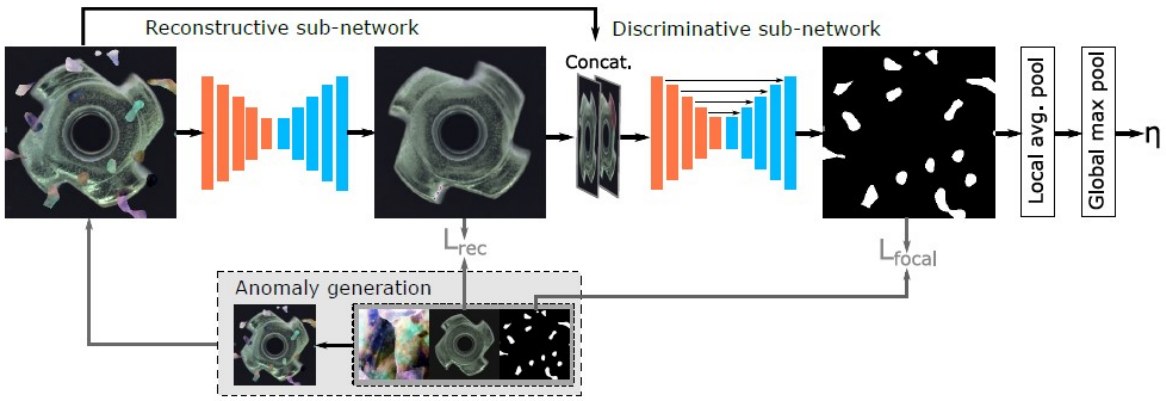


Figure 27: Reverse Distillation from One-Class Embedding Architecture. Source: [54].

4.5.1 Simulated anomaly generation

This method uses Perlin technique [75] to generate artificially anomalous samples from a non-defective dataset. Using a dataset $D = I_1, I_2, ..., I_n$, for each image sample $I_k \in \mathbb{R}^{w*h*c}$, where h is the height, w is the width and c is the number of the depth channels, there are generated an I_a image and a M_a mask.

As presented in the Figure 28, the anomalous images use a Perlin noise generator to capture a variety of anomaly shapes. From the textures dataset [76], a P image is selected and thresholded to obtain an M_a anomaly mask. After that, another image A from the textures dataset is sampled to define the anomalies texture. Image A is augmented using three distinct random augmentation functions to change the posterize, sharpness, solarize, equalize, brightness, color, and auto-contrast. Thereafter, the image A masks with the M_a and combines with the image I to create I_a anomalous samples.

$$I_a = \overline{M}_a \odot I + (1 - \beta)(M_a \odot I) + \beta(M_a \odot A) \tag{4.21}$$

The I_a artificial images is created following the Equation 4.21, where $\overline{M}_a$ is the inverse of M_a, $\odot$ is the element-wise multiplication operation and β, a random value $\in [0.1, 1.0]$, is the opacity parameter in the combination of A and I. The final result of this process is a clean image I, an anomalous artificial sample I_a, and the respective binary mask M_a that will train DRAEM.

4.5.2 Reconstructive sub-network

The reconstructive sub-network has an encoder-decoder architecture used to generate a new image in a non-defective distribution. The encoder-decoder uses an anomalous sample artificially generated and converts the defective patterns into a clean pattern. The goal of the reconstruction network is to reconstruct the original image I, without defects, from the artificial sample I_a. Using the result of the reconstructive sub-network, the SSIM loss and MSE loss are estimated using 4.22 and 4.23 respectively.

MSE loss is always an option in segmentation problems. However, MSE loss does not relate the independence between neighbours pixels. So a solution followed by the authors is use the MSE loss combined with the SSIM loss.

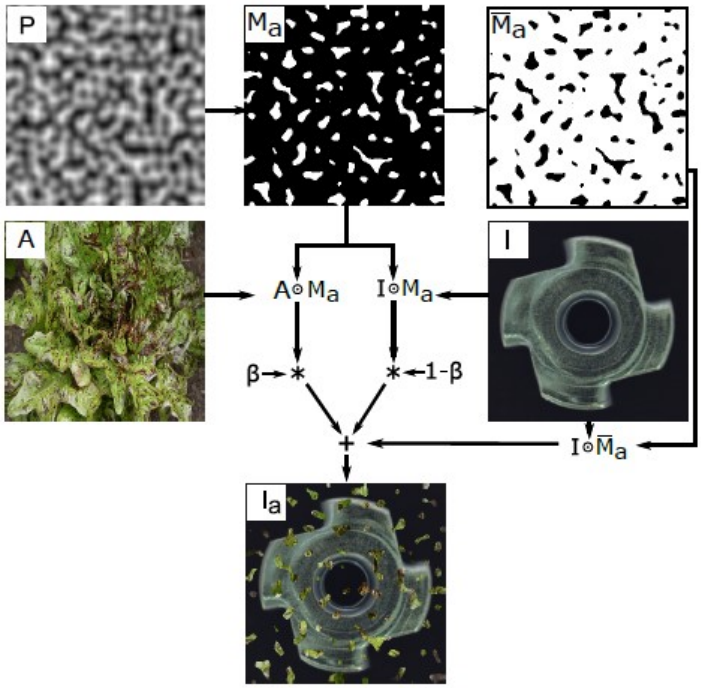


Figure 28: Simulated anomaly generation. Source: [55]

$$L_{SSIM}(I, I_r) = \frac{1}{N_p} \sum_{i=1}^{H} \sum_{j=1}^{W} 1 - SSIM(I, I_r)_{(i,j)} \tag{4.22}$$

$$L_{MSE}(I, I_r) = \frac{1}{N_p} \sum_{i=1}^{H} \sum_{j=1}^{W} 1 - MSE(I, I_r)_{(i,j)} \tag{4.23}$$

In the Equations 4.22 and 4.23, H and W are the height and width of image I. N_p is equal to the number of pixels in I. Ir is the reconstructed image output by the network. In the end, the two loss measures are combined to estimate the reconstruction loss using 4.24.

$$L_{rec}(I, I_r) = L_{SSIM}(I, I_r) + L_{MSE}(I, I_r) \tag{4.24}$$

In this dissertation, to implement DRAEM, it is necessary to create a reconstructive sub-network AE architecture. The architecture chose was available in their public repository [1]. The AE is composed by two networks, the encoder and decoder architecture, presented in the Table 3 and 4.

[1] https://github.com/VitjanZ/DRAEM/

Table 3: AE - Encoder Architecture.

Layer	Output size	Kernel	Parameters Stride	Padding
Input	256*256*3	-	-	-
Conv1	256*256*128	3*3	1	1
Conv2	256*256*128	3*3	1	1
MaxPool1	128*128*128	2*2	2	0
Conv3	256*128*128	3*3	1	1
Conv4	256*128*128	3*3	1	1
MaxPool2	64*64*256	2*2	2	0
Conv5	64*64*512	3*3	1	1
Conv6	64*64*512	3*3	1	1
MaxPool3	32*32*512	2*2	2	0
Conv7	32*32*1024	3*3	1	1
Conv8	32*32*1024	3*3	1	1
MaxPool4	16*16*1024	2*2	2	0
Conv9	16*16*1024	3*3	1	1
Conv10	16*16*1024	3*3	1	1

Table 4: AE - Decoder Architecture.

Layer	Output size	Kernel	Parameters Stride	Padding
Input	16*16*1024	-	-	-
Upsample1	32*32*1024	-	-	-
Conv1	32*32*1024	3*3	1	1
Conv2	32*32*1024	3*3	1	1
Conv3	32*32*512	3*3	1	1
Upsample2	64*64*512	-	-	-
Conv4	64*64*512	3*3	1	1
Conv5	64*64*512	3*3	1	1
Conv6	64*64*256	3*3	1	1
Upsample3	128*128*256	-	-	-
Conv7	128*128*256	3*3	1	1
Conv8	128*128*256	3*3	1	1
Conv9	128*128*128	3*3	1	1
Upsample4	256*256*128	-	-	-
Conv10	256*256*128	3*3	1	1
Conv11	256*256*128	3*3	1	1
Conv12	256*256*128	3*3	1	1
Conv13	256*256*3	3*3	1	1

4.5.3 Discriminative sub-network

The discriminative sub-network uses a concatenation between the anomalous image I_a and the reconstructed image I_r to obtain an anomaly score map. For this task, the authors use U-Net [25] to learn to distance between a clean pixel and an a anomalous pixel. With this architecture, the expected output is an anomaly score map M of the same size as I, attributing to each pixel the probability of being defective. During the training phase, the M map and the M_a mask will be used to estimate the Focal loss [77]. In the end, the reconstruction loss and the focal loss combine to estimate the DRAEM loss using Equation 4.25.

$$L(I, I_r, M_a, M) = L_{rec}(I, I_r) + L_{focal}(M_a, M) \tag{4.25}$$

Chapter 5

Experiments

In this chapter, there are presented the training and evaluation settings to execute a set of performance evaluation experiments using the Novelty Detection techniques on the leather defects detection problem.

5.1 Datasets

For this work, the experiments use two distinct datasets, one is the MVTec AD dataset [61], and the other is a self-made dataset created using Neadvance captured images.

MVTec AD is a dataset for anomaly detection methods. It has been used in the most recent researches as a benchmark image dataset. It contains 5000 images divided into fifteen distinct objects and categories, including leather samples. The MVTec leather dataset has 235 free defects images for training and 123 images for testing. The testing images have 42 non-defective images, 19 with color defects, 19 with cut defects, 17 with fold defects, 19 with glue defects, and 17 with poke defects. Each testing image has a respective ground truth mask. This dataset only has 2.7% anomalous pixels in the test set. Figure 29 presents an example of each defect type and a good sample, side by side with the respective ground truth mask.

The second dataset was developed by Neadvance. Neadvance is, according to their website, a "Machine Vision and Intelligent Systems company which develops computer vision and artificial intelligence

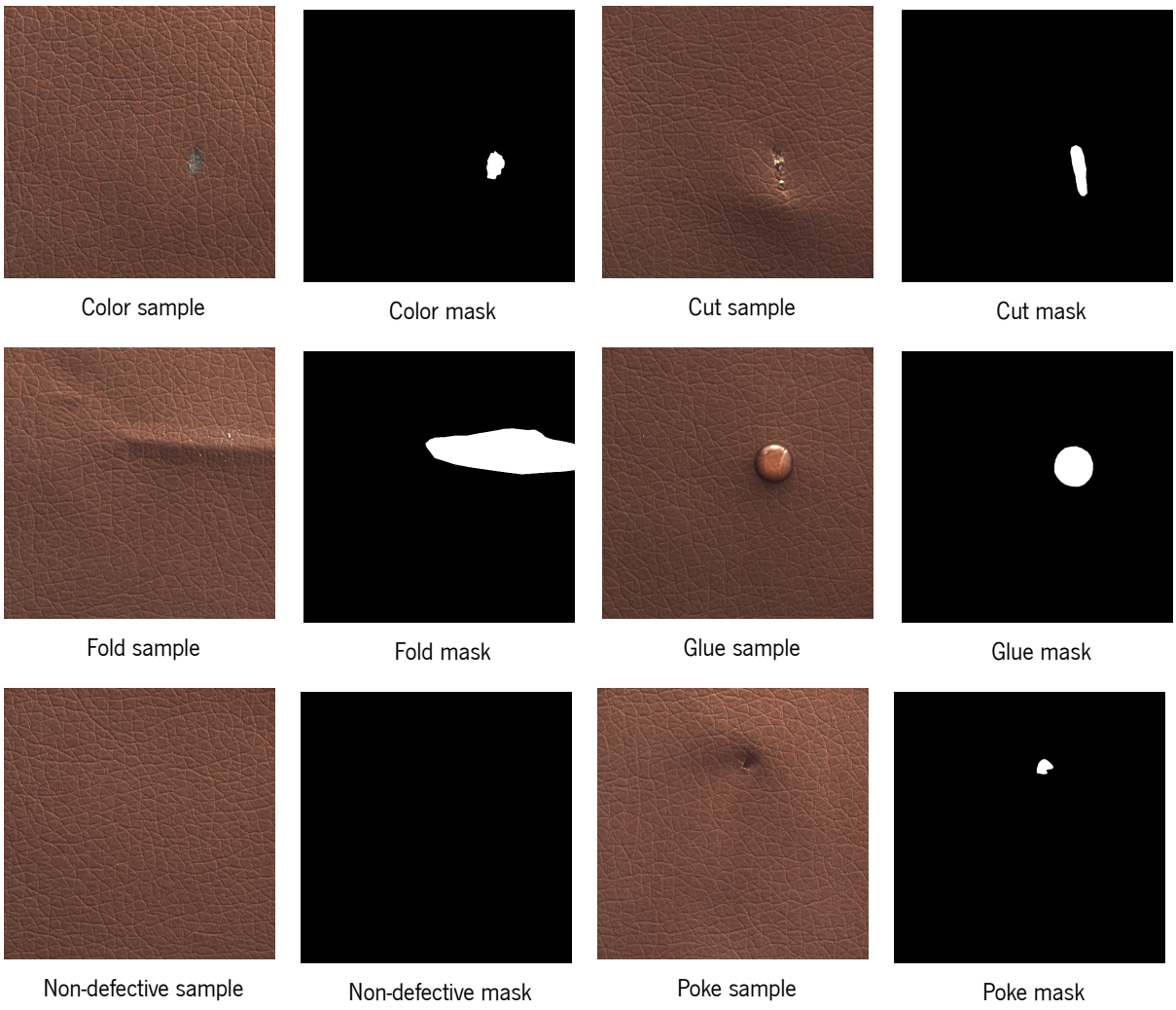


Figure 29: Leather MVTec AD samples and masks.

solutions and platforms for the digital and industrial revolution worldwide". Their dataset has 211 defective images, 40 with cut defects, 47 with hole defects, 52 with line defects, and 82 with wrinkle defects. The initial dataset does not have non-defective samples. So, the clean regions from the defective samples were cropped to obtain non-defective samples. For training and testing, 260 and 42 regions were cropped, respectively. The testing dataset includes 42 non-defective and 211 defective samples mentioned before. Figure 30 presents an example of each defect type and a good sample, side by side with the respective ground truth mask. For this dataset, it is crucial to highlight that these defects are more difficult to detect than MTVEC defects.

In this dissertation, to evaluate the techniques presented in Chapter 4, it is necessary to choose the evaluation metrics to use. For this problem, it is important to evaluate the techniques in two tasks, defects

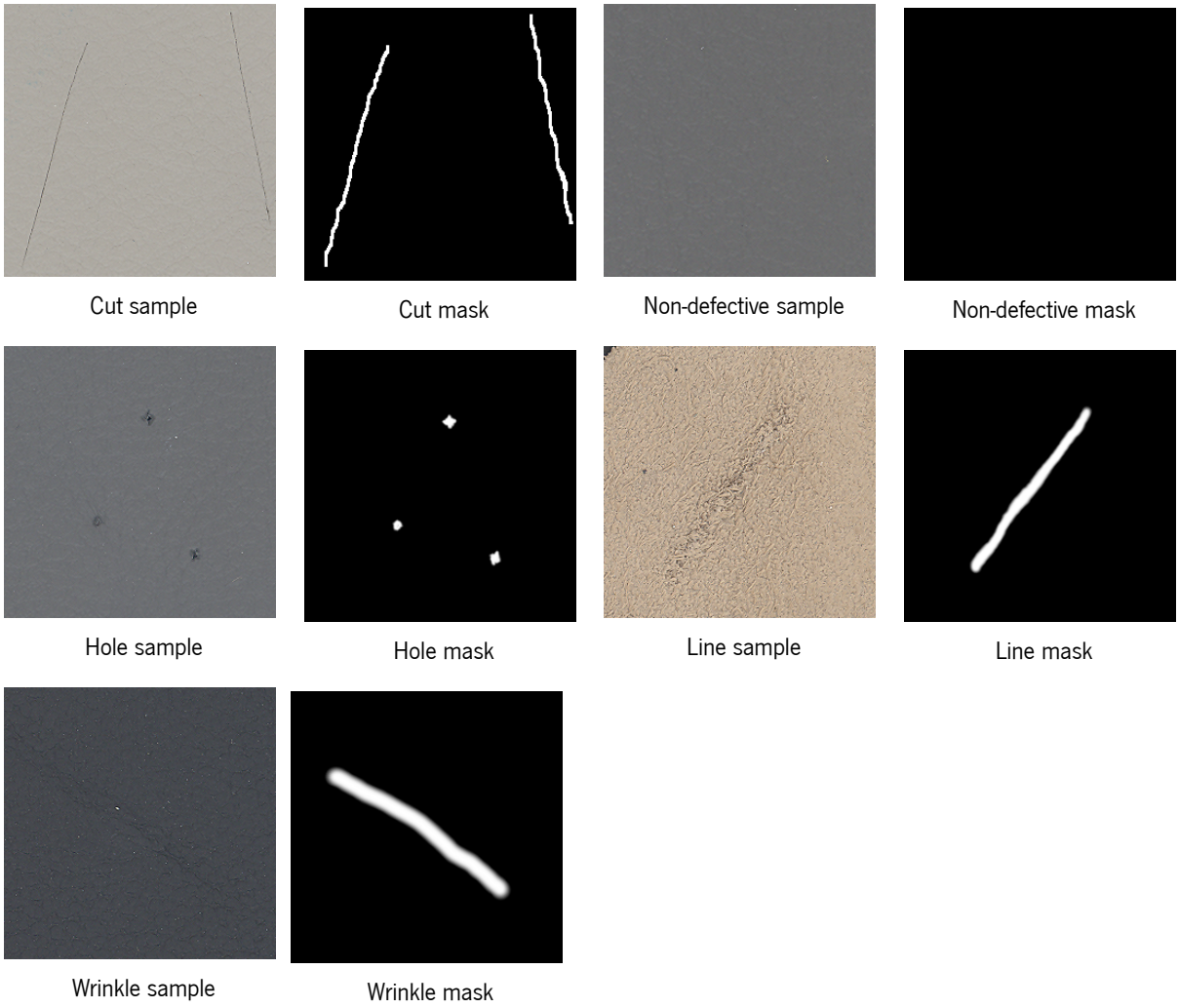


Figure 30: Neadvance samples and masks.

localization (segmentation) and detection (binary classification).

5.2 Segmentation metrics

As the output of the techniques is an anomaly score map, indicating the probability of each image pixel being anomalous, and the ground truth is a binary mask, it is necessary to use a threshold on the anomaly score map to evaluate the differences between the thresholded anomaly map and the ground truth mask.

A pixel prediction classifies as either a true positive (TP), false positive (FP), true negative (TN), or false negative (FN). A pixel prediction is TP when it is anomalous, and predicted as anomalous. A pixel prediction is FP when it is clean, and predicted as anomalous. TN if the pixel is clean and predicted as clean. And FN if the pixel is anomalous and predicted as clean.

One of the most used metrics in segmentation evaluations is the Intersection over Union (IoU) score (Equation 5.1), also known as the Jaccard index. The IoU estimates the intersection area of two regions divided by the area of two regions united. In this case, it divides the number of truly predicted anomalous (TP) pixels by the number TP plus the number of FP and FN.

$$IoU = \frac{TP}{TP + FP + FN} \tag{5.1}$$

The advantages of IoU metric are that they are easy and effective to calculate. However, considering every pixel as completely independent introduces a bias to large anomalous areas.

On the other hand, instead of considering every pixel as independent, the segmentation performance can be averaged over each ground truth mask component. This technique is substantially time-consuming. However, it is useful to consider the localization of smaller anomalies equally crucial as the localization of larger anomalies. In this work, the evaluation of each component is done using the per-region overlap (PRO), which is a benchmark metric for anomaly segmentation techniques.

To use this technique, each image i is divided into its k connected components. For each k component, the $C_{i,k}$ marks the defective pixels of image i and P_i marks the pixels predicted as defective. $C_{i,k}$ and P_i estimate the PRO using the Equation 5.2.

$$PRO = \frac{1}{N} \sum_{i} \sum_{k} \frac{|P_i \cap C_{(i,k)}|}{|C_{(i,k)}|} \tag{5.2}$$

The PRO averages the ratio between the intersection of P_i and $C_{i,k}$ with $C_{i,k}$ for all N ground truth components of all images used in the testing evaluation.

The evaluation metrics presented, until this moment, require a threshold to convert the anomaly score map into a binary mask. Finding the best threshold to classify a pixel as defective is not easy and a wrong threshold can ruin the evaluation of the techniques. So, it is also crucial to evaluate the techniques using a threshold-independent metric.

In ML, TPR (True Positive Rate) (Equation 5.3), also known as Recall, and FPR (False Positive Rate) (Equation 5.4) are simple metrics used to evaluate the results of the predictions. Using the TPR and FPR, the goal is to obtain a high TPR and a low FPR.

$$TPR = \frac{TP}{TP + FN} \tag{5.3}$$

$$FPR = \frac{FP}{FP + TN} \tag{5.4}$$

The threshold agnostic metric used, in this dissertation, to combine TPR and FPR is the area under the receiver operating characteristic curve (AUROC). The AUROC uses several thresholds $\in$]0,1[to binarize the scores maps and estimate the TPR and FPR for each threshold. In this way, it is possible to obtain the ROC curve that indicates the probability of TPR be higher than an FPR. In every experiment, the goal is to maximize the AUROC, as illustrated in Figure 31 by the orange curve. Also, like AUROC combines TPR and FPR, it works well when the dataset is not balanced, as in this dissertation.

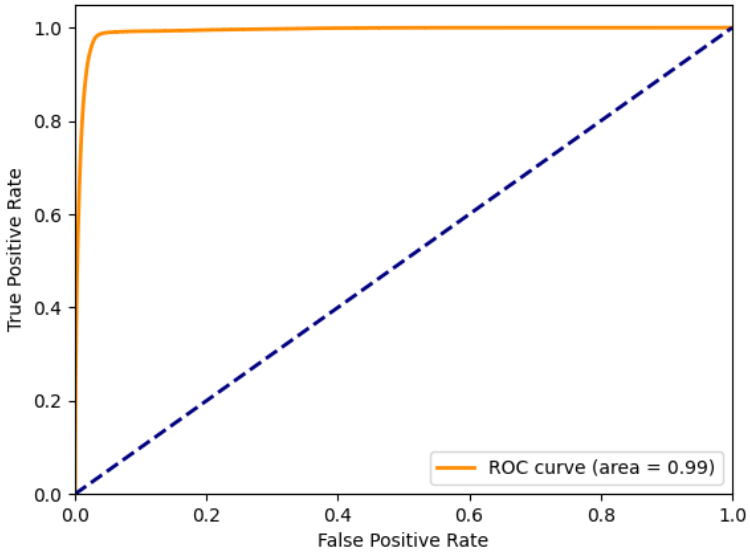


Figure 31: ROC curve example

5.3 Detection metrics

Beyond the segmentation metrics, in this work it is important to evaluate the defects detection performance. As in segmentation, the performance is evaluated using dependent and non-dependent threshold metrics. The chosen metrics are the F1-Score and the AUROC.

In this problem, it is important to reduce both FP and FN. The FP because if a leather sample is detected as defective and it is not, the leather will lose value and will not be used to produce leather goods, losing money. In other hand, the FN has to be reduced because it classifies a defective sample as clean, producing defective leather goods.

To ensure a reduced number of FP, the Precision metric (Equation 5.5) can be used, and to ensure a reduced FN number, the Recall can be used. A known metric capable to combine these two metrics is the F1-Score. To use F1-Score it is necessary to threshold the anomaly score maps, converting them into binary masks. So, if a binary mask has at least one defective pixel, it is considered defective. Otherwise, it is considered not defective. After that, the F1-Score can be estimated using Equation 5.6.

$$Precision = \frac{TP}{TP + FP} \tag{5.5}$$

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.6}$$

Like in the segmentation evaluation, it is crucial to evaluate the techniques using a threshold agnostic metric, in case the threshold is not well estimated. So, the AUROC is the metric chosen again. In this way, it is necessary to obtain a single value from the entire anomaly map to compare this value with the ground truth label.

As the anomaly score map indicates the probability of the image pixels being defective, for each anomaly map, the value selected to work as the predicted label is the maximum value of the anomaly score map. In other words, the value of the pixel with the highest probability of being defective.

5.4 Thresholds

To obtain a binary anomaly map from the score map, it is necessary to estimate a threshold. However, selecting a threshold value without knowing the defects that will appear in the future is not an easy task. In this way, in the extended version of Bergman et al. work [72], there are defined five different ways to estimate the binary threshold: maximum threshold, p-Quantile threshold, k-sigma threshold, and max area threshold.

As seen in Bergman's work, all threshold estimation techniques allow a certain amount of false positives. However, the p-Quantile threshold attempt to fix the false positive rate at one percent. So, in Bergman's work, the p-Quantile was used to estimate the threshold (T1). The p-Quantile selects a threshold such that a percentage p of the normal distribution pixels classify as anomalous (outlier). For this method, the p-value chosen was 99, the same used by Bergman, and it is the standard value in outlier detection. To estimate T1, it is necessary to use a set of non-defective images as the normal distribution.

Instead of using only non-defective images, the authors of CFLOW decided to select the threshold using defective and non-defective images. In this way, they estimate two distinct thresholds, one to maximize the F1 Score for the defects segmentation task (T2) and the other for the defects classification task (T3). For both tasks, the Precision-Recall curve, illustrated in Figure 32, is estimated to obtain the Precision, Recall, and threshold results.

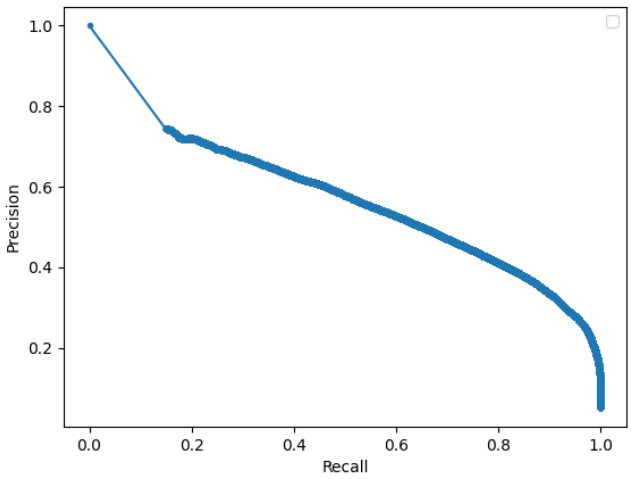


Figure 32: Precision Recall Curve.

After that, for each Recall and Precision values the F1 score values are estimated using Equation 5.6.

Then the index j of F1 Score maximum value is estimated and the threshold used will be the jth Precision-Recall threshold. The Figure 33 represents the F1 Score values versus the respective thresholds.

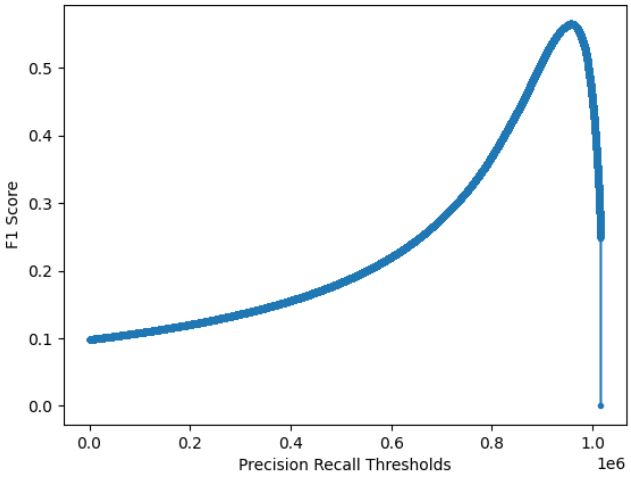


Figure 33: F1 Score vs Precision Recall thresholds.

5.5 Experiments definition

In this dissertation, to evaluate the novelty detection techniques, there are proposed three distinct experiments using the Novelty Detection techniques:

- **Experiment 1** - Train and evaluate the techniques with MVTEC and with Neadvance;
- **Experiment 2** - Train with MVTEC and evaluate with Neadvance, and vice versa;
- **Experiment 3** - Train the techniques with both datasets and evaluate for MVTEC and Neadvance;

The first experiment is a baseline experiment to evaluate the ability of the Novelty Detection techniques detects defects with the same leather pattern as used during the training. The second experiment is performed to evaluate the generalization ability of these techniques. Check if the techniques can detect defects from samples with different patterns than the used during the training. The third experiment was performed to verify if a larger dataset, combining the color patterns from both datasets can obtain better results than Experiment 1.

These experiments were performed following the next settings. In the training process, the optimizing functions and the loss functions are the same as presented by the techniques authors. At the beginning of the training process, the training dataset was split into three, 70% for training, 15% for validation, and 15% to estimate the T1. Also, the testing dataset was split in two parts, 15% to estimate T2 and T3, and 85% to evaluate the techniques.

The techniques architectures train to fit into the training split, and the validation split works in the model validation, which is useful in the early stopping and in the model checkpoints. In these experiments, to avoid overfitting, early stopping is applied. Early stopping avoids the model overfitting into training data. In this way, if the validation loss does not improve along 20 epochs, the training process stops. In ML it is crucial saving distinct architecture checkpoints during the training. After each training epoch, if the actual model is better than the last checkpoint model, the actual model is saved as the new checkpoint. To evaluate if the new model is better than the previous checkpoint, it is necessary to evaluate the loss reduction. This step could use the training split, in this way, it would evaluate if the model fits in the training data, which can cause overfitting. So it is necessary to validate the model using a distinct data split.

The configurations for the techniques training are the number of epochs, batch size, learning rate, and image size. So, in this work, the techniques train during 300 epochs with a 0.01 learning rate using reshaped images of 256*256 pixels. As the techniques presented have distinct architectures, each has distinct memory requirements. So, each one have a distinct batch size.

As mentioned before, in ML the hardware requirements are critical. So, in these experiments, the hardware configuration of the machine used is Intel(R) Xeon(R) CPU E5-2680 V4@2.4GHZ and NVIDIA GeForce GTX 1080Ti 11 GB.

Chapter 6

Results

In this chapter, there are presented the localization and detection tasks results for the experiments presented before. As Experiment 1 works as a baseline evaluation, it is presented not only the quantitative results but also the qualitative and complexity results. For the remaining of the experiments, only the quantitative results obtained from the localization and detection metrics are presented. In the end, it is presented a discussion about the obtained results.

6.1 Experiment 1

The results are divided into quantitative, qualitative and complexity results. The quantitative results present the results of evaluation metrics for both datasets. The qualitative results present the image results and succinct analysis of the defects detection results. In the end, the complexity results present which techniques require less computational power to work.

6.1.1 Quantitative results

Table 5 compares the localization results of all techniques with the MVTEC dataset. As IOU and PRO metrics depend on a good threshold estimation, this analysis starts by comparing the AUROC results. As expected, the MSE AE performs worst than the SSIM AE. In the state-of-the-art, the MSE AE has 75% AUROC and the SSIM AE 78% AUROC. In this experience, they obtain 83.49% AUROC and 94.18% AUROC

respectively. This can be justified by the increasing number of epochs and for the pos-processing applied. As the used architecture classifies the borders pixels as defective, a clean border procedure [1] was applied . The CFLOW outperforms all the other techniques with 99.57% AUROC. Nonetheless, STFPM (98.91% AUROC) and Reverse (99.31% AUROC) obtain great results too. Analyzing the IOU results, excluding SSIM AE, all the other techniques perform better when using a segmentation optimized threshold T2. In this case, DRAEM outperforms all the other techniques with 38.97% IOU using T2. In the PRO evaluation, the T1 threshold outperform the T2 threshold, except when using the DRAEM technique. These results indicate how critical is the threshold estimation process. Using a distinct threshold, the results can vary a lot. As seen with DRAEM results, T2 achieves better results for both metrics, IOU and PRO.

Table 5: MVTEC dataset localization results.

	Metric (%)				
Model	**IoU T1**	**IoU T2**	**PRO T1**	**PRO T2**	**AUROC**
MSE AE	1.63	14.24	58.39	24.40	83.49
SSIM AE	31.33	29.41	47.24	42.01	94.18
CFLOW	7.28	22.54	99.98	95.65	99.57
STFPM	4.39	22.32	99.42	43.16	98.91
Reverse	1.23	21.49	99.99	79.81	99.31
DRAEM	1.49	38.97	2.34	75.49	94.80

As expected, all techniques reach the same performance as in their respective original researches. So, it is interesting to analyze if the techniques reach the same performance with a distinct dataset. In this way, Table 6 presents the localization results with the Neadvance dataset. In terms of AUROC, the results are lower. However, CFLOW, STFPM and Reverse still outperform the other techniques with 72.52%, 71.40%, 74.17% of AUROC respectively. Also, in this table, the MSE AE still has the worst AUROC result with 56.85%. With Neadvance dataset, the threshold-dependent metrics have worse results than with MVTEC dataset. This means that the predicted masks are very different from the ground truth masks. The IoU values achieve the maximum using the Reverse architecture wih 4.78% IoU using T1 and 9.89% IoU using T2. The PRO metric results follow the IoU results. However, it is notorious the higher values for DRAEM PRO results with 72.37% using T1 abd 82.43% using T2.

[1] https://scikit-image.org/docs/stable/api/skimage.segmentation.html#skimage.segmentation.clear_border

Table 6: Neadvance dataset localization results.

	Metric (%)				
Model	**IoU T1**	**IoU T2**	**PRO T1**	**PRO T2**	**AUROC**
MSE AE	0.48	0.55	2.79	4.79	56.85
SSIM AE	6.24	0.65	12.04	32.63	68.07
CFLOW	1.92	0.01	14.27	17.11	72.52
STFPM	2.18	7.61	8.03	48.26	71.40
Reverse	4.78	9.89	51.71	28.03	74.17
DRAEM	0.01	0.01	72.37	82.43	46.77

After analyzing the localization results, it is crucial to evaluate the detection performance because there are problems where only the detection task is relevant. In this case, the Table 7 and Table 8 show the results of defect detection task. Table 7 shows the results using the MVTEC and for this dataset, the CFLOW outperforms all the other techniques with AUROC=100% and with the F1-Score=95.36% using the optimized threshold T3. The F1-Score results from STFPM with T1 and with T3 are very distinct, using T1 STFPM achieves 46.60% F1-Score and using T3 achieves 86.76%. This significant margin helps to understand the importance of the threshold estimation process. This is also evident in DRAEM results, DRAEM achieves 88.76% F1-Score, close to maximum, using the threshold optimized T3. Using T1 the result is the second worst with 63.06% F1-Score.

Table 7: MVTEC dataset detection results.

	Metric (%)		
Model	**F1-Score T1**	**F1-Score T3**	**AUROC**
MSE AE	83.62	82.84	86.40
SSIM AE	92.13	90.57	91.38
CFLOW	82.02	95.36	100
STFPM	46.60	86.76	97.08
Reverse	68.92	91.67	99.91
DRAEM	63.06	88.76	99.23

In the Neadvance detection results (Table 8), the MSE AE surprises, reaching the highest F1-Score value, 80.24% with the T3. However, with the threshold independent metric, the MSE AE (75.56% T1 and 80.24% T3), outperforms the SSIM AE (53.13% T1 and 55.51% T3) and the DRAEM (1.08% T1 and 4.39% T3). Relatively to the other techniques, the T1 and T3 F1-Score values do not differ a lot. Evaluating the AUROC results of this table, STFPM, Reverse and DRAEM outperforms the other techniques with 77.41%, 77.05%, 77.74% AUROC respectively.

Table 8: Neadvance dataset detection results.

	Metric (%)		
Model	**F1-Score T1**	**F1-Score T3**	**AUROC**
MSE AE	75.56	80.24	68.14
SSIM AE	53.13	55.51	62.95
CFLOW	70.51	70.53	70.79
STFPM	21.78	33.03	77.41
Reverse	70.75	70.75	77.05
DRAEM	1.08	4.39	77.74

6.1.2 Qualitative results

Evaluation metrics are used to quantify the performance of the novelty techniques. Beyond that, it is a good habit, to verify the performance of the techniques using visual examples. In this way, Appendix A presents the results of all techniques. As both datasets are large to be presented in this dissertation, only the results of nine defective images are compared. Each image represents one defect category of both datasets, five belonging to MVTEC and four belonging to Neadvance dataset. For each technique and defect, it is presented the anomaly score map and three images with the predicted boundary (denoted with a red line). Each of these three images obtains the defective mask by applying a distinct threshold (T1,T2,T3) to the anomaly score map. The anomaly score maps presented are grayscale images with pixel values in the [0,1] range. However, grayscale is not the best color map to show the results. So, the anomaly scores maps use Viridis color map from the Matplotlib library[2]. The Viridis colors change between purple, blue, green, and yellow colors. The purple colors represent values close to zero and the yellow colors represent values close to one. This means that the yellow pixels have a high probability of being defective.

In this dissertation, it is important to analyze the anomaly scores maps because they represent the probability of each pixel being defective. So, in a defective image score map, the anomalous regions should have higher scores and non-defective regions should have low scores. If the anomaly score map creates a good margin between the non defective and defective pixels scores, it is easier to estimate a threshold.

Starting by analyzing the results of MVTEC samples, the color sample defect is easily detected by all techniques (Section A.1). There are some cases where the thresholds were not able to detect or segment the color defect. However, for each technique, there is at least one threshold able to detect the defective

[2]https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.viridis.html

region. For these samples, the optimized thresholds perform better than T1. It is important to highlight that the DRAEM anomaly map (Figure 42) is the anomaly map that better distinguishes between the two classes, improving the segmentation task. The results for cut sample (Section A.2) are also good. However, AE does not perform well. In the MSE AE, any threshold was capable to detect or segment the defective region (Figure 43). In the SSIM AE, the thresholds were capable to detect the defects (Figure 44). However, the segmentation of defects does not perform well. The next defect results to analyze are the fold results (Section A.3). For this defect, DRAEM using T3 (Figure 54) was the technique that better segmented the defective region. For this defect sample, T1 performs worse than T2 and T3. STFPM (Figure 52), Reverse (Figure 53), and DRAEM (Figure 54) does not localize the defective region using T1. The glue sample chosen is easy to detect (Section A.4). However, the defective region contain a small region with non defective pixels inside. So, for this sample, a good technique should be able to detect the small clean region inside of the defective area. So, analyzing the anomaly scores maps, only DRAEM gives low scores to non-defective region (Figure 60). In this way, DRAEM is the only technique that produces a mask that does not include the inner clean region. MSE AE is the only technique that does not attribute high scores to the defective region (Figure 55), in this way, it is not able to segment the defect. The remaining techniques create a good defect boundary. The last MVTEC defect sample to analyze is the poke. In general, all techniques produce good anomaly scores maps (Section A.5), excluding the MSE AE (Figure 61). However, it allows to create good masks, just equalized by the T2 and T3 masks from SSIM AE (Figure 62) and DRAEM (Figure 66).

After analyzing the MVTEC results, it is time to evaluate the Neadvance dataset results. Generally, the Neadvance results are very bad. However, there are some cases that are important to mention. In the previous dataset, DRAEM performs very well and it was the technique that produces the best qualitative results. For this dataset, the DRAEM performance is not so good. DRAEM produces anomaly scores maps that attributes high scores to every pixel, difficulting the creation of good masks. After this note, let's start by analyzing the cut defect sample results (Section A.6). The cut defect is visible to the human eye and the SSIM AE, CFLOW, STFPM, and Reverse techniques attribute high scores to the defective region. However, there is any threshold that outperforms the others. T1 was able to segment the cut defect using STFPM (Figure 70). Nonetheless, SSIM AE (Figure 68) and Reverse (Figure 71) produce a better masks using T3. In the case of the hole sample results (Section A.7), SSIM AE produces a good anomaly score map and masks (Figure 74) similar to the ground truth with the three thresholds. CFLOW map (Figure 75) highlights the defective region, but, the thresholds do not produce good masks. This sample is the only one from

Neadvance samples presented where DRAEM localizes the defect (Figure 78). The line sample result maps (Section A.8) are not good to detect the defects. Only, the SSIM AE produces good anomaly score map and mask, using the optimized segmentation threshold T2 (Figure 79). STFPM does not produce a good score map (Figure 82). However, T2 was well estimated, allowing the creation of a good mask. The wrinkle defect is the most difficult to detect, even for a specialized worker. So, the results (Section A.9) are the worst of all the previous samples. The techniques give similar probabilities for all pixels, impeding appropriate mask creation. The Neadvance qualitative performance confirms the quantitative results, where the segmentation tasks perform badly with very low scores in the threshold-dependent metrics.

6.1.3 Complexity results

In real-time solutions, the complexity of each technique is crucial. To work in real-time, the number of predicted frames per second (FPS) has to be high as possible, in other words, the inference time has to be low. Also, it is crucial to use a technique that does not require a lot of memory, in this way, it can be deployed and executed in low-computational power machines.

Analyzing the Table 9, it is clear that techniques require more time to infer a MVTEC sample than a Neadvance sample. This fact is caused by the MVTEC batch loading time, because, MVTEC images have 1024*1024 resolution and the Neadvance images have 256*256 resolution. So, the batch loading spends more time because it has to resize the MVTEC images to the 256*256 resolution. This operation is mandatory because, in these experiences, the architectures were trained using 256*256 images.

In this experience, the MSE AE was the fastest technique with 42.25 FPS using the MVTEC and 85.94 FPS using the Neadvance dataset. It is fastest than SSIM AE because estimating the anomaly score map is more time-consuming than MSE. The SSIM estimates the difference between the original image and the reconstructed image using a sliding window, and that is time-consuming. DRAEM also achieves good results because it does not compare the teacher and student maps to obtain the final anomaly score map. And these operations increase the inference time, reducing the number of FPS.

Table 9: Complexity results (MVTEC/Neadvance).

Model	Inference time	FPS
MSE AE	2.93 / 2.94	42.25 / 85.94
SSIM AE	4.24 / 5.84	29.28 / 43.31
CFLOW	10.29 / 12.63	12.04 / 20.02
STFPM	5.63 / 6.17	21.99 / 40.96
Reverse	10.62 / 16.9	11.66 / 14.89
DRAEM	3.57 / 4.50	34.69 / 56.10

As mentioned in the Chapter 4, the computational complexity of these solutions should be as low as possible. So, Figure 34 illustrates the average GPU memory utilization percentage for each technique. As expected, the MSE AE was the technique that uses less GPU, only 22%. The CFLOW and STFPM have an acceptable GPU utilization with 62% and 75%, respectively.

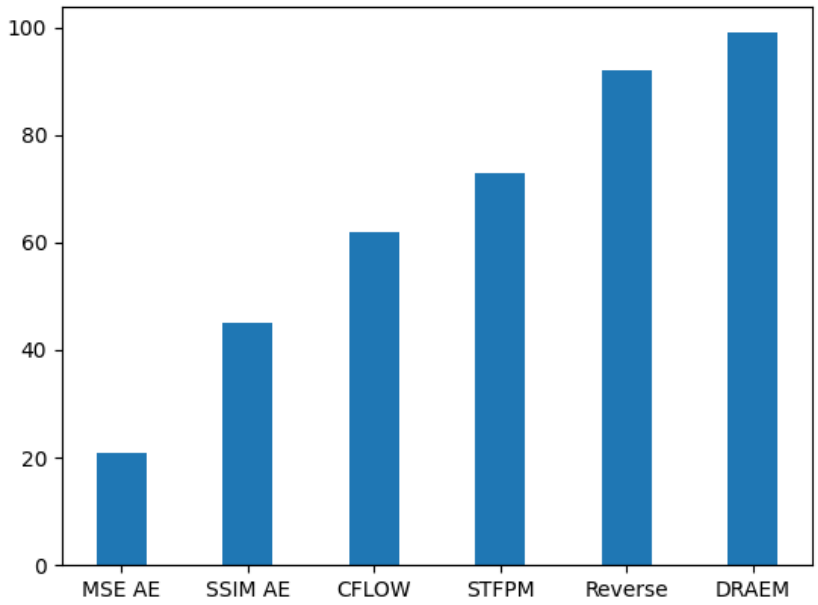


Figure 34: Average GPU memory utilization (%) for each technique.

The Reverse and DRAEM can not be executed in low-computational power machines. The other techniques can be further optimized by increasing the batch size, as they did not fully utilize the available GPU memory. On the other hand, Reverse and DRAEM memory allocation was already very close to the system's limits.

6.2 Experiment 2

This section presents the quantitative results of Experiment 2, evaluating if the Novelty Detection techniques can detect defects on a dataset different from the used during the training. Beyond be analyzed the results, in this section, it will be compared with the results obtained during the Experiment 1. Compare if the MVTEC results using models trained with MVTEC are similar to the MVTEC results using models trained with Neadvance, and vice-versa.

6.2.1 Quantitative results

Before starting to present the results, it is important to highlight that the thresholds used in this experiment were obtained during Experiment 1. So for the MVTEC results using Neadvance models, the thresholds T1, T2, and T3 used were obtained from Neadvance. And for the Neadvance results using MVTEC models vice-versa.

Table 10 presents the localization results of the MVTEC dataset using models trained with the Neadvance dataset. Start by analyzing the AUROC results, it seems that SSIM AE, CFLOW, STFPM and Reverse achieve good AUROC values (95.06%, 97.89%, 91.93%, 94.27%), close to the AUROC results from Table 5, trained and tested with MVTEC (94.18%, 99.57%, 98.91%, 99.31%). However, the MSE AE AUROC dropped from 83.49% to 57.77%. DRAEM was the technique with the highest AUROC drop, from 94,80% to 38.79%. This drop can be justified by the inability of DRAEM to learn the Neadvance patterns, as seen in Table 6 DRAEM presents the worst AUROC results. Relatively to the threshold-dependent metrics, for most of the techniques, the results are very bad. This can be justified by the bad threshold estimation because, for this experiment the threshold used was estimated to work on the training dataset, which differs from the testing dataset. However, SSIM AE keeps similar results. In the Table 5, SSIM AE achieves 31.33% IoU using T1, 29.41% IoU using T2, 47.24% PRO using T1 and 42.01% using T2. In this table, SSIM AE achieves 15.02% IoU using T1, 33.27% IoU using T2, 19.45% PRO using T1, and 46.02% using T2. This table induces that some models have the generalization ability, they just need to find thresholds for the testing dataset to achieve the Table 5 similar results.

Table 10: MVTEC dataset localization results using Neadvance as training dataset.

	Metric (%)				
Model	**IoU T1**	**IoU T2**	**PRO T1**	**PRO T2**	**AUROC**
MSE AE	0.01	1.95	0.47	2.52	57.77
SSIM AE	15.02	33.27	19.45	46.09	95.06
CFLOW	0.01	1.73	0	0	97.89
STFPM	4.26	3.54	6.43	4.23	91.93
Reverse	0.01	1.30	5.89	9.23	94.27
DRAEM	0.02	0.42	7.14	9.05	38.79

The Table 11 presents the Neadvance localization results using MVTEC as training dataset. In general, the results presented in this table are not good, following the results of Table 6. Once again, the AUROC results have a small drop in most of the models, (from 56.85%, 68.07%, 72.52%, 71.40%, 74.17%, 46.77% to 47.39%, 61.07%, 50.77%, 57.06%, 69.05%, 47.90%, on MSE AE, SSIM AE, CFLOW, STFPM, Reverse and DRAEM, respectively). This drop is acceptable, considering that the patterns used during the training, MVTEC patterns, just have one color and testing patterns have distinct colors. Analyzing the IoU and PRO results, it is clear that the results are very bad for both thresholds.

Table 11: Neadvance dataset localization results using MVTEC as training dataset.

	Metric (%)				
Model	**IoU T1**	**IoU T2**	**PRO T1**	**PRO T2**	**AUROC**
MSE AE	0.02	0.03	3.58	8.37	47.39
SSIM AE	1.07	3.68	9.80	7.19	61.07
CFLOW	0.07	0.02	0	0	50.77
STFPM	0.04	4.89	3.11	4.51	57.06
Reverse	0.04	0.02	8.82	6.37	69.05
DRAEM	0.05	0.02	5.42	7.62	47.90

Now it is time to analyze the results for the defects detection task on the MVTEC dataset using Neadvance as the training dataset, presented in Table 12. Such as in the previous tables, let's start by analyzing the AUROC metric. Surprisingly, the reconstructive-based techniques, MSE AE and SSIM AE overcome all the other techniques with 93.41% and 95.82% AUROC respectively. DRAEM AUROC also claims attention because it achieves 41.00%, being the worst AUROC of all techniques. This result follow the AUROC result for DRAEM on Table 10, both obtain the worst value in the respective tasks. Regarding F1-Score, the reconstructive-based techniques also obtain better results. Obtaining the best results for F1-Score using

T3 with MSE AE and SSIM AE (90.43% and 95.82% respectively). CFLOW also obtains a satisfactory AUROC value, however, the F1-Scores values presented are equal to 0%. So, it can be said that CFLOW has generalization ability, even obtaining very bad F1-Score results, because these results are influenced by the threshold estimation.

Table 12: MVTEC dataset detection results using Neadvance as training dataset.

	Metric (%)		
Model	**F1-Score T1**	**F1-Score T3**	**AUROC**
MSE AE	32.73	90.43	93.41
SSIM AE	71.33	83.02	95.82
CFLOW	0	0	89.37
STFPM	57.75	0	63.32
Reverse	0	0	68.61
DRAEM	4.25	0	41.00

The next table presents the Neadvance dataset detection results using models trained with the MVTEC dataset. Comparing the results of this table with Table 8, which presents the Neadvance dataset detection result. In this table, the MSE AE, SSIM AE, and DRAEM follow the AUROC results of Table 8, even if not been trained for the Neadvance dataset. In this table 69.63%, 62.14%, and 77.74% AUROC were obtained for MSE AE, SSIM AE, and DRAEM. Whereas, in the Table 8 the AUROC results for MSE AE, SSIM AE and DRAEM were 68.14%, 62.95% and 77.74%, respectively. In this table, it is important to highlight the SSIM AE results for F1-Score with 51.16% for T1 and 62.28% for T3. Obtaining better results on Neadvance dataset than the Table 8 with 53.13% for T1 and 55.51% for T3.

Table 13: Neadvance dataset detection results using MVTEC as training dataset.

	Metric (%)		
Model	**F1-Score T1**	**F1-Score T3**	**AUROC**
MSE AE	0	54.66	69.63
SSIM AE	51.16	62.28	62.14
CFLOW	0	0	46.66
STFPM	0	0	69.41
Reverse	0	0	58.36
DRAEM	2.80	0	77.74

6.3 Experiment 3

This section presents the quantitative results of Experiment 3, evaluating the localization and detection performances of techniques trained with MVTEC and Neadvance samples joined. The results of Experiment 3 will be compared with the results obtained in Experiment 1 to evaluate if training using both datasets improve the final results.

6.3.1 Quantitative results

Before starting to present the results, it is important to highlight how the thresholds used in this experiment were obtained. The T1 threshold was obtained during the Experiment 3 training techniques using both MVTEC and Neadvance samples. Moreover, the T2 and T3 thresholds were estimated during the evaluation phase using samples from the dataset used for testing.

Let's start by analyzing the results of Table 14, which presents the MVTEC localization results using models trained with MVTEC and Neadvance. In general, most of the techniques obtain excellent AUROC results. However, the MSE AE performance reduce from 83.49% to 67.49% when compared the values presented in Table 5 with the values of this table. The AUROC results of CFLOW, STFPM, and Reverse follow the AUROC results of Table 5. In this table, the DRAEM AUROC result is disappointing. Comparing with the Table 5, DRAEM AUROC drops almost 40%. This drop can be justified by the inability of DRAEM to learn the Neadvance samples, as shown in Table 6. Duringtraining, DRAEM did not learn to find the optimized parameters for the Neadvance samples, it also was not able to find parameters that match both datasets. Now, it is time to analyze the results of the threshold-dependent metrics. For the IoU metric, SSIM AE obtains the best result using T2 with 35.44% IoU. Also for the PRO metric, SSIM AE was the technique that achieved the best result with 57.13% using T2.

Table 14: MVTEC dataset localization results using MVTEC + Neadvance as training dataset.

	Metric (%)				
Model	**IoU T1**	**IoU T2**	**PRO T1**	**PRO T2**	**AUROC**
MSE AE	0.04	12.12	0.01	18.35	67.49
SSIM AE	23.14	35.44	31.74	57.13	91.02
CFLOW	1.34	23.08	0	9.62	99.52
STFPM	14.19	21.47	59.80	23.70	96.46
Reverse	9.06	22.70	41.76	49.28	99.93
DRAEM	0.14	1.43	0.12	2.81	55.09

The next table presents the Neadvance localization results using techniques trained with MVTEC and Neadvance (Table 15). Analyzing the AUROC results, it seems that this table follows the results from Table 6. In the case of SSIM AE, AUROC increases from 68.07% to 71.26%. Also, in the case of CFLOW, it increases from 72.52% to 74.35%. However, the STFPM and Reverse AUROC results drop. The STFPM AUROC from 71.40% to 66.69% and Reverse from 74.17% to 73.25%. It seems that training this technique with both datasets did not improve the localization results. This can be justified by the difficulty of Neadvance defects be detected.

Table 15: Neadvance dataset localization results using MVTEC + Neadvance as training dataset.

	Metric (%)				
Model	**IoU T1**	**IoU T2**	**PRO T1**	**PRO T2**	**AUROC**
MSE AE	0.01	0.75	2.41	22.56	55.22
SSIM AE	5.57	6.77	26.9	14.83	71.26
CFLOW	0.01	3.03	0	32.87	74.35
STFPM	4.24	6.51	9.03	15.96	66.69
Reverse	3.01	7.71	21.93	32.12	73.25
DRAEM	0.01	0.01	0.01	9.68	54.01

The next table presents the MVTEC detection results using techniques trained with MVTEC and Neadvance (Table 17). The results of this table do not equal the MVTEC detection results trained only with MVTEC samples (Table 7). Even though the AUROC results do not be so high as expected, SSIM AE and CFLOW still reach excellent AUROC values, 92.30%, and 99.73%, respectively. On other hand, the F1-Score results using T3 achieve good results, close to those presented in Table 7. In the case of SSIM AE and CFLOW, the F1-Score results using T3 overcome the previous results. Increasing from 90.57 to 93.59 SSIM AE F1-Score value and from 95.36% to 95.95% F1-Score.

Table 16: MVTEC dataset detection results using MVTEC + Neadvance as training dataset.

	Metric (%)		
Model	**F1-Score T1**	**F1-Score T3**	**AUROC**
MSE AE	28.89	78.12	86.60
SSIM AE	77.78	93.59	92.30
CFLOW	77.78	93.59	92.30
STFPM	79.17	77.61	81.17
Reverse	83.62	80.00	77.85
DRAEM	22.99	0	55.80

The next table is the last table to analyze. The next table presents the MVTEC detection results using

models trained with both datasets. It presents satisfactory AUROC results, achieving the maximum with DRAEM, 78.69%. Comparing these results with the presented in Table 8, SSIM AE, CFLOW and DRAEM outperform the AUROC results. Relatively to the F1-Score results using T3, SSIM AE, CFLOW, STFPM, and Reverse present better results. For the T1, there also have improvements on SSIM AE, STFPM, and DRAEM.

Table 17: Neadvance dataset detection results using MVTEC + Neadvance as training dataset.

	Metric (%)		
Model	**F1-Score T1**	**F1-Score T3**	**AUROC**
MSE AE	62.22	67.35	65.81
SSIM AE	66.90	60.15	68.65
CFLOW	0	80.69	73.30
STFPM	38.63	35.40	67.42
Reverse	70.13	85.39	66.30
DRAEM	36.30	0	78.69

6.4 Discussion

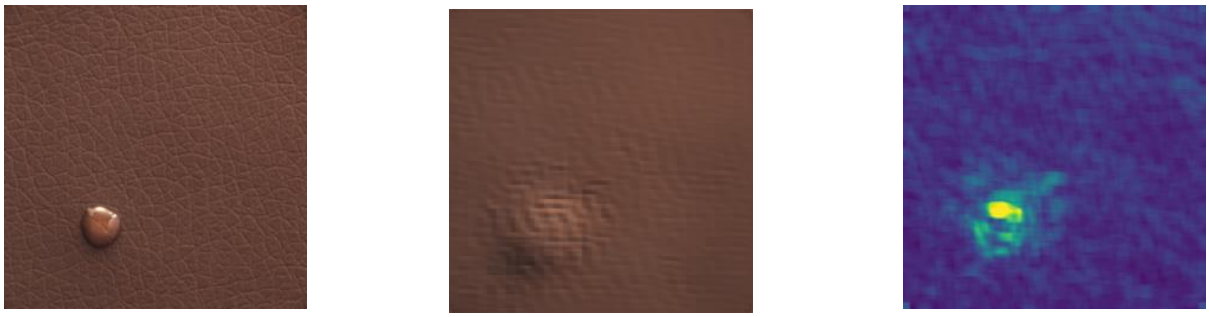
After the experiments analysis, it is time to combine the ideas that arise from the previous analyzes to discuss the advantages and disadvantages of each technique. In the end, there are presented the techniques that should be used for distinct scenarios.

6.4.1 Experiment 1

The AE is a reconstructive technique that learned to reconstruct non-defective samples, in this way, AE should not be able to reconstruct defective regions. Thus, when comparing the reconstructed image with the original, the differences should be notorious. Analyzing Figure 35, the image reconstructed by MSE AE presents a non-defective pattern. As the MSE AE does not reconstruct the defective region, it should be easy to detect the dissimilarities between this two images. However, MSE does not have the ability to do it. SSIM solves this problem, as presented in Figure 36, the reconstructed image was not able to reconstruct the defective regions to a non-defective pattern. So, SSIM extract the structural information to obtain a good anomaly score map, allowing the detection and localization of most defects. As seen previously, MSE AE does not present good results on leather defects detection. Even MSE AE presents low inference time and requiring low percentage of GPU utilization, MSE AE should not be used.

Original Reconstructed Map

Figure 35: Glue sample result using MSE AE.



Original Reconstructed Map

Figure 36: Glue sample result using SSIM AE.

In this dissertation, three feature-extraction based techniques were tested: CFLOW, STFPM, and Reverse. The CFLOW is the technique in the state-of-the-art with the highest segmentation AUROC for the MVTEC dataset [78] and this experiment confirms that. In this experience, the CFLOW has 100% of detection AUROC with the MVTEC samples. However, analyzing the segmentation and detection results for the Neadvance dataset, the CFLOW is not the best technique. Also, CFLOW does not produce the best anomaly score maps, difficulting the scores maps threshold.

The STFPM and Reverse are the two teacher-student architectures used. As Reverse uses an OCBE, the time complexity increases relatively to STFPM. The Reverse Distillation has higher percentage of GPU utilization and requires more time to infer each image. As described in Section 4.4, Reverse Distillation emerged to outperform knowledge distillation techniques like STFPM, where the student has the same architecture as the teacher. So, Reverse Distillation proposes a reverse architecture for the student produces distinct features. Analyzing the global quantitative results, it is not clear that Reverse Distillation outperforms the STFPM. However, analyzing the threshold independent metric, the AUROC increases 0.4% and 3.77% from STFPM to Reverse Distillation on MVTEC and Neadvance localization task, respectively.

DRAEM is the last technique to analyze. This technique presents the lowest inference time for both datasets. However, it requires the entire GPU utilized in this experiment because the two architectures used on DRAEM are very deep. On the MVTEC dataset, DRAEM performs well, achieving quantitative results close to the best. In this technique, it is important to highlight the anomaly scores maps generated, attributing high scores to the defective pixels and low scores to the non-defective pixels. This point improves the quality of the predicted mask obtained because the threshold can assume a large range of values.

After this discussion, it is necessary to define in which real-world scenarios these techniques should be used. In a scenario where a reduced inference time is the most critical point, the MSE AE could be an option. However, even though MSE AE has the highest number of FPS and the lowest percentage of GPU utilization, the MSE should not be used because it does not present a good performance on defects detection. So, DRAEM could be a good option because it has the second highest number of FPS. Nevertheless, DRAEM does not perform well on the Neadvance dataset. So, DRAEM only should be an option to achieve a low inference time when the real-world scenario has leather samples only following one color, like MVTEC. When the scenario has more than one color, STFPM should be used, because it presents better anomaly scores maps than CFLOW and has lower complexity than Reverse Distillation.

DRAEM and STFPM are the recommended techniques to use in the previously mentioned scenarios. However, the previous reflection does not take into account the hardware requirements. The techniques percentage of GPU utilization by descendent order is DRAEM, REVERSE, STFPM, CFLOW, SSIM AE, and MSE AE. As MSE AE presents bad results, SSIM AE should be used when the hardware resources available are limited.

6.4.2 Experiment 2

The main goal of Experiment 2 was to evaluate the generalization ability of the novelty detection techniques. Check if the techniques can detect defects from samples with different patterns than those used during the training. In other words, the techniques used can be generalized solutions for different scenarios.

Relatively to the MVTEC dataset, it presents AUROC segmentation results similar to Experiment 1, only DRAEM does not obtain a good AUROC. As mentioned before, this can be justified by the inability of DRAEM to learn the Neadvance patterns during the training. In this way, if did not learn the Neadvance normal samples during the training, will not be able to detect the defective patterns on the MVTEC dataset

during testing. SSIM AE, CFLOW, STFPM and Reverse present good AUROC values, however, the threshold-dependent results are very bad. So, it is advised to estimate a new optimized threshold T2 to perform better predictions. In this way, it will be possible to obtain better defective masks. In terms of detection tasks, MSE AE, SSIM AE, and CFLOW present the best results. Nevertheless, it is necessary to estimate new T3 values for the CFLOW. The F1-Score using T3 is below expectations.

Relatively to the Neadvance dataset, the AUROC results are close to those obtained during Experiment 1 and in this way, it could be said that the results reveal some generalization ability. However, without a satisfactory performance on the Neadvance dataset in both Experiments, it can not be assumed to trust conclusions. Also, it is necessary to express that during Experiment 1, the Neadvance evaluation was performed using techniques trained with the non-defective cropped samples, obtained from defective Neadvance samples. So, only if Experiment 1 with Neadvance was trained using a high-quality dataset, it could be possible to trust on generalization ability.

In this experiment, SSIM AE and CFLOW present good generalization ability on the MVTEC dataset for both tasks. However, as seen, the threshold-dependent metrics results are very bad, so, it is necessary to estimate new optimized thresholds to obtain better masks. To conclude the discussion of Experiment 2, it is possible to use novelty detection techniques trained with distinct color patterns to detect defects. However, the quantitative results are low. So, it is recommended to train the novelty detection techniques to predict samples with the same color as the one used during the training, if there are enough image samples available.

6.4.3 Experiment 3

Experiment 3 pretended to verify if joining both datasets improve the quantitative results on MVTEC and Neadvance datasets (Experiment 1). It is expected that this improvement not only be caused by the increasing number of training samples but also, by the use of samples with distinct color patterns.

The MVTEC results do not reveal a clear improvement in localization and defect detection tasks. Furthermore, the DRAEM results for both tasks have a drop, when compared with Experiment 1. The Neadvance results also do not allow to verify that joining both datasets improve the results.

So, a conclusion that arises from this experiment is that the increase in training complexity, caused by the increasing number of image samples, did not improve the results. In this way, it seems that it is better

to train the techniques only with samples from the scenario where these techniques will be deployed.

Chapter 7

Conclusion

The leather inspection process is a critical process in the leather industry. A good inspection process should avoids the production of leather goods with anomalous areas. As most industries use specialized hand labor to detect defects in leather, this makes the defects detection task slow. In this way, as technology has evolved in the last decades, it is essential to automatize this process. The solution is to apply the most recent techniques to automatize this process.

In this way, this dissertation explores distinct techniques to automatize this task. The anomaly detection state-of-the-art solutions perform well using Machine Learning and Computer Vision techniques. However, most of them are supervised techniques, requiring a labeled dataset to train. Exploring supervised architectures was not an option for this dissertation because there are no available supervised datasets, and the cost of acquiring a new dataset is very high. Also, the supervised techniques were excluded because there are an infinite number of leather defect types. So, supervised architectures cannot learn all the defect types. In this way, the solution found to these constraints was to use an unsupervised technique, known as novelty detection.

Novelty detection techniques try to learn the normal pattern of the leather samples during the training process. So, as the technique learns the normal pattern, it is expected that the techniques fail in the anomalous regions. Based on this perspective, this work experiments six distinct techniques to solve the problem using a small dataset composed of non-anomalous samples.

A detailed discussion for each method were already presented in the results chapter. So, it is important to highlight if the goals of this dissertation were achieved. In this work, the proposed main goal was to solve this problem using novelty detection techniques, and it was achieved as presented in the results in Chapter 6. The second goal was to find solutions that use the minimal computational power. This goal was also achieved by techniques as SSIM AE and MSE AE.

In order to combine the conclusions of all experiments of this dissertation, for the defects detection problem, it is recommended to detect defects using novelty techniques trained with samples with the same color features as the deployment scenario. This can be assumed by the results from the Experiment 2 and Experiment 3 that, in general, did not present good results as Experiment 1. So, to conclude, this dissertation argues that distinct techniques should be used in distinct scenarios. In the case where the computational power available is low, SSIM AE should be the technique to use. Despite MSE AE also requiring less computational power, it does not present a good performance in defect detection. In the case there is enough computational power to execute, the DRAEM should be the chosen option, if the real-world scenario just has one color sample. In any other case, the STFPM should be the option chosen.

The work presented in this dissertation solves the leather detection problem. However, every day, new researches present novelty detection techniques that overcome the previous state-of-the-art techniques. So, for future work, the continuous upgrading of leather detection solutions using recent novelty detection techniques is mandatory. New ways to estimate the binary threshold, to convert the anomaly score maps into a binary mask, should also be explored. Also, it is crucial to continue looking for solutions with low computation requirements. Most of the time, these solutions are applied in small computers such as raspberries that do not have the required hardware to implement the techniques presented in this work. On other hand, the presented solutions can perform better if the training dataset was bigger. In this way, it is necessary to invest in leather image capture, and if there is enough budget to obtain an annotated dataset, it will be interesting to explore supervised techniques.

Bibliography

[1] *Africa Leather and Leather Products Institute (ALLPI) - HIDES AND SKINS IMPROVEMENT HANDBOOK: TRAINER'S MANUAL - HIDES AND SKINS IMPROVEMENT HANDBOOK: TRAINER'S MANUAL*. url: https://allpi.int/courses-and-publications/reports/manuals/hides-and-skins-improvement.

[2] *Leather Inspection Table: A Big Idea Made Simple | Simplified Building*. url: https://www.simplifiedbuilding.com/projects/leather-inspection-table-a-big-idea-made-simple.

[3] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3 (3 July 1959), pp. 210–229. issn: 0018-8646. doi: 10.1147/RD.33.0210. url: http://ieeexplore.ieee.org/document/5392560/.

[4] Alberto Garcia-Garcia et al. *A Review on Deep Learning Techniques Applied to Semantic Segmentation*. 2017. doi: 10.48550/ARXIV.1704.06857. url: https://arxiv.org/abs/1704.06857.

[5] Konstantina Kourou et al. "Machine learning applications in cancer prognosis and prediction". In: *Computational and Structural Biotechnology Journal* 13 (2015), pp. 8–17. issn: 2001-0370. doi: https://doi.org/10.1016/j.csbj.2014.11.005. url: https://www.sciencedirect.com/science/article/pii/S2001037014000464.

[6] Taeho Jo, Kwangsik Nho, and Andrew J. Saykin. "Deep Learning in Alzheimer's Disease: Diagnostic Classification and Prognostic Prediction Using Neuroimaging Data". In: *Frontiers in Aging Neuroscience* 11 (2019). issn: 1663-4365. doi: 10.3389/fnagi.2019.00220. url: https://www.frontiersin.org/article/10.3389/fnagi.2019.00220.

[7] Li Yang et al. “Sentiment Analysis for E-Commerce Product Reviews in Chinese Based on Sentiment Lexicon and Deep Learning”. In: *IEEE Access* 8 (2020), pp. 23522–23530. doi: 10.1109/ACCESS.2020.2969854.

[8] Sorin Grigorescu et al. “A survey of deep learning techniques for autonomous driving”. In: *Journal of Field Robotics* 37.3 (2020), pp. 362–386. doi: https://doi.org/10.1002/rob.21918. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21918. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21918.

[9] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education, 2021. isbn: 9781292401171. url: https://books.google.pt/books?id=cb0qEAAAQBAJ.

[10] *Semi-Supervised Learning*. MIT Press, 2010, pp. 221–232. doi: 10.7551/MITPRESS/9780262033589.001.0001.

[11] J. R. Quinlan. “Induction of Decision Trees”. In: *Mach. Learn.* 1.1 (1986), pp. 81–106. issn: 0885-6125. doi: 10.1023/A:1022643204877. url: https://doi.org/10.1023/A:1022643204877.

[12] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in machine learning. Elsevier Science, 1993. isbn: 9781558602380. url: https://books.google.pt/books?id=HExncpjbYroC.

[13] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. “Training algorithm for optimal margin classifiers”. In: *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* (1992), pp. 144–152. doi: 10.1145/130385.130401.

[14] John H. Murphy. “An Overview of Convolutional Neural Network Architectures for Deep Learning”. In: 2016.

[15] S. Velliangiri, S. Alagumuthukrishnan, and S. Iwin Thankumar Joseph. “A Review of Dimensionality Reduction Techniques for Efficient Computation”. In: *Procedia Computer Science* 165 (Jan. 2019), pp. 104–111. issn: 1877-0509. doi: 10.1016/J.PROCS.2020.01.079.

[16] M. Turk and A. Pentland. “Eigenfaces for Recognition”. In: *Journal of Cognitive Neuroscience* 3 (1 Jan. 1991), pp. 71–86. issn: 0898-929X. doi: 10.1162/JOCN.1991.3.1.71. url: http://direct.mit.edu/jocn/article-pdf/3/1/71/1932018/jocn.1991.3.1.71.pdf.

[17] Pradeep Rai and Shubha Singh. “A Survey of Clustering Techniques”. In: *International Journal of Computer Applications* 7 (12 Oct. 2010), pp. 1–5. doi: 10.5120/1326-1808. url: https://www.researchgate.net/publication/49586251_A_Survey_of_Clustering_Techniques.

[18] J. B. MacQueen. “Some Methods for Classification and Analysis of MultiVariate Observations”. In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by L. M. Le Cam and J. Neyman. Vol. 1. University of California Press, 1967, pp. 281–297.

[19] Gaowei Zhang, Yue Pan, and Limao Zhang. “Semi-supervised learning with GAN for automatic defect detection from images”. In: *Automation in Construction* 128 (2021), p. 103764. issn: 0926-5805. doi: https://doi.org/10.1016/j.autcon.2021.103764. url: https://www.sciencedirect.com/science/article/pii/S0926580521002156.

[20] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521 (7553 May 2015), pp. 436–444. issn: 14764687. doi: 10.1038/NATURE14539. url: https://www.researchgate.net/publication/277411157_Deep_Learning.

[21] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics 1980 36:4* 36 (4 Apr. 1980), pp. 193–202. issn: 1432-0770. doi: 10.1007/BF00344251. url: https://link.springer.com/article/10.1007/BF00344251.

[22] Le Cun et al. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Advances in Neural Information Processing Systems* 2 (1989).

[23] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86 (11 1998), pp. 2278–2323. issn: 00189219. doi: 10.1109/5.726791.

[24] Saleh Albelwi and Ausif Mahmood. “A Framework for Designing the Architectures of Deep Convolutional Neural Networks”. In: *Entropy* 19.6 (2017). issn: 1099-4300. doi: 10.3390/e19060242. url: https://www.mdpi.com/1099-4300/19/6/242.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9351 (May 2015), pp. 234–241. issn: 16113349. url: https://arxiv.org/abs/1505.04597v1.

[26] *Morphological Transformations*. url: https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html.

[27] Bu¨lent Sankur. "Survey over image thresholding techniques and quantitative performance evaluation". In: *Journal of Electronic Imaging* 13 (1 Jan. 2004), p. 146. issn: 1017-9909. doi: 10.1117/1.1631315.

[28] Nobuyuki Otsu. "A threshold selection method from gray level histograms". In: *undefined* SMC-9 (1 1979), pp. 62–66. issn: 00189472. doi: 10.1109/TSMC.1979.4310076.

[29] Choonjong Kwak, José A. Ventura, and Karim Tofang-Sazi. "Automated defect inspection and classification of leather fabric". In: *Intelligent Data Analysis* 5 (4 2001), pp. 355–370. issn: 15714128. doi: 10.3233/ida-2001-5406.

[30] Saket Bhardwaj and Ajay Mittal. "A Survey on Various Edge Detector Techniques". In: *Procedia Technology* 4 (Jan. 2012), pp. 220–226. issn: 2212-0173. doi: 10.1016/J.PROTCY.2012.05.033.

[31] Lawrence G. Roberts. "Machine Perception of 3-{D} Solids". In: *OEOIP* (1965), pp. 159–197. url: https://dspace.mit.edu/bitstream/handle/1721.1/11589/33959125-MIT.pdf.

[32] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).

[33] J. M. S. PREWITT. "Object enhancement and extraction". In: *Picture Processing and. Psychopictorics* (1970). url: https://ci.nii.ac.jp/naid/10017095478/en/.

[34] "Texture and color feature extraction for classification of melanoma using SVM". In: *2016 International Conference on Computing Technologies and Intelligent Data Engineering, ICCTIDE 2016* (Oct. 2016). doi: 10.1109/ICCTIDE.2016.7725347. url: https://www.researchgate.net/publication/309588827_Texture_and_color_feature_extraction_for_classification_of_melanoma_using_SVM.

[35] Robert Haralick, K. Shanmugam, and Ih Dinstein. "Textural Features for Image Classification". In: *IEEE Trans Syst Man Cybern* SMC-3 (Jan. 1973), pp. 610–621.

[36] D. Gabor. “Theory of communication. Part 1: The analysis of information”. In: *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering* 93 (26 Nov. 1946), pp. 429–441. doi: 10.1049/JI-3-2.1946.0074.

[37] Malathy Jawahar, N. K.Chandra Babu, and K. Vani. “Leather texture classification using wavelet feature extraction technique”. In: *2014 IEEE International Conference on Computational Intelligence and Computing Research, IEEE ICCIC 2014* (Sept. 2015). doi: 10.1109/ICCIC.2014.7238475.

[38] S. Arivazhagan and L. Ganesan. “Texture classification using wavelet transform”. In: *Pattern Recognition Letters* 24 (9-10 June 2003), pp. 1513–1521. issn: 0167-8655. doi: 10.1016/S0167-8655(02)00390-2.

[39] Siddharth Misra and Hao Li. “Chapter 9 Noninvasive fracture characterization based on the classification of sonic wave travel times”. In: *Machine Learning for Subsurface Characterization* (2020), pp. 243–287. doi: 10.1016/B978-0-12-817736-5.00009-0. url: https://app.dimensions.ai/details/publication/pub.1121887371.

[40] Choonjong Kwak, Jose A. Ventura, and Karim Tofang-Sazi. “A neural network approach for defect identification and classification on leather fabric”. In: *Journal of Intelligent Manufacturing 2000 11:5* 11 (5 Oct. 2000). Igual ao ”Automated defect inspection and classification of leather fabric”mas usa NN e arvores de decisao, pp. 485–499. issn: 1572-8145. doi: 10.1023/A:1008974314490. url: https://link.springer.com/article/10.1023/A:1008974314490.

[41] Lidiya Georgieva, Kaloyan Krastev, and Nikola Angelov. “Identification of surface leather defects”. In: (2003), pp. 303–307. doi: 10.1145/973620.973670.

[42] H. Pistori et al. “Defect detection in raw hide and wet blue leather”. In: *undefined* (2006).

[43] Li Jian, Han Wei, and He Bin. “Research on inspection and classification of leather surface defects based on neural network and decision tree”. In: *2010 International Conference on Computer Design and Applications, ICCDA 2010* 2 (2010). doi: 10.1109/ICCDA.2010.5541405.

[44] Sri Winiarti et al. “Pre-trained convolutional neural network for classification of tanning leather image”. In: *International Journal of Advanced Computer Science and Applications* 9 (1 2018), pp. 212–217. doi: 10.14569/IJACSA.2018.090129.

[45] Sze-Teng Liong et al. “Integrated Neural Network and Machine Vision Approach For Leather Defect Classification”. In: (May 2019). url: https://arxiv.org/abs/1905.11731v1.

[46] Y. S. Gan et al. “Automated leather defect inspection using statistical approach on image intensity”. In: *Journal of Ambient Intelligence and Humanized Computing 2020 12:10* 12 (10 2020), pp. 9269–9285. issn: 1868-5145. doi: 10.1007/S12652-020-02631-6. url: https://link.springer.com/article/10.1007/s12652-020-02631-6.

[47] Jiehang Deng et al. “A novel framework for classifying leather surface defects based on a parameter optimized residual network”. In: *IEEE Access* 8 (2020), pp. 192109–192118. doi: 10.1109/ACCESS.2020.3032164.

[48] Masood Aslam et al. “Ensemble convolutional neural networks with knowledge transfer for leather defect classification in industrial settings”. In: *IEEE Access* 8 (2020), pp. 198600–198614. doi: 10.1109/ACCESS.2020.3034731.

[49] Sumita Dixit et al. “Toxic hazards of leather industry and technologies to combat threat: a review”. In: *Journal of Cleaner Production* 87 (C Jan. 2015), pp. 39–49. issn: 0959-6526. doi: 10.1016/J.JCLEPRO.2014.10.017.

[50] Paul Bergmann et al. “MVTEC ad-A comprehensive real-world dataset for unsupervised anomaly detection”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June (June 2019), pp. 9584–9592. issn: 10636919. doi: 10.1109/CVPR.2019.00982.

[51] Marco A F Pimentel et al. “A review of novelty detection”. In: *Signal Processing* 99 (2014), pp. 215–249. doi: 10.1016/j.sigpro.2013.12.026. url: http://dx.doi.org/10.1016/j.sigpro.2013.12.026.

[52] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. *CFLOW-AD: Real-Time Unsupervised Anomaly Detection with Localization via Conditional Normalizing Flows*. 2021. doi: 10.48550/ARXIV.2107.12571. url: https://arxiv.org/abs/2107.12571.

[53] Guodong Wang et al. *Student-Teacher Feature Pyramid Matching for Anomaly Detection*. 2021. doi: 10.48550/ARXIV.2103.04257. url: https://arxiv.org/abs/2103.04257.

[54] Hanqiu Deng and Xingyu Li. “Anomaly Detection via Reverse Distillation from One-Class Embedding”. In: (2022). doi: 10.48550/ARXIV.2201.10703. url: https://arxiv.org/abs/2201.10703.

[55] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. *DRAEM – A discriminatively trained reconstruction embedding for surface anomaly detection*. 2021. doi: 10.48550/ARXIV.2108.07610. url: https://arxiv.org/abs/2108.07610.

[56] Markos Markou and Sameer Singh. "Novelty detection: a review—part 1: statistical approaches". In: *Signal Processing* 83.12 (2003), pp. 2481–2497. issn: 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2003.07.018. url: https://www.sciencedirect.com/science/article/pii/S0165168403002020.

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[58] Kazuma Takahara, Shuhei Ikemoto, and Koh Hosoda. "Reconstructing State-Space from Movie Using Convolutional Autoencoder for Robot Control". In: *Intelligent Autonomous Systems 15*. Ed. by Marcus Strand et al. Cham: Springer International Publishing, 2019, pp. 480–489. isbn: 978-3-030-01370-7.

[59] Lovedeep Gondara. "Medical Image Denoising Using Convolutional Denoising Autoencoders". In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. 2016, pp. 241–246. doi: 10.1109/ICDMW.2016.0041.

[60] J.K. Chow et al. "Anomaly detection of defects on concrete structures with the convolutional autoencoder". In: *Advanced Engineering Informatics* 45 (2020), p. 101105. issn: 1474-0346. doi: https://doi.org/10.1016/j.aei.2020.101105. url: https://www.sciencedirect.com/science/article/pii/S1474034620300744.

[61] Paul Bergmann et al. "Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders". In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019. doi: 10.5220/0007364503720380. url: https://doi.org/10.5220%2F0007364503720380.

[62] Paul Bergmann et al. "Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders". In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019. doi: 10.5220/0007364503720380. url: https://doi.org/10.5220\%2F0007364503720380.

[63] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. doi: 10.1109/TIP.2003.819861.

[64] Amrutbhai N. Patel and Dhara Shah. “SSIM based image quality assessment for vector quantization based lossy image compression using LZW coding”. In: 2015.

[65] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. doi: 10.1109/TIP.2003.819861.

[66] Marco Rudolph et al. *Fully Convolutional Cross-Scale-Flows for Image-based Defect Detection*. 2021. doi: 10.48550/ARXIV.2110.02855. url: https://arxiv.org/abs/2110.02855.

[67] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008. url: http://arxiv.org/abs/1706.03762.

[68] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[69] Sergey Zagoruyko and Nikos Komodakis. “Wide Residual Networks”. In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Edwin R. Hancock Richard C. Wilson and William A. P. Smith. BMVA Press, 2016, pp. 87.1–87.12. isbn: 1-901725-59-6. doi: 10.5244/C.30.87. url: https://dx.doi.org/10.5244/C.30.87.

[70] Andrew Howard et al. “Searching for MobileNetV3”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1314–1324. doi: 10.1109/ICCV.2019.00140.

[71] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. doi: 10.48550/ARXIV.1503.02531. url: https://arxiv.org/abs/1503.02531.

[72] Paul Bergmann et al. “The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection”. In: *International Journal of Computer Vision* 129 (4 Apr. 2021), pp. 1038–1059. issn: 15731405. doi: 10.1007/S11263-020-01400-4/TABLES/8. url: https://link.springer.com/article/10.1007/s11263-020-01400-4.

[73] Taesung Park et al. “Semantic Image Synthesis with Spatially-Adaptive Normalization”. In: (2019). doi: 10.48550/ARXIV.1903.07291. url: https://arxiv.org/abs/1903.07291.

[74] Thomas Schlegl et al. *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. 2017. doi: 10.48550/ARXIV.1703.05921. url: https://arxiv.org/abs/1703.05921.

[75] Ken Perlin. "An Image Synthesizer". In: 19.3 (1985), pp. 287–296. issn: 0097-8930. doi: 10.1145/325165.325247. url: https://doi.org/10.1145/325165.325247.

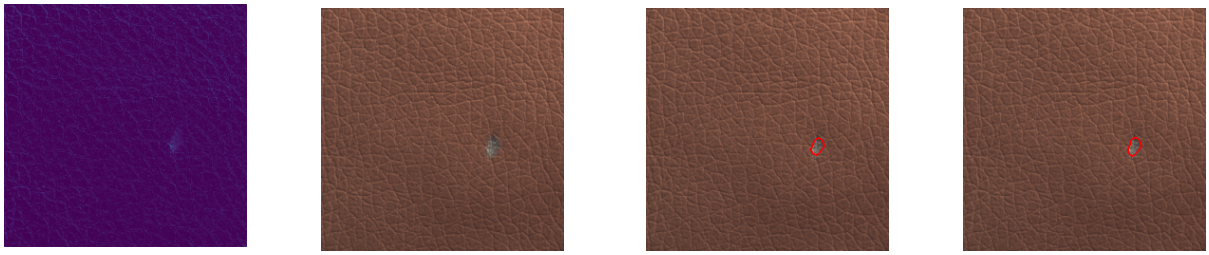
[76] Mircea Cimpoi et al. *Describing Textures in the Wild*. 2013. doi: 10.48550/ARXIV.1311.3618. url: https://arxiv.org/abs/1311.3618.

[77] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2999–3007. doi: 10.1109/ICCV.2017.324.

[78] *MVTec AD Benchmark (Anomaly Detection) | Papers With Code*. Accessed: 2022-07-13. url: https://paperswithcode.com/sota/anomaly-detection-on-mvtec-ad.

Appendix A

A.1 MVTEC color sample results



Map T1 T2 T3

Figure 37: MVTEC color sample results with MSE AE.

Map T1 T2 T3

Figure 38: MVTEC color sample results with SSIM AE.

Figure 39: MVTEC color sample results with CFLOW.

Figure 40: MVTEC color sample results with STFPM.

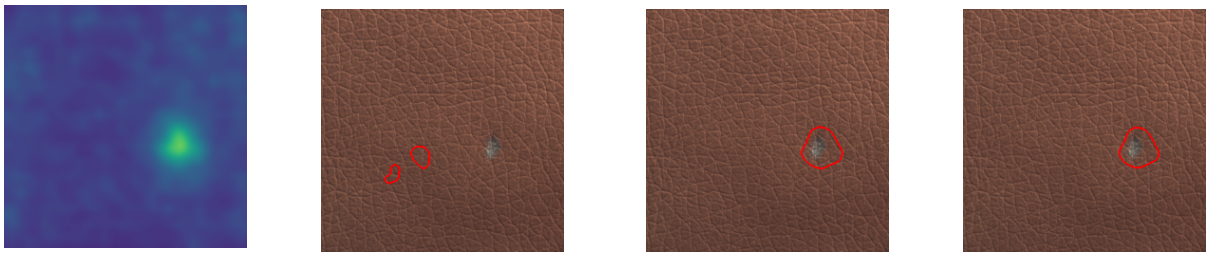


Figure 41: MVTEC color sample results with Reverse.

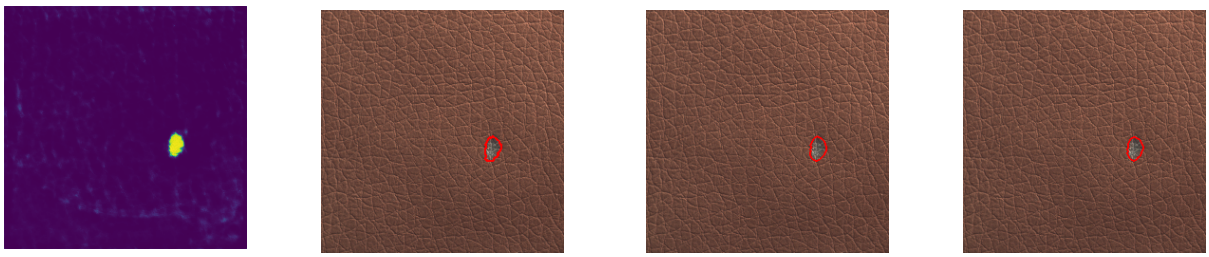
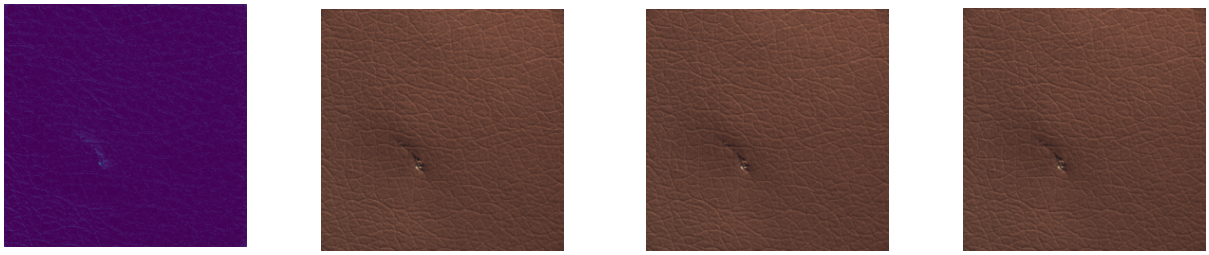


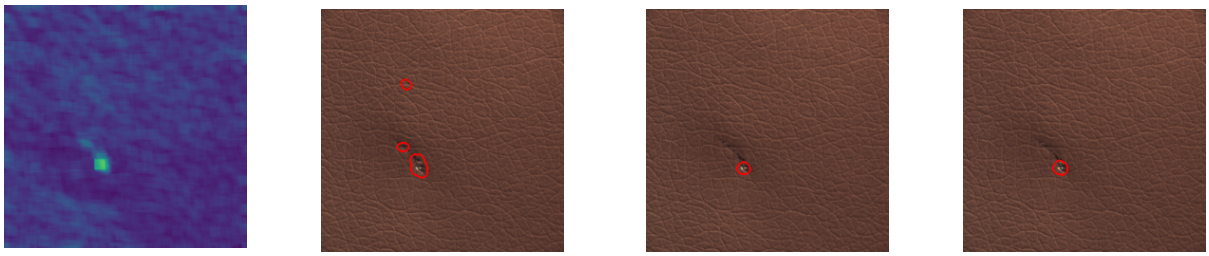
Figure 42: MVTEC color sample results with DRAEM.

A.2 MVTEC cut sample results



Map T1 T2 T3

Figure 43: MVTEC cut sample results with MSE AE.



Map T1 T2 T3

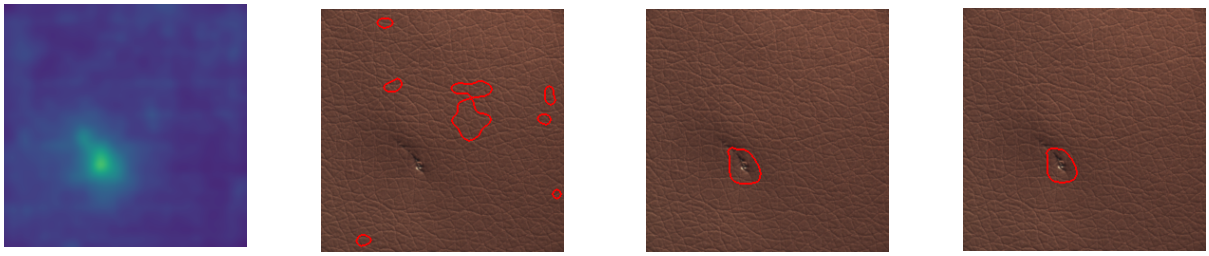
Figure 44: MVTEC cut sample results with SSIM AE.

Map T1 T2 T3

Figure 45: MVTEC cut sample results with CFLOW.

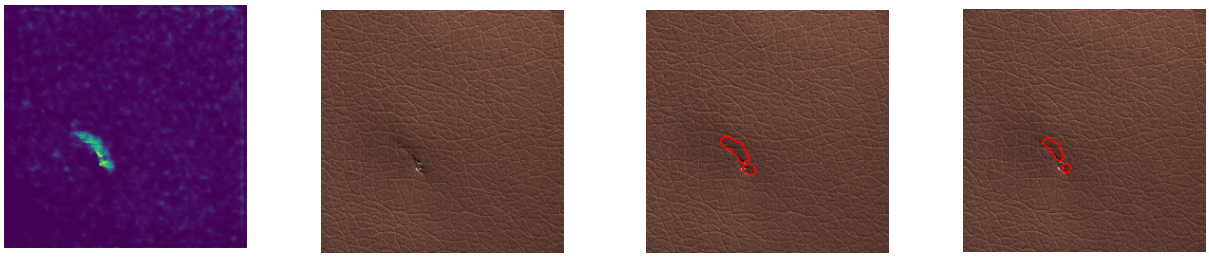
Map T1 T2 T3

Figure 46: MVTEC cut sample results with STFPM.



Map T1 T2 T3

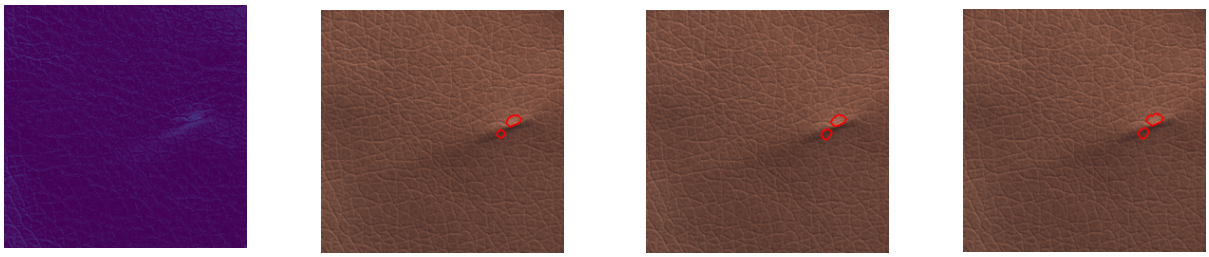
Figure 47: MVTEC cut sample results with Reverse.



Map T1 T2 T3

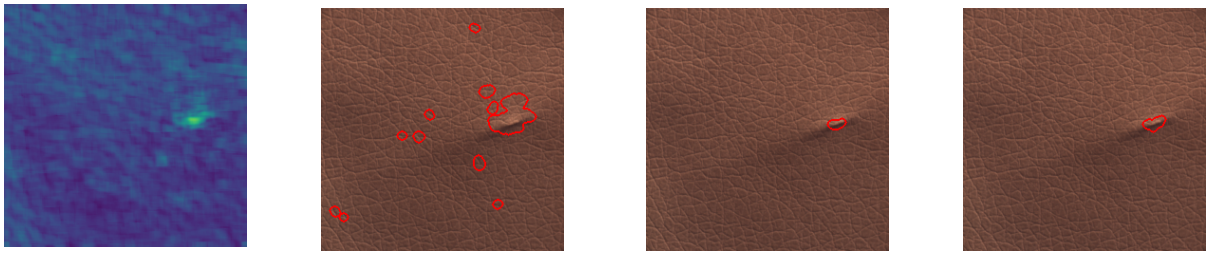
Figure 48: MVTEC cut sample results with DRAEM.

A.3 MVTEC fold sample results



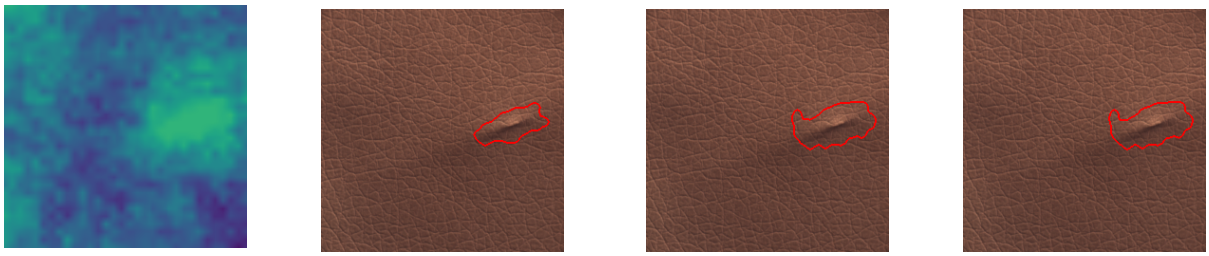
Map T1 T2 T3

Figure 49: MVTEC fold sample results with MSE AE.



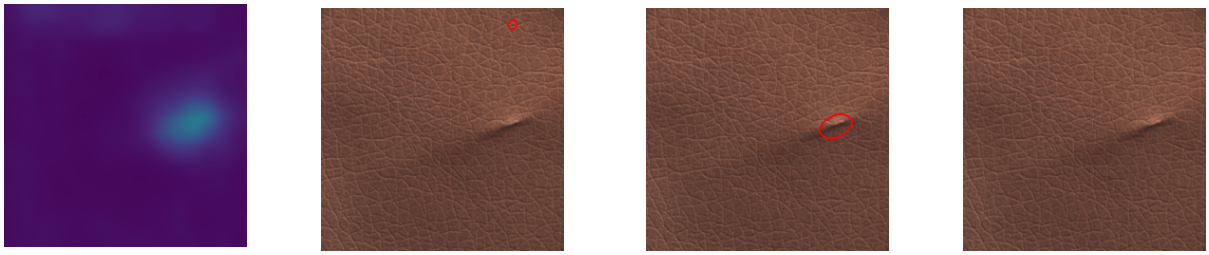
Map T1 T2 T3

Figure 50: MVTEC fold sample results with SSIM AE.



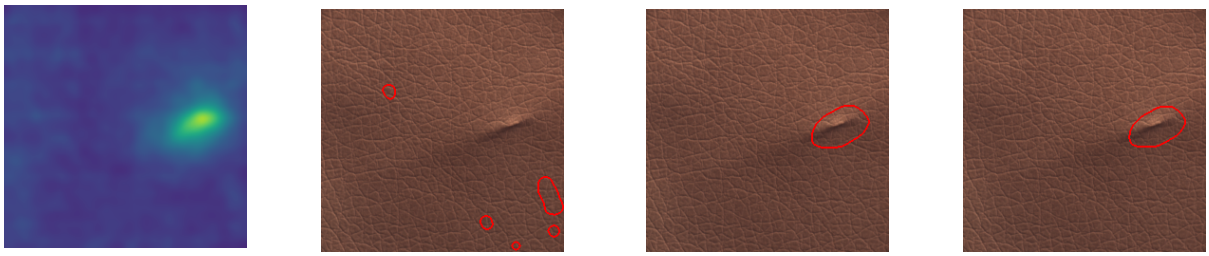
Map T1 T2 T3

Figure 51: MVTEC fold sample results with CFLOW.



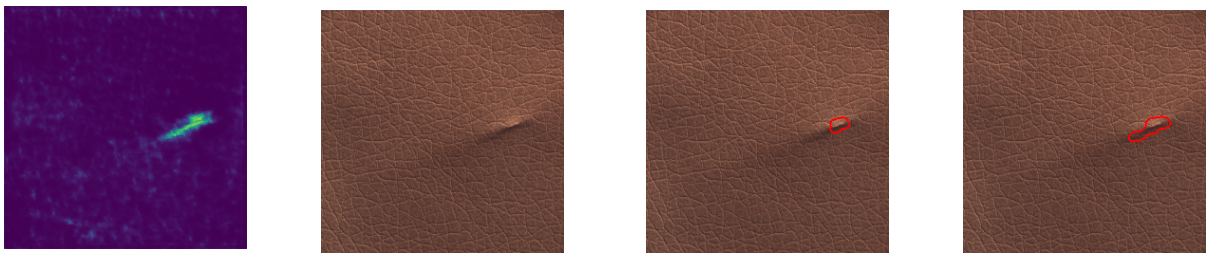
Map T1 T2 T3

Figure 52: MVTEC fold sample results with STFPM.



Map T1 T2 T3

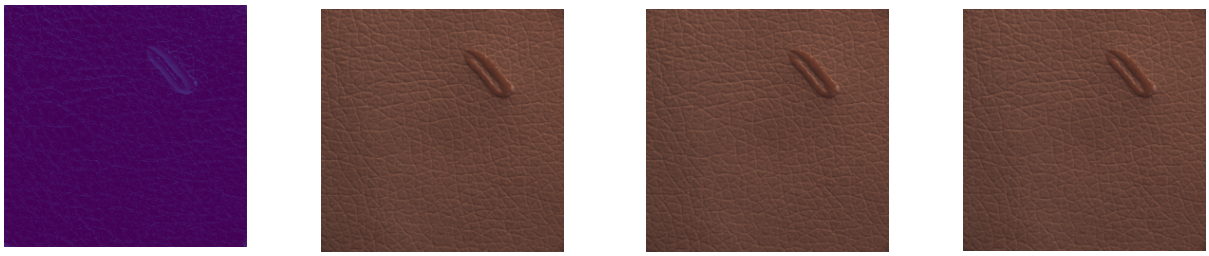
Figure 53: MVTEC fold sample results with Reverse.



Map T1 T2 T3

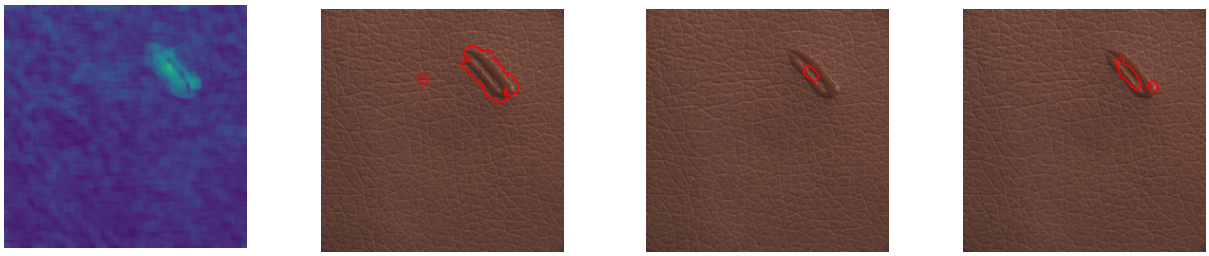
Figure 54: MVTEC fold sample results with DRAEM.

A.4 MVTEC glue sample results



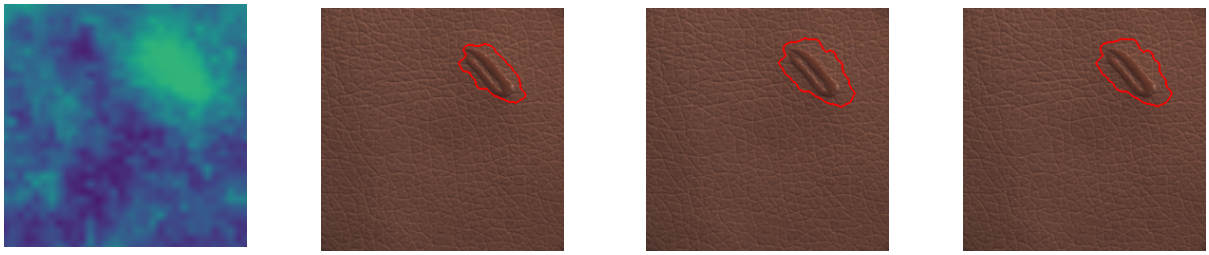
Map T1 T2 T3

Figure 55: MVTEC glue sample results with MSE AE.



Map T1 T2 T3

Figure 56: MVTEC glue sample results with SSIM AE.



Map T1 T2 T3

Figure 57: MVTEC glue sample results with CFLOW.

Map T1 T2 T3

Figure 58: MVTEC glue sample results with STFPM.

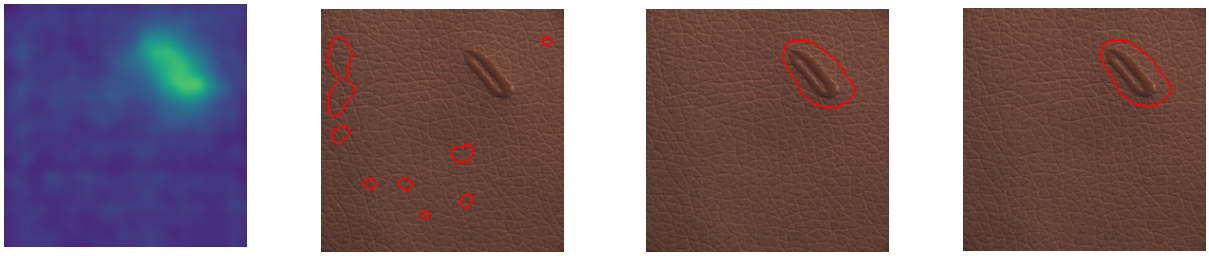


Figure 59: MVTEC glue sample results with Reverse.

Figure 60: MVTEC glue sample results with DRAEM.

A.5 MVTEC poke sample results

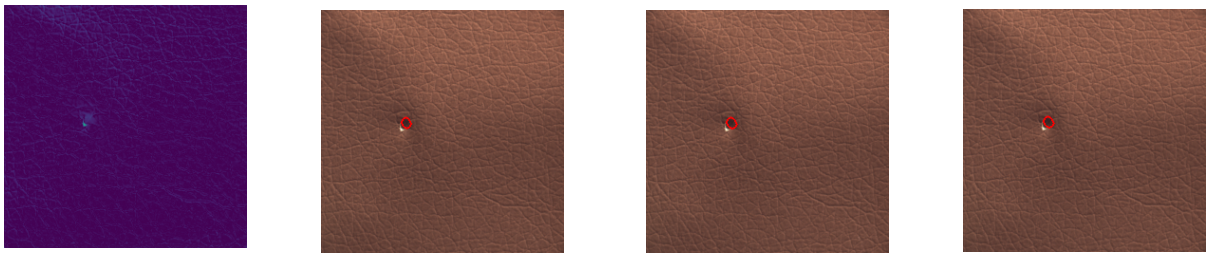


Figure 61: MVTEC poke sample results with MSE AE.

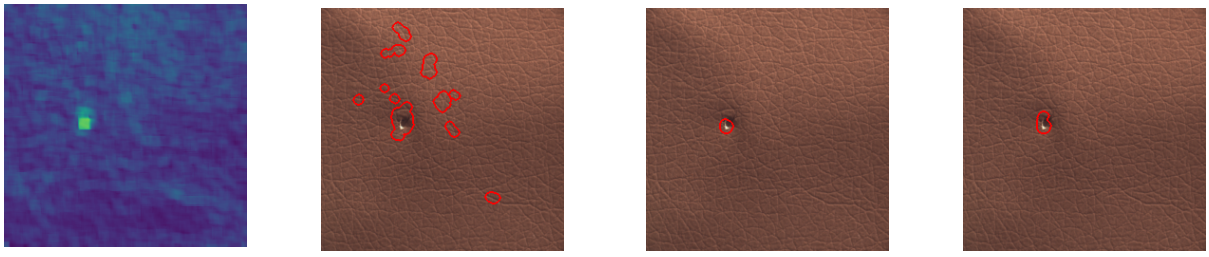


Figure 62: MVTEC poke sample results with SSIM AE.

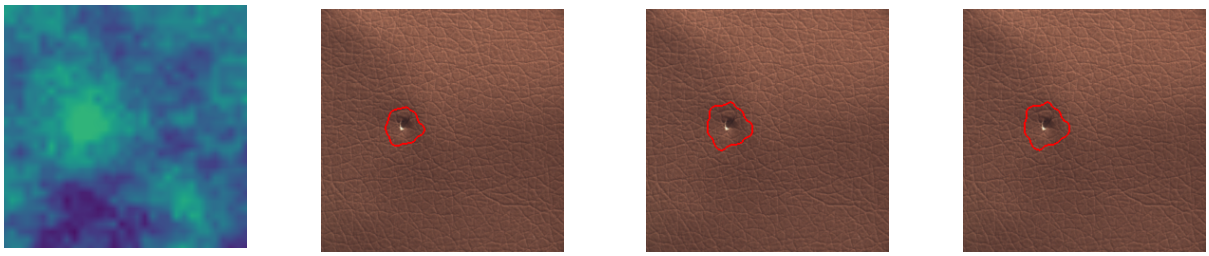


Figure 63: MVTEC poke sample results with CFLOW.

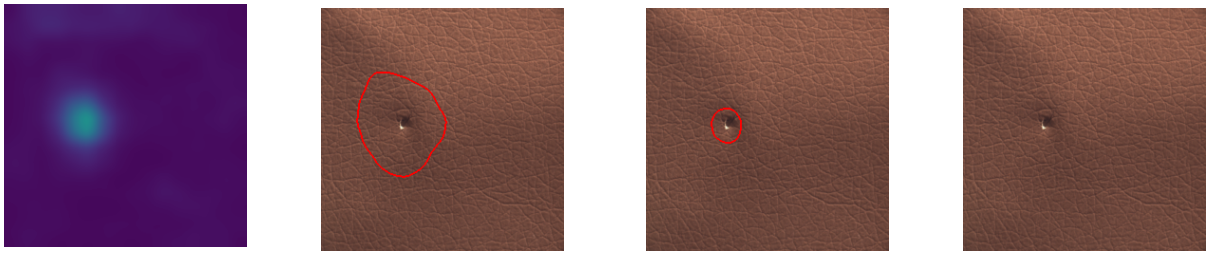


Figure 64: MVTEC poke sample results with STFPM.

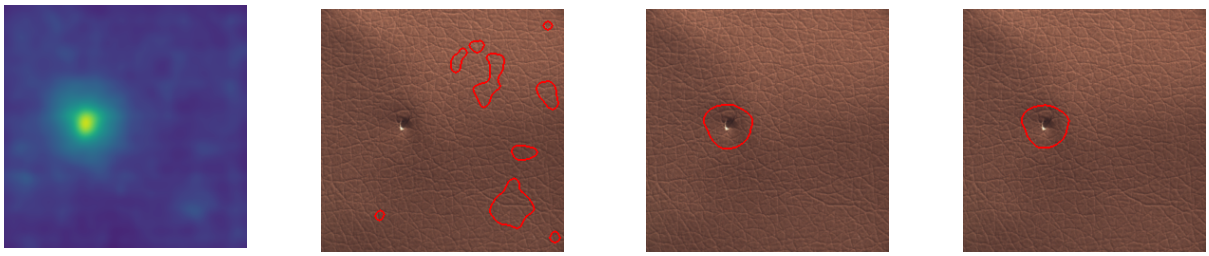


Figure 65: MVTEC poke sample results with Reverse.

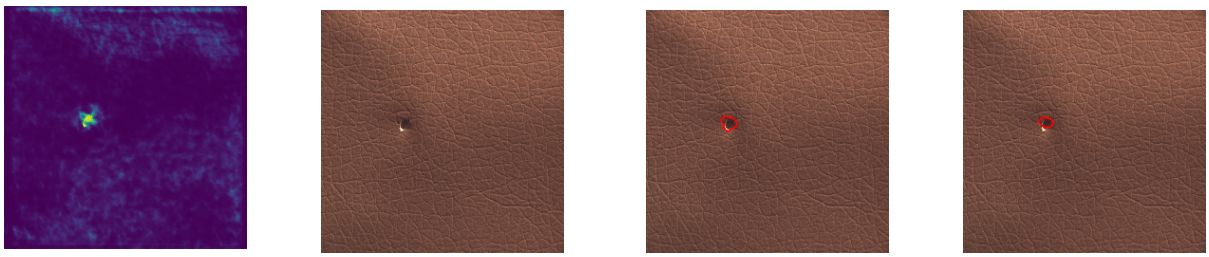


Figure 66: MVTEC glue sample results with DRAEM.

A.6 Neadvance cut sample results

Figure 67: Neadvance cut sample results with MSE AE.

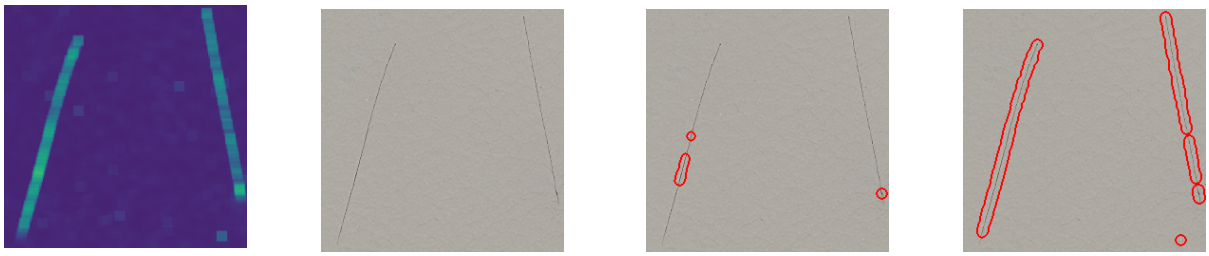


Figure 68: Neadvance cut sample results with SSIM AE.

Figure 69: Neadvance cut sample results with CFLOW.

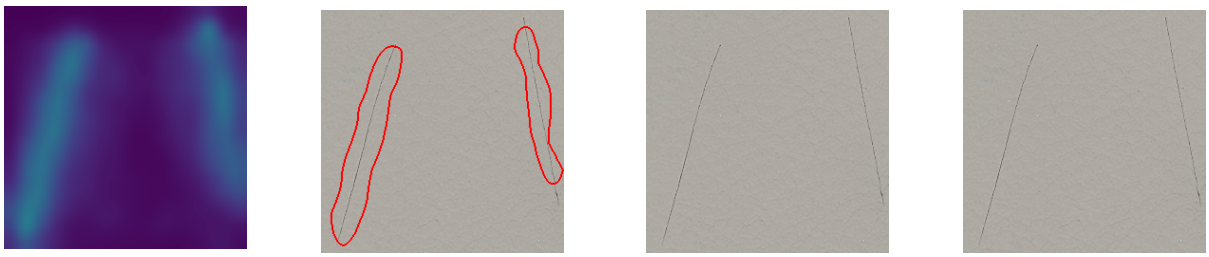


Figure 70: Neadvance cut sample results with STFPM.

Map T1 T2 T3

Figure 71: Neadvance cut sample results with Reverse.

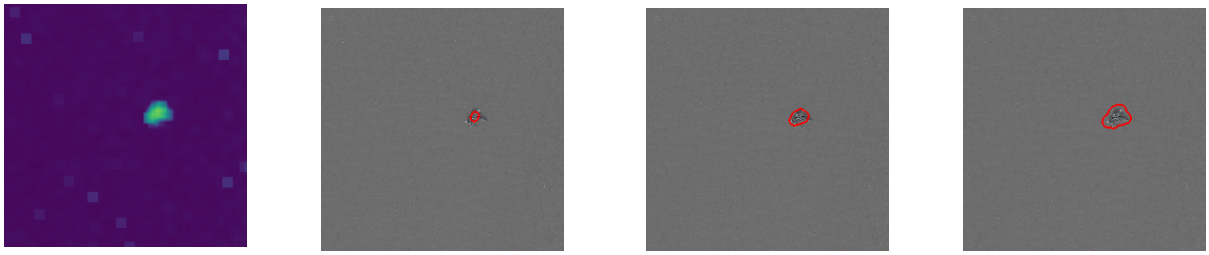
Map T1 T2 T3

Figure 72: Neadvance cut sample results with DRAEM.

A.7 Neadvance hole sample results

Map T1 T2 T3

Figure 73: Neadvance hole sample results with MSE AE.



Map T1 T2 T3

Figure 74: Neadvance hole sample results with SSIM AE.

Figure 75: Neadvance hole sample results with CFLOW.

Figure 76: Neadvance hole sample results with STFPM.

Figure 77: Neadvance hole sample results with Reverse.

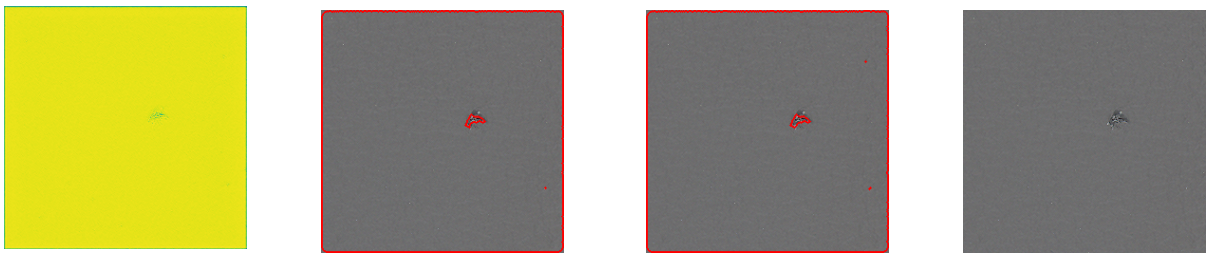


Figure 78: Neadvance hole sample results with DRAEM.

A.8 Neadvance line sample results

Figure 79: Neadvance line sample results with MSE AE.

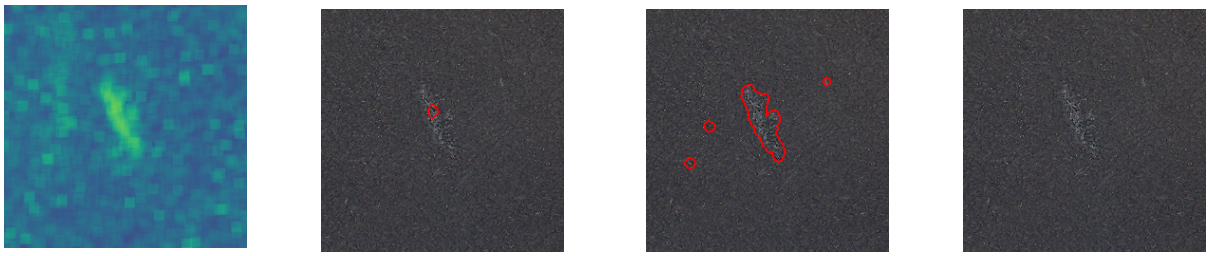


Figure 80: Neadvance line sample results with SSIM AE.

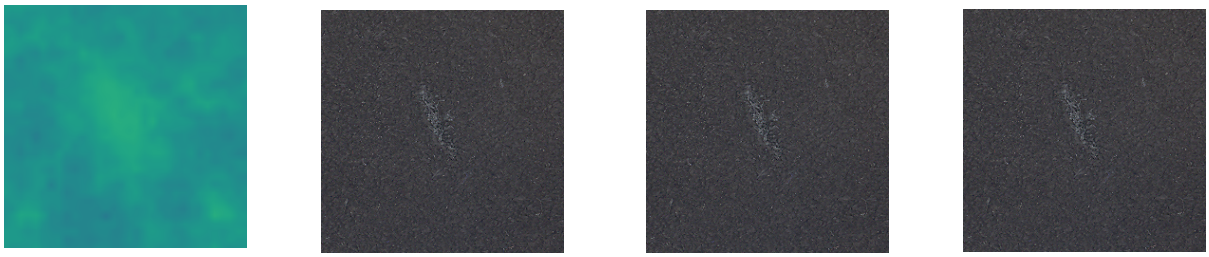


Figure 81: Neadvance line sample results with CFLOW.

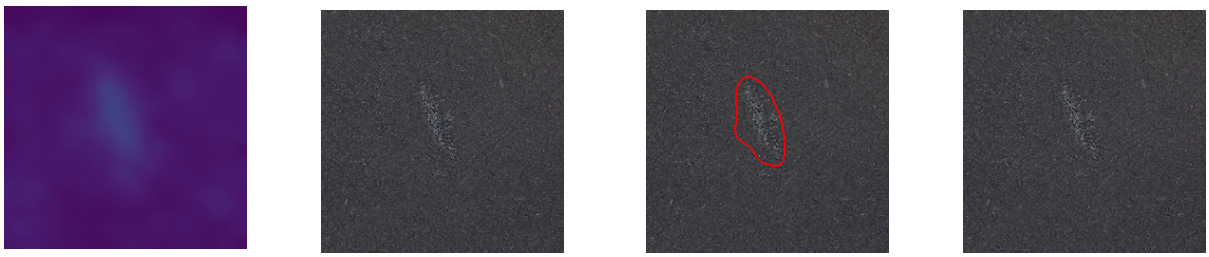


Figure 82: Neadvance line sample results with STFPM.

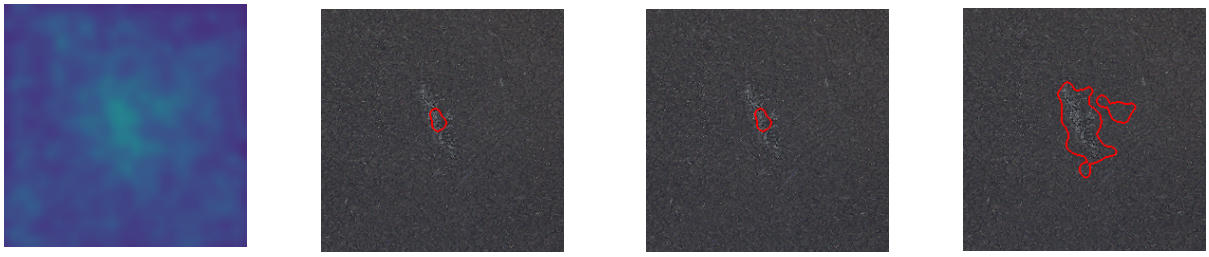


Figure 83: Neadvance line sample results with Reverse.

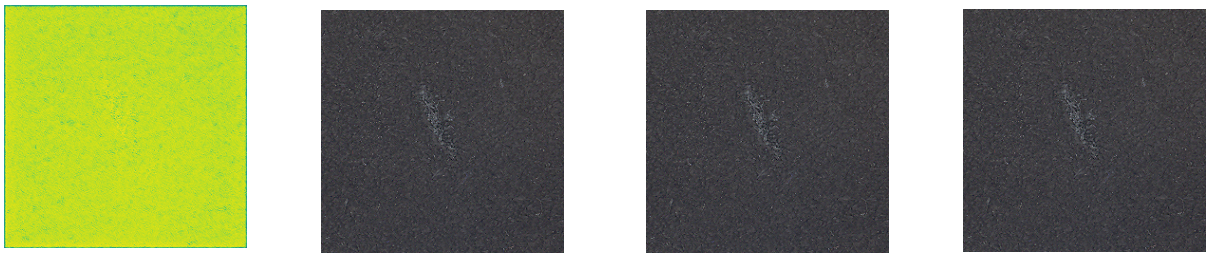


Figure 84: Neadvance line sample results with DRAEM.

A.9 Neadvance wrinkle sample results

Figure 85: Neadvance wrinkle sample results with MSE AE.

Figure 86: Neadvance wrinkle sample results with SSIM AE.

Figure 87: Neadvance wrinkle sample results with CFLOW.

Figure 88: Neadvance wrinkle sample results with STFPM.

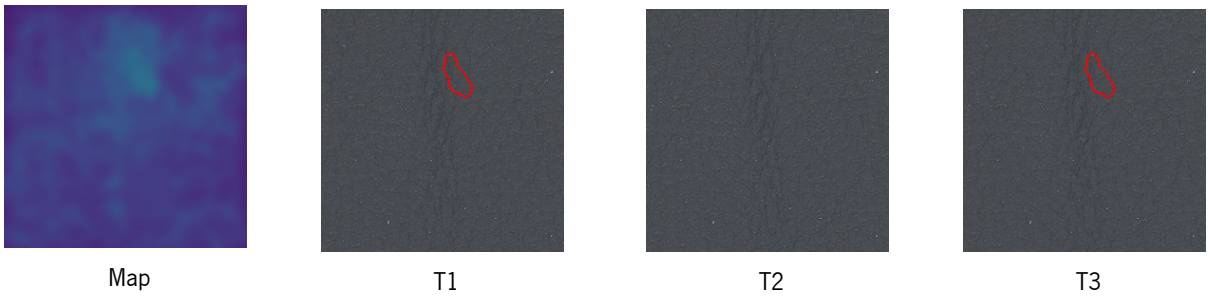


Figure 89: Neadvance wrinkle sample results with Reverse.

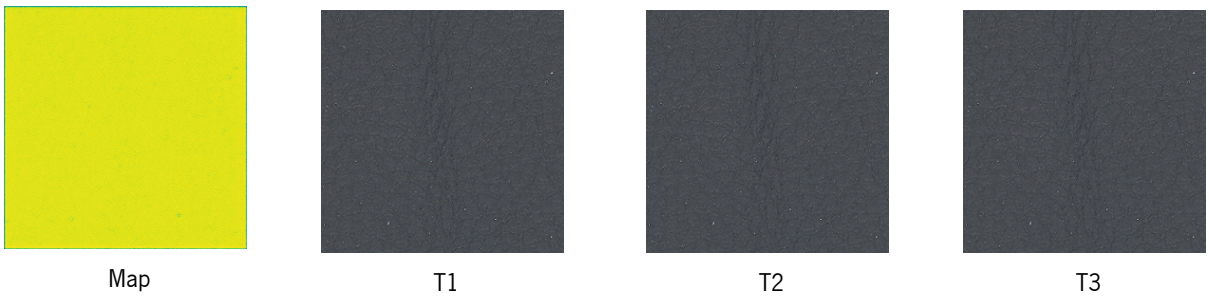


Figure 90: Neadvance wrinkle sample results with DRAEM.