ISSN 1688-2784

Universidad de la República
Facultad de Ingeniería

Deep Image Generative Modeling and Statistical Testing for Industrial Anomaly Detection

Thesis presented to Facultad de Ingeniería de la Universidad de la República by

Matías Tailanian

in partial fulfillment of the requirements for the degree of Doctor en Ingeniería Eléctrica.

Thesis advisors

Pablo Musé ..... Universidad de la República & Digital Sense
Álvaro Pardo ..... Universidad Católica del Uruguay & Digital Sense

Thesis Committee

Matías Di Martino ..... Universidad Católica del Uruguay
Thibaud Ehret ..... ENS Paris Saclay, France
Jean-Michel Morel ..... City University of Hong Kong
Gregory Randall ..... Universidad de la República, Uruguay

Academic Director

Pablo Musé ..... Universidad de la República

Montevideo
Tuesday 17th September, 2024

This page was intentionally left blank.

Thanks

Thanks to Oli, who came into our lives during this period, to change and put everything into perspective. And thanks to Gabi for your love, patience and belief in me, making this journey more bearable and meaningful.

I would also like to thank my family and friends, who accompanied me during this whole process, which was very difficult, but it was much more enjoyable.

I extend an special thanks to my advisors Pablo Musé and Álvaro Pardo, whose contributions have been fundamental. They have shared their extensive knowledge, enthusiasm, dedication, time, and a lot of patience. We have shared the most interesting and infinite discussions, generating an inabarcable amount of ideas, some of which ended up in tangible pieces of work.

On the other hand, I would like to thank the School of Engineering of Universidad de la República, Uruguay, for the solid technical basis provided and for the possibility of doing this thesis. To Digital Sense for the support, help, and possibilities opened. And to the Agencia Nacional de Investigación e Innovación, Uruguay, for the grant that partially funded this research.

Last but not least, I express my gratitude to the committee members for their willingness to review this manuscript. Your expertise and thoughtful consideration will undoubtedly contribute to the quality and completeness of this work.

This page was intentionally left blank.

Abstract

This thesis addresses the challenge of anomaly detection in images, for industrial applications. It explores advanced methodologies employing both classical image processing techniques and modern generative modeling approaches, specifically focusing on Normalizing Flows and Diffusion Models.

As anomalies are rare by definition, collecting normal samples is generally easier and more feasible in industrial settings than acquiring comprehensive datasets with labeled anomalies. Therefore, the focus of this research is on unsupervised methods, and one-class methods, where the idea is to model the "normality" and detect anomalies as everything that deviates from this model.

Initially, a multi-scale anomaly detection method based on classical image processing techniques is proposed, leveraging an *a contrario* approach to control the number of false alarms. Subsequently, a novel method called U-Flow is introduced, which employs a U-shaped architecture in Normalizing Flows to achieve anomaly detection with automatic thresholding. Then, this thesis further explores the use of Diffusion Models for anomaly detection, presenting the Diffusion Anomaly Detection (DAD) method. This work incorporates score-based generative models and inpainting techniques to refine anomaly detection capabilities. Additionally, a new diffusion-based method called RIFA (Random Inpainting For Anomaly detection) is proposed as a completely unsupervised alternative. Finally, the techniques and knowledge gained from Diffusion Models are applied to a completely different application: counter-forensics.

Throughout the whole thesis, a special emphasis is placed on bridging the gap between theoretical research and practical industrial applications, setting the theoretical foundations for obtaining automatic segmentations of anomalies, by performing statistical tests and controlling the number of false alarms using the *a contrario* framework.

Experimental results on standard datasets validate the effectiveness of the proposed methods, highlighting substantial performance gains in some cases. The final chapter applies the best-performing method to two industrial problems: quality control in manufacturing leather samples for the upholstery industry, and defect detection in fruits, demonstrating its practical viability and impact on improving quality control processes in these industries.

In addition, this research contributes to the open-source community with several code repositories and has resulted in four published papers so far, and hopefully, more will follow. Future work will particularly focus on improving likelihood estimation with Diffusion Models and expanding its applicability to other industrial domains.

Contents

This page was intentionally left blank.

Chapter 1

Introduction

Recently, the integration of advanced technologies such as artificial intelligence (AI) has revolutionized industrial operations, leading to increased efficiency, safety, and productivity. Among the various applications of AI, computer vision and image processing stand out as crucial in automating and optimizing visual inspection tasks.

The main goal of this thesis is to explore and propose advanced methodologies for anomaly detection in images specifically tailored to industrial environments. Over the past three years, I have had the privilege of engaging in this dynamic field both as a Ph.D. researcher and a practitioner in the industry. This dual role has helped me to bridge theoretical research with practical, real-world industrial applications. Guided by personal backgrounds and interests, as well as recent work experience in the area, this thesis was conceived from the beginning to employ two approaches: a classical image processing approach and a more recent machine learning approach. Moreover, it seeks to amalgamate both approaches into methods that can exploit both benefits, attempting to close the classic gap that often separates them and makes them seem independent.

Specifically, the primary objective of this thesis is to advance the state-of-the-art in anomaly detection in industrial images, focusing on the following key areas:

- **Classical image processing for anomaly detection**: Exploring traditional image processing techniques to detect anomalies in images.
- **Deep learning for anomaly detection**: Investigating the use of convolutional neural networks and other deep learning architectures to detect anomalies in images automatically. This involves exploring various network architectures, training strategies, and more.
- **Unsupervised and semi-supervised learning**: Given the scarcity of labeled anomaly data in many industrial contexts, this thesis examines the potential of unsupervised and semi-supervised learning techniques. These approaches aim to identify anomalies without extensive labeled datasets, leveraging the intrinsic patterns in normal data to detect deviations.

- **Application-specific solutions**: Tailoring anomaly detection techniques to specific industrial applications, such as quality control in manufacturing, predictive maintenance, and safety monitoring. This includes case studies and pilot implementations that demonstrate the practical viability and benefits of the proposed methods.

Importantly, the machine learning models used in this work are not treated as black boxes. Instead, their mathematical underpinnings and the models behind them are thoroughly analyzed and discussed.

All the projects share a common factor: using the a contrario theory to perform statistical hypothesis tests. Thus, the intention to combine both approaches is evident even in this amalgamation.

Subsequent chapters will detail the methodologies, experiments, and results that underscore this work's contributions. They ultimately aim to pave the way for more intelligent and autonomous industrial systems.

1.1 Anomaly detection

Anomaly detection refers to identifying patterns in data that do not conform to expected behavior. In the context of image analysis, anomalies can manifest as defects, irregularities, or any deviations from the norm that might indicate issues in manufacturing processes, equipment malfunction, or potential safety hazards. Various examples of different anomalies in industrial images are shown in Figure 1.1. All anomalies are very different from each other. Some of them are easy to identify, but for others, problem-specific information is needed to understand what an anomaly and normality are. Traditional anomaly detection methods in industrial settings often rely on manual inspection, which can be labor-intensive, time-consuming, and prone to human error. The advent of AI and machine learning offers promising alternatives that can significantly enhance these inspections' accuracy, speed, and reliability.

Anomaly detection is often approached as a one-class problem due to several key factors related to the nature of the data and the practicalities of real-world applications. Anomalies are inherently rare, occurring significantly less frequently than normal events or conditions. This rarity makes it challenging to gather a comprehensive set of abnormal samples. Furthermore, anomalies can manifest in numerous and unpredictable ways, leading to high variability in their appearance, and complicating the collection of representative examples.

Labeling anomalies is another significant challenge. In industrial settings, identifying and labeling defective products or abnormal conditions often requires expert knowledge and a considerable amount of time. On the other hand, normal samples are typically abundant and easier to label accurately.

In many applications, particularly in industrial environments, the primary focus is ensuring systems operate normally and efficiently. By modeling normal behavior, systems can effectively flag deviations as potential anomalies, aligning with practical needs in quality control, maintenance, and safety monitoring. One-class approaches focus on learning the characteristics of normal data, which

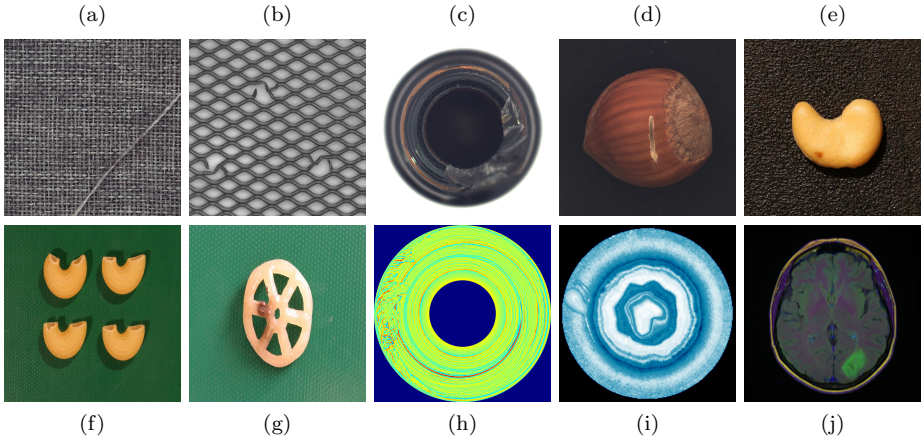


Figure 1.1: Example images with different kinds of anomalies from different datasets. Image (a), (b), (c), and (d) correspond to different categories of the MVTec-AD dataset [13] (carpet, grid, bottle, and hazelnut, respectively). Images (e), (f), and (g) are examples from the VISA dataset [180] (categories cashew, macaroni1, and fryum, respectively). Images (h) and (i) are examples from the Bean Tech dataset [106], and (j) from LGG-MRI [21].

tends to be more consistent and homogeneous. This allows the model to generalize well to unseen normal samples and identify deviations from this learned representation as anomalies.

Given these considerations, one-class anomaly detection methods are particularly well-suited for these scenarios, where normal samples are plentiful, anomalies are rare and varied, and the goal is to maintain normal operations by identifying deviations. This approach enables the effective detection of anomalies without requiring an exhaustive and often impractical collection of abnormal samples.

1.2 Industrial problems addressed

As previously mentioned, this thesis was developed while working in parallel in the industry at Digital Sense. In some of the projects addressed, we were able to apply the algorithms created for this thesis to real-world applications. This allowed us to validate and refine the methodologies, ensuring their practical viability and effectiveness in industrial settings.

Particularly noteworthy are the following two projects, which are explained below:

- Bader: defect detection in leather, and
- Sienz: detection of problematic areas in fruits.

1.2.1 Bader: defect detection in leather

Bader is a world-leading producer of leather upholstery for the high-end automotive industry. Some of their clients are Audi, BMW, Lamborghini, Porsche and Tesla.

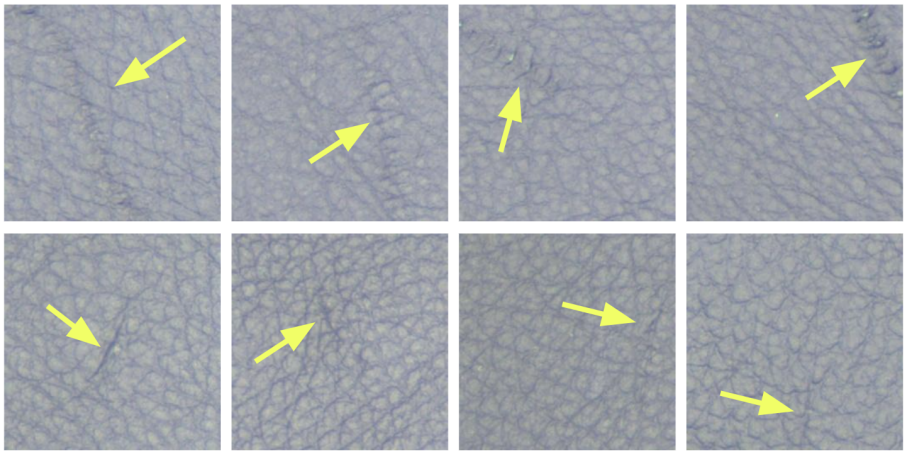


Figure 1.2: Example images from Bader project: detecting defects in leather samples. The images from the top row show defects introduced during the manufacturing process, while the bottom row shows defects that come from wounds that the animal had. Some defects are evident, but others are very difficult to see with the naked eye. An arrow indicating the anomaly was added to each image to ease their identification.

Each vehicle manufacturer allows a small percentage of defects in the leather pieces they buy. When this margin is exceeded, the entire batch of leather pieces is returned, incurring large losses for the producing companies.

Defects are manually searched in all samples at various points in the process, which is a bottleneck in the plant's production capacity. Moreover, this manual process suffers from at least these important drawbacks:

- Criteria difference,
- Difficulties in personnel availability,
- Performance degradation in the afternoon, when workers are more tired.

Example images from this project are shown in Figure 1.2. As can be seen, although some defects are easy to see, others are extremely difficult to detect. This difficulty gets even harder when we compare the images with the anomaly-free samples shown in Figure 1.3. Comparing both sets of images, we conclude that the line that divides the anomalies from the normal texture is very thin and sometimes fuzzy.

1.2.2 Sienz: detection of problematic areas in fruits

Sienz is a forward-thinking technology provider specializing in packaging plant solutions. They are currently developing an advanced automated quality control line for citrus packaging.

Citrus fruit defects can vary in appearance and location, sometimes subtle or beneath the skin. Maintaining high efficiency is also crucial to prevent slowing

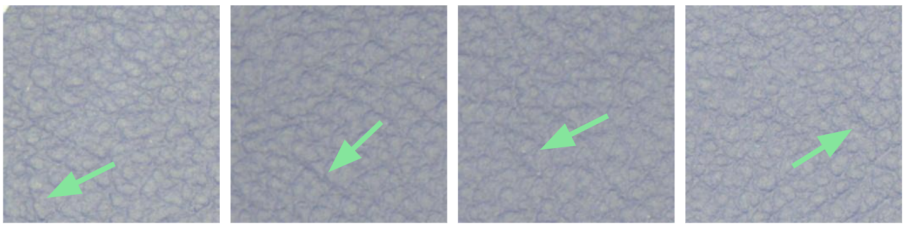


Figure 1.3: Example anomaly-free images from Bader project. The texture of the normal samples is diverse, and it is easy to confound some normal structures with anomalies. Green arrows indicate some normal structures that are similar to some anomalies.

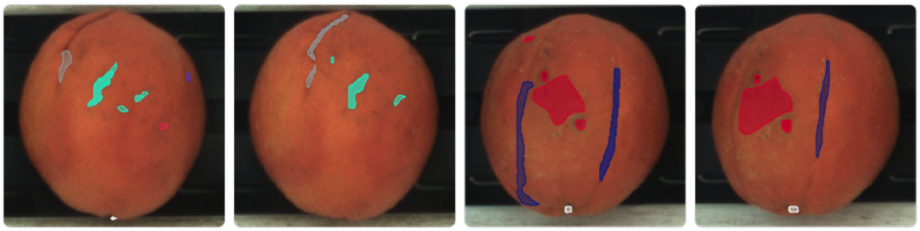


Figure 1.4: Oranges examples from the Sienz project, depicting label inconsistencies. The first two images correspond to one orange, and the last two to another orange. Two different labelings for each orange are shown, made by different operators. It can be easily seen that in both cases there are big differences in the labeling, which may create inconsistencies at the training time.

down the high-volume packaging line. To do so, fast computer vision algorithms are included in the line to classify citrus fruit into different quality brackets.

The overall project includes real-time detection and classification of around 20 types of defects on fruits of any kind. Each fruit is captured in motion, passing through a production line, from at least three different RGB cameras. Depending on the machine, there can also be one or two types of IR cameras.

The computer vision algorithm aims to detect and co-register defects in each image so that each defect is only detected once.

In this case, as in the previous project (Section 1.2.1), we also have the criteria difference problem. Different operators usually label the defects differently. Moreover, the same person sometimes labels different images of the same fruit differently. This behavior is depicted in Figure 1.4, where there are evident label inconsistencies between different views of the same fruit.

These label inconsistencies are one of the reasons why a one-class approach that only tries to characterize normality (i.e., healthy fruits) could be helpful, either to accelerate the labeling process in the case of a supervised approach or to consider semi-supervised strategies.

Toward the end of the manuscript, in Chapter 8, we will return to both applications to show some results obtained using the anomaly detection methods proposed in this thesis.

Figure 1.5: Anomaly detection pipeline for industrial applications. The process starts with a real-world object, followed by capturing measurements (such as images or sensor data), extracting relevant features, applying transformations or feature adaptation, and concluding with statistical testing to identify potential anomalies.

1.3 Pipeline

The complete pipeline we typically follow for solving an anomaly detection project is illustrated in Figure 1.5. Throughout this thesis, we will demonstrate that all the methods developed align with this pipeline. The primary stages are as follows:

- **Real-World Object**: Everything starts with a real sample. This is the physical system or industrial process we are monitoring for potential anomalies. The data collected here can represent machinery, production processes, or any tangible elements being observed for irregular behavior. In the case of the Bader project, for example, the objects are the leather samples that are put into a specifically designed cabin, for controlling the acquisition conditions.

- **Measurements (Images)**: Data is gathered from the system, often in the form of sensor readings, images, or other measurements that reflect the state of the object. These measurements capture the observable variables that may indicate normal or abnormal behavior. Following the same example, in the Bader project, these measurements are the images taken for each sample inside the cabin.

- **Features**: From these measurements, relevant features are extracted. These features, based on visual aspects of a product (e.g., texture or shape), provide a structured way of representing the data for analysis. Depending on the method, these features could come from very different techniques and, therefore, have different characteristics.

- **Transformations / Feature Adaptation**: The extracted features are processed or transformed to ensure that they are suitable for analysis. This stage might involve different transformations or mappings, and the main idea is to make the features more interpretable or prepare them for the next stage: statistical testing.

- **Statistical Testing**: Finally, statistical analysis is performed to detect anomalies. In most cases found in the literature, the outcome of anomaly detection methods is an anomaly map, which assigns a score to each pixel indicating its level of abnormality. However, in practical applications, we often require an actual segmentation of the anomaly—classifying

each pixel as either anomalous or not. Instead of determining thresholds through supervised learning, our methods rely on a multiple-hypothesis testing approach, specifically the *a contrario* framework, to compute automatic segmentation thresholds.

This structured approach allows for a systematic way of analyzing industrial data and detecting anomalies, ensuring each step builds upon the previous one to facilitate accurate and timely identification of potential issues.

In the concluding chapter (Chapter 9), it will be explicit how each new method enhances specific stages of this process.

1.4 Thesis' structure

This manuscript is structured in the following chapters:

Chapter 2: Background. This chapter formally introduces the anomaly detection problem and gives a general overview of the state of the art. Later, in chapters 3 and 5, a more detailed state-of-the-art review is presented, with a more specific focus for each case. In addition, this chapter introduces deep generative modeling and the *a contrario* methodology used throughout this thesis. These two important topics are revisited later more in-depth, explaining the specific techniques that are employed.

Chapter 3: A Contrario Multi-Scale Anomaly Detection Method for Industrial Quality Inspection. This chapter reports the first anomaly detection method published during this thesis. It is an unsupervised algorithm built on traditional image processing methods. This work led to two publications. The first version of it [143] was published as a conference paper, and an extended version of it with some improvements was reported as a book chapter [144].

Chapter 4: Normalizing Flows. As one of the most important outcomes of this thesis uses Normalizing Flows as one of the key parts, we include a detailed description of this type of deep generative model in this chapter.

Chapter 5: U-Flow: A U-shaped Normalizing Flow for Anomaly Detection with Unsupervised Threshold. This chapter reproduces the content of the article published in Springer Journal of Mathematical Imaging and Vision [145]. In this work, a method using Normalizing Flows is able to derive an anomaly score based on the likelihood estimation of the test sample. Also, it incorporates a segmentation of anomalies using the *a contrario* methodology.

Chapter 6: Diffusion Models. Similarly to Chapter 4, this chapter presents a detailed introduction to Diffusion Models, as it is the basis of the methods described in the following chapters.

Chapter 7: DAD: Diffusion Anomaly Detection. We present a method that uses score-based Diffusion Models and, by solving an ordinary differential equation on the probability flow, produces a likelihood estimation for the test samples that can be used as an anomaly score and to perform anomaly segmentation. This is a preliminary version of a work that will be submitted for publication soon. This chapter also includes a completely unsupervised variant of the previous method, called RIFA: Random Inpainting For Anomaly detection. It is based on generating several samples by inpainting random blocks of the test image and evaluating the statistics of the difference with the input image. This work will also be submitted for publication in the coming weeks.

Chapter 8: Anomaly detection in two industrial problems. This chapter shows some results of the industrial applications presented in Section 1.2.

Chapter 9: Conclusion and Future Work. In this chapter, some conclusions and future work are presented.

Appendix A: Diffusion Models meet image counter-forensics. This appendix reproduces the work published in WACV 2024 [142]. It is included as an appendix as it addresses a task different from anomaly detection. Nonetheless, this work was also developed as a different application for Diffusion Models, a deeply studied technique for this thesis.

Appendix B: Datasets and Metrics. This chapter presents the datasets and metrics used to evaluate the different methods developed in this thesis. We focus on the most popular datasets in the literature to easily compare our methods with the state of the art. Regarding metrics, as we propose methods that not only obtain anomaly scores but also perform a segmentation of the anomalies, we present metrics for both assessing the obtained anomaly scores and the segmentation masks.

1.5 List of publications

- **Tailanian, M., Musé, P., & Pardo, Á. (2021). A multi-scale a contrario method for unsupervised image anomaly detection. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 179-184).**

  Anomalies can be defined as any non-random structure which deviates from normality. Anomaly detection methods reported in the literature are numerous and diverse, as what is considered anomalous usually varies depending on particular scenarios and applications. In this work we propose an a contrario framework to detect anomalies in images applying statistical analysis to feature maps obtained via convolutions. We evaluate filters learned from the image under analysis via patch PCA, Gabor filters and the feature maps obtained from a pre-trained deep neural net-

work (Resnet). The proposed method is multiscale and fully unsupervised and is able to detect anomalies in a wide variety of scenarios. While the end goal of this work is the detection of subtle defects in leather samples for the automotive industry, we show that the same algorithm achieves state-of-the-art results in public anomalies datasets.

- **Tailanian, M., Musé, P., & Pardo, Á. (2022). A Contrario multi-scale anomaly detection method for industrial quality inspection. In *Deep Learning Applications, Volume 4* (pp. 193-216). Springer Nature.**

  Anomalies can be defined as any non-random structure which deviates from normality. Anomaly detection methods reported in the literature are numerous and diverse, as what is considered anomalous usually varies depending on particular scenarios and applications. In this work we propose an a contrario framework to detect anomalies in images applying statistical analysis to feature maps obtained via convolutions. We evaluate filters learned from the image under analysis via patch PCA, Gabor filters and the feature maps obtained from a pre-trained deep neural network (Resnet). The proposed method is multiscale and fully unsupervised and is able to detect anomalies in a wide variety of scenarios. While the end goal of this work is the detection of subtle defects in leather samples for the automotive industry, we show that the same algorithm achieves state-of-the-art results in public anomalies datasets.

- **Tailanian, M., Gardella, M., Pardo, Á., & Musé, P. (2024). Diffusion Models meet image counter-forensics. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 3925-3935).**

  From its acquisition in the camera sensors to its storage, different operations are performed to generate the final image. This pipeline imprints specific traces into the image to form a natural watermark. Tampering with an image disturbs these traces; these disruptions are clues that are used by most methods to detect and locate forgeries. In this article, we assess the capabilities of Diffusion Models to erase the traces left by forgers and, therefore, deceive forensics methods. Such an approach has been recently introduced for adversarial purification, achieving significant performance. We show that diffusion purification methods are well suited for counter-forensics tasks. Such approaches outperform already existing counter-forensics techniques both in deceiving forensics methods and in preserving the natural look of the purified images. The source code is publicly available at `https://github.com/mtailanian/diff-cf`.

- **Tailanian, M., Pardo. Á., and Musé, P. (2024). U-flow: A u-shaped Normalizing Flow for anomaly detection with unsupervised threshold. Journal of Mathematical Imaging and Vision.**

In this work, we propose a one-class self-supervised method for anomaly segmentation in images that benefits both from a modern machine learning approach and a more classic statistical detection theory. The method consists of four phases. First, features are extracted using a multi-scale image Transformer architecture. Then, these features are fed into a U-shaped Normalizing Flow (NF) that lays the theoretical foundations for the subsequent phases. The third phase computes a pixel-level anomaly map from the NF embedding, and the last phase performs a segmentation based on the a contrario framework. This multiple-hypothesis testing strategy permits the derivation of robust unsupervised detection thresholds, which are crucial in real-world applications where an operational point is needed. The segmentation results are evaluated using the Mean Intersection over Union (mIoU) metric, and for assessing the generated anomaly maps, we report the area under the Receiver Operating Characteristic curve (AUROC), as well as the Area Under the Per-Region-Overlap curve (AUPRO). Extensive experimentation in various datasets shows that the proposed approach produces state-of-the-art results for all metrics and all datasets, ranking first in most MVTec-AD categories, with a mean pixel-level AUROC of 98.74%. Code and trained models are available at `https://github.com/mtailanian/uflow`.

Chapter 2

Background

This chapter presents a formal introduction to the anomaly detection problem and provides a brief overview of the state of the art. A more detailed review is reserved for Chapters 3 and 5, which correspond to two published papers and include comprehensive state-of-the-art reviews. Therefore, this chapter offers only a concise summary, highlighting the most important family of methods in the literature.

2.1 Anomaly Detection

Anomaly detection is a methodology used to identify data points, events, or observations that deviate significantly from the norm within a given dataset. Formally, anomaly detection can be defined as follows: Given a dataset $\mathcal{X} = \{x_1, x_2, \dots x_n\}$, where $x_i \in \mathbb{R}^d$ represents a data point in a d-dimensional space, the goal of anomaly detection is to find a subset $\mathcal{A} \subset \mathcal{X}$ consisting of data points that do not conform to the expected distribution or pattern of the normal data. These non-conforming data points are referred to as anomalies, outliers, or rare events. Mathematically, an anomaly can be defined as:

$$\mathcal{A} = \{x_i \in \mathcal{X} / f(x_i) > \epsilon\}, \tag{2.1}$$

where $f(x)$ is an anomaly score function that quantifies the degree of deviation of x from the norm, and ϵ is a threshold value such that data points with an anomaly score above ϵ are considered anomalies. The function $f(x)$ is typically derived from statistical models, machine learning algorithms, or other analytical techniques that characterize normal behavior based on the provided data.

2.2 Related Works

Anomaly detection is a well-studied problem in data analysis, with a broad range of methods developed over the years. These methods can be broadly categorized into several families based on their underlying principles and techniques. This section introduces these different family types of methods, providing a brief

overview of each.

Statistical methods for anomaly detection are among the earliest approaches that were developed. These methods assume that normal data points follow a certain statistical distribution, and anomalies are points that deviate significantly from this distribution. Key techniques in this category include parametric methods, which assume that the data follows a known parametric distribution (e.g., Gaussian distribution), and non-parametric methods, which do not assume a specific data distribution and instead rely on data-driven techniques such as kernel density estimation to model the data distribution and detect anomalies.

Proximity-based methods detect anomalies by examining the distances or similarities between data points. The fundamental idea is that normal data points are close to their neighbors, while anomalies are isolated. Prominent techniques in this category include distance-based methods, which calculate the distance between data points using metrics like Euclidean distance, and density-based methods, which estimate the local density of data points, identifying anomalies as points located in low-density regions. The LOF (Local Outlier Factor) algorithm is a well-known example.

Clustering-based methods group data points into clusters and identify anomalies as points that do not fit well into any cluster. Key approaches include partitioning methods, such as DBSCAN [54] clustering, which partition the data into clusters and identify points that do not belong to any cluster as anomalies, and hierarchical methods, which build a hierarchy of clusters and identify anomalies at various levels of the hierarchy.

Classification-based methods treat anomaly detection as a supervised classification problem. They require labeled data for training and can be further divided into binary classification methods, which train a classifier using labeled normal and anomaly data and identify anomalies based on the classifier's predictions, and one-class classification methods, which train a model using only normal data to recognize the normal class, and any deviations from this class are considered anomalies. The One-Class SVM (Support Vector Machine) is a common technique in this category.

Reconstruction-based methods rely on the ability to reconstruct normal data accurately. Anomalies are identified as points that are poorly reconstructed. Prominent approaches include Principal Component Analysis (PCA), which reduces the data to its principal components and considers points that cannot be well-represented by these components as anomalies, and non-linear auto-encoders, which use neural networks to learn to compress and then reconstruct the data, identifying anomalies through high reconstruction errors.

Deep learning methods leverage the power of neural networks to model complex data distributions and detect anomalies. These methods have gained popularity due to their ability to handle high-dimensional data. Key techniques

include deep auto-encoders, improving reconstruction quality for normal data and identifying anomalies through reconstruction errors, and Generative Adversarial Networks (GANs), which consist of a generator and a discriminator network where the generator tries to create realistic data, and the discriminator distinguishes between real and generated data, detecting anomalies based on the discriminator's performance.

Hybrid methods combine multiple approaches to leverage their strengths and improve anomaly detection performance. These methods often integrate statistical, proximity-based, and deep learning techniques to create robust detection systems.

2.3 Generative modeling

This section is particularly relevant because various generative models were studied and utilized throughout this thesis. Therefore, settling the basic concepts to understand the following chapters is important.

Generative models have emerged as a powerful class of machine learning algorithms capable of modeling complex data distributions. Unlike discriminative models, which aim to classify or predict outcomes based on input data, generative models focus on understanding and capturing the underlying data distribution. This capability allows generative models to create new data samples statistically similar to the original dataset.

2.3.1 Generative models taxonomy

The complete family of generative models is presented in Figure 2.1. Based on how they represent probability distributions, they can be easily classified into two types: likelihood-based models (explicit density) and implicit density models.

Likelihood-based models use (approximate) maximum likelihood to learn the distribution's probability density function. Examples of likelihood-based models include autoregressive models [90, 148], Normalizing Flow models [47, 48], energy-based models (EBMs) [93,139], and variational auto-encoders (VAEs) [81].

In implicit generative models, the probability distribution is implicitly represented by a model of the sampling process. The most visible example is generative adversarial networks (GANs) [63], which create new samples from the data distribution by modifying a random Gaussian vector using a neural network.

The models that are able to estimate the exact density directly give the value of $p(x)$. In contrast, the models that predict the density implicitly are not able to output the value of $p(x)$ but are still able to sample from it. The explicit density methods can be further divided into the ones that can approximate the probability density, such as Variational Auto-Encoders (VAEs), and the methods that can estimate the exact probability density, like Normalizing

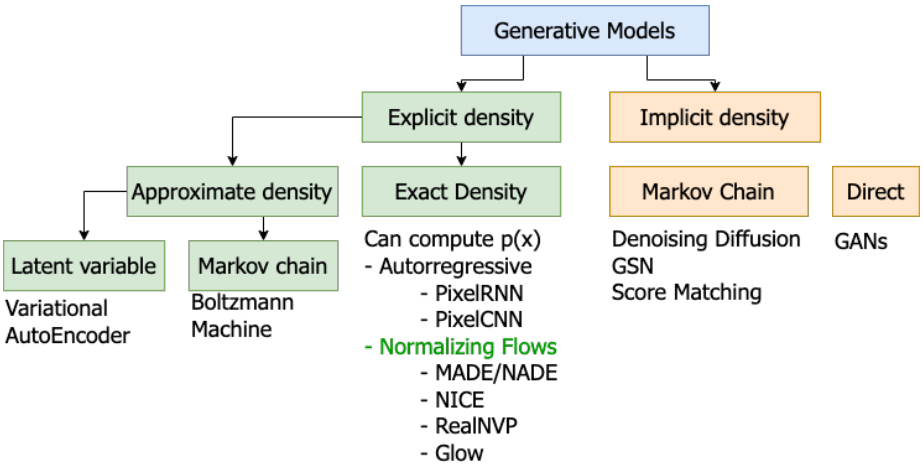


Figure 2.1: Generative models taxonomy. Diagram adapted from Ian Goodfellow's presentation in Khipu, 2019.

Flows (NF).

Being able to represent a given dataset with a probability distribution enables many possibilities, such as evaluating the likelihood at test time, which allows one to score how good the model is and to evaluate how probable it is that the tested data belong to the same distribution, or to compute conditional probabilities, which can be useful for example for building conditional models. In addition, knowing the probability of data enables us to easily generate new data by simply sampling from the learned distribution and to use some complexity measures like entropy or mutual information [75].

Figure 2.2 provides an overview of four different types of generative models: Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAEs), Flow-based models, and Diffusion models. Each model is represented by a diagram showing the key components and processes in generating new data samples.

GANs intelligently manage to convert the generation process, which is inherently unsupervised, into a supervised approach. The process involves adversarial training between a generator and a discriminator. By solving a *minimax* optimization problem, the Discriminator learns to distinguish between real and fake data, at the same time as the Generator learns to generate images each time closer to the real ones.

VAEs focus on maximizing the variational lower bound. This involves an encoder that maps input data to a latent space and a decoder that reconstructs the data from the latent representation. The training objective is to maximize the likelihood of the data under the model while ensuring that the latent space is structured.

Flow-based models use invertible transformations of distributions. The flow

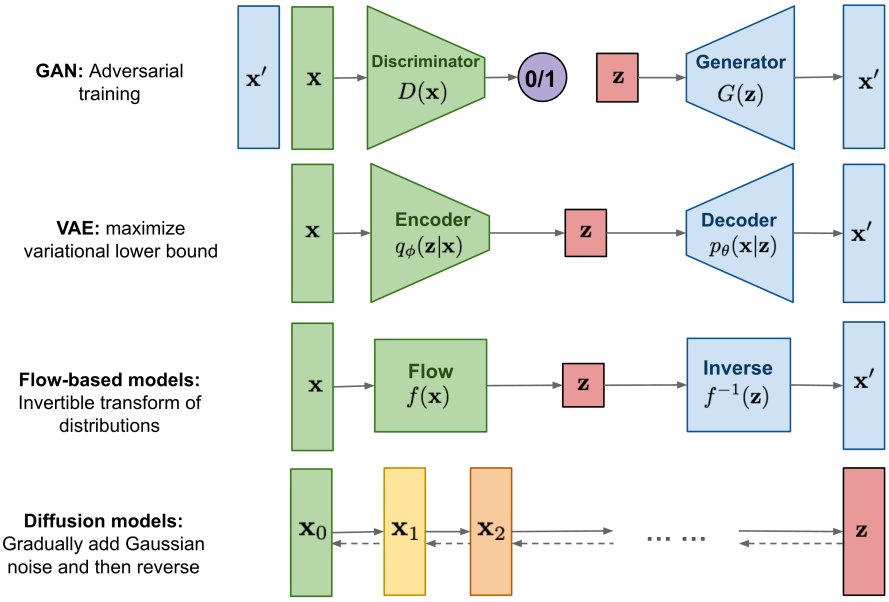


Figure 2.2: Comparison of four categories of generative models. Diagram taken from [159].

model transforms data into a latent space through invertible functions, and the inverse transformation is used to generate new data from the latent representation. These models allow for exact likelihood estimation and data generation.

Diffusion models involve a process that gradually adds Gaussian noise to the data step-by-step. The reverse process aims to reconstruct the original data from the noisy data, effectively denoising it. The final latent representation is obtained after a series of transformations. These models are particularly effective for generating high-quality samples by gradually reversing the noise addition process.

Normalizing flows and Diffusion Models are presented more in-depth in Chapters 4 and 6, respectively, as they were used in this thesis for developing anomaly detection methods.

2.4 The *a contrario* approach and number of false alarms

Throughout this thesis, concepts and methods based on the so-called *a contrario* theory are utilized and developed. Therefore, it is important to introduce the theory and explain some of its basic concepts.

The *a contrario* approach is based on the Helmholtz principle [46]. It states that we do not perceive any structure in a uniformly random image, or, said differently, whenever a significant deviation from randomness occurs, a structure

is perceived. In essence, it means we immediately perceive whatever could not happen by chance. Similarly, in the Gestalt theory, the non-accidentalness principle states that human perception leans on the non-casual arrangement of structures. The *a contrario* theory is a probabilistic formalization of this principle. Given the fact that we do not usually know what the anomalies look like, this theory is based on the modeling of the normality by defining a null-hypothesis ($\mathcal{H}_0$), also called *background model*, and basing the detection on how likely is an event E to happen under $\mathcal{H}_0$. One of its most useful characteristics is that it allows to automatically find a detection threshold by controlling the Number of False Alarms (NFA), defined as:

$$\mathrm{NFA}(E) = N_T \mathrm{P}_{\mathcal{H}_0}(E), \tag{2.2}$$

where N_T is the number of performed tests, and $\mathrm{P}_{\mathcal{H}_0}(E)$ is the probability of the event E to happen as a realization of the background model. Besides providing an automatic threshold, the NFA value itself has a clear, intuitive meaning: it is an estimation of the expected number of times, among the tests that are performed, that a certain pattern (or event) could occur by a mere random arrangement, namely under $\mathcal{H}_0$. A low NFA value means that the observed pattern is too regular to be generated by the background model and therefore indicates a possible anomaly.

Formally, we say an event E is ϵ-meaningful if its expected number of occurrences is less than ϵ under the normality assumption. Consequently, ϵ provides an upper bound on the expected number of false detections we can obtain on anomaly-free images. We usually simply say the event is meaningful when $\epsilon \leq 1$ [46]. Moreover, this threshold can be tuned for any application-specific needs. For instance, in an industrial environment with the goal of detecting sample defects, an average value of one false detection per image may not be a good choice, as it would raise too many false alarms in normal images (one per image, on average). Further discussion on this point are addressed in Section 5.4.1.

The *a contrario* theory is particularly useful in applications where the cost of false positives is high, and it is crucial to maintain a low false alarm rate. By using the NFA as a criterion, the method provides a principled way to balance sensitivity (detecting true anomalies) and specificity (avoiding false alarms). It has been successfully applied to derive unsupervised statistical detection thresholds in a wide variety of detection problems, such as alignments for line segment detection [151], clustering [23], image forgery detection [61], and even anomaly detection [42, 65], to name a few.

The approach we propose in this work follows the same rationale that [53], where the authors revisit several anomaly detection methods from the perspective of the *a contrario* framework to derive unsupervised statistical detection thresholds.

Chapter 3

A Contrario multi-scale anomaly detection method for industrial quality inspection

Anomalies can be defined as any non-random structure which deviates from normality. Anomaly detection methods reported in the literature are numerous and diverse, as what is considered anomalous usually varies depending on particular scenarios and applications. The work we present in this chapter proposes an a contrario framework to detect anomalies in images by applying statistical analysis to feature maps obtained via convolutions. We evaluate filters learned from the image under analysis via patch PCA, Gabor filters, and the feature maps obtained from a pre-trained deep neural network (ResNet). The proposed method is multi-scale and fully unsupervised and is able to detect anomalies in a wide variety of scenarios. While the end goal of this work is the detection of subtle defects in leather samples for the automotive industry, we show that the same algorithm achieves state-of-the-art results in public anomalies datasets.

3.1 Introduction

Anomaly detection is an active field that has been studied for decades, motivated by a wide variety of practical applications, ranging from robotics and security to health care. One of the most relevant applications is automatic or aided product quality inspection in industrial production.

In most situations it is very difficult or even impossible to collect a statistically relevant sample of all types of anomalies, given its rare nature and intra-class variability. In addition, in most cases, anomalies are very subtle, and the line that separates them from normality is thin and fuzzy. These kinds of problems are typically formulated as one-class classification problems [110], where the strategy is based on learning and characterizing the statistics of normality.

The key advantage of this scheme is that only "normal" (i.e., anomaly-free) samples are needed for the training process. Then, newly tested samples are assigned an anomaly score that represents the deviation of the tested sample from the characterized normality.

A very different situation arises when normality and abnormality are not absolute notions but relative to the sample itself. Indeed, in many scenarios, the very same pattern can be considered normal or abnormal when it occurs within two different samples. This situation is depicted in Figure 3.1, which shows some examples of the data considered for the application that motivates the present work. In this application, which will be described in detail in Section 3.5, the goal is to segment defects on processed leather pieces for the automotive industry. The leather could be colored with a wide variety of tones and may present different textures, and the strength of the texture engraving may also differ. Note, for example, that the anomalous pattern in Figure 3.1(b) should be considered normal if it was present in the sample of Figure 3.1(a).

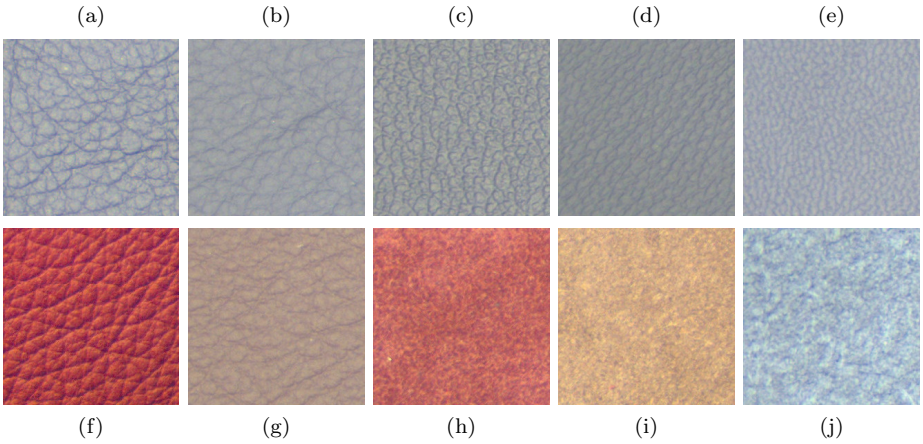


Figure 3.1: [Best viewed in color] Motivating application: defect detection in leather samples. Example images showing the diversity of samples. Image (a) and (b) correspond to the same texture, with different engraving strength. The defect present in (b) may not be considered an anomaly if it was in (a). Images (c), (d) and (e) are examples of different textures. (f) and (g) show different color possibilities, and (h), (i) and (j) correspond to the down side of leather samples.

In summary, the problem we aim to tackle is that normality can only be modeled from a single image sample, and we have no prior knowledge of whether this sample contains anomalies. These conditions frame our solution to a small set of methods, discarding all techniques based on learning from any set of images, and those that extract statistics from a set of anomaly-free samples. To this end, we developed a completely unsupervised algorithm that can handle all these variations and detect anomalies on both sides of the leather.

A well-established methodology for unsupervised anomaly detection under these conditions is the *a contrario* approach [46, 101]. This methodology is commonly used in anomaly detection and has proven to produce impressive results in many tasks, such as clustering, edges and line segments detection [22, 150], general point alignments [96], among others. In this work, inspired by this

methodology, we propose a method to detect anomalies after a step of feature extraction devised to enhance the difference between normal and anomalous patches in the image (Section 3.3).

Based on the fact that learning a statistical model of anomalies from data is not possible due to their very limited number of occurrences, the *a contrario* methodology, as other approaches, focuses on designing a *background model* or null hypothesis that characterizes normality. Following the non-accidentalness principle [161], the anomalies are detected as events such that the probability of occurrence under the background model is so small that they are much more likely to result from another cause. This principle is very intuitive. Suppose we observe a certain specific arrangement or structure in an image. The more rare this arrangement is, the less probable that it corresponds to a mere random arrangement, indicating a possible presence of an anomaly. Therefore, we need to evaluate the probability of that arrangement being generated by the background model. Another characteristic that makes the *a contrario* framework different and particularly useful is that it automatically fixes detection thresholds that control the number of false alarms (NFA) [46, 52, 91, 149], allowing not only to detect rare events in very diverse backgrounds but also to associate a rareness score. This score has a clear statistical meaning: it is an estimate of the number of occurrences of an observed event if it was produced by the background model. Thus, obtaining a very low NFA value means that it is very unlikely that the background model generated the event, and therefore, it is probably a meaningful event in the anomaly detection scheme.

In short, in this work, we develop a fully unsupervised multi-scale *a contrario* anomaly detection method that proves to be highly versatile with very good results. Our algorithm does not need any normal or anomalous datasets or any priors. In practice, this method is parameterless and benefits from all the computational power of neural network libraries.

The remainder of this chapter is organized as follows: in Section 3.2 we present an exhaustive review of state-of-the-art methods and compare their key elements. Then, we present our work as a general algorithm to detect anomalies, explaining the method and showing results in Sections 3.3, and 3.4. Finally, in Section 3.5, we apply the proposed method to our particular industrial application, and we present its results.

3.2 Related Work

There is a wide literature on anomaly detection. The most common approach to detect anomalies consists of modeling the distribution of normal samples by means of a background model of randomness. The design of this model is usually problem-specific. For instance, in [51] the normal samples are assumed to follow a Gaussian distribution, or in [18] the authors characterize normality by fitting different GMMs with dense covariance matrix to a 4-level image pyramid. As previously mentioned, the *a contrario* approach is based on the same principle, but differs from the others in the sense that it allows to derive detection thresh-

olds by estimating and controlling the expected number of occurrences of an event under the background model. An excellent example is the work by Ehret et al. [52], in which they eliminate the self-similarity of the image by averaging the most frequent patches using Non Local Means [19] , and perform a detection using the NFA score over the residual image, formed by noise and anomalies. In this way, the general background modeling problem gets reduced to a noise modeling problem, making the algorithm work with any kind of background.

More recently, new data-driven methods based on deep neural networks (DNN) were developed, taking advantage of the continuously growing computation capabilities. In most cases, they use only anomaly-free images, achieving impressive results in publicly available anomaly datasets. A comparison between some key elements of different state-of-the-art works is presented in Table 3.1. From this table it is easy to see that most of the methods operate with some kind of supervision, which is the main difference of our proposed algorithm. A common classification separates methods into two categories: *feature-similarity based* and *reconstruction based* methods.

Feature-similarity based methods. Works based on feature similarity start by building a new feature space, where it is claimed to be easier to identify anomalies. The second step is usually to define and measure some notion of similarity, which can range from a simple k-Nearest Neighbors (kNN) or some probability distribution fitting to some learned metric ad hoc to the specific problem. For example, in [34], the authors construct a pyramid of features using the activation maps of a Wide-ResNet50, pre-trained on ImageNet, and use these feature maps for finding: i) the K nearest anomaly-free images, and ii) the pixel level anomaly segmentation. For the latter, they perform a multi-image correspondence of sub-images, matching the target sub-image with all sub-images of the K nearest neighbors. The anomaly score is finally obtained by averaging the distance between the sub-image of the target image and each one of its K neighbors. In [44], excellent results are reported by modeling normality with a multivariate Gaussian distribution for each patch position, using the output of a pre-trained CNN, and measuring the Mahalanobis distance to normality for each patch. Normality is characterized for each patch using the output of a pre-trained CNN. At test time, an anomaly score is assigned to each patch of a test sample based on the Mahalanobis distance to normality for the same corresponding spatial region (patch). In [100] a fully convolutional neural network is trained, where the output features preserve the spatial information, and are indeed a down-sampled version of an anomaly heatmap. By using an HSC (Hyper-Sphere Classifier) loss, the nominal samples are encouraged to be mapped near the center of the new space, and the anomalous samples away. The network is trained with both normal and anomalous samples. Anomalous samples could be synthetic, but the authors found that by using even a few examples of real labeled anomalies, the method performed much better. In [112] the authors extract the features from a pre-trained DNN and build a dictionary over the features, which is subsequently used to measure the distance of the target image to normality. A student-teacher framework is used in [153], where the student learns the distribution of anomaly-free images by matching their features with the teacher network accordingly. At test time, the feature maps of

the two networks for a given target image are compared, and an anomaly score is obtained from their distance. In [172] the idea of *Deep-SVDD* [124] is extended by constructing a hierarchical encoding of the image patches. The *Deep-SVDD* idea of mapping normal images near the center of a hyper-sphere in the feature space is improved by training an encoder to gather semantically similar patches. The output anomaly map is generated by computing the distance of each patch of the target image to the nearest normal pre-computed patch in the feature space. The score for each pixel is the average of the scores of all patches it belongs to. The algorithm performs multi-scale inspection, and the results are aggregated by element-wise multiplication to generate the final anomaly map. The authors show that the performance of the whole algorithm is improved by adding a self-supervised learning term to the loss function. The hard boundary of SVDD associated with the hyper-sphere can cause the model to overfit the training data. Therefore, in [28], the authors propose to replace it with a Gaussian SVDD (GSVDD) and treat its mean and covariance as latent variables to be estimated. They try to reduce the likelihood of an anomalous image embedding laying between normal samples by interpolating the latent embedding of different input images, making the distribution of the normal samples denser. The GSVDD is driven by the adversarially constrained auto-encoder interpolation (ACAI) [15], and by doing so, it generates a robust estimation of the normal image distribution.

Reconstruction-based. Generative models, as VAEs [81] or GANs [63], are based on the idea that the network trained only on non-defective samples will not be able to reconstruct the anomalies at test time. In practice, they usually present high capacity and generalization power, leading to even a good reconstruction of the anomalies. In [116] the authors address this issue by stating the generative part as an inpainting problem. By building a network only with self-attention blocks, they are able to model successfully distant contextual information, and achieve good results. As stated in [14], better results could be obtained by measuring the reconstruction error with the structural similarity measure (SSIM). The authors show that they achieve better results compared to the classical l_2 distance. In [97] the authors design a special classification proxy task in order to be able to learn a deep representation in a self-supervised manner, by training the network with random cut-and-pasted regions over anomaly-free images. Once the representation is obtained, a one-class classifier is trained using the new representation. In [170] the authors propose to train an Auto-Encoder over the feature maps of a pre-trained network. Instead of reconstructing the image itself, the network performs a deep features reconstruction. Finally, they compute a pixel-level anomaly score by analyzing the difference between the features at the input and output of the Auto-Encoder. Unlike AEs and VAEs, which learn a direct representation from the image to the latent code, Generative Adversarial Networks (GANs) learn the mapping from the latent space to realistic normal images. As the latent space transitions are smooth, in [128] the authors propose an iterative method to find a latent code that generates a similar image to the target, enabling to find anomalies by looking at the reconstruction error, using a GAN.

Method	Anomaly map	Pre-trained	Normal img use	Anomaly img use	Thr	Super-vision	Proxy task	Multi-Scale
Texture-Insp [18]	Negative log-likelihood for trained GMM	No	Fit GMM	No	No	Sup.	No	4-layer pyramid
SPADE [34]	Euclidean distance to normal samples	Resnet	kNN	No	No	Sup.	No	DNN scales
Dict-Similarity [112]	Distance to normality	Yes	Build dict.	No	No	Sup.	No	Different patch sizes
FCDD [100]	Activation layer from network	No	FCN	Yes	No	Sup.	No	DNN scales
PaDiM [44]	Mahalanobis distance by patch	Resnet	Mah. dist.	No	No	Self-Sup.	No	DNN scales
Patch-SVDD [172]	Distance to normal samples	No	Encoder	No	No	Self-Sup.	Patch-gather	Arch design
RotDet [136]	KDE at each patch location	No	DNN	No	No	Self-Sup.	Rotation	DNN scales
STPM [153]	Student-teacher L2 distance	Resnet	Student net	No	No	Self-Sup.	No	DNN scales
Det-In-Noise [52]	Log-NFA map	Option	No	No	Yes	UnSup.	No	Downsampled images
SSIM-AE [14]	Reconstruction error	No	AE	No	No	Self-Sup.	No	AE architecture
AnoGan [128]	Reconstruction error	No	GAN	No	No	Self-Sup.	No	GAN architecture
In-Tra [116]	Reconstruction error	No	Transformer	No	No	Self-Sup.	Inpainting	Downsampled images
Cut-Paste [97]	One class over representation	No	DNN	No	No	Self-Sup.	Cut-Paste	Replicating
DFR [170]	Reconstruction error	VGG19	VAE	No	No	Self-Sup.	No	DNN scales
GBVS [68]	Equilibrium distribution of markov chain	No	No	No	No	UnSup.	No	Downsampled images
SALICON [72]	Activation layer from network	No	No	Yes	No	Sup.	No	Arch design
DRFI [77]	RF Regression	No	No	Yes	No	Sup.	No	Multi-Level segmentat.
Ours	Log-NFA map	Option	No	No	Yes	UnSup.	No	Downsampled images

Table 3.1: Related work comparison showing some of the key elements of each method: how they generate the final anomaly map, whether if they use some pre-trained network or not, how is the usage of normal and abnormal images, the kind of supervision, if they solve a proxy task and the way to achieve the multi-scale analysis.

When learning from one-class data, a usual approach consists of mapping normal images near some centroid in the latent space and expecting (but often with no guarantee) that an anomalous image will lay far from this centroid. There are many works that propose strategies to overcome this potential issue. Many of them create proxy tasks in order to train a self-supervised network. Even though the specific proxy task could be very different from the final objective, in some cases, the network is proven to be very successful in extracting useful and informational features from the data. For example, in [136], the authors present a distribution-augmented contrastive learning technique via data augmentation that obstructs the uniformity of contrastive representations, making it easier to isolate outliers from inliers. A deep neural network is trained over the anomaly-free images in a self-supervised scheme by predicting the degree of rotation augmentation as the proxy task. In the second stage, a one-class classifier is built over the learned high-level data representation in order to obtain an anomaly score.

As we are especially interested in methods that can be used with no training and over a wide variety of different images as input, we also focus on the domain of *saliency detection*. For this work, we are only interested in textured images, where the random structures are uniformly distributed. In these cases, from the perception point of view, any structure or pattern that deviates from normality should also be the most salient part of the image. There is an extensive literature dedicated to this. To cite a few relevant works, a united approach to the activation and normalization steps of saliency computation is introduced in [68], by using dissimilarity and saliency to define edge weights on graphs that are interpreted as Markov chains. In [77], the salient object detection is formulated as a regression problem in which the task consists of learning a regressor that directly maps the regional feature vector to a saliency score. In [72], the authors propose to estimate saliency by combining multiple popular Deep Neural Networks (DNN) architectures for object recognition, that are known to encode powerful semantic features. They found that a good compromise between semantic representation and spatial information for saliency prediction is to use the last convolutional layer.

Although there is a vast literature related to anomaly detection, we found that only a few algorithms are able to perform the detection with no prior knowledge of the specific task/problem, and most of them try to model normality from a set of normal images, often needing them to be aligned and acquired in very controlled situations. In this work, we develop a simple and effective unsupervised algorithm that only uses the input image, with no need for any prior knowledge or training data, and estimates a "rareness score" given by the Number of False Alarms (NFA) for each pixel/region of the image. This algorithm was developed to address a specific task, but it has proven to work well in a wide variety of different problems, with no modifications.

3.3 Method

The kind of anomalies we wish to detect could be very diverse. In this work, we concentrate on low-level anomalies (by opposite of semantic anomalies). Using classic image processing techniques, we achieved a simple, fast, and yet very effective method that has proven to work successfully in very different kinds of images, as will be shown in Section 3.4. As this method was developed to solve an industrial problem, both accuracy and speed are of major importance.

A key and reasonable assumption in our approach is that low-order moments of the set of normal samples can be statistically characterized from the whole dataset, i.e., that anomalies are so rare that they do not affect these statistics. Under this assumption, the anomaly detection method we propose relies on a statistical characterization of the normal samples from the whole dataset, which we call *normality*, and a statistical distance of test samples to normality. This distance will then be used to evaluate how likely an observed sample is normal, and to derive a statistical test to detect anomalies.

Samples are characterized by a set of features that will be extracted to be uncorrelated. We show that these features follow a Gaussian distribution, which allows us to easily define a distance to normality as a Mahalanobis distance, and to ensure that this distance follows a χ^2 distribution.

The proposed method therefore consists of five stages, illustrated in 3.2: A) Image pyramid generation by scale decomposition, to detect anomalies of different sizes; B) Feature Extraction; C) Mahalanobis distance to normality; D) NFA computation; and E) Generation of the final anomaly map, by combining the results obtained for each different scale.

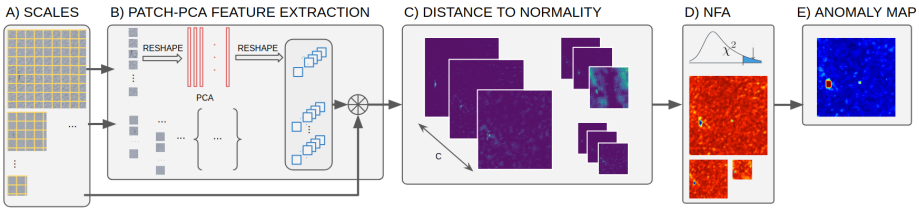


Figure 3.2: Method summary. Our method can be divided into 5 stages. Stage A) consists of creating a pyramid of images of different scales. In stage B) we obtain the filters by performing a Patch-PCA transformation and use them to filter the input image. At the beginning of stage C) we have one feature map for each filter and for each input channel, and we combine them by computing one Mahalanobis distance for each channel. Stage D) performs the computation of the Number of False Alarms. The Stages B), C), and D) are computed for each scale independently, and all results are combined in stage E).

For the feature extraction and the NFA computation (stages B and C), different variants are explored. In the remainder of this section we describe all the stages of the proposed approach.

3.3.1 Image decomposition: scales' pyramid

Since anomalies can occur at any scale, it is crucial to detect them no matter their size. This is performed by means of a multi-scale approach based on an image pyramid where each new scale is half the width and the height of the previous one.

3.3.2 Feature Extraction

Three alternatives for the feature extraction stage are considered.

Patch-PCA. The procedure described here is applied to each scale obtained from the scales' pyramid decomposition, as shown in Stage B in Figure 3.2.

To represent the local statistics of the pixels in the acquired images, such as their surrounding texture, we associate a patch of $s \times s$ pixels centered on each pixel. We treat each image channel separately, so for each channel c, each pixel location is characterized by a n-feature vector, where $n = s \times s$. We apply PCA to decorrelate the patch feature vector and treat the new coordinates independently.

PCA performs a linear transformation, $\mathbf{y}_i = P^T \mathbf{x}_i$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $i \in \{1 \dots N\}$ is a set of observations (patches), $\mathbf{y}_i \in \mathbb{R}^m$ with $m \leq n$ their projections, and the columns $\{\mathbf{p}_i\}_{i=1}^n$ of P the PCA eigenvectors. The element j in vector $\mathbf{y}_i$ is the result of the inner product between $\mathbf{x}_i$ and $\mathbf{p}_j$. Since these projections are inner products over patches, they can be efficiently computed as 2D convolutions with the eigenvectors as kernels, using common deep learning libraries. Extracting the PCA eigenvectors can also be seen as a methodology for finding the filters used to extract features based only on the input image. This characteristic is crucial in our application, as it defines specific adapted filters for each image. Once we have obtained the filters, we convolve the input image by channel with each filter (e.g., a projection into the PCA space), obtaining a feature vector of size m for each of the c channels, for each pixel.

ResNet feature maps. Instead of applying PCA to the output of the scales' pyramid, the PCA could be applied to the activation maps resulting from feeding a convolutional neural network with the input image. The advantage of this approach is that these activation maps gather a set of salient features of the image. More precisely, the set of features we consider are obtained from a ResNet-50 network pre-trained on ImageNet, keeping the output of the last convolution of three layers: *layer1/conv3, layer2/conv3, layer3/conv3.* As the input of the network is the entire three-channel image, the filtering is not performed separately for each channel (as in the Patch-PCA approach), and at the end of this stage, we obtain a feature vector of size equal to the sum of the number of filters in each convolutional layer that was considered. After applying PCA to the filtering output, we also keep the first m components.

Gabor filters. Instead of performing the scale decomposition and computing the Patch-PCA descriptors per image, we propose to use a set of (fixed, signal-independent) Gabor filters of different sizes. Note that in this case, the resulting features are not uncorrelated. However, results presented in Section 3.4 show

that this modification still achieves good performance while significantly reducing the computational burden from the image-dependent PCA computation. In this case, we follow the same procedure as in Patch-PCA: the extracted features are the filter responses for each channel of the input image.

3.3.3 Distance to normality

As previously mentioned, at this point we are supposed to have a set of uncorrelated features. This is true for the case of Patch-PCA and ResNet feature extractor, in the latter thanks to the PCA performed over the activation maps. For the case of Gabor filtering, since these filters are not orthogonal to each other, the resulting features do present some degree of correlation; nevertheless, we take the dare to analyze it anyway, as we believe it is still a valuable approach due to its simplicity.

For each pair of features ($\mathbf{a}$, $\mathbf{b}$), we define the distance between its corresponding image regions $\mathbf{x^a}$ and $\mathbf{x^b}$ as:

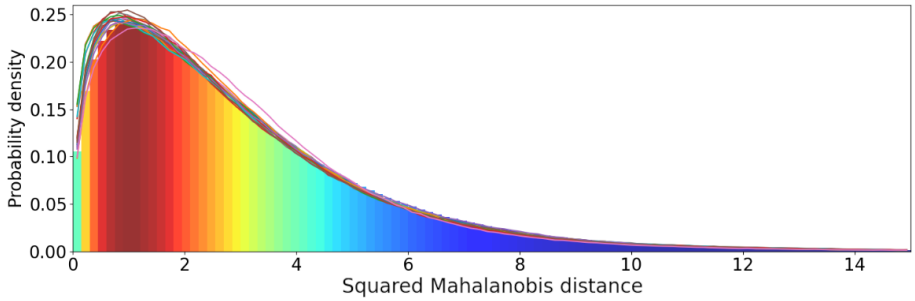
$$d^2(\mathbf{x^a}, \mathbf{x^b}) = \sum_{i=1}^{m} (a_i - b_i)^2.$$

To define a distance to normality based on the previous metric, we need to characterize the set of normal samples and their corresponding feature vectors (obtained with any of the feature extraction methods described in the previous section). Assuming that the size of the anomaly is small compared to the size of the image, the low-order statistics of the normal samples can be accurately characterized from the whole image. In what follows, we will describe the computation of the distance to normality for the features extracted with Patch-PCA and Resnet methods since both end up with uncorrelated feature vectors due to the use of PCA. Then, at the end of this section, we will comment on the case of Gabor feature extraction.

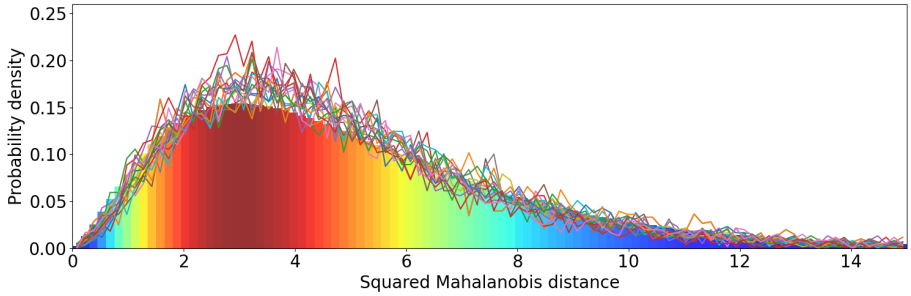
Let μ_i and σ_i^2, $i = 1, \ldots, m$, denote the mean and the variance of the top m PCA components of the feature vectors. We define the squared distance of a sample to normality as:

$$d^2(\mathbf{x^a}, \mathit{normality}) = \sum_{i=1}^{m} \frac{(a_i - \mu_i)^2}{\sigma_i^2}. \tag{3.1}$$

This quantity corresponds to the Mahalanobis distance of the sample $\mathbf{x^a}$ to the mean of the normal samples, restricted to the top m principal directions. One of the main advantages of this simple notion of distance to normality is that, if we now consider samples $\mathbf{x^a}$ to be random, its statistical characterization is straightforward. Indeed, we have tested empirically that each PCA component follows a Gaussian distribution; knowing that these components are uncorrelated, if we further assume they are independent, the distance to normality (3.1) follows a χ^2 distribution with m degrees of freedom. Figure 3.3 shows the distribution of distances from image samples to normality for several test images, and its comparison to the $\chi^2(m)$ distribution. Note that the fit is remarkably accurate.



(a) Patch-PCA features, keeping the first $m = 3$ components.



(b) ResNet-based features, for the first $m = 5$ components of the PCA.

Figure 3.3: Distribution of distances to normality for the Patch-PCA and ResNet-based features. Colored bars represent the χ^2 theoretical distribution. Solid-colored lines are the normalized histograms of distances to normality for different images.

For simplicity, in the case of the method based on Gabor filters, we will assume uncorrelated components and apply the same computation of the distance to normality explained above. While not true, as already explained, this deviation appears to be not significant enough to affect the results while allowing a computationally efficient implementation.

3.3.4 Number of False Alarms

In this section, we propose three different strategies, sharing a common methodology, to detect anomalies in images of textures, using the distance to normality defined in the previous subsection. The detection methodology we follow is the *a contrario* framework. As explained in Section 2.4, it is a multiple hypothesis testing strategy that detects meaningful events (image structures) as large deviations from a background model or null hypothesis $\mathcal{H}_0$, which in our case models normality. Following the *a contrario* approach [46], to each event E we inspect, we associate a rareness score called Number of False Alarms (NFA) . The lower the NFA of an event is, the more unlikely that this event was produced by the background model, being more likely to result from another cause. If we set the threshold on the NFA to $\epsilon = 1$, we should detect on average one false alarm, and the rest of the detections should correspond to actual anomalies.

We now present the three different detection strategies based on the *a contrario* framework. It is usually the case where the anomaly only shows in one particular channel or scale (e.g. some color anomaly). Therefore, in order to keep all possible detections, we keep the minimum of all NFAs across all channels and scales. The output of this stage is one NFA map for each scale.

NFA by pixels In this strategy, for each pixel $\mathbf{x^a}$ in the image, characterized by its feature vector $\mathbf{a}$, we define

$$\mathrm{NFA}(\mathbf{x^a}) = HW(1 - \mathrm{CDF}_{\chi^2(m)}(d^2(\mathbf{x^a}, \mathit{normality}))),$$

where H and W denote the height and width of the image, and $\mathrm{CDF}_{\chi^2(m)}$ is the Cumulative Density Function of a χ^2 distribution of order m. This quantity corresponds to the number of events we test (all the pixels in the image) times the probability of a $\chi^2(m)$ random variable being larger than its observed squared distance to normality.

NFA by blocks In this per-block strategy, the objective is to detect anomalies as conspicuous accumulations of pixels exhibiting a large distance from normality within image blocks. This algorithm begins with the extraction of candidate regions for each channel by thresholding each squared Mahalanobis distance $d^2(\mathbf{x^a}, \mathit{normality})$ with a threshold τ, which can be easily set. Since under the normality hypothesis, these distances are assumed to follow a $\chi^2(m)$ distribution, we set τ such that the corresponding p-value is $p = 0.01$. In this case, our normality assumption ($\mathcal{H}_0$) is that pixels such that their distance to normality exceeds τ are uniformly distributed on the whole image. Following the idea in [61], we base this NFA criterion on the detection of high concentrations of these candidates as follows.

Considering a block B of size $w \times w$, and the set of candidates $\mathcal{L}_B = \{\mathbf{x^a} \in B \text{ s.t. } d^2(\mathbf{x^a}, \mathit{normality}) > \tau\}$ within the block, we define a new random variable, $U_{i,j}$, with the following Bernoulli distribution:

$$U_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{L}_B \\ 0 & \text{otherwise.} \end{cases}$$

To search for unusual concentrations of candidate pixels, we need to evaluate the probability

$$P(U \geq |\mathcal{L}_B|), \quad \text{with } U = \sum_{i,j} U_{i,j} \text{ following a binomial law.}$$

Evaluating this probability is not straightforward, as the $U_{i,j}$ are not independent. To overcome this issue, we would have to sub-sample the Mahalanobis distance image, using a factor that ensures that two neighbor pixels in the new image can be considered independent, i.e. keeping one of each s pixels in each direction (the size of the patch for the Patch-PCA feature extraction, or the size of the filter for Gabor filtering). Once we have the sub-sampled image and we have independent pixels, we could use the previous NFA by pixel approach. But instead of doing that, we can use a trick based on the intuition that we

are observing $|\mathcal{L}_B|^2/s^2$ independent meaningful events out of a total of w^2/s^2 possibilities. Note that if were actually doing the sub-sampling, we probably would not find the exact average value ($|\mathcal{L}_B|^2/s^2$), depending on the grid origin we choose. In fact, we will only find this value if the meaningful events are uniformly distributed. In the case they are not, we would obtain some sub-samplings where we find more events than the average and some others where we find less. In any case, we can ensure that by choosing the average value, we will have at least one sub-sampling as significant as the one we are evaluating. Therefore, the probability of observing at least $|\mathcal{L}_B|/s^2$ distances greater than τ in block B, is given by the tail of the binomial law, $\mathcal{B}\left(\frac{w^2}{s^2}, \frac{|\mathcal{L}_B|}{s^2}, p\right)$. The number of blocks B to be tested is HW/w^2. Testing all $s \times s$ sub-sampled grids per block leads to a total number of $N_T = HWs^2/w^2$ tests. Finally, the NFA associated with a block B is given by

$$\text{NFA}(B) = \frac{HW}{w^2} s^2 \mathcal{B}\left(\frac{w^2}{s^2}, \frac{|\mathcal{L}_B|}{s^2}, p\right).$$

NFA by regions As in the NFA by blocks strategy, here we also look for a conspicuous accumulation of pixels for which the distance to normality is sufficiently large.

The previously presented NFA computation techniques apply to pixels and square blocks, but this variant extends the possibilities by enabling the computing of the NFA of arbitrarily sized and arbitrarily shaped regions. It is inspired on the work by Grompone et al. [149]. One of the difficulties of computing the NFA value over regions of any shape resides in the calculation of the number of tests. To do so, we need to quantify all possible arrangements of pixels with any shape and of all possible sizes. To build these regions, Grompone et al. propose to work with 4-connectivity regions $\mathcal{R}$, for which the approximate number of possible configurations with $N_{\mathcal{R}}$ pixels is given by (see [76]):

$$\#\text{configurations} \approx \alpha \frac{\beta^{N_{\mathcal{R}}}}{N_{\mathcal{R}}},$$

where $\alpha = 0.316915$ and $\beta = 4.062570$. The construction of the regions is based on a greedy algorithm proposed in [149] but with some differences. First, we use the Mahalanobis distance as input and consider the pixels for which the distance to normality is larger than a fixed threshold. Second, and more importantly, we add a region-growing criterion based on the NFA value itself. In order to decide if a pixel is to be added to a certain region, we evaluate the significance (NFA value) of the region with and without the pixel and only add it if it makes the whole region more significant, i.e., lowers the NFA value. The original algorithm is designed for single-channel images, so we apply it for each filter response separately and then combine all the NFAs by keeping the minimum between all results for each position. The pixels that were not considered in any region are assigned with the maximum NFA value calculated, as they are the less significant ones.

We implemented this variant of the NFA computation only for the ResNet-based feature maps, which is the best-performing combination among the methods presented so far, as will be shown in Section 3.4.

As explained before, once we obtain the feature maps (in this case obtained only from *layer1[0]/conv3*), we decorrelate them using PCA and compute the Mahalanobis distances. Note that the feature maps have a spatial resolution much lower than the input image. In order to work at full resolution, we up-sample the Mahalanobis distances to the input size and compute the NFA. By doing so, we are introducing some dependency between neighbor pixels. To overcome this issue, we proceed in a way similar to the NFA by blocks strategy, by considering the receptive field $\tilde{s}$ of the network to ensure independence again. Reasoning in a similar way, by assuming that we observe $N_{\mathcal{R}}/\tilde{s}^2$ pixels, we can ensure that there is at least one event as significant as the one we are computing. The final NFA value computed is therefore:

$$\mathrm{NFA}(\mathcal{R}) = \frac{HW}{\tilde{s}^2}\,\alpha\frac{\beta^{N_{\mathcal{R}}/\tilde{s}^2}}{N_{\mathcal{R}}/\tilde{s}^2}\left(1-\mathrm{CDF}_{\chi^2(N_{\mathcal{R}}/\tilde{s}^2)}\Big(\frac{1}{\tilde{s}^2}\sum_{(i,j)\in\mathcal{R}} d_{i,j}^2\Big)\right),$$

where $d_{i,j}^2$ denotes the squared distance to normality associated to pixel (i,j).

3.3.5 Final anomaly map

At this point, we have obtained one NFA map for each scale. The next step is to combine them in order to generate a unique map gathering the information of all scales. To do so, we follow the intuition that if some structure results to be anomalous at a certain scale, it has to be marked as anomalous in the final map. Therefore we compute the final NFA map as the minimum value of NFA of each scale for each pixel. These maps are combined by up-sampling smaller scales to match the original size. Finally, we translate this NFA map into a rareness measure by simply defining the Anomaly Score as $AS = -\log_{10}(\mathrm{NFA})$, which provides a more intuitive map where the larger the value associated with a structure, the higher its rareness.

3.4 Results

To the best of our knowledge, there are few truly unsupervised anomaly detection methods that can work only with the image being inspected, with no prior training or information about normal samples, as shown in Table 3.1.

	AE-Ssim [14]	AE-L2 [14]	Ano-Gan [128]	Cnn-Dict [112]	Tex-Insp [18]	DR-FI [77]	GB-VS [68]	Det-Noi. [52]	Ours PCA	Ours PCA†	Ours Ga-bor	Ours Ga-bor†	Ours Res-Net	Ours Res-Net‡
Carp	**0.87**	0.59	0.54	0.72	**0.88**	0.72	0.45	0.57	0.76	0.65	0.77	0.69	**0.94**	**0.94**
Grid	**0.94**	0.90	0.58	0.59	0.72	0.58	0.73	0.68	0.88	0.91	0.90	0.83	**0.92**	**0.96**
Leath	0.78	0.75	0.64	0.87	**0.97**	0.68	0.86	0.81	**0.98**	0.87	0.95	0.92	**0.99**	**0.99**
Tile	0.59	0.51	0.50	**0.93**	0.41	0.72	0.49	0.53	0.69	**0.84**	0.79	0.80	0.77	**0.87**
Wood	0.73	0.73	0.62	**0.91**	0.78	0.67	0.79	0.57	0.86	**0.87**	**0.87**	0.84	0.86	**0.92**
Mean	0.78	0.70	0.58	0.80	0.75	0.67	0.66	0.63	0.83	0.83	**0.86**	0.82	**0.90**	**0.94**
Rank	8	10	14	6	9	11	12	13	5	4	**3**	7	**2**	**1**

Table 3.2: Area under the receiver operating characteristic curve (ROC AUC). In red the best score, blue the second best and green the third. The ranking of the algorithms is done by sorting the performance and summing the positions for each algorithm. The one with the lowest value is the best one. For our methods, † indicates the Block NFA variant, ‡ indicates the Region NFA variant, and no symbol corresponds to the Pixel NFA variant.

In order to compare our proposed method with the state of the art, we use MVTec AD [13], a recent anomaly dataset that simulates faults or anomalies in industrial conditions. The MVTec AD dataset contains several subsets divided into two categories: texture and objects. We focus on the texture category, as we are interested in the detection of low-level anomalies. Also, one of these subsets has special importance for us, as it considers anomalies in leather samples. MVTec AD also provides several non-faulty images, which most state-of-the-art methods use for learning the normality. In order to extend the comparison, we also include the baseline methods, even if they require training on normal-only samples and the comparison is not completely fair in their favor. Results are shown in Table 3.2. To perform the comparison, we use the area under the receiver operating characteristic curve (ROC AUC), as it is the most widely used metric in the literature and enables us to compare with state-of-the-art results.

The results in Table 3.2 corresponding to the Block-NFA version are indicated with a single cross, and results corresponding to Region-NFA with a double cross; the others use Pixel-NFA. The proposed algorithm achieves state-of-the-art results, even when compared with some methods that clearly benefit from using anomaly-free images for training. In some subsets our methods manage to obtain the best results. As mentioned before, we pay special attention to the leather case since it is related to our industrial application (see next section). For this dataset, our approach achieves the highest score. Furthermore, if we consider the average score over all subsets, all variants of the proposed algorithm perform the best, among which stands out the ResNet variant, which not only produces the best results but is also the fastest variant of our approach. In addition, the final improvement added to the ResNet version, which enables computing the NFA for arbitrarily shaped and arbitrarily sized regions, has proven to outperform all the other methods, even exhibiting the best performance on almost all subsets. Figure 3.4 shows several example images alongside their final anomaly map, for one example of each defect type of each dataset. Both the ground truth (in green) and the detections with log-NFA=0 (in blue) are drawn over the original image.

As mentioned before, one of the benefits of using the *a contrario* framework is the simplicity of setting the threshold for the defects segmentation. As the metric we used to compare with all methods is independent of the threshold, in Figure 3.5 we show the ROC curves obtained for the leather dataset with the top three performing variants of our proposed algorithm (for each different feature extractor), and indicate with a star the point where AS=0, i.e. NFA=1. It can be seen that values close to AS = 0 lay near the optimal point in the ROC curve. Also, another interesting verification is to estimate the GAP between the anomaly scores of the faulty and non-faulty regions. In order to do so, we show in Figure 3.6 the distribution of the anomaly scores for all leather samples, overlapped with the cumulative density functions for both classes, making it clear that there is a good separation between them. Additionally, we evaluate this GAP numerically by computing the difference between the median values of the anomaly scores AS of the two classes for all variants, obtaining the results shown in Table 3.3.

In all cases, we obtained a very good GAP and well-separated classes. De-

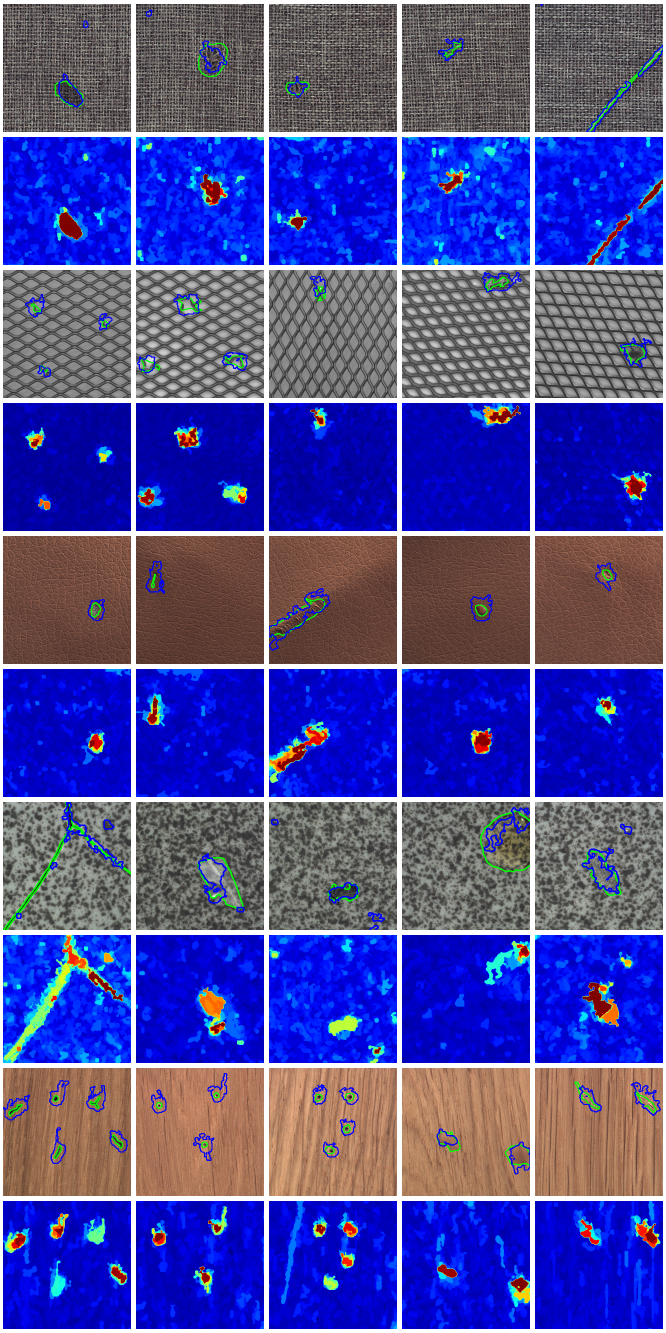


Figure 3.4: For each textured dataset in MVTec AD (carpet, grid, leather, tile, and wood), we show one image with each type of defect and their corresponding anomaly maps with the *ResNet+RegionNFA* method. The ground truth in is shown green, and the detection with log-NFA=0 in blue, superimposed to the original image.

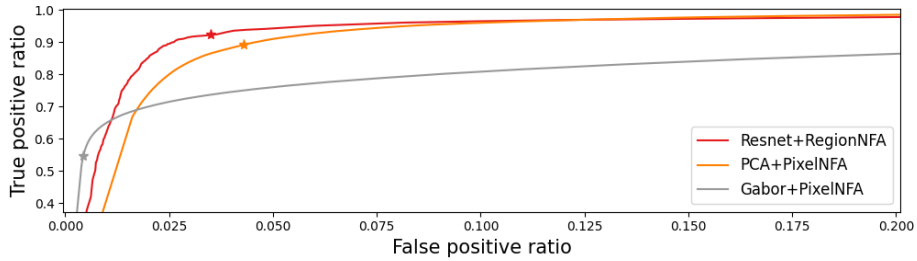


Figure 3.5: ROC curve for the leather subset of MVTec AD, for the best-performing variant for each feature extractor of the proposed method. The value with AS=0 (NFA=1) is indicated with a star. Note that the points AS=0 lay close to the optimal points of the ROC curves.

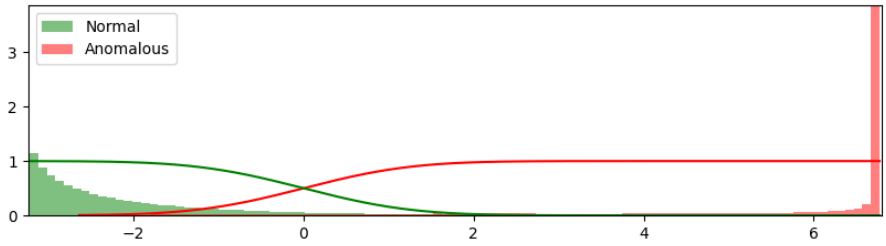


Figure 3.6: Distribution of the Anomaly Score AS $= -\log(\text{NFA})$ values for normal and anomalous pixels of the leather subset of MVTec AD. overlapped with the cumulative density function for both classes. Note the large gap between classes, and the adequate choice of NFA values close to AS=0 to derive the threshold.

pending on the feature extraction method used, the AS gap is greater in one case using BlockNFA and in the other using PixelNFA. The best GAP is obtained for the combination Resnet+RegionNFA, which is also the best-performing method.

For the PCA-based variant, we used $m = 45$, $s = 17$, and 4 scales. For the Gabor variant $m = 72$, s ranging from 7 to 31, and 4 scales. The Block-NFA is computed considering blocks of size $w = 51$ with a stride of 10 pixels.

Table 3.3: GAP between the anomaly scores of the faulty and non-faulty regions for the different combinations of our method.

Method	GAP
PCA+PixelNFA	2.25
PCA+BlockNFA	4.06
Gabor+PixelNFA	9.12
Gabor+BlockNFA	4.62
ResNet+PixelNFA	3.15
ResNet+RegionNFA	11.83

3.5 Anomaly detection in leather for the automotive upholstery industry

In this section, we present the industrial problem that motivates this work, carried out for a world-leading producer of leather upholstery for the high-end automotive industry[1].

The raw skin of animals can present multiple kinds of defects caused by the living animal itself (e.g. tick, lice and mite marks, scars), or in the subsequent processes that transform raw leather into textured leather pieces ready to be installed in a vehicle. Various types of defects (holes, machinery drag, etc.) may be introduced in each step of the process, such as differences in the processes of tanning, differences in the roller pressing for texturing (Figure 3.1(a)-(b)), etc. The size and shape of defects vary widely (from bites, which are millimeters in size, to wrinkles occupying most of the piece). In industries that work with leather, these defects must be detected as soon as possible in the production line to take the corresponding actions (avoiding cutting on the defect, leaving it in inconspicuous areas, or even discarding the pieces). Having a human operator to perform the quality control has several downsides, such as the inability to maintain the necessary attention level for long periods, criterion differences between different operators, and, of course, the time and workforce costs.

Each vehicle manufacturer tolerates a small margin of defects in the leather pieces that they buy. When this margin is exceeded, the entire batch of leather pieces is returned, incurring large losses for the producing companies. On the other hand, production is limited by the number of personnel available for inspection. Since defects are searched manually in all samples at various points in the process, this is a bottleneck in the plant's production capacity.

As the defects are often very hard to see to the naked eye, a dedicated acquisition setup was designed. For each skin sample, a total number of 5 images are acquired with a single camera pointing at the sample at nadir, using different lightning conditions and orientations. More precisely, a diffuse light is located next to the camera, and 4 directional grazing lights are located in each corner of the table. In this way, some defects that are very subtle but have little relief can also stand out. A typical set of images acquired with this setup is shown in Figure 3.7, top row.

3.5.1 Pre-processing

The algorithms presented in Section 3.3 can be fed either with all 5 raw images or with the images resulting from this pre-processing step. Contrary to the most common use of PCA, where only the first components are kept, in this case, we are more interested in the last ones. Indeed, the first components correspond to the directions of the largest variance and, therefore, encode global information about the image, such as color or texture. Since anomalies are particular local structures, they are mostly present in the last components. Figure 3.7 depicts a visual example. The first row corresponds to the 5 raw images, and the second row to the PCA components (the first image corresponds to the first

[1]The project was developed by Digital Sense in collaboration with Pensur.

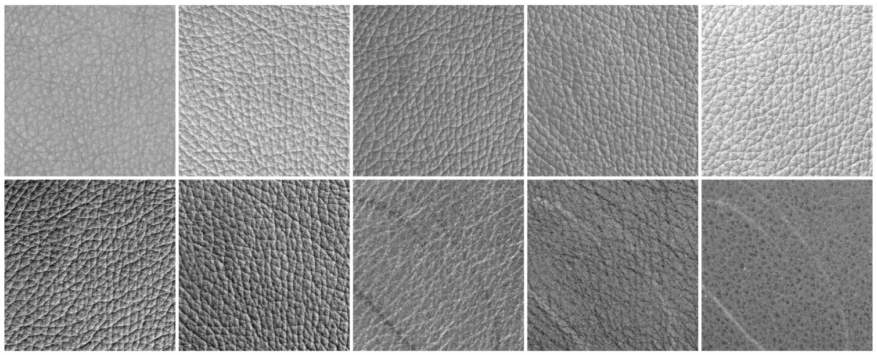


Figure 3.7: Top row: original images. Bottom row: Output of the preprocessing step. Note how the PCA-based preprocessing better manages to single out potential anomalies.

Table 3.4: ROC AUC results for the considered industrial application: defect detection in leather samples.

Method	ROC AUC
PCA+PixelNFA	89.53
PCA+BlockNFA	78.63
Gabor+PixelNFA	85.67
Gabor+BlockNFA	78.41
ResNet+PixelNFA	87.84
ResNet+RegionNFA	87.92

eigenvector). This example shows that the anomaly is very hard to visualize in all the raw images but stands out more clearly in the last 3 components of the projected space.

3.5.2 Industrial application results

For this industrial application, no reliable anomaly segmentation is available. We only were able to gather a limited number of defective samples that were coarsely annotated during the normal inspection work. We present the ROC AUC results for all our method variants and some examples for visual inspection in Figure 3.8, showing the input image, its NFA maps, and their corresponding segmentation with NFA = 1 for some variants of the proposed method: *PCA+PixelNFA*, *Gabor+BlockNFA*, *ResNet+PixelNFA* and *ResNet+RegionNFA*. With all the variants of our method, we succeed in detecting all defects for these samples. For the Block-NFA variant, we obtain a coarser map compared to Pixel-NFA, caused by the computation by blocks. On one hand, this characteristic makes the map cleaner and smoother, but on the other hand, it may cause the evaluation metric used to drop, as it depends on a fine segmentation of the anomaly.

Since we have an input of 5 images (1 diffuse light and 4 grazing directional lights), we cannot directly compare other state-of-the-art methods for this

dataset, as they expect a single image as input. Moreover, for the ResNet variant of our method, we had to include a modification, feeding the ResNet independently for each image and concatenating the output feature maps. Also, for these examples, we use only the output of the first layer of ResNet, *layer1/conv3*. The ROC AUC results obtained are presented in Table 3.4. The best performance is achieved by the method PCA+PixelNFA, followed by ResNet+RegionNFA. The main drawback of PCA+PixelNFA is that we must compute the PCA for each image before filtering. On the other hand, ResNet+RegionNFA filtering is faster overall, but we still need to compute the PCA after filtering. Gabor+PixelNFA, although its performance (in terms of AUC) is lower, is a very simple method that does not require PCA after filtering.

We tested our algorithm in production with thousands of leather samples in the modality of assessment for the operator. More important than the results commented above, the first results obtained in production demonstrate that this tool provides useful assistance to the operators, indicating problematic zones and making their work easier and faster.

Conclusions

In the work presented in this chapter, we have proposed a fully unsupervised *a contrario* method for anomaly detection, which aims at detecting defects in leather samples from the automotive industry, achieving industry standards. Although the method was designed for this particular application, its foundations are general enough and proved to perform successfully in a wide variety of defects in images of different scenarios. The proposed approach outperforms other state-of-the-art methods on the MVTec dataset and has proven to work especially well for leather samples. Of all the proposed variants, the one that clearly stands is using the ResNet feature extraction with the Region NFA computation. It achieves the best score in almost all datasets, and it is ranked as the best overall. In addition, this method obtained the best separation between normal and defective samples, as can be deduced by the GAP analysis. Last but not least, the natural threshold (log-NFA=0) represents a good choice for easily fixing the detection threshold, as shown in Figure 3.5.

Future potential improvements include substituting PCA with non-linear transformations obtained from auto-encoders and/or Normalizing Flows trained on normal data. In this setting, the multi-scale approach may be included inherently in the network by means of a U-shaped network. Another line of work to be explored is the combination of the *a contrario* approach with deep neural networks in order to output detections with a confidence score or to train the ResNet we utilized with leather anomaly-free samples in order to obtain more specific feature extraction for our task.

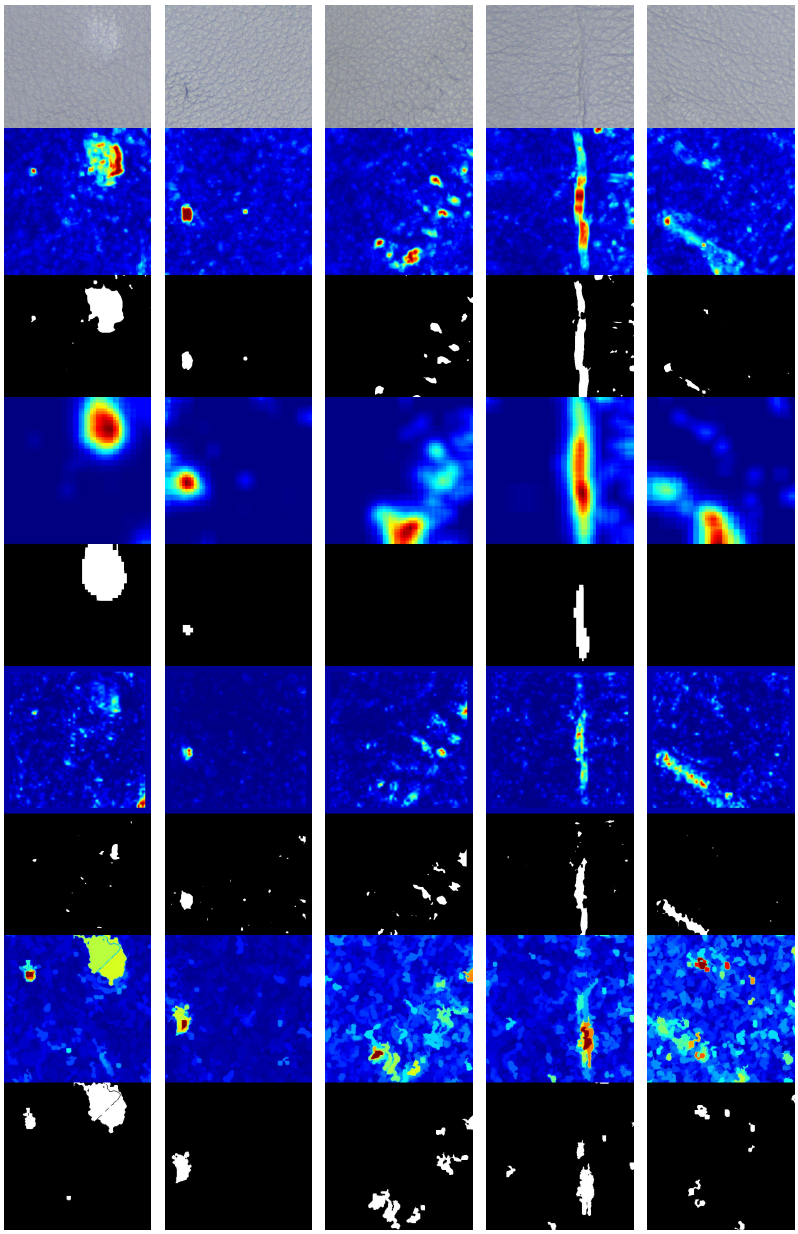


Figure 3.8: Results on our industrial data. First row: original diffuse image. Following rows show the anomaly score map and the segmentation with AS=0, for some variants of our proposed method: *PCA+PixelNFA*, *Gabor+BlockNFA*, *ResNet+PixelNFA*, and *ResNet+RegionNFA*. All defects are detected in all cases, and AS=0 provides a good choice for the detection threshold.

Chapter 4

Normalizing Flows

The method detailed in the previous chapter employs classical image processing techniques for feature extraction and transformation, with anomaly detection based on a multiple-hypothesis testing approach. This method uses the *a contrario* framework, assuming that the data follows a Gaussian distribution. Initially, this may appear to be a typical density estimation problem. However, the real challenge lies in the high-dimensionality of the feature space, where traditional techniques struggle to represent distributions accurately. In high-dimensional settings, the density becomes sparse and low valued in the whole space. Anomalies, which by definition occur in low-probability regions, are particularly elusive in these spaces where likelihood values are uniformly low. In such a setting, generative models emerge as a crucial tool capable of overcoming the limitations of traditional techniques. Specifically, Normalizing Flows provide an elegant solution to the low-density challenge by allowing for exact likelihood computation in high-dimensional spaces. They are a powerful and flexible class of probabilistic models capable of transforming simple probability distributions (such as a Gaussian) into more complex ones through a series of invertible and differentiable mappings. These mappings allow Normalizing Flows to effectively capture the complexity of real-world data, even in high-dimensional spaces, while maintaining the ability to compute exact likelihoods. This capability is crucial for anomaly detection, as it ensures that low-probability regions where anomalies are expected to occur can be accurately identified and measured.

This chapter aims to provide a comprehensive understanding of Normalizing Flows, covering their theoretical foundations and mathematical formulation. It lays the groundwork for understanding the techniques used in the following chapter, which presents one of the most important methods developed in this thesis, U-Flow.

4.1 Background

Flow-based generative models have been relatively overlooked compared to the more prominent GANs [63] and VAEs [81]. However, they offer some inter-

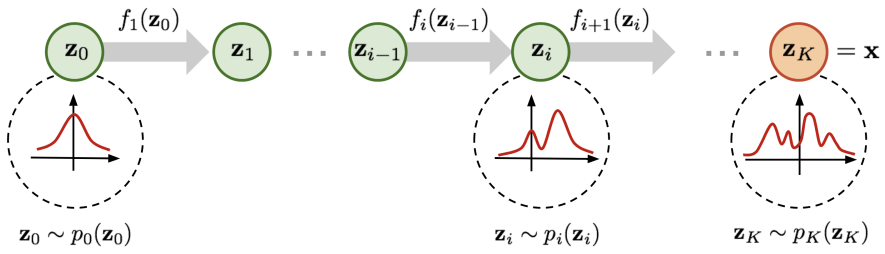


Figure 4.1: Figure taken from [158]. Illustration of a Normalizing Flow model, transforming a simple distribution $p_0(z_0)$ to a complex one $p_K(z_K)$, step by step.

esting advantages. Normalizing Flows (NFs) can perform exact latent-variable inference and log-likelihood evaluation, unlike VAEs, which optimize a likelihood bound, and GANs, which directly lack an encoder to obtain a latent variable. This ensures accurate inference and enables optimization of the exact log-likelihood of the data rather than just a lower bound. Flow-based generative models demonstrate efficiency in both inference and synthesis tasks. Autoregressive models, such as Pixel-CNN [148], are also reversible, but their synthesis process is challenging to parallelize, often resulting in inefficiency and long-running times [82].

As previously mentioned, a Normalizing Flow transforms a simple base distribution $p_Z(z)$, typically a Gaussian distribution, into a more complex target distribution $p_X(x)$ through a series of invertible and differentiable mappings $f_1, f_2, \ldots, f_K$. The transformation can be represented as:

$$x = f_K \circ f_{K-1} \circ \cdots \circ f_1(z). \tag{4.1}$$

A graphical representation is also included in Figure 4.1, where we use $z = z_0$, and $x = z_K$.

All functions f_i are invertible, which means that we can map the data x back to the latent space z, as

$$z = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_K^{-1}(x). \tag{4.2}$$

The idea is simple: We start with a known and easy-to-use distribution and locally transform it step by step using invertible functions until we obtain an approximation of our data distribution. Therefore, it is important to understand how to modify density functions when passing through one of these functions f_i. To do so, we introduce the change of variable theorem in the following section.

4.1.1 Change of variable

Given a random variable (RV) Z with a known probability distribution $Z \sim \pi(z)$ and a transformation $x = f(z)$, our aim is to determine the probability density function (PDF) of x, denoted as $p(x)$, in terms of the PDF of z.

For an intuitive presentation, let us consider the simplest case: a one-dimensional RV that distributes according to a uniform distribution on a given support. When we modify the support of the uniform distribution by some factor, we need to adjust the probability density value accordingly to maintain probability. This adjustment is captured by the relationship

$$\pi(z) = p(x) \left| \frac{dx}{dz} \right|. \tag{4.3}$$

Here, $\left|\frac{dx}{dz}\right|$ represents the absolute value of the derivative of the transformation $f(z)$, ensuring that we account for both positive and negative changes in probability density. Ultimately, this adjustment preserves the integral value, ensuring that the total probability remains unchanged despite the transformation. From (4.3) it follows that $p(x) = \pi(z) \left|\frac{dz}{dx}\right|$, and using that f is invertible we finally obtain that

$$p(x) = \pi(z) \left| \frac{dz}{dx} \right| = \pi(f^{-1}(x)) \left| \frac{df^{-1}}{dx} \right| = \pi(f^{-1}(x))|(f^{-1})'(x)|. \tag{4.4}$$

The multivariate case has a similar formula:

$$\begin{aligned} \mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x}) \\ p(\mathbf{x}) = \pi(\mathbf{z}) \left| \det \frac{d\mathbf{z}}{d\mathbf{x}} \right| = \pi(f^{-1}(\mathbf{x})) \left| \det \frac{df^{-1}}{d\mathbf{x}} \right|, \end{aligned} \tag{4.5}$$

where $\det \frac{\partial f}{\partial \mathbf{z}}$ is the determinant of the Jacobian of function f. Just as a reminder, the Jacobian is the following matrix:

$$\mathbf{J} = \frac{df}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}. \tag{4.6}$$

Now, let's explore how to apply the change of variable theorem in the context of Normalizing Flows. In the following section, we will demonstrate how this approach enables the determinant of the Jacobian matrix to be propagated through the entire network. This capability lets us directly optimize the exact likelihood and train our network.

4.1.2 Normalizing Flows optimization

For convenience, in the following we use indistinctly $\pi(\mathbf{z})$ and $p_0(\mathbf{z}_0)$. Using the change of variable theorem, for each variable z_i in the chain, we know that

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right|. \tag{4.7}$$

Then, operating and taking logarithm to both sides, we obtain

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|. \tag{4.8}$$

The previous expression lets us understand the relationship between each pair of consecutive variables in the chain. Therefore, we expand this relationship step by step until we reach the initial distribution z_0, as follows [158]:

$$\begin{aligned} \mathbf{x} = \mathbf{z}_K &= f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0). \\ \log p(\mathbf{x}) = \log p_K(\mathbf{z}_K) &= \log p_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\ &= \log p_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\ &= \dots \\ &= \log p_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|. \end{aligned} \tag{4.9}$$

This is probably the most important equation for Normalizing Flows, relating a known probability density function p_0 to the distribution of our data $p(x)$. For p_0 a Gaussian distribution is usually considered, mainly because it is a well-understood distribution with well-defined mathematical properties and closed-form expressions, and because sampling is computationally efficient, among others.

With the expression (4.9), the log-likelihood of the train data is now tractable. Therefore, we can directly use it for the optimization itself. The loss for flow-based generative models is therefore:

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}). \tag{4.10}$$

Note that (4.9) is easy to compute if functions f_i satisfy two main properties:

- Easy to invert,
- Their Jacobian determinants are easy to compute,

as the other term is just evaluating the Gaussian PDF on the latent variables. The next question is how to build these functions f_i. Basically, any function that fulfills the two conditions above might be suitable. Much effort has been put into designing these functions to obtain the best results possible. The following presents the most important ones for this thesis.

4.1.3 Affine coupling layer

A predecessor of the affine coupling layer was initially introduced in NICE [47] and subsequently improved upon in RealNVP [48] and Glow [82]. The core concept is to achieve an invertible transformation by keeping half of the data unchanged while transforming the other half based on the unchanged part. This approach ensures that the inverse transformation is straightforward, as the first half of the data remains the same, and we can compute the transformations again. This simple trick eliminates the need to invert the applied transformation,

allowing for the use of arbitrarily complex operations. Simple neural networks are usually used for these transformations.

The affine coupling layer from Glow, which is the best-performing one and the one used in this thesis. Specifically, the input x is split into two parts: x_a and x_b. The affine coupling layer applies a transformation to x_b using a scaling function $s(x_a)$ and a translation function $t(x_a)$ conditioned on x_a. The output y consists of $y_a = x_a$ and $y_b = x_b \odot s(x_a) + t(x_a)$. Figure 4.2 shows a diagram of this layer.

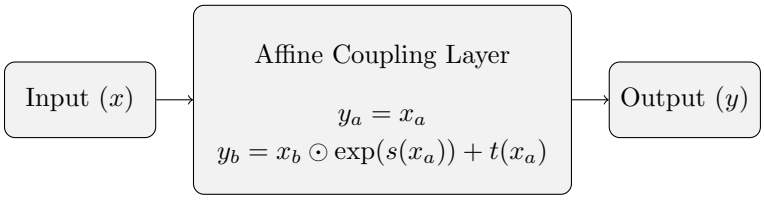


Figure 4.2: Diagram of the Affine Coupling Layer.

These operations can be easily inverted for the reverse process. One can compute x from y as:

$$\begin{aligned} x_a &= y_a, \\ x_b &= (y_b - t(y_a)) \odot \exp(-s(y_a)). \end{aligned} \tag{4.11}$$

The forward and reverse processes are depicted in Figure 4.3, where we can see that both $s(\cdot)$, and $t(\cdot)$ are run in the direct direction for both cases (forward and reverse propagation), and therefore there is no need to invert them.

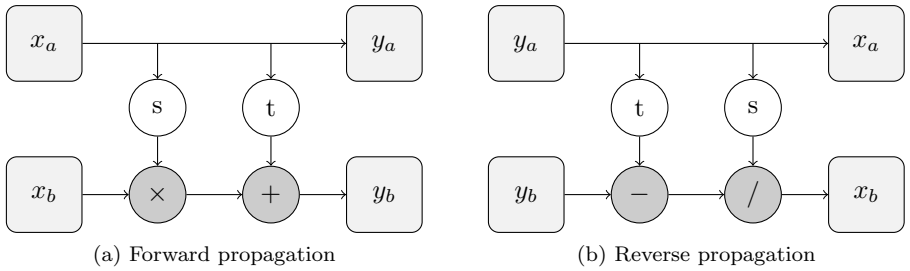
4.1.4 Glow

From the explanation above, it is evident that using only the affine coupling layer may not be enough, as half of the input remains unchanged. Therefore, some kind of permutation is needed so that different input parts remain unchanged in each layer. In RealNVP [48], the authors proposed to use fixed permutations, while in Glow [82] they replaced them by 1×1 convolutions.

Basically, a Glow block consists of three steps: an activation normalization (actnorm), an invertible 1×1 convolution, and an affine coupling layer, as shown in Figure 4.4, and explained in the following.

Activation Normalization (ActNorm). Activation normalization, or "actnorm", performs an affine transformation using scale and bias parameters per channel, similar to batch normalization. However, it is designed to work with a mini-batch size of 1. The parameters are trainable and initialized so that the mini-batch of data has a mean of 0 and a standard deviation of 1 after actnorm.

Invertible 1x1 Convolution. In RealNVP, the channel order is inverted between each pair of affine coupling layers, so in this way, all dimensions are altered. A 1×1 convolution with the same number of input and output channels



(a) Forward propagation (b) Reverse propagation

Figure 4.3: Diagram of the affine coupling layer defined in Glow [82]

generalizes any permutation of channel ordering. Using these results, Glow's authors propose using an invertible 1×1 convolution to reorder channels, ensuring all data dimensions can be altered.

Affine Coupling Layer. This layer is exactly the same as the one presented in the previous section, allowing for efficient and invertible transformations of the input data.

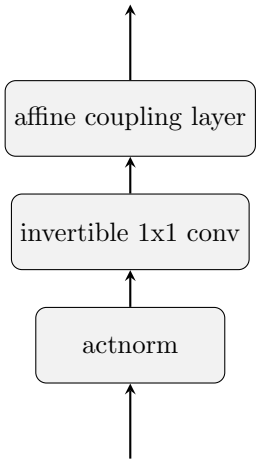


Figure 4.4: Diagram of the layers in a Glow [82] layer.

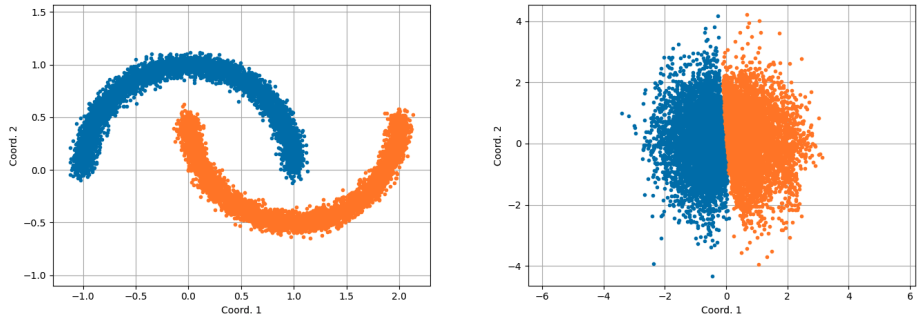
As explained before, we need the functions f_i to be easily invertible and their Jacobian determinants to be easy to compute. As a summary, Table 4.1 shows all the expressions needed for the computation of a Glow block.

4.2 Example

To illustrate how Normalizing Flows work, let us consider the simple toy dataset Two Moons. The idea of this example is to train a NF to learn the mapping

Block	Function	Reverse Function	Log-determinant
Actnorm	$\forall i,j:$ $y_{i,j} = \mathbf{s} \odot x_{i,j} + \mathbf{b}$	$\forall i,j:$ $x_{i,j} = (y_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \sum(\log \lvert\mathbf{s}\rvert)$
1x1 conv.	$\forall i,j:$ $y_{i,j} = \mathbf{W} x_{i,j}$	$\forall i,j:$ $x_{i,j} = \mathbf{W}^{-1} y_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \sum(\log \lvert\mathbf{s}\rvert)$
Affine coupling layer	$x_a, x_b = \text{split}(x)$ $(\log s, t) = \text{NN}(x_b)$ $s = \exp(\log s)$ $y_a = s \odot x_a + t$ $y_b = x_b$ $y = \text{concat}(y_a, y_b)$	$y_a, y_b = \text{split}(y)$ $(\log s, t) = \text{NN}(y_b)$ $s = \exp(\log s)$ $x_a = (y_a - t)/s$ $x_b = y_b$ $x = \text{concat}(x_a, x_b)$	$\sum(\log(\lvert\mathbf{s}\rvert))$

Table 4.1: Description, functions, reverse functions, and log-determinants of the layers used in the Glow block. Table adapted from [82].



(a) True distribution: Two Moons.

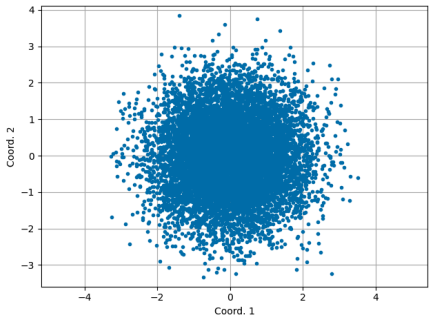
(b) After inference with NF, samples are close to a Gaussian distribution.

Figure 4.5: **Inference**. Sampling from the true distribution, in this case, the Two Moons distribution, and transforming samples through the whole network until we obtain samples close to a Gaussian distribution. Samples are colored according to each cluster in the original distribution (each of the moons).

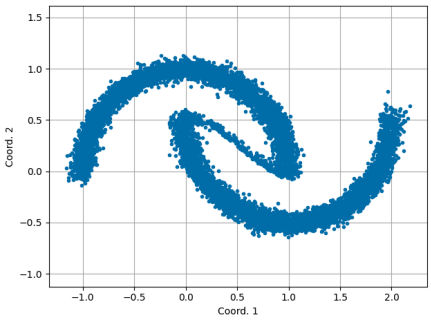
from the Gaussian distribution to the Two Moons distribution.

For this example, the NF is constructed by concatenating eight Glow blocks. At the end of the training phase, we obtain the results shown in Figures 4.5 and 4.6 for inference and generation, respectively.

This simple experiment also helps us understand how we will use Normalizing Flows in the context of anomaly detection. Let us assume we are in the situation of Figure 4.5(a), where we have many "normal" samples. In this case, the notion of normality would be a good fit for the Two Moons dataset. In this case, a normal point could be any of the plotted blue or orange points. Passing all these points through the network leads to points following a Gaussian distribution, as shown in Figure 4.5(b). Now, let's imagine we have an anomalous data point far from both moons. Getting a notion of how far this point is



(a) Initial points following a Gaussian distribution.



(b) Transformed points. The output distribution is very close to the original data distribution, the Two Moons distribution.

Figure 4.6: **Generation**. For the generative part of the Normalizing Flows, we start by sampling from (usually) a standard Normal distribution and then pass these data points through the network. The invertible functions f_i sequentially transform the data points until we obtain samples following a distribution close to the original distribution of our training data. Note that, in this case, the moons are linked by a small path. This is because the functions f_i smoothly deform the space and can only generate connected clusters.

could be difficult, especially if working in a high-dimensional space. Normalizing Flows makes this task really easy, as an anomalous sample in the original data space would be mapped far from the origin in the latent space, where normal data is expected to follow a standard Normal distribution. Measuring how far this point is from the normal data is straightforward in the latent space, as we can estimate the likelihood of the sample belonging to the Normal distribution using an analytical expression.

In a general setting, we can assume each point i is represented by a vector $Z_i = \{z_i^1, \dots, z_i^C\}$. We can compute the likelihood of this point of belonging to the Normal distribution directly evaluating the following expression:

$$\begin{aligned} l_i &= \frac{1}{\sqrt{2\pi}^C} \exp\left(-\frac{1}{2} Z_i^T Z_i\right) \\ &= \frac{1}{\sqrt{2\pi}^C} \exp\left(-\frac{1}{2} \sum_{k=1}^{C} (z_i^k)^2\right). \end{aligned} \tag{4.12}$$

Now that the concepts of Normalizing Flows have been introduced, we can move to the following chapter, which presents a method for anomaly detection and segmentation based on Normalizing Flows. As will be explained later, for computing an anomaly map, we use the estimation of the likelihood with an expression very similar to the one presented in 4.12.

Chapter 5

U-Flow: A U-shaped Normalizing Flow for Anomaly Detection with Unsupervised Threshold

This chapter presents a work in which we propose a one-class self-supervised method for anomaly segmentation in images that benefits both from a modern machine learning approach and a more classic statistical detection theory. The method consists of four phases. First, features are extracted using a multi-scale image Transformer architecture. Then, these features are fed into a U-shaped Normalizing Flow (NF) that lays the theoretical foundations for the subsequent phases. The third phase computes a pixel-level anomaly map from the NF embedding, and the last phase performs a segmentation based on the a contrario framework. This multiple hypothesis testing strategy permits the derivation of robust unsupervised detection thresholds, which are crucial in real-world applications where an operational point is needed. The segmentation results are evaluated using the Mean Intersection over Union (mIoU) metric, and for assessing the generated anomaly maps, we report the area under the Receiver Operating Characteristic curve (AUROC), as well as the Area Under the Per-Region-Overlap curve (AUPRO). Extensive experimentation in various datasets shows that the proposed approach produces state-of-the-art results for all metrics and all datasets, ranking first in most MVTec-AD categories, with a mean pixel-level AUROC of 98.74%.

Code and trained models are available at `https://github.com/mtailanian/uflow`.

5.1 Introduction

Detecting image anomalies is a long-standing problem that has been studied for decades. In recent years, this problem has received a growing interest from the computer vision community, motivated by applications in various fields, ranging from surveillance and security to even health care. One of the most common applications, and the one that we are especially interested in, is automatizing product quality control in an industrial environment. In this case, it is often very hard (sometimes even unfeasible) to collect and label a good amount of data representing all kinds of anomalies since these are rare structures by definition. For this reason, the major effort on anomaly detection methods has been focused on learning only from the normal samples (i.e., anomaly-free images) and detecting the anomalies as everything that deviates from what has been learned. This approach is found in the literature under different names, usually called one-class classification, self-supervised anomaly, or novelty detection.

There is a wide literature on anomaly detection, and many approaches have been proposed. One of the most common classic approaches consists of embedding the training images in some latent space and then evaluating how far a test image is from the manifold of normal images. Other approaches focus on learning the probability density function of the training set. Among these, three types of generative models have become popular in anomaly detection in the last five years. On the one hand, Generative Adversarial Networks [63, 128] implicitly learn this probability, being able only to sample from it. On the other hand, Variational Auto-Encoders [81, 171] can explicitly estimate this probability density, but they can only learn an approximation, as they maximize the evidence lower bound. And finally, Normalizing Flows [47, 48, 82] are able to learn the exact probability density function explicitly. Being able to do so has several advantages, as it permits to estimate the likelihood and score of how probable it is that the tested data belongs to the same training distribution. A significant advantage we exploit in this work is the possibility of developing formal statistical tests to derive unsupervised detection thresholds. This is a crucial feature for most real-world problems where a segmentation of the anomaly is needed. To this end, we propose a multiple hypothesis testing strategy based on the *a contrario* framework [46].

Common approaches based on VAEs [14, 62, 179] and GANs [2, 127, 128] have demonstrated poor performance in the industrial defect detection case, where anomalies are actually very similar to the normal parts of the images. To enhance the performance in these cases, instead of grounding the detection on the RGB image itself, in recent years, many works have introduced a feature extraction stage using a pre-trained network, usually on ImageNet [125], and perform the anomaly detection in the feature space [79, 95, 123, 170, 173, 178].

In this work, we propose **U-Flow**, a new self-supervised method that ensembles the features extracted by multi-scale image transformers into a U-shaped Normalizing Flow's architecture and applies the *a contrario* methodology for finding an automatic segmentation of anomalies in images. Our work achieves state-of-the-art results, widely outperforming previous works in terms of $mIoU$ for the segmentation, and exhibiting top performance in localization metrics in terms of $AUROC$ and $AUPRO$. The main contributions in U-Flow are:

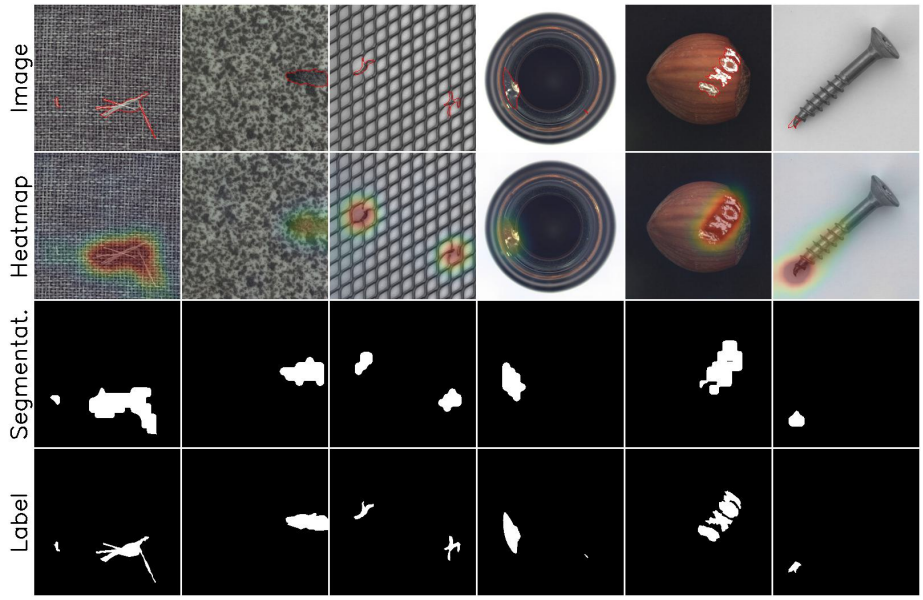


Figure 5.1: Anomalies detected with the proposed approach on MVTec-AD examples from different categories. Top row: original images with ground truth segmentation. Second row: corresponding anomaly maps. Third row: automatic segmentations. Last row: ground truth masks.

- For the feature extraction, we propose a multi-scale vision Transformer architecture based on the CaiT Transformers [146], pre-trained independently for each scale, which we refer to as **MS-CaiT**. Results clearly show that it better captures the multi-scale information and increases performance, comparing it both against other neural networks' feature extractors and other multi-scale Transformers.

- We leverage the well-known advantages of U-like architectures in the setting of Normalizing Flows, proposing a novel architecture and solving non-trivial technical details for merging the scales in an invertible way that better aggregates the multi-scale information elegantly and efficiently while imposing statistical mutual independence within and across scales.

- We propose, for the first time, to take advantage of the statistical properties of Normalizing Flows to derive unsupervised thresholds for anomaly detection and segmentation. More precisely, the resulting multi-scale characterization of the anomaly-free images as a white Gaussian process in the latent space allows us to propose a multiple hypothesis testing strategy that leads to an automatic detection threshold to segment anomalies, outperforming state-of-the-art methods.

- Although the method is designed for industrial defect detection tasks, we show that it also exhibits excellent performance on other benchmark

datasets of different natures.

- Results are easily reproducible. Code is available on GitHub, and also integrated into *Anomalib* library [1].

In short, we propose an easy-to-train method that does not require any parameter tuning or modification to the architecture (as opposed to other state-of-the-art methods), that is fast, accurate, and provides an unsupervised anomaly segmentation. Example results are shown in Figure 5.1.

The remainder of this chapter is organized as follows. In Section 5.2, we discuss previous work related to the proposed approach. Details of the method are presented in Section 5.3. In Section 5.4, we compare the performance of our anomaly detection approach with a large number of state-of-the-art methods on several benchmark datasets of different nature: MVTec-AD [13], BeanTech [106], LGG MRI [21], and ShanghaiTech Campus [99]. An ablation study that analyzes the role of each component of the proposed architecture is presented in Section 5.5. We conclude in Section 5.9.

5.2 Related work

The literature on anomaly detection is vast, with numerous proposed approaches. In this work, we focus on one-class self-supervised learning for industrial inspection, where a key element is to learn only from anomaly-free images. Within this group, methods can be further divided into several categories in many different ways.

A large number of methods fall under the category of representation-based techniques. These methods work by embedding samples into a latent space equipped with a specific metric. Several methods discussed in depth in Section 3.2 belong to this category. We recall some of them as they will be used for comparison with the method proposed in this chapter. PatchSVDD [172] constructs a hierarchical encoding of image patches, while SPADE [35] and PaDiM [44] build representations using standard feature extraction networks. PatchCore [121] further improves SPADE and PaDiM by estimating anomalies with a larger nominal context and making it less reliant on image alignment. Additionally, in [147], the authors aim to learn more representative features from normal images by using three encoders with three classifiers. In CFA [95], the authors propose a method to construct features using a patch descriptor and a scalable memory bank. They design a special loss function to map features to a hyper-sphere, densely clustering the representation of normal images. The method can also incorporate a few abnormal samples during training to increase the distance to the normal samples in the latent space.

Other methods manage to cast this problem into a self-supervised one by creating a proxy task [97], or rely on the concept that training a network exclusively with anomaly-free images should enable the network to accurately reconstruct normal images while failing to reconstruct anomalies since it has never encountered them during training. As discussed in Section 3.2, achieving this balance (where only normal structures are generated) can be challenging.

Methods often produce either poor-quality reconstructions or exhibit excessive generalization, resulting in well-reconstructed anomalies. This concept can be implemented in various ways, such as using a teacher-student scheme [169], or encoder-decoder architectures [14, 171, 179]. To address the generalization issue in reconstruction, the authors in [62] limit the reconstruction capability by incorporating memory modules in the latent space.

Regarding the use of Normalizing Flows (NF) in anomaly detection, a few methods have recently been proposed with impressive results. DifferNet [122] applies an average pooling to the features extracted with a pre-trained network and processes the resulting vectors in a one-dimensional NF, losing important contextual and spatial information. The authors try to alleviate this problem by passing different rotated image versions through the network at the cost of increasing the computational complexity. The method performs only anomaly detection, but a rough localization can also be obtained by back-propagating the negative log-likelihood through the network. CFlow [66] also uses a one-dimensional NF but includes the spatial information using a positional encoding as a conditional vector. On the other hand, FastFlow [173] directly uses a two-dimensional NF for each scale independently and averages the results at the end. CS-Flow [123] uses the same NF block as in [122] and [173], but for the affine layer, the information of the different scales are merged, by summing an upsampled/downsampled version of the feature volumes from the other scales. This method extends DifferNet and is also meant only for anomaly detection. However, as all the network's layers are convolutional, the spatial information is preserved, and an anomaly map can be obtained with the forward pass. The results for anomaly detection are excellent, but as the extracted features are of a small spatial size, the performance in localization could be better.

In this work, we further improve the performance of these methods for anomaly localization by proposing a multi-scale Transformer-based feature extractor and by equipping the NFs with a fully invertible U-shaped architecture that effectively integrates the information at different scales in a very natural and elegant manner. These two ingredients allow us to embed these multi-scale features in a latent space where intra and inter-scale features are guaranteed to be independent and identically distributed Gaussian random variables. Finally, we design a detection methodology from these statistical properties that leads to statistically meaningful anomaly scores and unsupervised detection thresholds.

5.3 Method

The proposed method is depicted in Figure 5.2 and consists of four main phases: 1) Feature Extraction, 2) U-shaped Normalizing Flow, 3) Anomaly score map, and 4) *A contrario* anomaly segmentation. All phases are presented below.

5.3.1 Phase 1: Feature Extraction

Since anomalies can emerge in various sizes and forms, collecting image information at multiple scales is essential. To do so, the standard deep learning strategy is to use pre-trained CNNs, often a VGG [132] or any variant of the

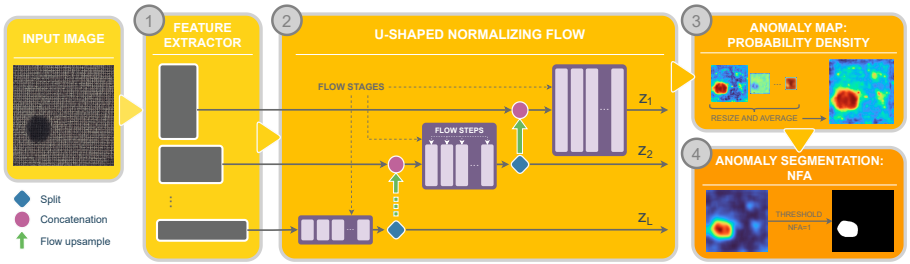


Figure 5.2: The method consists of four phases. (1) *Multi-scale feature extraction:* a rich multi-scale representation is obtained with MS-CaiT by combining pre-trained image Transformers acting at different image scales. (2) *U-shaped Normalizing Flow:* By adapting the widely used U-like architecture to NFs, a fully invertible architecture is designed. This architecture is capable of merging the information from different scales while ensuring independence in both intra- and inter-scales. To make it fully invertible, split and invertible up-sampling operations are used. (3) *Anomaly score map generation:* an anomaly map is generated by associating a likelihood-based anomaly score to each pixel in the test image. (4) *Anomaly segmentation:* Besides generating the anomaly map, we also propose to adapt the *a contrario* framework to obtain an automatic threshold by controlling the allowed number of false alarms.

ResNet [70] architectures, to extract a rich multi-scale image feature representation, by keeping the intermediate activation maps at different depths of the network. More recently, with the development of image vision Transformers, some architectures such as ViT [50] and CaIT [146] are also being used, but in these cases, a single feature map is retrieved. The features generated by vision Transformers are proven to better compress all multi-scale information into a single feature map volume, compared to the standard CNNs. However, this notion of a unique multi-scale feature map was challenged by MViT [56], which proved that combining the standard CNNs' idea of having multi-scale feature hierarchies with the Transformer models, the Transformers' features could be further enhanced. This work was followed by MViT2 [98], which was shown to outperform its predecessor and all other vision Transformers in various tasks. In our work, we also propose to construct a multi-scale Transformer architecture but following a different strategy. Specifically, we propose MS-CaiT, an architecture that employs CaIT Transformers at different scales, independently pre-trained on ImageNet [125], and combines them as the encoder of a U-Net [120] architecture. Despite its simplicity, an ablation study presented in Section 5.5.2 shows that this combination strategy leads to better results than using each of the Transformers alone, far better than other ResNet variants, and even better than MViT2. We conjecture that the fact that the CaIT networks were trained independently could be beneficial because it might give the network more flexibility to concentrate on different structures in each Transformer.

As a final comment regarding the feature extraction stage, it is important to mention that, notwithstanding all the results reported in Section 5.4 were obtained using exactly the same architecture configuration, the proposed architecture, and the code are agnostic to the feature extractor, and can be used with different extractors and different numbers of scales. For specific adaptation to certain situations and datasets, it could be better to use another extractor, for example, if computational power is very tight (e.g., in an industrial environ-

ment), probably at the expense of some decrease in performance.

5.3.2 Phase 2: Normalizing Flow

Normalizing Flows [118] are a family of generative models that are trained by directly maximizing the log-likelihood of the input data and which have the particularity of learning a bijective mapping between the input distribution and the latent space. Using a series of invertible transformations, the NF can be run in both directions. The forward process embeds data into the latent space and can serve as a measure of the likelihood. The reverse process starts from a predefined distribution (usually a standard Normal distribution) and generates samples following the learned data distribution.

The rationale for using NFs in an anomaly detection setting is straightforward. The network is trained using only anomaly-free images, and in this way, it learns to transform the distribution of normal images into a white Gaussian noise process. At test time, when the network is fed with an anomalous image, it is expected that it will not generate a sample with a high likelihood, according to the white Gaussian noise model. Hence, a low likelihood indicates the presence of an anomaly.

This second phase is the only one that is trained. It takes the multi-scale representation generated by the feature extractor as input and performs a sequence of invertible transformations on the data using the NF framework.

State-of-the-art methods following this approach are centered on designing or trying out different multi-scale feature extractors. In this work, we further improve the approach by proposing a new feature extractor and a multi-level deep feature integration method that aggregates the information from different scales using the well-known and widely used UNet-like [120] architecture. The U-shape comprises the feature extractor as the encoder and the NF as the decoder. We show in the ablation study of Section 5.5.1 that the U-shape is a beneficial aggregation strategy that further improves the results.

The 2D NF uses only invertible operations and treats each pixel location independently, only merging the information between different channels for each pixel location. As it is implemented as one unique computational graph, we obtain a fully invertible network. Moreover, optimizing the whole flow all at once has a crucial implication: the Normalizing Flow generates a mutually independent embedding not only within each scale but also across different scales, unlike other state-of-the-art flow-based methods. It is worth mentioning that in most works reported so far in the literature, the anomaly score is computed first at each scale independently, using a likelihood-based estimation, and finally merged by averaging or summing the result for each scale. Because of the lack of independence between scales, these final operations, although achieving very good performance, lack a formal probabilistic interpretation. The NF architecture proposed in this work overcomes this limitation; it produces statistically independent multi-scale features, for which the joint likelihood estimation becomes trivial.

Architecture. The U-shaped NF is compounded by a number of *flow stages*,

each one corresponding to a different scale, whose size matches the extracted feature maps (see Figure 5.2). For each scale starting from the bottom, i.e., the coarsest scale, the input is fed into its corresponding *flow stage*, which is essentially a sequential concatenation of *flow steps*. The output of this *flow stage* is then split in such a way that half of the channels are sent directly to the output of the whole graph, and the other half is up-sampled to be concatenated with the input of the next scale, and proceed in the same fashion. The up-sampling is also performed in an invertible way, as proposed in [74], where pixels in groups of four channels are reordered in such a way that the output volume has four times fewer channels and double width and height.

Each *flow step* has a size according to its scale and is compounded by Glow blocks [82]. Each step combines an Affine Coupling Layer [47], a permutation using 1×1 convolutions, and a global affine transformation (ActNorm) [48]. The Affine Coupling Layer uses a small network with the following layers: a convolution, a ReLU activation, and one more convolution. Convolutions have the same amount of filters as the input channels, and kernels alternate between 1×1 and 3×3.

To sum up, the U-shaped NF produces L white Gaussian embeddings $z^1, \ldots, z^L$, one for each scale l, $z^l \in \mathbb{R}^{C_l \times H_l \times W_l}$. Here, we denote the spatial location by (i, j), and by k the channel index in the latent tensors. Its elements

$$z^l_{ijk} \;\sim\; \mathcal{N}(0, 1), \tag{5.1}$$

$1 \leq i \leq H_l$, $1 \leq j \leq W_l$, $1 \leq k \leq C_l$, are mutually independent for any position, channel and scale i, j, k, l.

5.3.3 Phase 3: Likelihood-based Anomaly Score

This phase of the method is to be used at test time when computing the anomaly map. It takes as input the embeddings $\left\{z^l_{ijk}\right\}$ and produces an anomaly map based on the likelihood computation. Thanks to the statistical independence of the features produced by the U-shaped NF, the joint likelihood of a test image under the anomaly-free image model is the product of the standard normal marginals. Therefore, to build this map, we associate to each pixel (i, j) in the test image a likelihood-based Anomaly Score similar to those in [66, 173]:

$$AS(i, j) = 1 - \frac{1}{L} \sum_{l=1}^{L} \exp\left(-\frac{1}{2C_l} \sum_{k=1}^{C_l} (\tilde{z}^l_{ijk})^2 \right), \tag{5.2}$$

where $\tilde{z}^l_{ijk}$ is a bilinearly upsampled version of z^l_{ijk} at the original $H \times W$ resolution.

Note that the exponential term of the above expression does not correspond exactly to the anomaly-free likelihood since, for each scale, an average in the channel direction is computed instead of the sum. The reason for this is mainly computational: using the sum produces extremely small values close to the floating point error. Using the sum has another undesirable effect: the impact of the different scales in the total anomaly score is not equalized, as the number of channels C_l may significantly differ. Although these modifications

lead to an Anomaly Score that lacks a precise probabilistic interpretation, this computation produces high-quality anomaly maps and increases performance in practice.

In addition, in the following section, we introduce a more formal and statistically meaningful result, using the *a contrario* theory, that permits to derive an unsupervised detection threshold on the *Number of False Alarms (NFA)*, and produces an anomaly segmentation mask.

5.3.4 Phase 4: *A contrario* anomaly segmentation

Besides generating the anomaly map discussed in the previous section, we present a new method for computing an automatic segmentation of the anomalies based on the *a contrario* approach. In this approach, an estimate of the Number of False Alarms (NFA) is associated with each anomaly candidate, and detections are obtained by thresholding the NFA [46]. The proposed method has no parameters to tune and uses level sets organized into a tree structure to segment the anomalies automatically.

As explained in Chapter 2 (Section 2.4), to each event E we associate a NFA value as $\mathrm{NFA}(E) = N_T \mathrm{Pr}_{\mathcal{H}_0}(E)$. In the remainder of this section, we explain how we construct the null hypothesis $\mathcal{H}_0$, estimate the probability $\Pr(E)$, and determine the number of tests N_T. Note that, as the events E corresponding to anomalies can have arbitrary sizes and shapes, we must test all possible configurations of connected sets of pixels within every image. But this is computationally very inefficient and possibly even impossible in a reasonable amount of time. Therefore, we propose to use the anomaly map to select the regions to be tested from the set of connected components of its level sets. These components are naturally ordered by inclusion, providing a convenient hierarchical representation that can be organized in a tree structure.

Detecting anomalies from the level sets of the anomaly map

In image processing, level sets are the basis for a wide variety of morphological filters, and, as they are closely related to object boundaries, they have been used for edge detection [24, 45], registration [109, 111], image quantization [7], and segmentation [168], among others. All the information of a gray-valued image $u(i, j)$ is contained in a set of binary images obtained by thresholding at different levels. Since no information is lost, level sets provide a complete representation: any image can be easily reconstructed from the whole family of its (upper) level sets [130]

$$\mathcal{L}_\lambda = \{(i,\, j) \mid u(i,\, j) \geq \lambda\}\,. \tag{5.3}$$

We denote by $\mathcal{L}_{\lambda,c}$ the c-th connected component of $\mathcal{L}_\lambda$.

Level sets naturally define a hierarchical representation, ordered by their geometrical inclusion. In the case of the upper level sets, each connected component defined by a certain level is included in another connected component defined by a lower level. Therefore, they can be naturally embedded in a tree representation.

(a) Example image u used to build the tree. Each pixel has its value annotated.

10	5	7
9	4	5
6	3	9

(b) Same image u, annotated with the raveled index of each pixel.

0	1	2
3	4	5
6	7	8

(c) Example tree of connected components built from the upper level sets of (a). Each node is represented by a set of indexes from (b), and each row corresponds to a different level set, according to the values of (a).

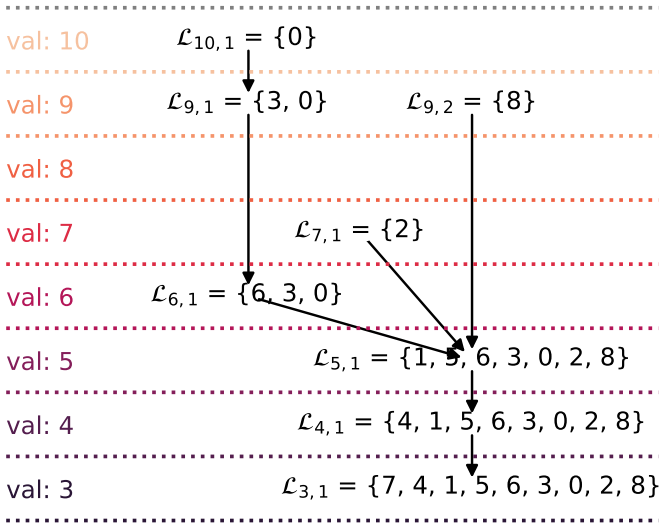


Figure 5.3: Tree of connected components of the upper level sets: the hierarchical representation based on the level sets used to retrieve the most significant regions to be tested for anomaly segmentation.

We treat all scales produced by the U-Flow independently and use these properties to build one *tree of connected components* for each scale. For simplicity, in this section, we present the method for one scale, and later in Section 5.3.4, we explain how to combine the results from different scales into a final set of detections. In the following, we omit the superscript l used to denote the embedding scale.

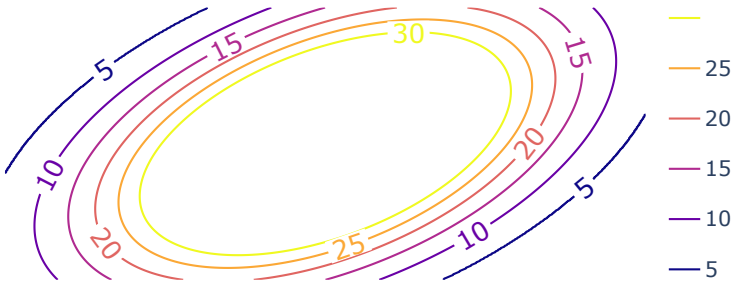


Figure 5.4: Example level lines of a branch of the tree of connected components.

Specifically, for each scale, we propose to use the upper level sets of an image defined from the NF embeddings as

$$u(i,j) = \sum_{k=1}^{C} (z_{ijk})^2, \tag{5.4}$$

where C is the number of channels of the embedding. Note that for building the tree of connected components, as we are only interested in the level sets, using $u(i,j)$ is equivalent to using the Anomaly Score (5.2) for a single scale l since this score is an increasing function of $u(i,j)$.

Figure 5.3 shows a toy example that illustrates the procedure. Figure 5.2a represents the image u, with values $u(i,j)$ denoted by numbers inside each pixel. Figure 5.2b shows the raveled indexes that serve as labels for the tree nodes shown in Figure 5.2c. The tree is built so that the nodes are given by the connected components of the level set $\mathcal{L}_{\lambda,c}$, and the edges represent the inclusion, connecting level set components from two different levels.

The tree of connected components allows us to significantly reduce the number of image regions we effectively test. Instead of computing the NFA of all possible N_T connected regions of any shape and size, which would be intractable, we will limit this computation to the connected components of the upper level sets in the tree. By construction, these regions are expected to be good candidates for anomalies.

To compute the NFA of a connected component $\mathcal{L}_{\lambda,c}$ in the tree, we need to evaluate the Probability of False Alarm (PFA), i.e., the probability of occurrence of a region like the observed $\mathcal{L}_{\lambda,c}$ under the background model of normality. In the absence of anomaly, the embedding produced by the U-Flow model is $z^l_{ijk} \sim \mathcal{N}(0,1)$, i.i.d. Therefore, variables $u(i,j)$ as defined in (5.4) are identically distributed as a *Chi-Squared* distribution of order C. Then, as the minimum pixel value in the connected component $\mathcal{L}_{\lambda,c}$ is λ, it follows that its probability of false alarms is given by

$$\begin{aligned} \mathrm{PFA}(\mathcal{L}_{\lambda,c}) &= \Pr\left(\min_{(i,j)\in\mathcal{L}_{\lambda,c}} u(i,j) \geq \lambda \right) \\ &= \left(1 - \underset{\chi^2(C)}{CDF}(\lambda) \right)^{|\mathcal{L}_{\lambda,c}|}, \end{aligned} \tag{5.5}$$

where $|\mathcal{L}_{\lambda,c}|$ denotes the number of pixels in $\mathcal{L}_{\lambda,c}$. This measure of meaningfulness of a region presents a natural balance between growing the regions (and with that increasing $|\mathcal{L}_{\lambda,c}|$) and reducing the minimum value of its pixels (i.e., reducing λ). This balance is desirable to delineate the abnormal regions well.

For efficiency purposes, instead of computing the *CDF* in (5.5), we use the Chernoff bound [30] for a $\mathcal{X}^2(C_l)$ distribution obtaining the final base 10 logarithm of the PFA value:

$$\begin{aligned} &\log(\mathrm{PFA}(\mathcal{L}_{\lambda,c})) = \\ &\quad = |\mathcal{L}_{\lambda,c}| \frac{C_l}{2\ln(10)} \left(1 + \ln\left(\frac{\lambda}{C_l}\right) - \frac{\lambda}{C_l}\right). \end{aligned} \tag{5.6}$$

Once we have computed the log(PFA) values of all connected components in the tree, to obtain the final regions, we iteratively perform the following *prune* and *merge* procedures until convergence (until the tree does not change anymore).

Procedure 1: PFA Prune. This procedure aims to filter a set of nested connected components, keeping only the most significant one. To do so, we directly use the log(PFA) value of each node (each connected component), and select the lowest one. Starting from each leaf, we identify a connected section of the tree where all nodes have exactly one predecessor (except for the starting leaf itself). The concentric level lines defined by such a branch are illustrated by the example in Figure 5.4. In other words, we identify which of these connected components is the most significant one, as it may better delineate the anomalous region. After determining which node to preserve, the tree is pruned so that just the chosen node is kept, and all other branch nodes are removed. In the example of Figure 5.2c, we would consider the branch $b_1 = \{\mathcal{L}_{10,1}, \mathcal{L}_{9,1}, \mathcal{L}_{6,1}\}$, evaluate the log(PFA) value for each node, and prune the tree, keeping only the one with lowest value, i.e. the most significant region, denoted as $\mathcal{L}_{b_1}$.

Procedure 2: PFA Merge. The second procedure consists of merging leaf nodes with the same successor in case the latter is more significant than all others. In this case, all leaf nodes are removed from the tree, and we only keep their successors. In the example of Figure 5.2c, this procedure would analyze if the node $\mathcal{L}_{5,1}$ (the successor) is more significant than $\mathcal{L}_{b_1}$ (the leaf predecessor node resulting from *Procedure 1*), $\mathcal{L}_{9,2}$, and $\mathcal{L}_{7,1}$. In case $\mathcal{L}_{5,1}$ has the lowest log(PFA) value between all these nodes, the other nodes are removed from the tree. Otherwise, this procedure makes no changes to the tree.

Note that in the case some actual merge is performed in *Procedure 2*, it could generate a new branch with more than one node, starting from a new leaf, where all nodes in the branch have only one predecessor (except the starting leaf itself). Therefore, we iteratively apply both procedures until no more changes are performed over the tree. At the end of this iterative process, we use the resulting leaf nodes of the tree as the final regions and assign them the log(PFA) value of the corresponding connected component, generating a log(PFA) heatmap per scale.

log(NFA) computation

The scales are merged by upsampling the resulting log(PFA) heatmaps at different scales l, up to the original input size (H, W), and keeping the minimum value at each pixel position (i, j):

$$\log\left(\mathrm{PFA}(i,j)\right) = \min_{l} \left\{ \overset{H,W}{\uparrow} \left(\log\left(\mathrm{PFA}^{l}\right)\right)(i,j) \right\}, \tag{5.7}$$

where $\overset{H,W}{\uparrow}(\cdot)$ represents the bilinear upsample up to the original size.

Finally, for the number of tests N_T, we need to quantify all possible regions, or arrangements of connected pixels of any size and shape, located in any part of the image. Considering 4-connectivity, these groups of connected pixels correspond to the figures called *polyominoes*, and a good approximation for the number of polyominoes of r pixels is given by $\alpha\beta^r/r$, with $\alpha = 0.316915$, and $\beta = 4.062570$ (see [76,149]). Therefore, considering all possible region sizes and shapes in all possible spatial positions of all scales, results in

$$N_T = \sum_{l} H_l\, W_l \sum_{r=1}^{H_l W_l} \alpha \frac{\beta^r}{r}. \tag{5.8}$$

Finally, the log(NFA) value is given by

$$\log(\mathrm{NFA}(i,\, j)) = \log(N_T) + \log(\mathrm{PFA}(i,\, j)). \tag{5.9}$$

5.4 Experimental Results

The proposed approach was tested and compared with state-of-the-art methods, by conducting extensive experimentation on several datasets: **MVTec-AD** [13], **BeanTech** (BT) [106], **LGG-MRI** (MRI) [21], and **ShanghaiTech Campus** (STC) [99].

MVTec-AD is the most widely used benchmark for anomaly detection, as most state-of-the-art anomaly detection methods report their performances on this dataset. It consists of 15 categories of textures and objects, simulating real-world industrial quality inspection scenarios with 5,000 images. Each category has normal (i.e., anomaly-free) images for training, and the testing set consists of images with different kinds of anomalies and some more normal images. **BeanTech** is also a real-world industrial anomaly dataset. It contains 2830 images of three categories, including normal and defective images.

Besides testing the proposed method on the industrial inspection task (which is the motivation and focus of this work), we also include experimentation with data from other fields to demonstrate the generalization capability of our method. To do so, we test and compare our method on datasets from the medical and surveillance fields. **LGG-MRI** contains images of lower-grade gliomas (brain tumors), with images of the tumors' evolution over time, from 110 patients of 5 institutions, taken from The Cancer Genome Atlas. Lastly, the

ShanghaiTech Campus dataset contains frames extracted from videos of 13 different scenes of surveillance cameras of the mentioned campus.

For assessing the anomaly maps defined in (5.2), we adopt the $AUROC$ metric (the area under the Receiver Operating Characteristic curve) at a pixel level, as it is the most widely used metric in the literature. We also consider $AUPRO$ (the area under the per-region-overlap curve) since it ensures that both large and small anomalies are equally important. For assessing the anomaly masks (after thresholding), we use the $mIoU$ metric (Mean Intersection over Union). Finally, even though the image-level anomaly detection task is not the primary focus of this work, we have included the image-level $AUROC$ results in Section 5.6 because we understand that it is important to check that any anomaly localization method should also perform well when used to classify an entire image as anomalous or normal (the detection task).

5.4.1 Results on the MVTec-AD dataset

Before proceeding to present the results on this dataset, it is worth mentioning that since the trained models for FastFlow, CFlow, and CS-Flow are not available, it was necessary to re-train them for all the categories. To ensure a fair comparison, we performed numerous training runs until the obtained $AUROC$ were comparable to the ones reported in the original papers. Note that for FastFlow and CFlow, a different set of hyper-parameters is chosen for each category, sometimes varying even the number of *flow steps* and the image resolution, while for our method, the architecture is always kept the same.

The results obtained for the anomaly maps assessment in terms of $AUROC$ and $AUPRO$ are shown in Table 5.1 and Table 5.2, respectively. U-Flow achieves state-of-the-art results, outperforming all previous methods on average for $AUROC$. Considering the $AUPRO$ metric, our results rank first among the flow-based methods and second overall, achieving the best results for several categories. It is worth emphasizing that FastFlow and CFlow results are reported using different hyperparameters that even change the architecture, and this is somehow unfair in their favor. Our results are always superior to CS-Flow, which also uses a multi-scale strategy inside the NF.

In addition, besides obtaining excellent results for $AUROC$ and $AUPRO$ in the anomaly localization task, our method presents another significant advantage with respect to all others: it produces a fully unsupervised segmentation of the anomalies and significantly outperforms its competitors, as shown in the next section.

For visual assessment, we include Figure 5.5, which presents typical example results of anomalous images from various categories and their comparison with other methods, and Figure 5.6, which shows typical examples of normal images in order to check that the anomaly maps present low values and that the detection method does not produce false positives.

Category	Patch SVDD	SPADE	PaDiM	Cut Paste	Patch Core	PEFM	Fast Flow	CFlow	CS-Flow	U-Flow (ours)
Carpet	92.60	97.50	99.10	98.30	98.90	99.00	99.40	99.25	95.23	**99.42**
Grid	96.20	93.70	97.30	97.50	**98.70**	98.48	98.30	**98.99**	94.46	98.49
Leather	97.40	97.60	98.90	99.50	99.30	99.24	99.50	**99.66**	89.32	**99.59**
Tile	91.40	87.40	94.10	90.50	95.60	95.19	96.30	**98.01**	90.71	**97.54**
Wood	90.80	88.50	94.90	95.50	95.00	95.27	**97.00**	96.65	86.29	**97.49**
Av. texture	93.68	92.94	96.86	96.26	97.50	97.44	**98.10**	**98.51**	91.20	**98.51**
Bottle	98.10	98.40	98.30	97.60	98.60	98.11	97.70	**98.98**	88.52	**98.65**
Cable	96.80	97.20	96.70	90.00	98.40	96.58	98.40	97.64	96.08	**98.61**
Capsule	95.80	99.00	98.50	97.40	98.80	97.94	**99.10**	98.98	98.13	**99.02**
Hazelnut	97.50	**99.10**	98.20	97.30	98.70	98.78	**99.10**	98.89	96.13	**99.30**
Metal nut	98.00	98.10	97.20	93.10	98.40	96.89	98.50	**98.56**	95.98	**98.82**
Pill	95.10	96.50	95.70	95.70	97.10	96.67	**99.20**	98.95	95.83	**99.35**
Screw	95.70	98.90	98.50	96.70	**99.40**	98.93	**99.40**	98.86	98.28	**99.49**
Toothbrush	98.10	97.90	98.80	98.10	98.70	98.28	**98.90**	**98.93**	97.39	98.79
Transistor	97.00	94.10	97.50	93.00	96.30	96.58	97.30	**97.99**	96.34	**97.87**
Zipper	95.10	96.50	98.50	**99.30**	98.80	98.29	98.70	**99.08**	95.82	98.60
Av. objects	96.72	97.57	97.79	95.82	98.32	97.71	98.63	**98.69**	95.85	**98.85**
Av. total	95.71	96.03	97.48	95.97	98.05	97.62	98.45	**98.63**	94.30	**98.74**

Table 5.1: MVTec-AD results: pixel-level *AUROC*. The two best results for each row are in bold. Comparison with the best performing methods: Patch SVDD [172], SPADE [35], PaDiM [44], Cut Paste [97], Patch Core [121], PEFM [152], Fast Flow [173], CFlow [66], and CS-Flow [123]. Our method outperforms all previous methods, with an average value of 98.74%.

Category	SPADE	PaDiM	PEFM	FastFlow	CFlow	CS-Flow	U-Flow (ours)
Carpet	94.70	96.20	96.75	97.30	**97.70**	83.55	**98.49**
Grid	86.70	94.60	**97.21**	94.79	**96.08**	78.83	95.46
Leather	97.20	97.80	98.91	**99.16**	**99.35**	81.05	98.40
Tile	75.90	86.00	91.10	86.90	**94.34**	72.97	**93.01**
Wood	87.40	91.10	95.77	94.14	**95.79**	60.49	**95.80**
Av. Texture	88.38	93.14	95.95	94.46	**96.65**	75.38	**96.23**
Bottle	95.50	94.80	**95.92**	91.55	**96.80**	64.14	95.64
Cable	90.90	88.80	**97.73**	**94.38**	93.53	84.07	93.73
Capsule	**93.70**	93.50	92.11	91.22	93.40	89.30	**95.02**
Hazle Nut	95.40	92.60	**97.99**	97.31	96.68	82.77	**97.35**
Metal Nut	**94.40**	85.60	93.88	91.05	91.65	76.93	**94.32**
Pill	94.60	92.70	96.18	**96.80**	95.39	75.61	**97.18**
Screw	**96.00**	94.40	95.73	67.00	95.30	92.21	**97.33**
Toothbrush	93.50	93.10	**96.21**	92.22	**95.06**	78.48	88.80
Transistor	87.40	84.50	90.84	91.76	81.40	**91.87**	**91.88**
Zipper	92.60	95.90	**96.45**	95.19	**96.60**	85.84	95.47
Av. Objects	93.40	91.59	**95.30**	91.05	93.58	82.12	**94.67**
Av. Total	91.73	92.11	**95.52**	92.18	94.60	79.87	**95.19**

Table 5.2: MVTec-AD results: pixel-level *AUPRO*. The two best results for each category are shown in bold. Comparison with the best-performing method available: SPADE [35], PaDiM [44], PEFM [152], and the best flow-based methods: FastFlow [173], CFlow [66], and CS-Flow [123]. Patch Core [121] is excluded from the table as it only reports a total average of 93.50%. Our method also achieves state-of-the-art performance for this metric, even obtaining the best result for some categories.

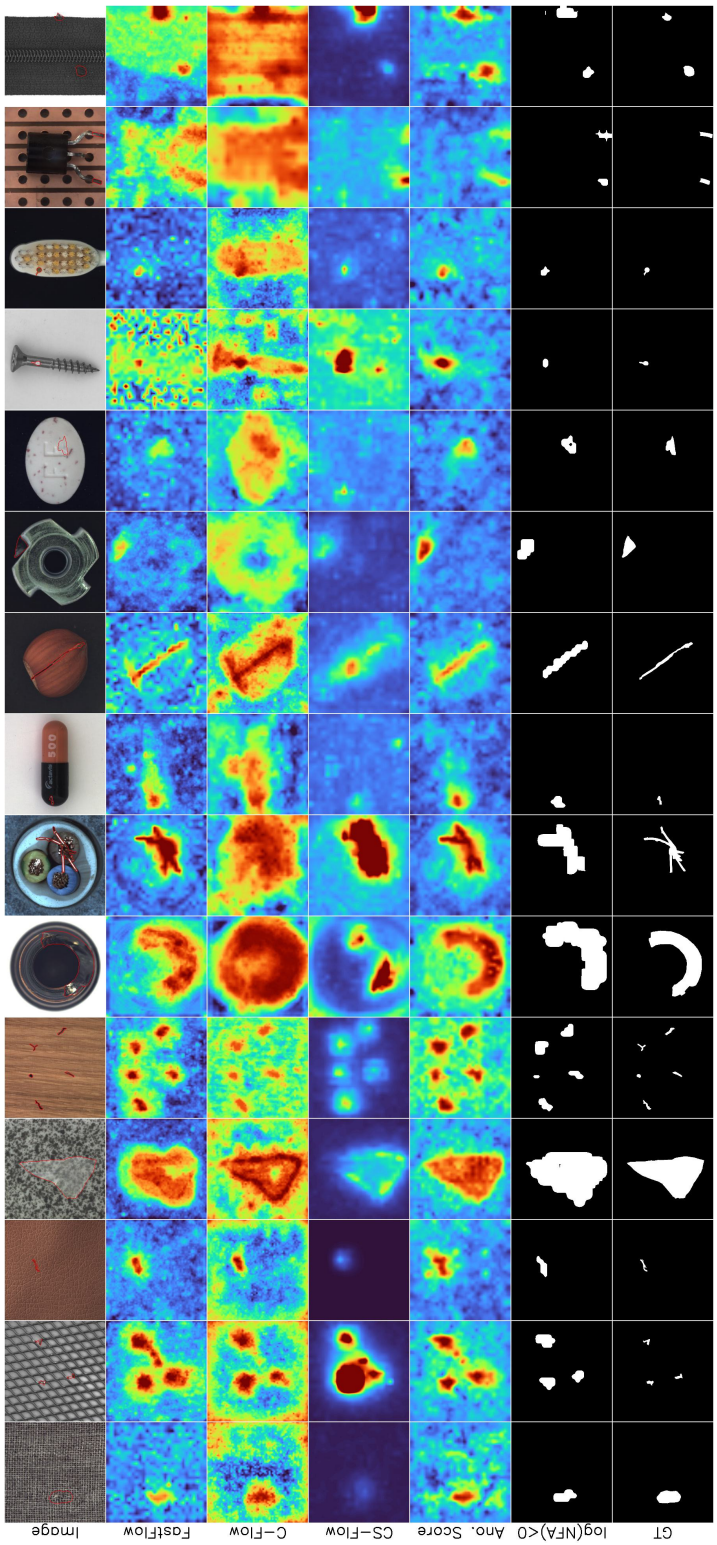


Figure 5.5: Example results for all MVTec categories. The first row shows the example images with the ground truth over-imposed in red. The results for FastFlow, CFlow, and CS-Flow are shown in the second, third, and fourth rows. The next two rows correspond to our method: the anomaly score defined in (5.2), and the segmentation obtained with the automatic threshold $\log(\text{NFA}) < 0$. The last row presents the segmentation masks for an easy comparison. While other methods achieve a very good performance, in some cases, they present artifacts and over-estimated anomaly scores. Our anomaly score achieves very good visual and numerical results, spotting anomalies with high confidence. Finally, the segmentation with the automatic threshold on the NFA is also able to spot and accurately segment the anomalies. All detections of these examples exhibit very low $\log(\text{NFA})$ values, ranging from -50 to -1515.

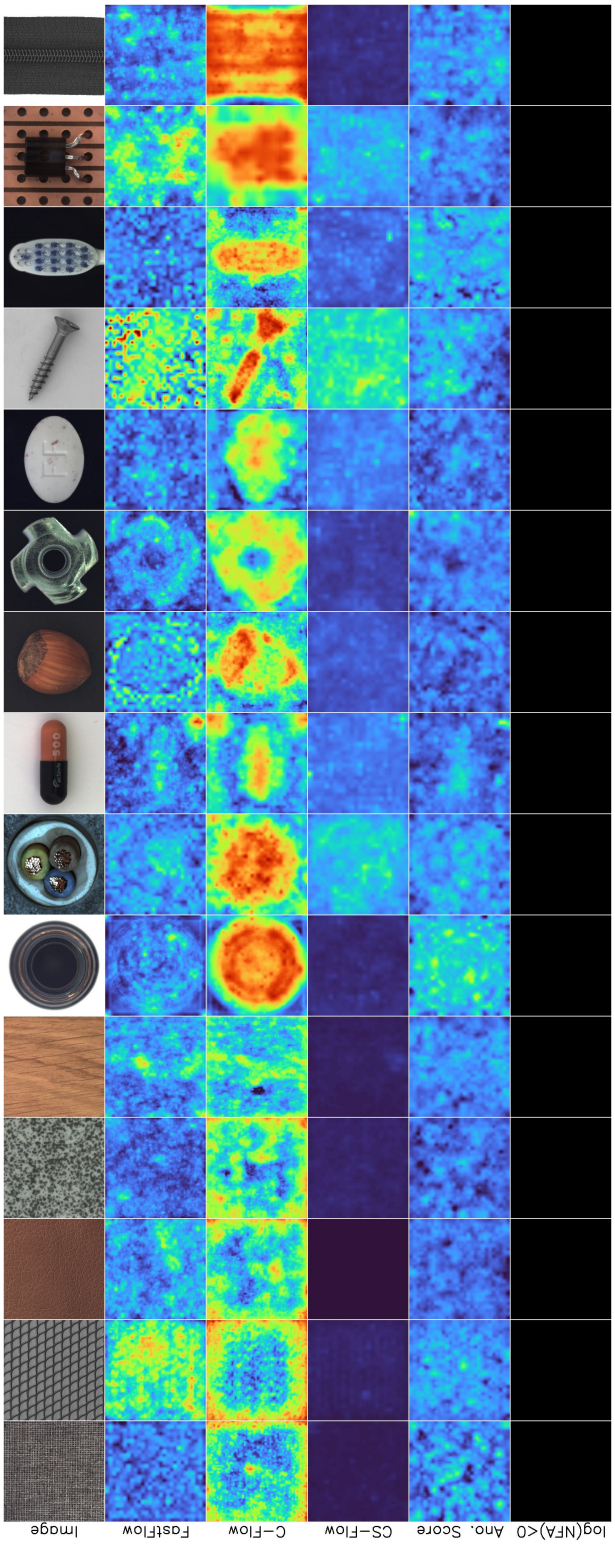


Figure 5.6: Normal image examples for all MVTec-AD categories. As can be seen, we always predict low values in the anomaly maps, and no detections are made.

Segmentation results

Providing an operation point is crucial in almost any industrial application. Most recent deep learning industrial anomaly detection methods in the literature focus on generating anomaly maps and evaluating them using the $AUROC$ metric; they do not provide detection thresholds or anomaly segmentation masks. Some methods do provide segmentation results, but these are not consistent with a realistic scenario: they are obtained using an *oracle*-like threshold that is computed by maximizing some metric over the test images [66, 121, 147].

In this section, we report results concerning anomaly segmentation based on the standard unsupervised threshold for meaningful events: NFA ≤ 1 ($\log(\text{NFA}) \leq 0$). As detailed in Section 5.3.4, this threshold implies that, theoretically, we allow one false anomaly detection per image on average. Moreover, this threshold can be tuned for any application-specific needs. For instance, in an industrial environment with the goal of detecting sample defects, an average value of one false detection per image may not be a good choice, as it would raise too many false alarms in normal images (one per image, on average). Instead, in this case, the threshold could be customized to anticipate, for example, a designated frequency of false alarms within a defined timeframe. However, as we will show next, the proposed method is inherently very robust to the threshold choice, and a much more restrictive threshold on the false positives can be considered without affecting the true positives' rate.

For evaluating the anomaly segmentation performance, we propose to use the $mIoU$ metric. Compared to IoU, using the average of all images ($mIoU$) prevents large anomaly regions from having more weight in the overall metric than small ones, and therefore, this metric better reflects how good the methods are in detecting all anomalies. The results for the $mIoU$ metric are shown in Table 5.3. We limit this comparison to the flow-based methods. As the state-of-the-art methods to which we compare do not provide detection thresholds, we adopt two strategies: *(i)* we compute an *oracle*-like threshold which maximizes the $mIoU$ for the testing set, and *(ii)* we use a *fair* strategy that only uses training data to find the threshold. In the latter, the threshold is set to allow on average one false positive in the training (anomaly-free) images, as it would be analogous to setting NFA ≤ 1 false alarm.

Visual examples of the obtained detections can also be found in Figure 5.5 for anomalous images, and in Figure 5.6 for normal images (which present no detections).

As seen in Table 5.3, our automatic thresholding strategy significantly outperforms all others, even when compared with their *oracle*-like threshold.

Category	Oracle threshold				Fair threshold			
	Fast Flow	**C-Flow**	**CS-Flow**	**Ours**	**Fast Flow**	**C-Flow**	**CS-Flow**	**Ours**
Carpet	0.5451	0.4519	0.3521	**0.5664**	0.4267	0.2393	0.3420	**0.5663**
Grid	**0.4664**	0.4077	0.3498	0.4316	0.2235	0.3413	0.3488	**0.4307**
Leather	**0.5483**	0.4818	0.2986	0.4242	**0.5241**	0.3096	0.2823	0.4183
Tile	0.5564	0.3705	0.4906	**0.6229**	0.4342	0.3105	0.2544	**0.6188**
Wood	0.4792	0.3767	0.3400	**0.5523**	0.4584	0.2433	0.0806	**0.5427**
Av. Texture	0.5191	0.4177	0.3662	**0.5195**	0.4134	0.2888	0.2616	**0.5154**
Bottle	0.5750	0.6503	0.4098	**0.6617**	0.4053	0.6434	0.3935	**0.6564**
Cable	0.5375	0.4009	0.5018	**0.6514**	0.4807	0.2299	0.3668	**0.6491**
Capsule	0.3439	0.3040	0.2739	**0.4182**	0.3160	0.3002	0.2669	**0.4124**
HazleNut	0.4723	0.5610	0.4654	**0.6546**	0.4671	0.4831	0.2747	**0.6511**
MetalNut	0.5801	0.3981	0.4631	**0.6307**	0.5801	0.2998	0.4165	**0.6267**
Pill	0.4616	0.3121	0.2324	**0.4815**	0.4051	0.3096	0.2260	**0.4803**
Screw	0.2562	0.3039	0.3137	**0.3811**	0.2375	0.2755	0.3049	**0.3726**
Toothbrush	0.3625	0.3347	0.3160	**0.4188**	0.3007	0.2416	0.3016	**0.4121**
Transistor	0.6337	0.6000	0.6622	**0.7381**	0.0597	0.2981	0.6516	**0.7354**
Zipper	**0.5772**	0.5001	0.4092	0.5460	0.4184	0.2403	0.4082	**0.5395**
Av. Object	0.4800	0.4365	0.4048	**0.5582**	0.3671	0.3322	0.3611	**0.5536**
Total	0.4930	0.4302	0.3919	**0.5453**	0.3825	0.3177	0.3279	**0.5408**

Table 5.3: Segmentation *mIoU* comparison for MVTec-AD, with the best flow-based methods in the literature: FastFlow [173], CFlow [66], and CS-Flow [123], for the oracle-like and fair thresholds defined in Section 5.4.1. Our method largely outperforms all others, and even exhibits a better performance comparing the proposed automatic threshold with their oracle-like threshold.

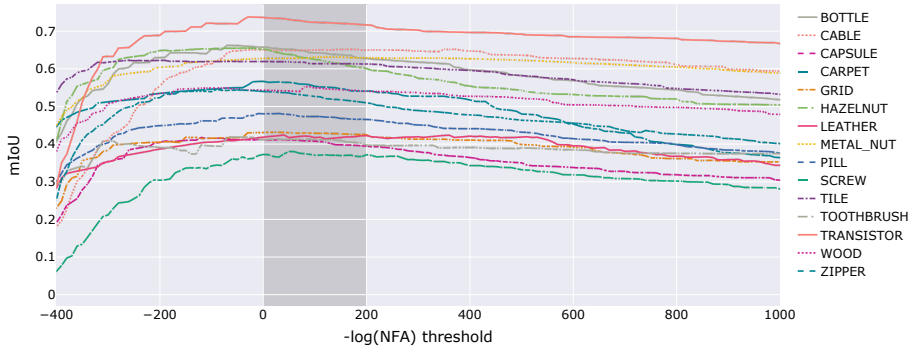


Figure 5.7: Segmentation results (*mIoU*) as a function of the $-\log(\text{NFA})$ threshold for all MVTec-AD categories. The automatic threshold $-\log(\text{NFA}) = 0$ is always very close to the optimal threshold for each category (which corresponds to the maximum of each curve). In addition, there is a wide range of thresholds, approximately indicated with the grayed-out region in the chart, for which the performance remains almost the same, indicating that this strategy is robust and, in practice parameter-free.

Additionally, it is important to note that our results obtained with the oracle-like and automatic thresholds are very close. This demonstrates the validity of the proposed statistical derivation and the inter-scale independence achieved by the proposed architecture. To further illustrate this point and to show the robustness of the anomaly detection method with respect to the threshold on the NFA, Figure 5.7 depicts the $mIoU$ for all the MVTec-AD categories, as a function of $-\log(\text{NFA})$ threshold, and Figure 5.8 shows a detailed view. These graphs show that the unsupervised threshold $\log(\text{NFA}) \leq 0$ is always near the optimal point (the oracle threshold), corresponding to the maximum $mIoU$, that is reported in Table 5.3 (noted as "Oracle threshold $\rightarrow$ Ours"). The fact that these thresholds are close is a confirmation that the theoretical densities are close to the sample densities. Not only are the oracle and the automatic thresholds close, but the variation in $mIoU$ is very low in a wide range of possible thresholds (all curves show a wide flat area), showing that this detection strategy is robust and parameter-free in practice.

As a verification, we additionally include a numerical evaluation of the proposed method's performance under different false alarm thresholds in Table 5.4. This table exhibits the $mIoU$ obtained scores across all MVTec categories, delineated across multiple rows corresponding to different NFA thresholds. Specifically, these thresholds encompass tolerances allowing an average of one false alarm per image, one false alarm per ten images, and progressively extending to one false alarm per one million images. As can be easily seen, particularly when considering the overall average displayed in the last column, the results demonstrate near parity across all thresholds. When allowing one false alarm per image on average (with the threshold set as $\log(NFA) \leq 0$), the resulting average $mIoU$ was found to be 0.541. Similarly, when permitting one false alarm per one million images on average (threshold $\log(NFA) \leq -6$), the obtained

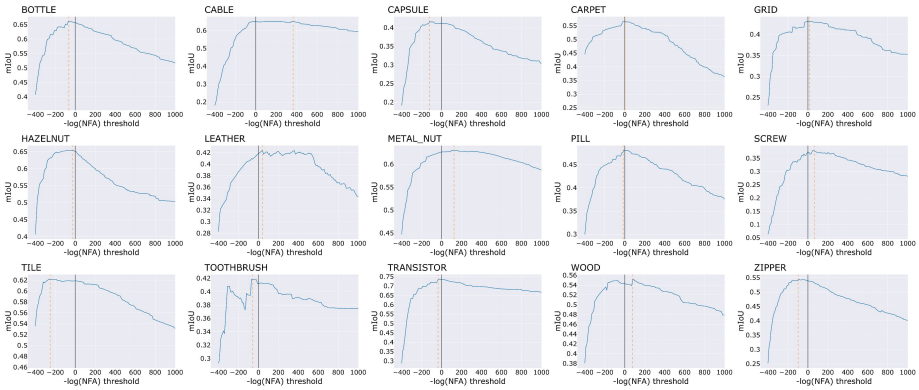


Figure 5.8: Segmentation performance (*mIoU*) as a function of the detection threshold over $-\log(\text{NFA})$, for all MVTec categories. The automatic threshold $\log(\text{NFA}) = 0$ is represented by the black solid line, and the "oracle" threshold with the dashed orange line. The latter corresponds to the best possible threshold to maximize *mIoU*. For all categories there is a wide range of thresholds with almost no variation in the performance metric, indicating that the method is robust in the selection of the threshold. Also, the "oracle" threshold lays almost always very close to the automatic threshold, supporting the theory behind the *a contrario* methodology. Note that for the cases where we obtained the most different thresholds (automatic vs. oracle), the obtained *mIoU* values are actually almost the same.

$mIoU$ was almost identical, at 0.540.

The robustness of our method to threshold variations can be attributed to its underlying approach. It utilizes a hierarchical structure based on the set of connected components derived from level sets, identifying maximal significant regions, defined as regions that potentially encompass lesser significant ones and are not encompassed by any higher significant region. Consequently, our method does not explicitly evaluate all potential regions. This characteristic contributes to computational efficiency and enhances the robustness of threshold selection. Each detected region encompasses a set of regions, and the ultimately identified regions are large deviations from the background model. Finally, we conducted detections on all anomaly-free test images to evaluate the empirical false alarm rate. Out of 467 images analyzed, 6 exhibited false alarms, of which 4 were identified as genuine yet unlabeled anomalies. Those images are presented in Figure 5.9. The remaining 2 instances were confirmed as true false alarms, attributable to significant deviations from the null hypothesis observed in the Normalizing Flow embedding. It is worth mentioning that these two false alarms are two sets of detections, as discussed before, suggesting that the actual average false alarm rate could be closer to the theoretical one.

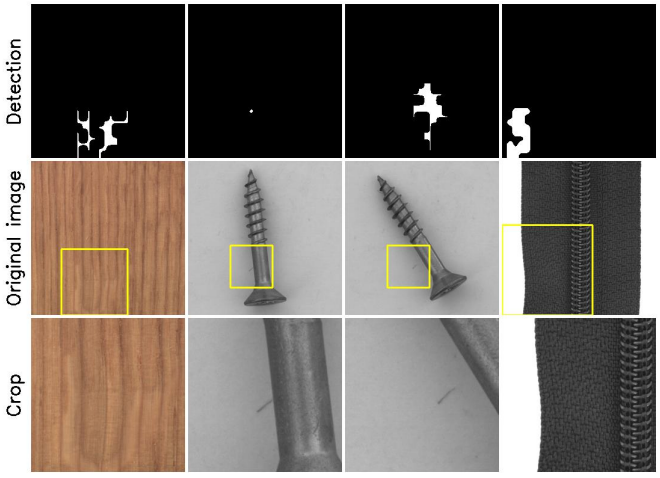


Figure 5.9: From all anomaly-free images in the test set of the MVTec dataset, the proposed detection method finds six false alarms, from which four actually correspond to real anomalies that are not labeled as such. This figure shows for these four images the detection using the automatic threshold ($\log(NFA) \leq 0$) in the first row, the original image in the middle row, and a crop containing the anomaly in the last row to ease the visualization.

NFA Thr.	Carp.	Grid	Leat.	Tile	Wood	Bott.	Cable	Caps.	HNut	MNut	Pill	Screw	Toot.	Tran.	Zipp.	**Avg.**
1	0.566	0.431	0.418	0.619	0.543	0.656	0.649	0.412	0.651	0.627	0.480	0.373	0.412	0.735	0.539	**0.541**
1/10	0.566	0.431	0.419	0.619	0.542	0.656	0.649	0.411	0.651	0.627	0.480	0.373	0.412	0.735	0.539	**0.541**
1/100	0.566	0.431	0.419	0.619	0.542	0.656	0.649	0.411	0.651	0.627	0.480	0.373	0.412	0.735	0.539	**0.541**
1/1000	0.566	0.431	0.419	0.619	0.542	0.656	0.649	0.410	0.651	0.627	0.480	0.373	0.412	0.735	0.539	**0.541**
1/10000	0.566	0.431	0.419	0.619	0.542	0.656	0.649	0.410	0.650	0.627	0.480	0.373	0.412	0.734	0.539	**0.540**
1/100000	0.566	0.431	0.419	0.619	0.542	0.656	0.649	0.410	0.650	0.627	0.480	0.373	0.412	0.734	0.539	**0.540**
1/1000000	0.566	0.431	0.419	0.619	0.542	0.656	0.649	0.410	0.650	0.627	0.480	0.372	0.412	0.734	0.539	**0.540**

Table 5.4: *mIoU* values for different NFA thresholds for all categories. Tested thresholds are $NFA \leq 1/10^i$, with $i \in [0, 6]$, or what is the same: $\log(NFA) \leq -i$. Theoretically, using the threshold $1/10^i$, we would obtain, on average one false alarm every 10^i images under the null hypothesis. For example, for the first threshold, we expect to have one false alarm per image, while in the last threshold ($1/10^6$), we expect one false alarm every one million images. It is evident that in practice nearly all tested threshold values result in the same *mIoU*s readings. By identifying the most significant region for each tree branch, the PFA tree introduced in Section 5.3.4 makes this threshold extremely robust.

5.4.2 Few-Shot learning

Industrial anomaly detection faces significant challenges in real-world scenarios where not only collecting defective samples is impractical, but gathering a large dataset of normal samples can also be challenging, particularly at the initial stages of new production lines or products. This scenario is especially common at project kickoffs, where production volumes are low or still being adjusted. Given these constraints, supervised approaches are unfeasible, driving the need for one-class anomaly detection methods that can perform effectively even with limited numbers of normal samples. Using a few-shot approach addresses this challenge, enables model deployment with minimal initial data, and supports an iterative improvement cycle: as more normal samples are collected and validated through the detection process, the model can be retrained to enhance its performance. This bootstrapping capability is particularly valuable in industrial settings, as it allows for a gradual refinement of the detection system while maintaining operational capabilities from day one. To validate our method's practicality and demonstrate its robustness under a few-shot setting, we conducted an extensive evaluation where the number of training images was progressively reduced. Table 5.5 and Figure 5.10 illustrate the $AUROC$ scores for each category in the MVTec AD dataset, comparing results when training with the full set of images against using only 50, 20, 10, 5, 2, and even 1 image per category. Additionally, in Table 5.5 we present the performance drop for each case, allowing for a detailed understanding of how the reduction in training data affects the detection capabilities.

Remarkably, the model demonstrates significant resilience, with only marginal performance degradation when fewer images are used. For instance, training with just 20 images results in a minor average performance drop of approximately 0.67%, indicating that the method can still generalize well with limited data. Even under more extreme conditions, such as training with a single image, the average performance drop is only 3.45%. This result is especially impressive, as few-shot learning typically struggles with large performance declines, particularly in highly varied industrial anomaly detection tasks.

In Figure 5.10, we can clearly observe a wide range in which performance remains stable despite significantly reducing the number of training images. Interestingly, even when training with just 5, 2, or 1 image(s), performance drops minimally for some categories, while the decline is more pronounced for others. Notably, the categories with the greatest performance drop are those characterized by lower self-similarity or those that require more domain-specific knowledge. This observation is quite intuitive and suggests that the method's robustness can be attributed to the multi-scale feature extraction combined with the Normalizing Flow architecture. This combination allows the model to effectively capture the underlying distribution of normal data, even from a limited number of samples, while maintaining invariance to transformations such as rotation and translation.

Training Images	Complete Set	50		20		10		5		2		1	
	AUROC	AUROC	Diff	AUROC	Diff	AUROC	Diff	AUROC	Diff	AUROC	Diff	AUROC	Diff
Carpet	99.42	99.41	0.01	99.40	0.02	99.31	0.11	99.31	0.11	99.31	0.11	99.30	0.12
Grid	98.49	98.34	0.15	98.24	0.25	98.18	0.31	97.89	0.60	96.76	1.73	95.90	2.59
Leather	99.59	99.50	0.09	99.57	0.02	99.57	0.02	99.53	0.06	99.34	0.25	99.24	0.35
Tile	97.54	96.74	0.80	95.94	1.60	96.54	1.00	94.15	3.39	95.45	2.09	96.44	1.10
Wood	97.49	97.24	0.25	96.76	0.73	96.58	0.91	95.96	1.53	95.76	1.73	95.72	1.77
Av. Texture	**98.51**	**98.25**	**0.26**	**97.98**	**0.52**	**98.04**	**0.47**	**97.37**	**1.14**	**97.32**	**1.18**	**97.32**	**1.19**
Bottle	98.65	98.60	0.05	98.57	0.08	98.57	0.08	98.63	0.02	98.68	-0.03	98.57	0.08
Cable	98.61	98.42	0.19	97.99	0.62	97.51	1.10	96.20	2.41	92.77	5.84	89.46	9.15
Capsule	99.02	98.75	0.27	98.81	0.21	98.89	0.13	98.78	0.24	98.40	0.62	97.14	1.88
Hazelnut	99.30	99.23	0.07	99.22	0.08	99.18	0.12	99.03	0.27	97.32	1.98	96.38	2.92
Metal Nut	98.82	98.57	0.25	98.39	0.43	97.84	0.98	98.19	0.63	97.06	1.76	96.10	2.72
Pill	99.35	99.23	0.12	99.25	0.10	99.08	0.27	98.50	0.85	97.00	2.35	95.91	3.44
Screw	99.49	98.92	0.57	97.69	1.80	90.66	8.83	90.62	8.87	90.46	9.03	90.51	8.98
Toothbrush	98.79	98.77	0.02	98.79	0.00	98.73	0.06	98.83	-0.04	98.38	0.41	97.95	0.84
Transistor	97.87	97.27	0.60	96.13	1.74	94.52	3.35	93.14	4.73	87.43	10.44	84.21	13.66
Zipper	98.60	98.55	0.05	96.26	2.34	96.27	2.33	96.31	2.29	96.34	2.26	96.44	2.16
Av. Objects	**98.85**	**98.63**	**0.22**	**98.11**	**0.74**	**97.13**	**1.73**	**96.82**	**2.03**	**95.38**	**3.47**	**94.27**	**4.58**
Av. Total	**98.74**	**98.50**	**0.23**	**98.07**	**0.67**	**97.43**	**1.31**	**97.00**	**1.73**	**96.03**	**2.70**	**95.28**	**3.45**

Table 5.5: Few-Shot Performance Results (*AUROC*). This table provides an analysis of how performance degrades when using fewer images to train the anomaly detection model. For each category, the *AUROC* scores are reported when training with varying numbers of images. The first column presents the results from training on the complete set of training images (as shown in Table 5.1). Subsequent columns show the results for models trained with 50, 20, 10, 5, 2, and even just one training image(s). The difference between the performance of each few-shot case and the complete set is also indicated. The table highlights the robustness of the model, as the performance remains nearly unchanged across a wide range of image counts. For example, when training with only 20 images, there is a minimal average performance drop of around 0.67%, and even when trained with a single image, the average performance decrease is only 3.45%, demonstrating the method's effectiveness in few-shot settings.

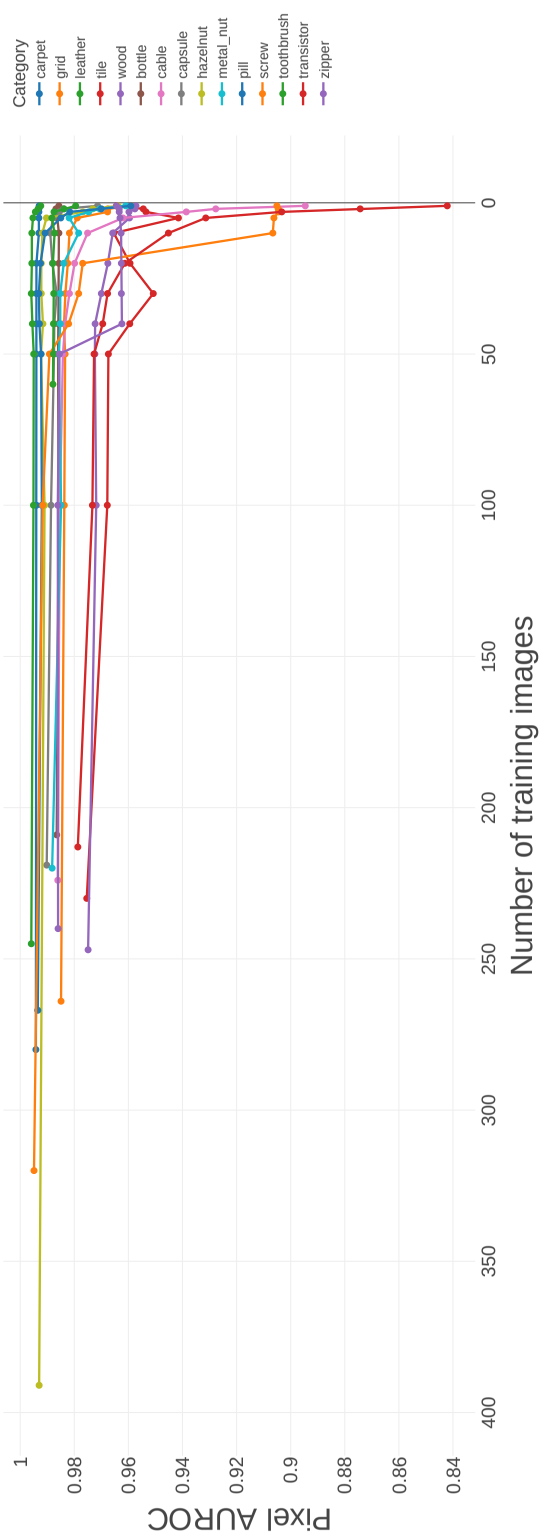


Figure 5.10: Few-shot learning performance of U-Flow on the MVTec AD dataset. This figure illustrates how U-Flow maintains high performance in all categories, even when the number of training images is drastically reduced. For many categories, the drop in performance is minimal. For instance, training with 20 images results in a drop of just 0.67%, or training just five images results in an average drop of only 1.73%, and even with only 1 image, the average performance drop is just 3.45%. While some categories, such as Screw and Transistor, experience larger performance drops due to their inherent complexity and lower self-similarity, others, like Bottle and Capsule, show almost no degradation, demonstrating U-Flow's robustness in few-shot scenarios and highlighting its ability to generalize effectively with limited data.

In conclusion, beyond the excellent results shown previously, we have demonstrated in this section that U-Flow not only achieves state-of-the-art performance but also exhibits remarkable robustness and stability in few-shot scenarios, where only a small number of samples are available for training. This capability significantly enhances its practical applicability, especially in industrial environments where acquiring large amounts of labeled data can be both difficult and costly.

5.4.3 Analysis of the Normalizing Flow embeddings

To complement the results, in this section, we propose to analyze visually the embeddings produced by the NF, by mapping them to a three-dimensional space using the *T-distributed Stochastic Neighbor Embedding* (T-SNE) algorithm. Results for various categories are shown in Figure 5.11. Each dot in this 3D space represents a different test image, and they are colored according to the type of defect. Green dots are always normal samples (i.e. anomaly-free images).

We could apply T-SNE directly to the entire volume z to generate this mapping. But instead, aiming to reduce the input dimensionality, we apply an intermediate step. Recalling that for each scale l, we have an embedding z_l of size $[C_l, H_l, W_l]$, we keep for each $k = \{1, \ldots, C_l\}$ the mean and standard deviation of the squared values of the volume, constructing a feature vector of dimension $2 * C_l$ for each image. The final feature vector, concatenating vectors from all scales, has size $2 \sum_l C_l$. These feature vectors are used as input for the T-SNE algorithm.

It is easy to observe that both normal samples and defect types reveal a clustering structure. Even though the method was aimed at separating abnormal samples from normal samples (green dots), it is interesting to note that this representation evidences that there is also a good separation between different types of defects, giving a hint that this technique could also be potentially applied for defects classification.

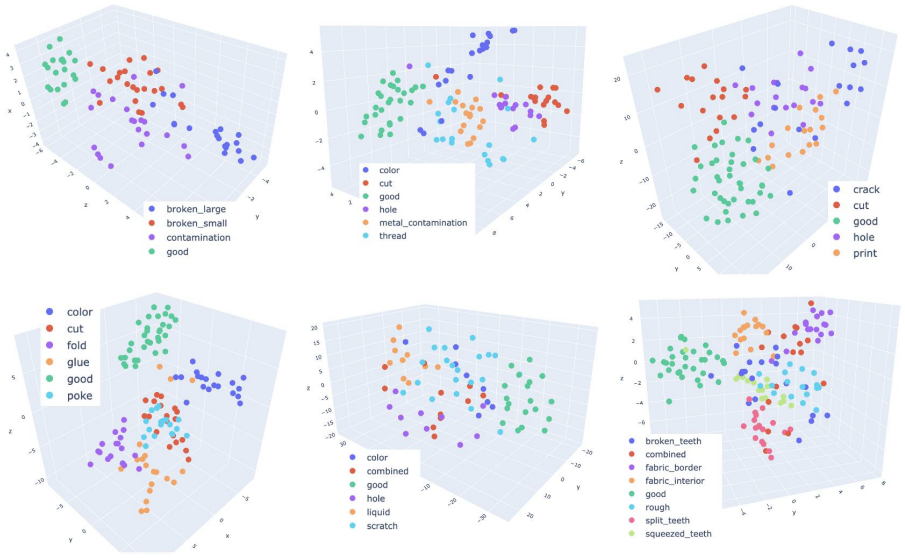


Figure 5.11: T-SNE embeddings. Top row: bottle, carpet and hazelnut. Bottom row: leather, wood and zipper. Normal samples are always in green, and other colors correspond to different defect types. It can be seen as a very good separation not only for the normal samples but also for each different type of defect.

		MRI	STC	BT01	BT02	BT03	BT AV.
AUROC	**FastFlow**	82.51	95.51	95.98	96.69	99.32	97.33
	CFlow	86.41	**96.90**	94.29	96.13	99.58	96.67
	CS-Flow	81.82	86.68	91.66	94.89	99.10	95.22
	U-Flow (ours)	**93.55**	87.19	**97.70**	**96.93**	**99.76**	**98.13**
AUPRO	**FastFlow**	47.48	73.67	76.23	62.39	96.60	78.41
	CFlow	59.47	**82.05**	60.19	54.72	98.32	71.08
	CS-Flow	37.30	54.04	73.75	47.18	94.45	71.79
	U-Flow (ours)	**71.74**	47.54	**80.06**	**66.49**	**98.83**	**81.79**
mIoU Oracle	**FastFlow**	0.2181	0.4840	0.4813	**0.2929**	0.9297	**0.5680**
	CFlow	0.2182	0.4831	0.4267	0.2782	0.9323	0.5457
	CS-Flow	0.2181	0.4895	0.3776	0.2439	0.9390	0.5202
	U-Flow (ours)	**0.2279**	**0.5052**	**0.4880**	0.2620	**0.9414**	0.5638
mIoU Fair	**FastFlow**	0.1947	0.4703	0.4408	**0.2890**	0.9129	0.5476
	CFlow	**0.2175**	0.4831	0.4227	0.2686	0.9149	0.5354
	CS-Flow	0.2060	0.4211	0.3522	0.1353	0.8738	0.4538
	U-Flow (ours)	0.1467	**0.4950**	**0.4667**	0.2448	**0.9330**	**0.5482**

Table 5.6: Results for LGG MRI (MRI) [21], ShanghaiTech Campus (STC) [99], and Bean-Teach (BT) [106] datasets. Comparison with best-performing flow-based models: FastFlow [173], CFlow [66], and CS-Flow [123]. Our method obtains the best results for almost all metrics and datasets, outperforming the other methods. Both *AUROC* and *AUPRO* refer to the pixel-level metric (localization task). For *mIoU*, we present the results using both the "Fair" and the "Oracle"-like thresholds.

5.4.4 Results on other datasets

In this section, we present the results obtained on the other datasets: Bean-Teach (BT), LGG MRI (MRI), and ShanghaiTech Campus (STC). Although our work is motivated by the industrial inspection task, we also evaluate the performance on datasets that are very different from that scenario. The results reported here demonstrate the robustness and generalization capability of the proposed approach (c.f. Table 5.6). For all datasets, we obtained excellent results, reaching top performance for almost all metrics and datasets. Again, all results were obtained by training the other methods with different hyperparameters to achieve the best possible results. In cases where some combinations of metrics and datasets were reported in the original articles, we confirmed that the same results had been obtained. Typical qualitative results are shown in Figure 5.12 for anomalous samples, and in Figure 5.13 for normal samples.

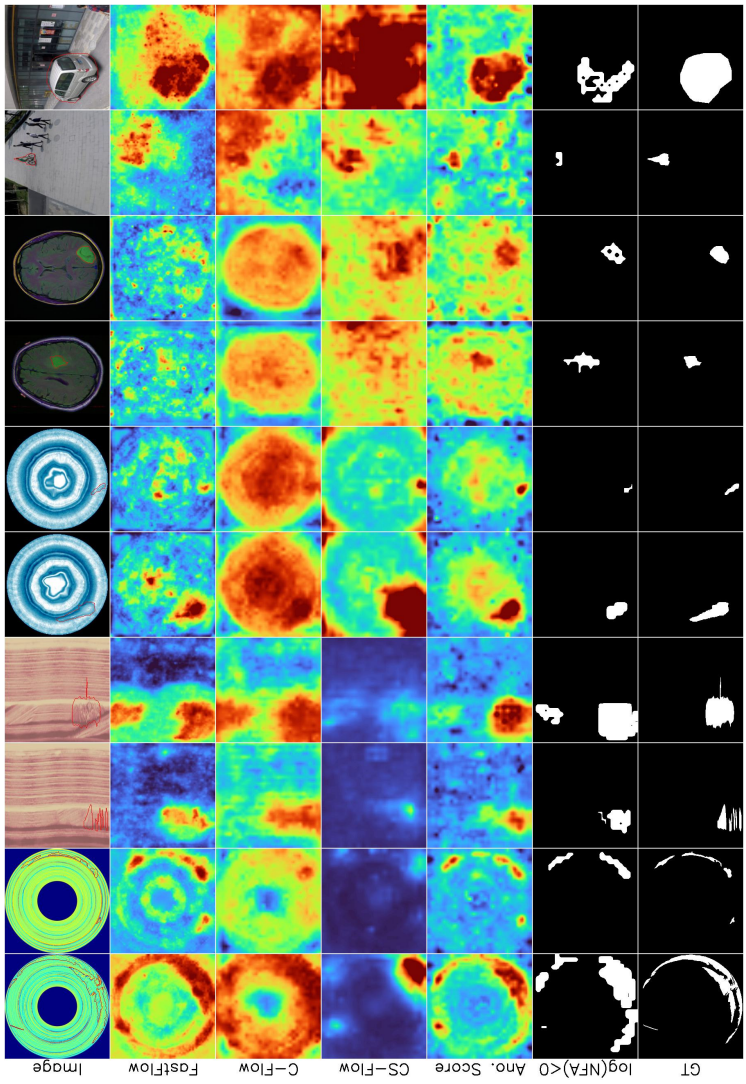


Figure 5.12: Examples of typical results on BeanTech, LGG MRI, and STC datasets. The first row shows the original images with over-imposed ground truth. The second, third, and fourth rows show the results of FastFlow, CFlow, and CS-Flow for comparison. The following two rows are the outputs of our method: the anomaly score defined in (5.2), and the anomaly segmentation with $\log(\text{NFA}) < 0$, and the last row is the ground truth. Again, our method achieves very good visual and numerical results and is able to detect anomalies with great confidence. All detections of these examples exhibit very low log(NFA) values, ranging from -56 to -1586.

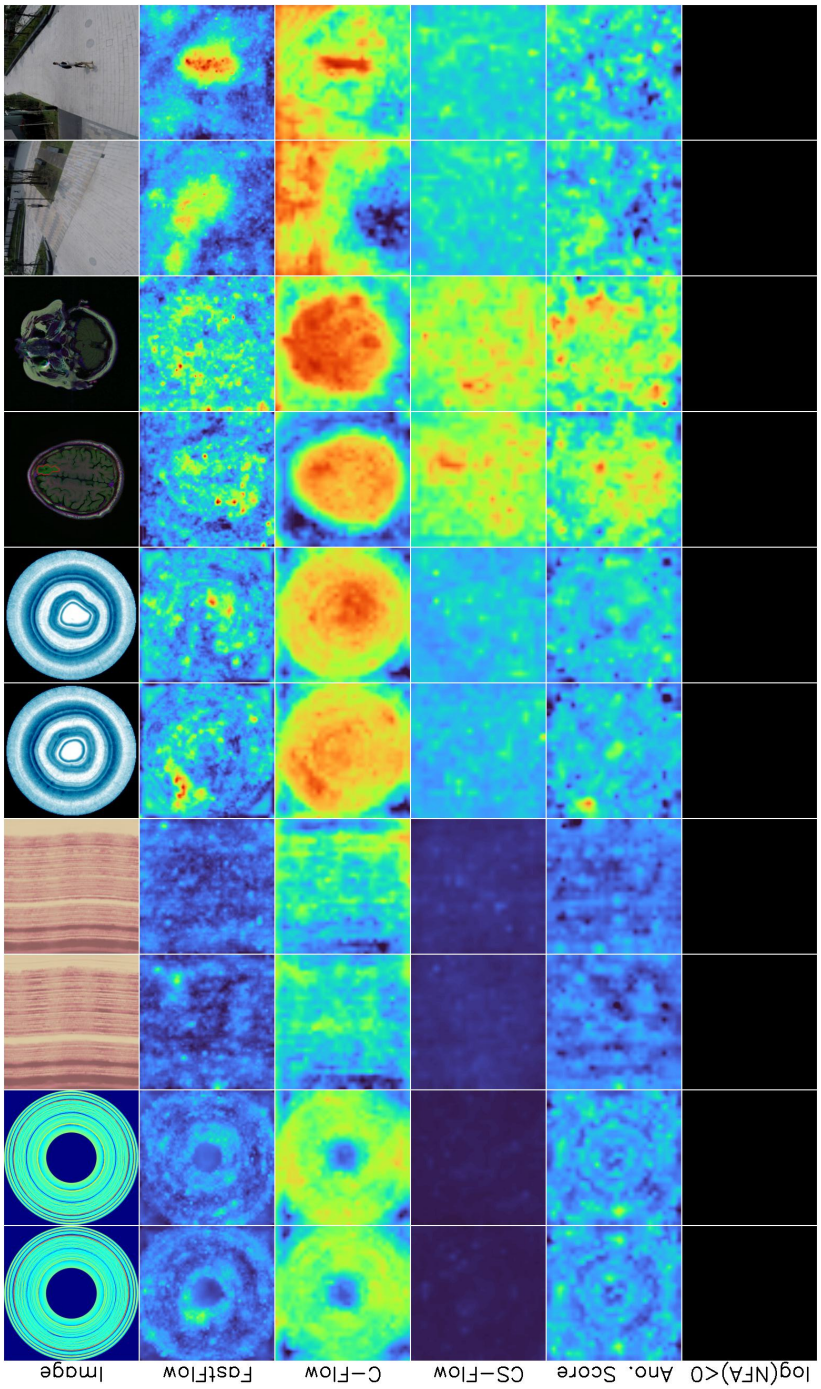


Figure 5.13: Normal images examples of BeanTech, LGG MRI, and STC.

5.4.5 Implementation details and complexity

The method was implemented in PyTorch [115], using PyTorch Lightning [55]. The NFs were implemented using the FrEIA Framework [6], and for all tested feature extractors, we used PyTorch Image Models [160]. In all cases, training was performed in a *GeForce RTX 2080 Ti*.

For the MS-CaIT, the input sizes are 448x448 and 224x224 pixels. The Normalizing Flow has 2 *flow stages* with 4 *flow steps* each.

As mentioned, our method only uses four *flow steps* in each scale. As a result, it has fewer trainable parameters than FastFlow and CFlow for the same feature extractors, as shown in Table 5.7.

Feature Extractor	**Fast Flow**	**CFlow**	**CS-Flow**	**U-Flow (ours)**
ResNet18	4.9 M	5.5 M	-	**4.3 M**
WideResnet50	41.3 M	81.6 M	-	**34.8 M**
EfficientNet	-	-	275 M	-
CaIT M48	14.8 M	10.5 M	-	**8.9 M**
MS-CaiT	-	-	-	**12.2 M**

Table 5.7: Complexity analysis: comparison of the number of trainable parameters for different feature extractors and methods.

Although all the results reported in this section were obtained using MS-CaiT as a feature extractor, we stress that the method and the code are developed agnostic to the feature extractor and can be easily changed according to the user's preference. In the following section, we present some results testing different feature extractors. In those cases, the number of scales and the volumes' sizes could vary depending on the chosen feature extractor.

5.5 Ablation Study

In this section, we study and provide unbiased assessments regarding some of the contributions of this work: the significance of the U-shaped architecture and the benefits of the multi-scale Transformer feature extractor MS-CaiT. Both results are shown all together in Table 5.8, and explained in the following sections.

5.5.1 Ablation: U-shape

One of the contributions of the presented work is proposing a multi-level integration mechanism by introducing the well-known U-shape architecture design to the Normalizing Flow's framework. To demonstrate that this architecture better integrates the information of the different scales, we compare the results obtained in terms of $AUROC$ against a modification of the architecture, in which each *flow stage* runs in parallel, and the per-scale anomaly maps are merged at the end by just averaging them, as done by other methods. Additionally, we compare the results obtained using each scale separately. The results, presented in the top half of Table 5.8, show that the U-merging strategy improves the

performance in almost all cases. Furthermore, this strategy, where the output of one scale inputs the next one, allows the use of less *flow steps* for each scale, resulting in a network with fewer parameters, as shown in Section 5.4.5.

5.5.2 Ablation: feature extraction with MS-CaiT

From the upper half of Table 5.8, it is also clear that utilizing image Transformers at various scales in MS-CaiT significantly improves the results, even if each of them already provides a multi-scale representation. Note that both scale-merging strategies outperform the results obtained by each individual Transformer.

In addition, in this section, we compare the results obtained by the proposed MS-CaIT feature extractor with the most common ResNet variants used in the literature: *ResNet-18* and *Wide-ResNet-50*, and the multi-scale Transformer MViT2. For all feature extractors, we extract features from all scales and proceed in the exact same way as presented before. Table 5.8's bottom half makes it clear that the MS-CaiT feature extractor performs far better than ResNet variants and MViT2. Note that, in favor of the ResNet extractors, for each category, we picked the best result obtained by both variants and also varied several hyper-parameters, for example, the amount of *flow steps* in each scale, while for MS-CaIT we always use the same exact architecture. Finally, note that our feature extractor always achieves better performance compared to MViT2, which is also a Transformer with a multi-scale hierarchy, probably getting some benefit from the training independence for each scale. The hyperparameter search for ResNet and MViT2 variants was performed with Optuna [3], using the TPE [156] sampler.

5.6 Image-level results

As mentioned in Section 5.4, this work focuses on the anomaly localization task. Nevertheless, we understand it is important to check that the proposed method also behaves well on the image-level detection task. Therefore, we present in Table 5.9 the results for the image-level $AUROC$ (detection task). To compute an image-level score, we simply used the maximum value of the anomaly score at the pixel level, defined in Section 5.3.3 (Eq. (5.2)). This simple strategy already achieves very good results: compared with all flow-based methods, our method ranks second, only behind FastFlow. However, it is worth noting that FastFlow actually uses different hyperparamenters for each category, even changing the architecture, which is an unfair test-set tuning.

For completeness, in Table 5.10 we also include the image-level results for all other tested datasets. Our method ranks first on average for LGG MRI and BeanTeach.

	Carp.	Grid	Leat.	Tile	Wood	Bott.	Cable	Caps.	HNut	MNut	Pill	Screw	Toot.	Tran.	Zipp.	Total
Scale 1	99.08	97.40	99.32	94.97	93.45	98.33	98.11	98.87	99.09	97.89	98.67	99.44	98.74	96.46	98.55	97.89
Scale 2	99.12	97.09	99.42	96.47	96.26	97.23	97.60	98.07	98.63	97.86	98.62	99.18	97.86	97.21	97.29	97.86
Avg.	**99.44**	98.25	99.52	97.27	96.40	98.61	98.50	98.85	99.16	98.29	99.12	**99.50**	98.78	97.66	**98.69**	98.54
Ours	99.42	**98.49**	**99.59**	**97.54**	**97.49**	**98.65**	**98.61**	**99.02**	**99.30**	**98.82**	**99.35**	99.49	**98.79**	97.87	98.60	**98.74**
ResNet	98.80	98.26	99.37	94.53	94.50	98.00	96.96	98.46	98.63	96.70	97.45	98.01	98.20	**98.38**	97.43	97.58
Arch.	wide	r18	r18	wide	r18	r18	wide	wide	wide	wide	wide	wide	r18	wide	wide	-
F. steps	6	6	4	8	6	4	4	4	4	4	4	6	6	4	4	-
MViT2	98.74	97.73	99.20	93.83	94.75	92.79	96.98	98.83	97.50	96.39	97.36	97.69	87.14	97.00	96.39	96.15

Table 5.8: Ablation results. The **middle row** (in-between horizontal lines), which shows the pixel-level *AUROC* results of our proposed method (U-Flow), serves for comparison for the top and bottom halves of the table. **Top half:** ablation study for the scale-merging strategy. Results using the anomaly scores generated by each scale independently, and a naive way of merging them (average). **Bottom half:** ablation study for the feature extractor. The proposed method (that uses MS-CaIT) is compared with ResNet and MViT2 feature extractors. For each category we show the *AUROC* of the best variant we could obtain, varying several hyper-parameters, such as the architecture (wide-resnet-50 or resnet-18) and the number of *flow steps*.

Category	P.SVDD	SPADE	Cut-Paste	Patch Core	PEFM	Fast Flow	CFlow	CS-Flow	U-Flow (ours)
Carpet	92.90	98.60	**100.0**	98.70	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Grid	94.60	99.00	96.20	98.20	96.57	99.70	97.60	99.00	**99.75**
Leather	90.90	99.50	95.40	**100.0**	**100.0**	**100.0**	97.70	**100.0**	**100.0**
Tile	97.80	89.80	**100.0**	98.70	99.49	**100.0**	98.70	**100.0**	**100.0**
Wood	96.50	95.80	99.10	99.20	99.19	**100.0**	99.60	**100.0**	99.91
Av. texture	94.54	96.54	98.14	98.96	99.05	**99.94**	98.72	99.80	99.93
Bottle	98.60	98.10	99.90	**100.0**	**100.0**	**100.0**	**100.0**	99.80	**100.0**
Cable	90.30	93.20	**100.0**	99.50	98.95	**100.0**	**100.0**	99.10	98.97
Capsule	76.70	98.60	98.60	98.10	91.90	**100.0**	99.30	97.10	99.56
Hazelnut	92.00	98.90	93.30	**100.0**	99.89	**100.0**	96.80	99.60	99.71
Metal nut	94.00	96.90	86.60	**100.0**	99.85	**100.0**	91.90	99.10	**100.0**
Pill	86.10	96.50	99.80	96.60	97.51	99.40	**99.90**	98.60	98.80
Screw	81.30	99.50	90.70	98.10	96.43	97.80	**99.70**	97.60	96.31
Toothbrush	**100.0**	98.90	97.50	**100.0**	96.38	94.40	95.20	91.90	91.39
Transistor	91.50	81.00	99.80	**100.0**	97.83	99.80	99.10	99.30	99.92
Zipper	97.90	98.80	**99.90**	98.80	98.03	99.50	98.50	99.70	98.74
Av. objects	90.84	96.04	96.61	**99.11**	97.68	99.09	98.04	98.18	98.34
Av. total	92.07	96.21	97.12	99.06	98.13	**99.37**	98.27	98.72	98.87

Table 5.9: MVTec-AD results: image-level *AUROC*. Comparison with best perfroming and flow-based methods: P.SVDD [172], SPADE [35], Cut-Paste [97], PatchCore [121], PEFM [152], FastFlow [173], CFlow [66], and CS-Flow [123]. Our method achieves state-of-the-art results also for the image-level metric (image-level *AUROC*). Among the flow-based methods, we rank second, only after FastFlow, which uses an unfair test-set tuning.

		MRI	STC	BT01	BT02	BT03	BT AV.
IMAGE	**FastFlow**	61.49	74.04	**100.0**	**89.13**	96.65	95.26
	CFlow	42.56	66.39	94.95	79.68	**99.96**	91.53
	CS-Flow	67.41	**98.46**	99.81	87.26	99.94	95.67
	U-Flow (ours)	**72.60**	64.59	99.42	88.72	99.72	**95.95**

Table 5.10: Image-level *AUROC* results for LGG MRI (MRI) [21], ShanghaiTech Campus (STC) [99], and BeanTeach (BT) [106] datasets. Comparison with best-performing flow-based models: FastFlow [173], CFlow [66], and CS-Flow [123].

5.7 Results on the VisA dataset

The VisA (Visual Anoamly) dataset, introduced in [180] and detailed in Appendix B, is a recent dataset, also designed for industrial anomaly detection. Therefore, it is also interesting to asses the performance on this dataset. Table 5.11 presents the preliminary results for the anomaly map generated by U-Flow, when training with default parameters. The performance of U-Flow on this dataset is also excellent, further confirming the robustness of the approach.

	AUROC	AUPRO
Candle	99.16	96.53
Capsules	99.49	89.76
Cashew	99.66	92.42
Chewinggum	99.34	90.81
Fryum	96.75	75.22
Macaroni 1	99.76	98.48
Macaroni 2	99.28	97.54
Pcb 1	99.78	95.33
Pcb 2	98.93	90.80
Pcb 3	98.98	88.52
Pcb 4	98.41	89.48
Pipe Fryum	99.64	93.87
Average	**99.10**	**91.56**

Table 5.11: U-Flow's anomaly map assessment for all categories in the VisA [180] dataset.

5.8 Supplementary experiments and qualitative results

In this section, we present a larger set of experimental results. We display two images for each category, with different types of defects. Texture examples are shown in Figure 5.14, and object examples are shown in Figures 5.15 and 5.16.

While visually inspecting the results, we found some cases where unlabeled anomalies were detected. A careful look reveals that these structures are actu-

ally different kinds of anomalies, and therefore it is actually correct to detect them. This kind of "false true detections" unfairly penalizes the method's performance.

Two such examples are shown in Figure 5.17. Figure 5.16a displays four images of the screw category, in which we can observe a subtle fluff in the background, almost unnoticeable to the naked eye, that correctly stands out in the anomaly score map. The two leftmost images correspond to images labeled as anomaly-free samples, while the two rightmost ones present labeled defects somewhere else that are also correctly detected. Similarly, Figure 5.16b shows four images of the toothbrush category that also present some defects in the background. As they are very difficult to see with the naked eye, we included a contrast-enhanced version of the image in the middle row for visualization purposes. Again, these defects are correctly detected, but they were not supposed to be there, and since they are not labeled, they penalize the results when computing the evaluation metrics.

5.9 Conclusion

In this work, we introduced a novel anomaly detection method that achieves state-of-the-art results on various datasets and even outperforms them in most cases. By using modern techniques with outstanding performance, such as Transformers and Normalizing Flows, we developed a method that exploits their characteristics to integrate them with classic statistical modeling. Our approach consists of four phases that follow one another, and we presented a clear and compelling contribution for each one: *(i)* We propose a new feature extractor using pre-trained Transformers to build a multi-scale representation. *(ii)* We integrate the U-shape architecture into the Normalizing Flow framework, creating a complete invertible architecture that settles the theoretical foundations for the NFA computation by ensuring independence in the embedding not only intra-scale but also inter-scales. *(iii)* We compute a likelihood-based anomaly score for each pixel with state-of-the-art performance in various datasets. And *(iv)* we propose a new anomaly detection method that permits the derivation of an unsupervised threshold based on the *a contrario* framework; this method exploits the statistical independence of the U-Flow embeddings to build a background model that produces excellent anomaly segmentation results.

The proposed approach was extensively evaluated and compared with the top-performing methods in the literature using different metrics, obtaining state-of-the-art results and, in most cases, even outperforming all previous methods. Finally, the method was also applied to several datasets from different domains with excellent results, demonstrating a strong generalization capability.

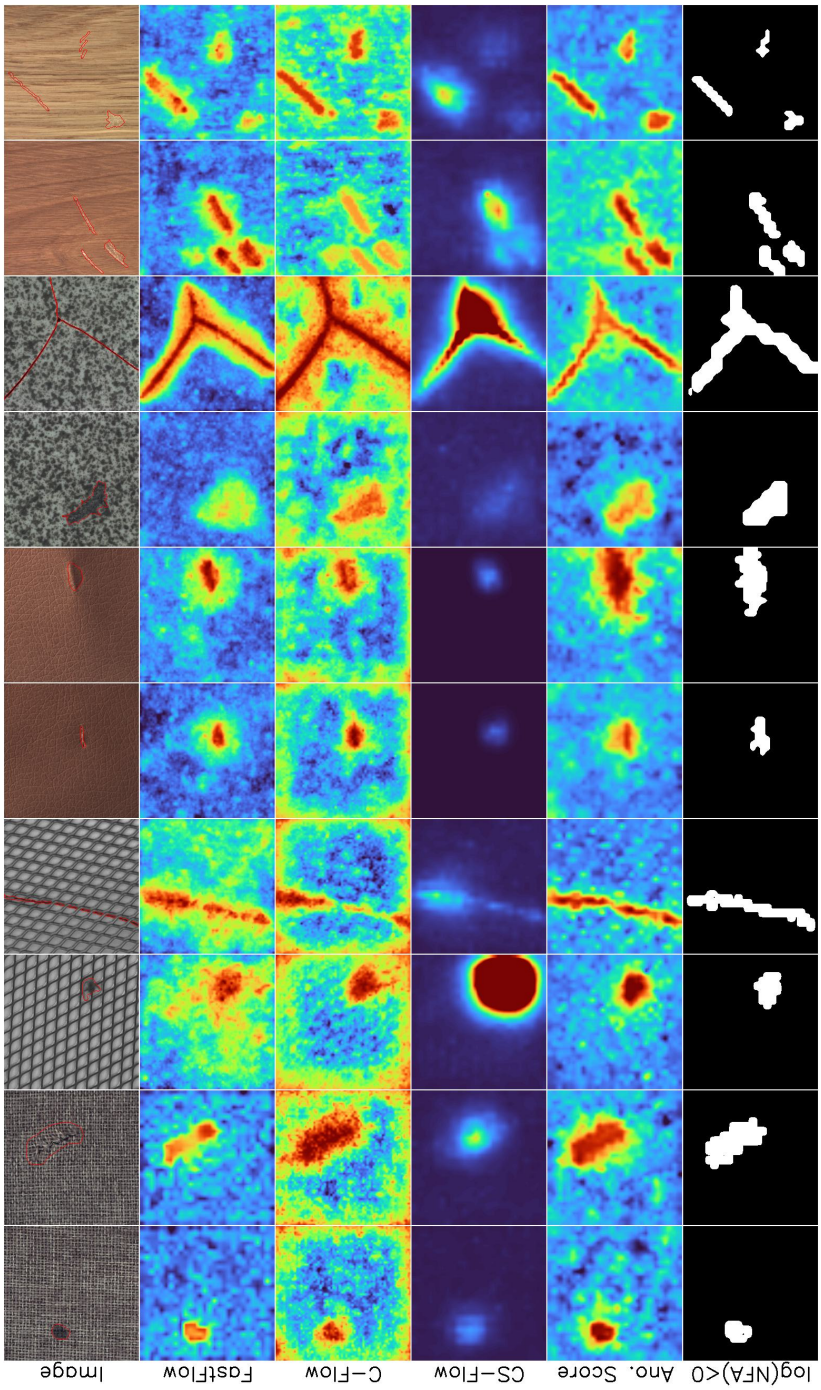


Figure 5.14: Textures: carpet, grid, leather, tile, and wood.

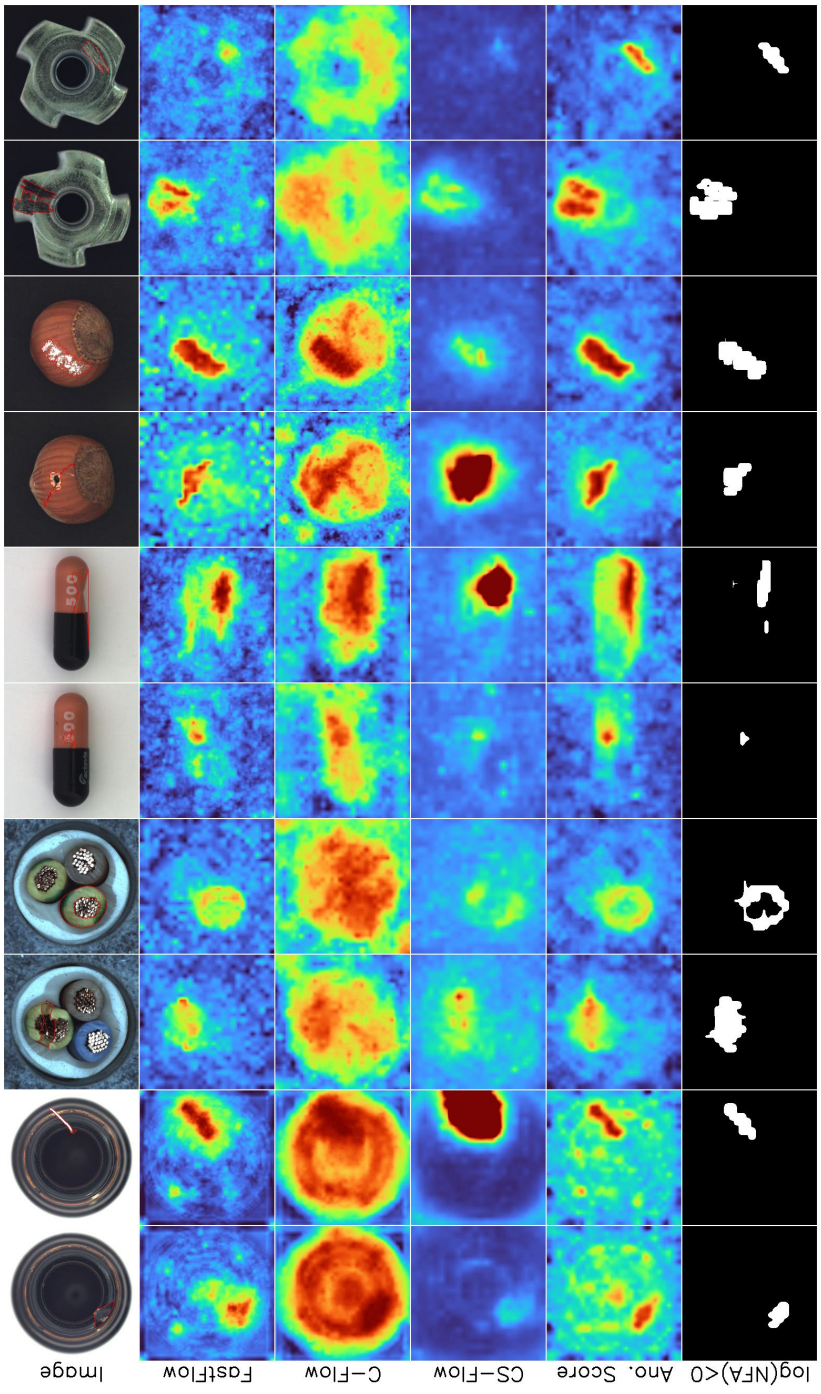


Figure 5.15: Objects 1: bottle, cable, capsule, hazelnut, and metal nut.

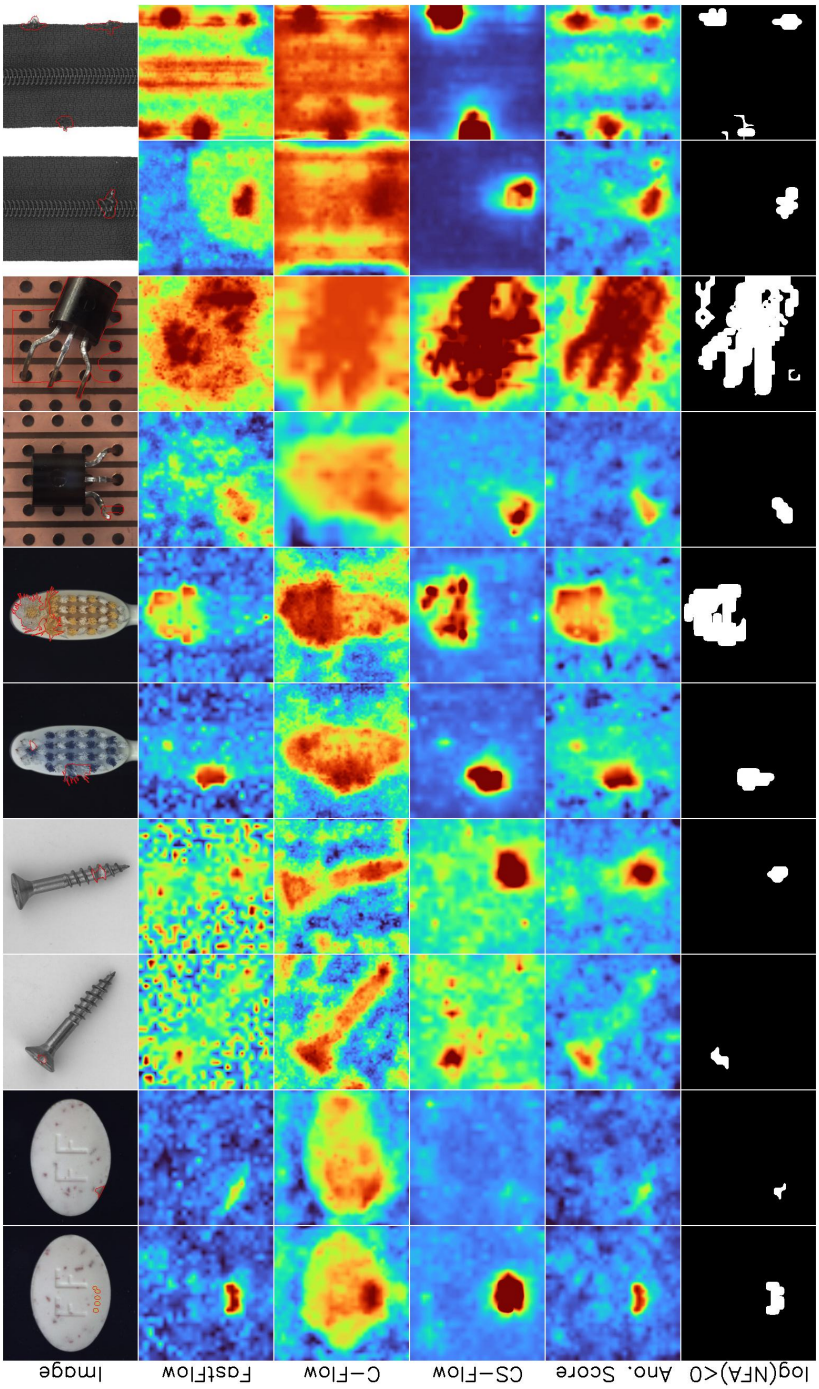
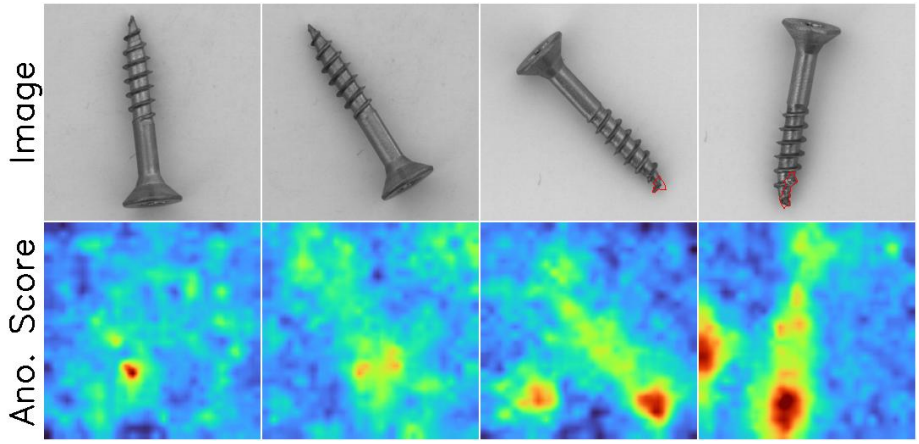


Figure 5.16: Objects 2: pill, screw, toothbrush, transistor, and zipper.

(a) Example of a subtle fluff in the background for the screw category



(b) Examples for the toothbrush category. Some defect in the background is being detected. We include the intermediate row with a contrast-enhanced version of the image, to better identify the defect.

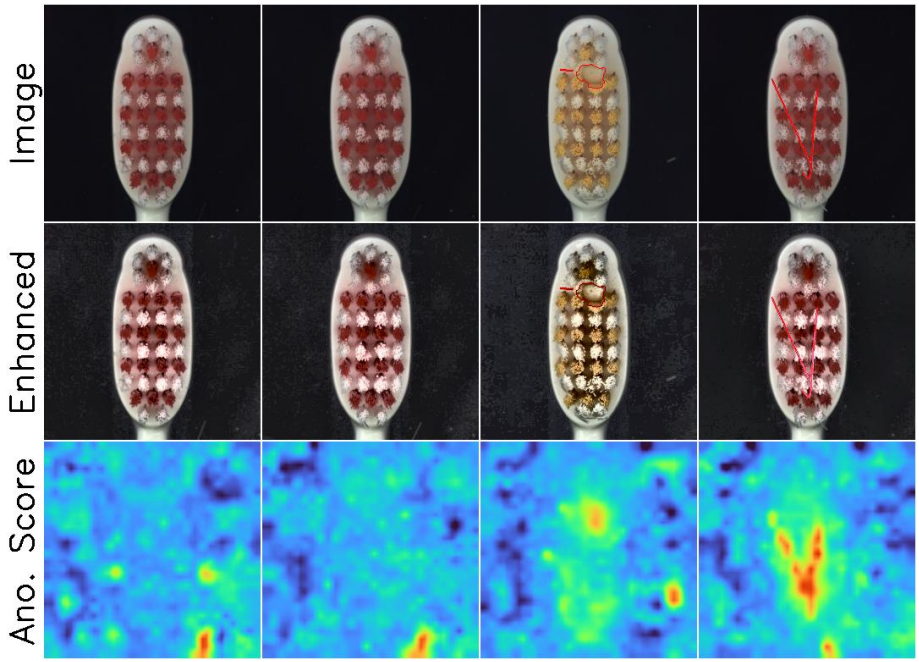


Figure 5.17: Examples of "false true" anomalies. Our method detects some anomalies that are not supposed to be there, and that are not labelled. For better visualization, we recommend to zoom-in.

Chapter 6

Diffusion Models

U-Flow (presented in Chapter 5) stands out as a remarkably effective method, obtaining state-of-the-art results in benchmark datasets and usually surpassing it. Moreover, it even demonstrates robustness in few-shot learning scenarios. Its multi-scale architecture integrates Normalizing Flows across various feature levels, which are extracted using pre-trained neural networks. This multi-scale approach allows U-Flow to capture important information at different levels of detail, which is crucial for detecting anomalies across diverse image types, from textures to objects. Despite these strengths, U-Flow faces challenges when dealing with very small defects. The reason is that while effective at capturing multi-scale patterns, the feature maps are small in spatial resolution, making it difficult to segment small anomalies with high precision.

To overcome this limitation, the next step is a shift toward pixel-level processing to achieve more granular and accurate anomaly segmentation. Given the success of generative models like Normalizing Flows in handling high-dimensional data, it was natural to explore another powerful generative modeling technique operating at a pixel level: Diffusion Models. These models have demonstrated impressive results in recent years, particularly in areas such as image synthesis and inpainting. While diffusion models are widely applied in creative tasks, their potential for likelihood estimation in anomaly detection remains relatively untapped. In this thesis, we aim to leverage the ability of diffusion models to compute likelihoods, which can be used for pixel-level anomaly detection. This novel approach provides the necessary precision to address the shortcomings in U-Flow's ability to accurately segment small defects, pushing the boundaries of generative modeling in anomaly detection tasks.

Using diffusion processes for modeling data generation can be linked to fundamental principles in statistical mechanics, where systems evolve over time according to stochastic differential equations (SDEs). These processes are characterized by their ability to model the gradual transformation of simple distributions into more complex ones, akin to the diffusion of particles in a medium.

The first significant application of these principles to generative modeling came with the work of Sohl-Dickstein et al. in their 2015 paper, "Deep Un-

supervised Learning using Nonequilibrium Thermodynamics" [135]. In this pioneering work, the authors proposed a framework where data is progressively corrupted by noise and then denoised through a learned reverse diffusion process. This approach demonstrated that it is possible to generate high-quality samples by reversing a diffusion-like process, marking the inception of modern Diffusion Models. Building on the foundational concepts, the field witnessed a substantial leap with the introduction of the Denoising Diffusion Probabilistic Models (DDPM) by Ho et al. in their 2020 paper, "Denoising Diffusion Probabilistic Models" [71]. This work refined the original ideas and introduced a practical and efficient training framework. One key innovation was the parameterization of the reverse diffusion process using deep neural networks, allowing for scalable and effective generative modeling. DDPMs demonstrated remarkable performance in generating high-fidelity images, rivaling and even surpassing contemporary generative models like GANs and VAEs in certain aspects. This breakthrough showcased the potential of Diffusion Models for a wide range of applications, from image synthesis to audio generation. Following the success of DDPMs, numerous variants and extensions have been proposed, each aiming to enhance the efficiency, scalability, and versatility of Diffusion Models. Notable contributions include:

- **Score-Based Generative Modeling** [138]: Introduced by Song and Ermon in 2019, this approach utilizes score matching to train generative models and has shown impressive results in generating diverse and high-quality samples.

- **Improved Sampling Techniques**: Researchers have explored various techniques to accelerate the sampling process in Diffusion Models, making them more practical for real-world applications.

- **Conditional Diffusion Models**: Extending the basic framework, conditional Diffusion Models have been developed to generate data conditioned on additional information, such as class labels or textual descriptions, enabling more controlled and versatile generation.

The advent of Diffusion Models has significantly impacted the field of generative modeling. Their ability to generate high-quality samples with a robust theoretical foundation has made them a valuable tool for researchers and practitioners.

In the following sections, we will delve deeper into the mechanics of Diffusion Models, exploring their mathematical formulation, training procedures, and practical implementations. Understanding these aspects will provide a comprehensive view of how Diffusion Models work and their potential for advancing generative modeling. In the following chapters, we will make use of all these concepts, applied to anomaly detection in Chapter 7, and counter-forensics tasks in Appendix A.

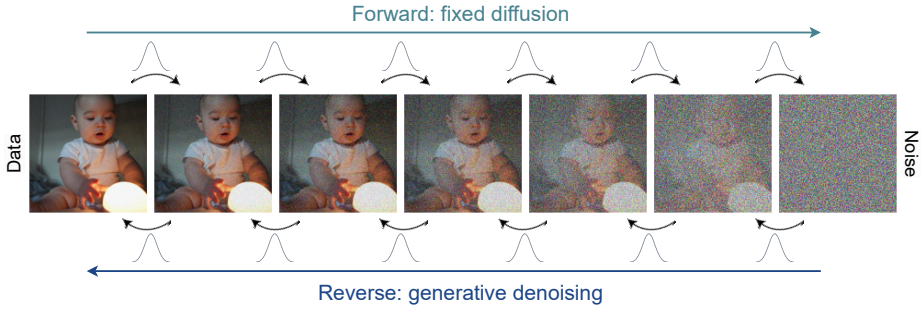


Figure 6.1: Diffusion diagram. The forward diffusion process starts with clean data and gradually adds a small amount of noise step by step until it ends with pure noise. On the other hand, the reverse generative process starts with pure noise, and during the training, the model learns to denoise the image step by step. In the end, the generative process generates a completely new image.

6.1 Introduction

In Diffusion Models, there are two main processes: the forward process and the reverse process. The forward process gradually transforms the original data into a simple noise distribution by iteratively adding small amounts of Gaussian noise. This process creates a sequence of progressively noisier versions of the data, eventually resulting in a distribution that resembles pure noise. This transformation is designed to be a Markov chain, where each step depends only on the previous one, and the noise addition is carefully controlled to ensure a smooth transition from the data to the noise distribution.

Conversely, the reverse process aims to recover the original data by denoising this sequence step-by-step. During the training phase, a neural network is trained to predict the noise added at each step of the forward process, effectively learning to reverse the diffusion. In the generative phase, the model starts with a sample from the noise distribution and iteratively applies the learned denoising steps to reconstruct the original data distribution. This reverse diffusion process allows the model to generate high-quality and complex data from simple initial noise, leveraging the learned structure and patterns of the original data.

Both processes are depicted in Figure 6.1.

6.2 Forward diffusion process

Given an image sampled from the real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, the forward diffusion process is defined by adding a small amount of Gaussian noise to the data point in T steps. This produces a sequence of noisy samples. The amount of noise added in each step is controlled by the variance schedule

$$\{\beta_t \in (0,1)\}_{t=1}^{T}\,, \tag{6.1}$$

which is fixed and predefined.

At every step, we assume we use a Normal distribution to generate a noisy image, conditioned on the image at the previous step. This Normal distribution is represented by q. It takes a sample at the previous step and generates the sample at the next step. Formally, it is defined as

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{1-\beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}\right). \tag{6.2}$$

Basically, at each step, we take the image from the previous step, rescale it, and add a small amount of noise. As we have this very simple expression for each step, we can also define the joint distribution of the whole chain, conditioned on the input sample. As each intermediate sample only depends on the sample in the previous step, the joint distribution has a very simple form, too:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)=\prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}). \tag{6.3}$$

But we actually do not need to go one step at a time in this Markov chain; we can directly jump from $\mathbf{x}_0$ to $\mathbf{x}_t$ for all time-steps t. Defining $\alpha_t = 1 - \beta_t$, and using the re-parametrization tick in the first step, we obtain [159]:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= \cdots \\
&= \sqrt{\prod_{s=1}^{T}\alpha_s}\ \mathbf{x}_0 + \sqrt{1-\prod_{s=1}^{T}\alpha_s}\ \boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \boldsymbol{\epsilon},
\end{aligned} \tag{6.4}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}\alpha_s$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In the operations above, we merged two Gaussians by summing their variances ($\bar{\boldsymbol{\epsilon}}_{t-2}$).

In summary, the distribution for the forward diffusion process for going from a clean data sample to its noisy version corresponding to time-step t is

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right). \tag{6.5}$$

And for sampling, we can use the re-parametrization trick again, as follows:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\boldsymbol{\epsilon}. \tag{6.6}$$

In the diffusion process, the noise schedule is designed so that $\bar{\alpha}_T \to 0$. In this way, we can assure that $q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \approx \mathcal{N}\left(\mathbf{x}_T ; \mathbf{0}, \mathbf{I}\right)$.

6.3 Reverse process

The reverse process aims to revert the noisy data back to the original data distribution. This process is also modeled as a Markov chain but in the reverse direction. It begins with a sample from the noise distribution and iteratively denoises it to generate a sample following the original data distribution. The goal of training a Diffusion Model is to learn these reverse transition probabilities such that the generated samples from the noise distribution are indistinguishable from the original data. The reverse process is crucial because it defines how effectively the model can transform noise back into meaningful data, making it an essential component of the generative capabilities of Diffusion Models.

The question is, how can we generate data from noise?

For computing the marginal probability distribution, we would need to integrate over all possible input images:

$$\underbrace{q\left(\mathbf{x}_t\right)}_{\substack{\text{Diffused}\\ \text{data dist.}}} = \int \underbrace{q\left(\mathbf{x}_0, \mathbf{x}_t\right)}_{\substack{\text{Joint}\\ \text{dist.}}} d\mathbf{x}_0 = \int \underbrace{q\left(\mathbf{x}_0\right)}_{\substack{\text{Input}\\ \text{data dist.}}} \underbrace{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)}_{\substack{\text{Diffusion}\\ \text{kernel}}} d\mathbf{x}_0; \quad (6.7)$$

and that is untractable. In order to generate data, we can start by sampling from the standard Gaussian distribution and then iteratively sample from the true denoising distribution $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$:

$$\begin{aligned} \mathbf{x}_T &\sim \mathcal{N}\left(\mathbf{x}_T; \mathbf{0}, \mathbf{I}\right), \\ \mathbf{x}_{t-1} &\sim q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right), \end{aligned} \quad (6.8)$$

but the problem is that we do not know the true denoising distribution. It can be mathematically proven that if β_t is small, the reversal of the diffusion process has an identical functional form as the forward process [59]. Our forward process is a Normal distribution, (6.2), and therefore, we can also approximate the reverse process using a Normal distribution.

In summary, the reverse denoising procedure starts from noise and learns to generate data by denoising. At every step, we approximate a Normal distribution to perform the denoising. Formally, the reverse process is represented as $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, where p_θ denotes the learned reverse transition probabilities parameterized by θ:

$$\begin{aligned} p(\mathbf{x}_T) &= \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) \\ p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}). \end{aligned} \quad (6.9)$$

The parameters of the reverse distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are estimated using a neural network, usually a U-Net [120], or a denoising auto-encoder. Given the noisy observation, this network is trying to estimate the most probable version of the denoised image. In other words, It estimates how does $\mathbf{x}_{t-1}$ looks like, in average, given $\mathbf{x}_t$. In the definition of (6.9), the network only estimates the mean μ_θ, but some works also include an estimation of the variance.

How can we parametrize the true denoising distribution?

In summary, we do not know the real denoising distribution $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$, but we know that it can be modeled as a Normal distribution. Unfortunately, we cannot easily estimate it, as we would need to integrate over all possible input data. But this density becomes tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}). \tag{6.10}$$

Moreover, we can approximate this distribution with $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, whose parameters are estimated using a neural network.

During the training, we encourage $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to resemble $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$, and for doing so we usually minimize the Kullback–Leibler (KL) divergence, which is a similarity measure between probability distributions. The fact that both are modeled as Gaussian distributions makes this computation really easy, as the KL divergence can be expressed in a closed analytical form. But, for doing so, we still need to parameterize $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$. Starting using the Bayes theorem, we know that

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) \frac{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)}, \tag{6.11}$$

and developing the expression of the Gaussian distribution, this yields

$$\begin{aligned} &q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \propto \\ &\quad \exp\left(-\frac{1}{2}\left(\frac{\left(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1}\right)^2}{\beta_t} + \frac{\left(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_{t-1}} - \frac{\left(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_t}\right)\right). \end{aligned} \tag{6.12}$$

Developing the squares and grouping in $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-1}^2$, leads to

$$\begin{aligned} &q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \propto \\ &\quad \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1} + C\right)\right), \end{aligned} \tag{6.13}$$

where C depends on $\mathbf{x}_t$ and $\mathbf{x}_0$, but does not depend explicitly on $\mathbf{x}_{t-1}$.

We know the expression in (6.13) follows a Gaussian distribution, so we can identify the variance as

$$\tilde{\beta}_t = 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = 1/(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t, \tag{6.14}$$

and the mean as

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) \\ &= (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\ &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0. \end{aligned} \tag{6.15}$$

Using (6.6),

$$\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t), \tag{6.16}$$

and plugging it to (6.15) gives

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t) \\ &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big).\end{aligned} \tag{6.17}$$

In summary, the true denoising distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is modeled as the Gaussian distribution given by

$$\begin{aligned}q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}) \\ \tilde{\boldsymbol{\mu}}_t &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big) \\ \tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t,\end{aligned} \tag{6.18}$$

and this distribution will be approximated with

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)), \tag{6.19}$$

using a neural network.

Side note: The first works directly tried to approximate $\tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0)$ with $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$. Later, it was empirically proven that estimating the noise itself, although exactly equivalent, helped to obtain better results in practice. Therefore most recent works focus on estimating $\boldsymbol{\epsilon}_t$ as $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$.

6.4 Loss

The objective is to maximize the likelihood of the data, according to the distribution p_θ; or equivalently, minimize the negative log-likelihood (NLL).

In this section, the derivation of the loss is presented following two different approaches: 1) By directly minimizing the NLL, as in VAEs, using the Evidence Lower BOund (ELBO), and 2) By minimizing the cross-entropy (CE) between q and p_θ.

Loss derivation: NLL approach

As the KL divergence is always non-negative by definition, we can relate the NLL with the divergence between the joint distributions $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)$ with a simple inequality. Using the definition of the KL divergence (D_{KL}) and

some operations (see [159]):

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\Big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big].
\end{aligned} \tag{6.20}$$

Finally, taking expectations w.r.t. $q(\mathbf{x}_0)$ on both sides and defining the right-hand side term as the variational lower bound loss yields

$$L_{\mathrm{VLB}} := \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0). \tag{6.21}$$

Loss derivation: CE approach

If we consider as learning objective loss the cross-entropy of $p_\theta(\mathbf{x}_0)$ relative to $q(\mathbf{x}_0)$, the same inequality (6.21) can be obtained using Jensen's inequality [159]:

$$\begin{aligned}
L_{\mathrm{CE}} &:= -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}\Big) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T}\Big) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big) \\
&= \mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big) \\
&\leq \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] = L_{\mathrm{VLB}}.
\end{aligned} \tag{6.22}$$

6.4.1 Computing the Variational Lower Bound

Operating on the expression of L_{VLB} (expressions (6.21) and (6.22)), this loss be can decomposed in three terms, as follows [71, 135, 159]:

$$
\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\cdot\frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \\
&\qquad\qquad + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big] \\
&= \mathbb{E}_q\Big[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} \\
&\qquad + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \\
&\qquad \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \\
&\Big].
\end{aligned}
\tag{6.23}
$$

The first term, L_T, is just the KL divergence between the diffusion kernel in the last step, and the base distribution. By definition, the diffusion kernel in T converges to the standard Normal distribution, which is the same distribution we chose for $\mathbf{x}_T$. Therefore, we can completely ignore this term for the training objective function. Note that actually, q has no trainable parameters here.

The last term, L_0, measures the likelihood of our input image given the noisy image at the first step under our denoising model, and is very small.

The second term is the KL divergence between $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, and it is the most important term. We know from Section 6.3, specifically in (6.18), that $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is modeled as a Gaussian distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) =$

$\mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$, with parameters

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_t &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big) \\ \tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t. \end{aligned} \tag{6.24}$$

And the denoising model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is also assumed to be Gaussian distribution with parameters $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ and σ_t (see (6.9)):

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}). \tag{6.25}$$

Therefore, this second term in the objective function is a KL divergence between two Gaussian distributions. And this can be computed analytically in closed form, reducing the whole computation to a simple L^2 norm between the means of both distributions ($\tilde{\boldsymbol{\mu}}_t$ and $\boldsymbol{\mu}_\theta$):

$$\begin{aligned} L_{t-1} &= D_{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) \\ &= \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] + C, \end{aligned} \tag{6.26}$$

where C is a constant.

In [71], the authors noted that we could also express these means in terms of $\mathbf{x}_t$ and the noise used to generate that data. Basically, if we have the noisy image and we want to find the cleaner image in the previous step, and we know the noise that was used to generate the sample, we can just subtract to $\mathbf{x}_t$ a scaled version of the noise, using (6.24). This, although mathematically equivalent, leads to better sample quality than directly estimating the means of the Normal distributions. Hence, what is actually done to train the denoising network is to estimate the noise $\boldsymbol{\epsilon}_\theta$, and with that, $\mu_\theta(\mathbf{x}_t, t)$ is obtained as follows:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right). \tag{6.27}$$

By doing so, L_{t-1} can be rewritten in terms of the noise, and using the re-parametrization trick, the following expression is obtained:

$$L_{t-1} = \mathbb{E}_{\substack{\mathbf{x}_0 \sim q(\mathbf{x}_0) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})}} \left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\alpha_t)}}_{\lambda_t} \left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta \left(\underbrace{\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t}_{\mathbf{x}_t}, t \right) \right\|^2 \right]. \tag{6.28}$$

Here, λ_t is a sort of weighting depending on time. For higher values of t, λ_t is small, and for small values of t, it is very high. This weighting actually ensures that the objective function is properly weighted for maximum data likelihood training and plays a very important role in data generation. There are other possible more advanced weightings schemes presented in [32]. However, in [71], the authors show that just setting $\lambda_t = 1$ leads to very high-quality samples,

resulting in the following simplified loss:

$$L_{t-1}^{\text{simple}} = \mathbb{E}_{\substack{\mathbf{x}_0 \sim q(\mathbf{x}_0) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbf{I}) \\ t \sim \mathrm{U}(1,T)}} \left[\left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta \left(\underbrace{\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t}_{\mathbf{x}_t}, t \right) \right\|^2 \right]. \tag{6.29}$$

6.5 Training and sampling

One might initially think the training process involves processing a single image and gradually adding noise, passing through the entire network sequentially. However, instead of this sequential processing, the procedure involves creating batches that contain different images, each with a different amount of noise added, according to different time steps. This approach is illustrated in Figure 6.2, and Algorithm 1.

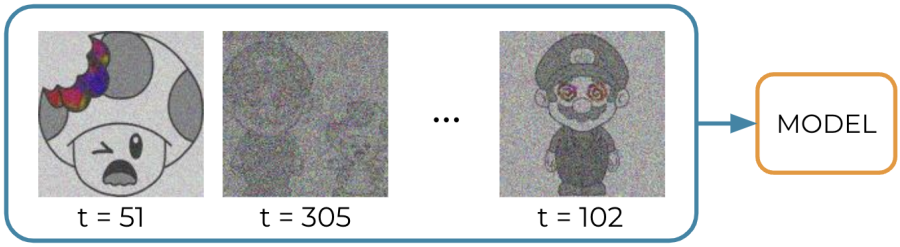


Figure 6.2: Example batch for training Diffusion Models. A batch of images is drawn from the dataset, and a different amount of noise is added to each according to randomly sampled time steps.

Algorithm 1 Training (from [71])

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on $\nabla_\theta \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2$
6: **until** converged

Actually, the training procedure is fairly straightforward: we begin by drawing a batch of samples from our dataset. Next, we randomly sample time steps from a uniform distribution ranging from 1 to T. We also sample noise from the standard Normal distribution. Using the re-parametrization trick, we generate the noisy samples at the chosen time steps and feed these samples into the denoising network, which predicts the added noise. The training is then performed using the L^2 norm of the difference between the added and predicted noises, as shown in Step 5 of Algorithm 1.

The sampling procedure, illustrated in Algorithm 2, is also very simple. It involves reversing the process used during training. Starting from the time

Algorithm 2 Sampling (from [71])

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

step T, we draw an initial sample from the standard Normal distribution. This sample is then iteratively passed through the denoising network in the reverse order, decrementing the time step at each iteration until reaching $t = 1$. At each time step t, the process includes two main steps:

- Compute the mean of the denoising model $\boldsymbol{\mu}_\theta$. It is derived by subtracting a scaled version of the predicted noise from the current sample $\mathbf{x}_t$. The scaling factor is determined by the time step t.
- Add noise to the adjusted sample, drawn from the standard Normal distribution, and appropriately scaled according to $t - 1$.

Both steps are performed together in Step 4 of Algorithm 2.

This iterative procedure effectively denoises the initial random sample, gradually transforming it into a sample from the learned data distribution. When the process reaches $t = 1$, the resulting sample resembles the data the model was trained on.

6.6 Implementation details

The network used in Diffusion Models to predict the noise added to the images at each time step during the forward process is usually a U-Net [120]. It is a type of convolutional neural network (CNN) originally designed for biomedical image segmentation. Still, it has since been adapted for various image processing tasks due to its efficiency and performance. It is composed of an encoder and a decoder. The encoder aims to capture the context of the input image by progressively downsampling the input, thus reducing its spatial dimensions while increasing the number of feature channels. The decoder reconstructs the spatial dimensions of the input image. Other key elements of the U-Net are the skip connections. At each level of the decoder, the feature maps are concatenated with the corresponding feature maps from the encoder. This helps the model retain high-resolution features and improves the quality of the output image.

The U-Net typically used in Diffusion Models retains the basic encoder-decoder structure with skip connections but includes additional mechanisms to incorporate the time step t as a conditioning variable. Specifically, the time step t is transformed into a higher-dimensional representation using a positional encoding scheme similar to those used in Transformer models. This encoding

helps represent the time step as a continuous variable the neural network can use. The encoded time-step is then concatenated with or added to the feature maps at various stages of the U-Net.

The inclusion of the time step in the network is of extreme importance. It provides context to the U-Net about where the current image is in the diffusion process, enabling it to dynamically adapt its processing to handle different amounts of noise, for example, learning to apply different filters, adjustments, and transformations based on the time-step. Additionally, it is very important to highlight the fact that having a single network with shared parameters between all time steps ensures consistency in the process. This is crucial, especially for the reverse process, where the model must progressively denoise the images in a coherent manner, starting from pure noise.

6.7 Acceleration techniques

Accelerating Diffusion Models is a critical area of research, as these models, while powerful, can be computationally intensive. Several techniques have been developed to speed up their training and sampling processes. The following is a list of some of the most important techniques, to mention a few, and it is not meant to be exhaustive.

Denoising Diffusion Implicit Models (DDIM) [137]: DDIMs improve the sampling speed of Diffusion Models by modifying the denoising process. Instead of requiring a large number of steps to gradually remove noise, DDIMs use a non-Markovian process that allows for fewer steps while still producing high-quality samples.

Latent Diffusion Models [119]: These models operate in a compressed latent space rather than in the original high-dimensional data space. By working in a lower-dimensional space, the computational cost is reduced significantly, which accelerates both training and sampling.

Score-Based Generative Models [80,138,140,177]: Also known as neural SDEs (stochastic differential equations), these models use a continuous-time approach to diffusion processes. By leveraging neural SDEs, it is possible to sample more efficiently, as they can be trained to follow a smooth trajectory in the data space. These models are introduced in more detail in the next chapter, as the final methods presented in this thesis make use of them and some unique advantages they have with respect to all other Diffusion Models so far.

This page was intentionally left blank.

Chapter 7

DAD: Diffusion Anomaly Detection

In the previous chapter, we delved into the foundational theory behind Diffusion Models, focusing on Denoising Diffusion Probabilistic Models (DDPM) [71]. We explored how DDPM operates in discrete time, progressively adding and removing noise to generate high-quality samples. In this chapter, we transition from discrete to continuous time models, introducing the theory behind score-based models [138, 140], and proposing a new method for anomaly detection using them.

Instead of adding noise with a finite number of noise distributions, continuous-time score-based Diffusion Models consider a continuum of distributions that evolve over time according to a diffusion process. This approach involves a smooth, continuous transformation where data points are progressively diffused into random noise. The evolution of these distributions is governed by a stochastic differential equation (SDE) that is predetermined and independent of the data, meaning it has no trainable parameters.

The fundamental concept of score-based models involves learning score functions, which are the gradients of log probability density functions. After learning these score functions, Langevin dynamics are used to generate new data samples. In fact, it can be shown that for continuous state spaces, the training objective of DDPM is implicitly computing the scores at each noise level. Therefore, DDPM can also be viewed as a type of score-based model.

One of the key advantages of score-based generative models is their capability of exact log-likelihood computation. So far, this is usually used to measure the model's performance, but in the context of this thesis, in this chapter we explore its utility for anomaly detection.

Additionally, score-based models can address inverse problems, such as image inpainting and super-resolution, without requiring model retraining, making them highly versatile and practical for numerous applications. This capability is also exploited for anomaly detection in this chapter.

7.1 Stochastic Differential Equation (SDE)

From the definition of the forward diffusion process in (6.2), and using the re-reparametrization trick, we know that

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(0,\mathbf{I}). \tag{7.1}$$

Then, assuming β_t is a discretization of a continuous function $\beta(t)$, and using the Taylor expansion when the step size $\Delta t \to 0$, we obtain[1]:

$$\begin{aligned}\mathbf{x}_t &= \sqrt{1-\beta(t)\Delta t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(0,\mathbf{I}) \\ &\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(0,\mathbf{I}).\end{aligned} \tag{7.2}$$

The forward process in the above equation has the form of an iterative update corresponding the following Stochastic Differential Equation (SDE):

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}_t, \tag{7.3}$$

where $\mathrm{d}\mathbf{w}_t$ represents the standard Wiener process (or Brownian motion). The first term of the above equation is usually called the **drift term** and pulls data toward the distribution mode, as the update direction is proportional to the negative of the state $\mathbf{x}_t$. The second term is the **diffusion term**, which injects noise.

It is worth mentioning that other differential equations are also being used to define other types of diffusion processes. Therefore, more generally, we can consider that the diffusion process is represented by the following SDE:

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)\,dt + g(t)\,d\mathbf{w}_t, \tag{7.4}$$

where $\mathbf{x}_t$ is the state of the system at time t, $f(\mathbf{x}_t, t)$ is the drift coefficient, and $g(t)$ is the diffusion coefficient.

For the forward diffusion process in (7.3), there exists another SDE that goes in the reverse direction and tracks the exact same probability distribution [4]:

$$\mathrm{d}\mathbf{x}_t = \underbrace{\left(-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\overbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}^{\text{Score function}}\right)}_{\text{Drift term}}\mathrm{d}t + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\mathbf{w}}_t}_{\text{Diffusion term}}, \tag{7.5}$$

where inside the drift term, we can identify the score function $\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)$. Knowing this score function allows us to generate data from noise. Therefore, the objective of score-based models is to learn the gradient of the log-probability density with respect to $\mathbf{x}_t$: $\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)$.

[1] Following the notation in [138]

7.2 Score matching network

In order to perform data synthesis, which involves generating new data, we require a method to compute the score function. The score function essentially captures the gradient of the log probability density with respect to the data. Since directly computing this score function is often infeasible, we adopt a common strategy from many other machine learning problems: we approximate it using a neural network.

Specifically, we aim to approximate the score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ by estimating the parameters $\boldsymbol{\theta}$ of a neural network $\mathbf{s}_{\boldsymbol{\theta}}$. The neural network $\mathbf{s}_{\boldsymbol{\theta}}$ is trained to produce an output that closely matches the true score function.

The primary approach to achieve this approximation is through direct regression. The objective is to minimize the difference between the neural network's output and the actual score function. This can be formally expressed by the following optimization problem:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)} \mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t)} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \right\|_2^2 . \tag{7.6}$$

In this expression, the expectation $\mathbb{E}_{t\sim\mathcal{U}(0,T)}$ represents averaging over the diffusion times t uniformly sampled from the interval $[0, T]$, and $\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t)}$ denotes averaging over the diffused data $\mathbf{x}_t$ sampled from the distribution $q_t(\mathbf{x}_t)$. The neural network $\mathbf{s}_{\boldsymbol{\theta}}$ is trained such that its output approximates the gradient of the log-density of the data, thus providing an estimate for the score function. However, (7.6) cannot be computed directly because the score of the marginal diffused density, $q_t(\mathbf{x}_t)$, is not tractable. Instead, we can leverage individual data points and focus on the conditional distribution $q_t(\mathbf{x}_t|\mathbf{x}_0)$. This approach leads to the denoising score-matching objective, which is formulated as:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)} \mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)} \mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0) \right\|_2^2 . \tag{7.7}$$

This method is advantageous because, despite focusing on the conditional distribution, the neural network still effectively learns to estimate the score of the marginal diffused data distribution. The neural network $\mathbf{s}_{\boldsymbol{\theta}}$ ultimately learns to estimate the score of the marginal diffused data distribution: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$.

Furthermore, this approach has a very interesting property that is worth mentioning: the **unique identifiability**. The learned denoising model that approximates the score function is uniquely determined by the data we use for training and the forward diffusion process. This has a crucial implication: for any given image, the encoding in the latent space is unique. Regardless of the architecture and initialization used, in the end, we obtain identical score function values and encodings in latent space, assuming we have sufficient training data, network capacity, and optimization accuracy.

7.3 Probability Flow ODE

Samplers based on Langevin Markov Chain Monte Carlo (MCMC) and stochastic differential equation (SDE) solvers present excellent sampling quality. How-

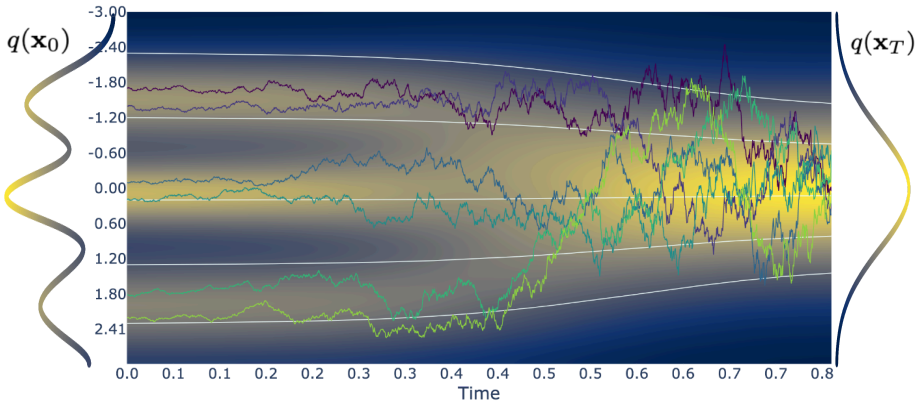


Figure 7.1: Diagram of score-based generative models through ODEs and SDEs. The background depicts the evolution of the density functions over time. The white lines represent the evolution following the Probability Flow ODE. The noisy colored lines are some examples of evolution according to the SDE. At the left-hand side of the chart, we have a representation of $q(\mathbf{x}_0)$, which was created as a mixture of three Gaussian distributions. At the right hand side of the chart is shown $q(\mathbf{x}_T)$, which is a standard Normal distribution.

ever, they lack the ability to compute the likelihood. To address this issue, this section presents a sampling technique based on ordinary differential equations (ODEs) that enables exact likelihood computation.

In [140], the authors demonstrate that it is possible to transform any SDE into an ODE without altering its marginal distributions. We can generate samples from the same distributions as the corresponding SDE by solving this ODE, known as the Probability Flow ODE. In the context of Diffusion Models, both the SDE and its corresponding ODE convert the same data distribution to the same prior distribution and vice versa, maintaining mathematically equivalent marginal distributions. In other words, solving the Probability Flow ODE yields trajectories with the same marginal distributions as those obtained from the SDE, but these new trajectories are deterministic, and enable for exact log-likelihood computation, as it is explained in the next section.

Figure 7.1 illustrates the trajectories of both SDEs and probability flow ODEs for a toy example, where the data distribution was created as a mixture of Gaussian distributions. The ODE trajectories are noticeably smoother than those of the SDE.

We are mostly interested in the reverse process, as it is what we need to generate data. For example, considering the reverse SDE presented in (7.5), the pair of corresponding SDE and ODE is:

$$\begin{aligned} \text{SDE: } \mathrm{d}\mathbf{x}_t &= -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\mathbf{w}}_t, \\ \text{ODE: } \mathrm{d}\mathbf{x}_t &= -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]\mathrm{d}t, \end{aligned} \tag{7.8}$$

and the general form is given by the following pair of equations:

$$\begin{aligned} \text{SDE: } \mathrm{d}\mathbf{x} &= \left[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t)\, \mathrm{d}\bar{\mathbf{w}}, \\ \text{ODE: } \mathrm{d}\mathbf{x} &= [\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t. \end{aligned} \tag{7.9}$$

Solving this ODE results in the same $q_t(\mathbf{x}_t)$ for all t, when initializing $q_T(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$.

Note that the SDE can be rewritten as

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t) \left[\mathbf{x}_t + \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)\right] \mathrm{d}t - \frac{1}{2}\beta(t) \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) + \sqrt{\beta(t)}\, \mathrm{d}\bar{\mathbf{w}}_t, \tag{7.10}$$

where the first term corresponds exactly to the probability flow ODE, and the other terms are related to the Langevin diffusion dynamics of the SDE. In practice, the latter terms can compensate eventual errors during the diffusion process,. Therefore, the sampling quality is usually better using the SDE than using the ODE. However, working with the ODE has some very interesting advantages that it is worth mentioning:

- Solving ODEs is much easier than solving SDE. There is a wide body of literature on how to solve ODEs quickly and efficiently. In practice, we can just use one of these solvers.
- The ODE can be run in both directions. From noise we can generate new samples following the data distribution. But we can also run it in the forward direction and map a given sample to the latent space of the Diffusion Model. This enables to obtain a representation of samples in the latent space, allowing for numerous applications, such as performing semantic interpolation in the latent space.
- Last but not least, working with ODEs allows us to compute the log-likelihood of the samples, which is what we need for performing anomaly detection in the context of this thesis.

The probability flow ODE can be seen as a continuous Normalizing Flow. However, in this case, the network is trained with score matching, while in continuous Normalizing Flows, for each train iteration, the whole trajectory has to be simulated and backpropagated, which is a much more difficult task.

7.4 Log-likelihood computation

This section introduces the log-likelihood computation using the Probability Flow ODE, which will be used in the proposed method explained in Section 7.6.

So far, we only considered the case where the diffusion coefficient $\sqrt{\beta(t)}$ (or $g(t)$ in the general form) is independent of $\mathbf{x}(t)$. But this framework can be extended to hold cases where it also depends on $\mathbf{x}(t)$. In this case, the forward process is governed by the following SDE:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + \mathbf{G}(\mathbf{x}, t)d\mathbf{w}, \tag{7.11}$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ and $\mathbf{G}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^{d \times d}$.

Its associated reverse-time SDE, according to [4], is given by:

$$d\mathbf{x} = \left\{\mathbf{f}(\mathbf{x}, t) - \nabla \cdot \left[\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top\right] - \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right\} dt + \mathbf{G}(\mathbf{x}, t) d\bar{\mathbf{w}}. \tag{7.12}$$

Finally, the Probability Flow ODE corresponding to the previous equation is

$$d\mathbf{x} = \left\{\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}\nabla \cdot \left[\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top\right] - \frac{1}{2}\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right\} dt. \tag{7.13}$$

See Appendices A and C of [140] for a detailed derivation.

As discussed previously, the denoising score matching network learns to approximate $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. Therefore, we can substitute the score by our network approximation, as follows:

$$\begin{aligned} d\mathbf{x} &= \tilde{\mathbf{f}}_\theta(\mathbf{x}, t)\, dt \\ \tilde{\mathbf{f}}_\theta(\mathbf{x}, t) &:= \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}\nabla \cdot \left[\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top\right] - \frac{1}{2}\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\top \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t). \end{aligned} \tag{7.14}$$

For the previous case, where $\mathbf{G}(\mathbf{x}, t) = g(t)$, we obtain that $\tilde{\mathbf{f}}_\theta(\mathbf{x}, t)$ has the following form:

$$\tilde{\mathbf{f}}_\theta(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t). \tag{7.15}$$

According to [27], we can compute the log-likelihood of $p_0(\mathbf{x})$ using the instantaneous change of variables formula, as follows:

$$\log p_0(\mathbf{x}(0)) = \log p_T(\mathbf{x}(T)) + \int_0^T \nabla \cdot \tilde{\mathbf{f}}_\theta(\mathbf{x}(t), t) dt. \tag{7.16}$$

Here, the log-likelihood in the data space $\log p_0(\mathbf{x}(0))$ is computed using the prior distribution for time T, and a correction term involving $\mathbf{x}(t)$, which is the solution of the Probability Flow ODE. However, computing the divergence $\nabla \cdot \tilde{\mathbf{f}}_\theta(\mathbf{x}(t), t)$ is usually too expensive in terms of computational cost. Therefore, in practice, the Skilling-Hutchinson trace estimator [73, 134] is used, which only involves computing the Jacobian of $\tilde{\mathbf{f}}_\theta(\mathbf{x}, t)$:

$$\nabla \cdot \tilde{\mathbf{f}}_\theta(\mathbf{x}, t) = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\boldsymbol{\epsilon}^\top \nabla \tilde{\mathbf{f}}_\theta(\mathbf{x}, t) \boldsymbol{\epsilon}], \tag{7.17}$$

where ϵ is a random variable for which $\mathbb{E}_{p(\boldsymbol{\epsilon})}[\boldsymbol{\epsilon}] = \mathbf{0}$, and $\mathrm{Cov}_{p(\boldsymbol{\epsilon})}[\boldsymbol{\epsilon}] = \mathbf{I}$.

The Skilling-Hutchinson estimator is unbiased, meaning that we can theoretically achieve any desired level of accuracy by sampling many $\boldsymbol{\epsilon}$ from $p(\boldsymbol{\epsilon})$ and averaging the results. In practice, evaluating $\boldsymbol{\epsilon}^\top \nabla \tilde{\mathbf{f}}_\theta(\mathbf{x}, t)$ is straightforward because we can leverage the automatic differentiation capabilities of machine learning frameworks, such as PyTorch's "autograd". This makes the process

efficient and easy to implement.

Although, in theory, we can achieve any desired level of accuracy, and the estimation is unbiased, we will see in the following sections of this chapter that the estimation has a high variance and is, in practice, very noisy.

7.5 Solving inverse problems with score-based models

Inverse problems involve inferring unknown parameters or inputs of a system based on observed outputs. In mathematical terms, considering a system governed by the relation $\mathbf{y} = \mathcal{F}(\mathbf{x})$, where $\mathbf{x}$ represents the unknown parameters or inputs, $\mathbf{y}$ denotes the observed outputs, and $\mathcal{F}$ is the forward model describing the system. The goal of an inverse problem is to determine $\mathbf{x}$ given $\mathbf{y}$ and knowledge of $\mathcal{F}$, i.e., to determine $p(\mathbf{x} \mid \mathbf{y})$, knowing $\mathcal{F}$ and $p(\mathbf{y} \mid \mathbf{x})$.

Inverse problems are inherently challenging due to their ill-posed nature. They often lack unique solutions, are sensitive to noise in the data, and require sophisticated mathematical and computational techniques to solve.

Score-based generative models are particularly well-suited for solving inverse problems, which are essentially Bayesian inference problems. Starting from Bayes formula and taking logarithm and gradients with respect to $\mathbf{x}$ at both sides, we obtain:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}). \tag{7.18}$$

With the score matching network we estimate $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, and as $\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$ is known, we can easily compute $\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y})$.

7.5.1 Inpainting with Score-Based Generative Models

Inpainting is the type of inverse problem that we are interested in this thesis, as will be explained when introducing the method in Section 7.6.

Inpainting with score-based generative models involves filling in missing or corrupted parts of an image using the learned score function. This process leverages the generative capabilities of the model to produce plausible completions that match the given context.

Let us assume we are to inpaint an image $\mathbf{x}$ that has some missing parts. These missing parts can be represented with a mask $\mathbf{m}$ where $\mathbf{m}_i = 1$ if pixel i is observed and $\mathbf{m}_i = 0$ if pixel i is missing. In this case, we can easily define the function $\mathcal{F}$ as the masking operation, and the observation $\mathbf{y}$ as

$$\mathbf{y} = \mathbf{m} \cdot \mathbf{x}, \tag{7.19}$$

where $\cdot$ is the point-to-point multiplication.

The general idea for doing the inpainting is to incorporate the known parts of the image into the inpainting process, ensuring that the generated content is consistent with the observed parts. During the reverse diffusion process, we ensure that the observed parts remain unchanged while the model generates content for the missing parts. The process start from the noisy image $\mathbf{y}_T$, and gradually denoise the image using the learned score function $\nabla \log p(\mathbf{y}_t)$. At each time step t, we ensure that the observed parts are enforced:

$$\mathbf{y}_{t-1} \leftarrow \mathbf{m} \cdot \mathbf{x} + (1 - \mathbf{m}) \cdot \mathbf{y}_{t-1}. \tag{7.20}$$

Finally, we repeat the reverse diffusion process iteratively, refining the generated parts of the image while keeping the observed parts fixed. This iterative refinement continues until the image is denoised, resulting in a plausible inpainting of the missing parts.

7.6 Method

In previous sections, we discussed two key advantages of score-based Diffusion Models that can be utilized for developing a novel anomaly detection method: their ability to estimate likelihood and their capability for inpainting. These two elements are fundamental to the method presented in this section: Diffusion Anomaly Detection (DAD).

Before discussing the details of the entire method, let us visualize the core idea and motivation behind it. For that purpose, Figure 7.2 shows the different stages of the procedure. Figure 7.2(a) shows the original image. Figure 7.2(b) represents the anomaly map (in this case generated by U-Flow, just as an example). This map is binarized in Figure 7.2(c), and this mask is used for inpainting in Figure 7.2(d). Finally, figures 7.2(e) and 7.2(f) show two difference measures between the inpainted and original images. This figure is meant just to illustrate the idea of the method. But, given that the same score-based Diffusion Model is able to estimate the likelihood and perform inpainting, the goal of this method is to do every step of this procedure using the same Diffusion Model.

While U-Flow and other Normalizing Flow-based methods achieve excellent results, they typically operate at a feature level with much lower spatial resolution than the original image. For instance, using M-CAIT as the feature extractor for U-Flow (as discussed in Chapter 5), the feature maps are sized 28×28 and 14×14 for an input image of 448×448. Since Normalizing Flows preserve spatial dimensions, the embedding and, consequently, the anomaly map remain at these reduced sizes.

In contrast, reconstruction-based methods are expected to be able to capture fine details and high-resolution information directly by operating at the pixel level. This is crucial for industrial anomaly detection, where even minor defects must be identified accurately. The high resolution provided by pixel-level methods ensures that subtle anomalies, which might be missed by lower-resolution feature-level methods, could be, in theory, effectively detected.

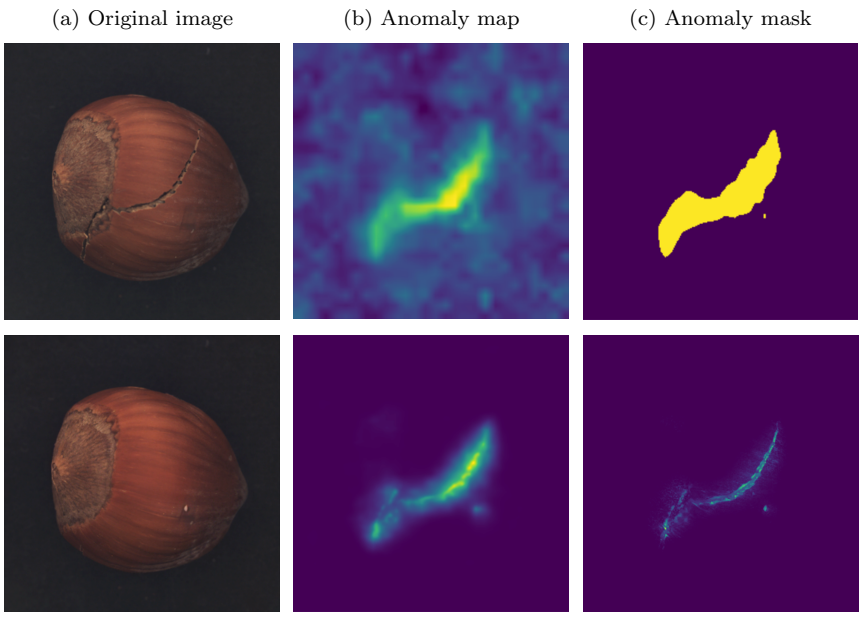


Figure 7.2: DAD intuition. This figure shows an overview of the idea behind the method. Starting from the original image (a), we first compute the likelihood for each pixel. In this case, it is shown in (b) the output of U-Flow [145], just as an example. Then, we build a mask (c) based on the likelihood just by thresholding the previous result and use it to inpaint. The inpainting result is shown in (d). Finally, to obtain a new anomaly map, we evaluate some measure of discrepancy/distance between the inpainted and original images. Two different examples are shown in (e) and (f), using the LPIPS metric and a combination of LPIPS and MSE, respectively.

As mentioned before, the proposed method has two main stages, both solved by the score-based Diffusion Model: one based on the likelihood estimation, and the other based on inpainting. Although the sampling quality is not a key feature directly involved in this method, it is important to asses the model capability of generating samples. It helps in verifying that the model is well-trained and is effectively learning and understanding the patterns present in the training images, by checking that is capable of generating high-quality, realistic outputs. This ensures a comprehensive understanding of the model's capabilities and areas for improvement. Figure 7.3 displays eight different generated samples from the trained model for each MVTec AD category, allowing for a visual evaluation of the generation quality. Additionally, a real image from the training set is included for comparison. The generation quality is generally high, demonstrating the model's ability to produce realistic samples. However, there are areas for improvement. For instance, in the grid category, some of the generated images clearly do not resemble the semantics of the training data (some not crossing grids). This could be an indicator that the model's performance in this category is not as strong as in others. However, as shown in the results of Section 7.9, the method actually performs really well in this category.

Specifically, the proposed method consists of training a score-based Diffusion

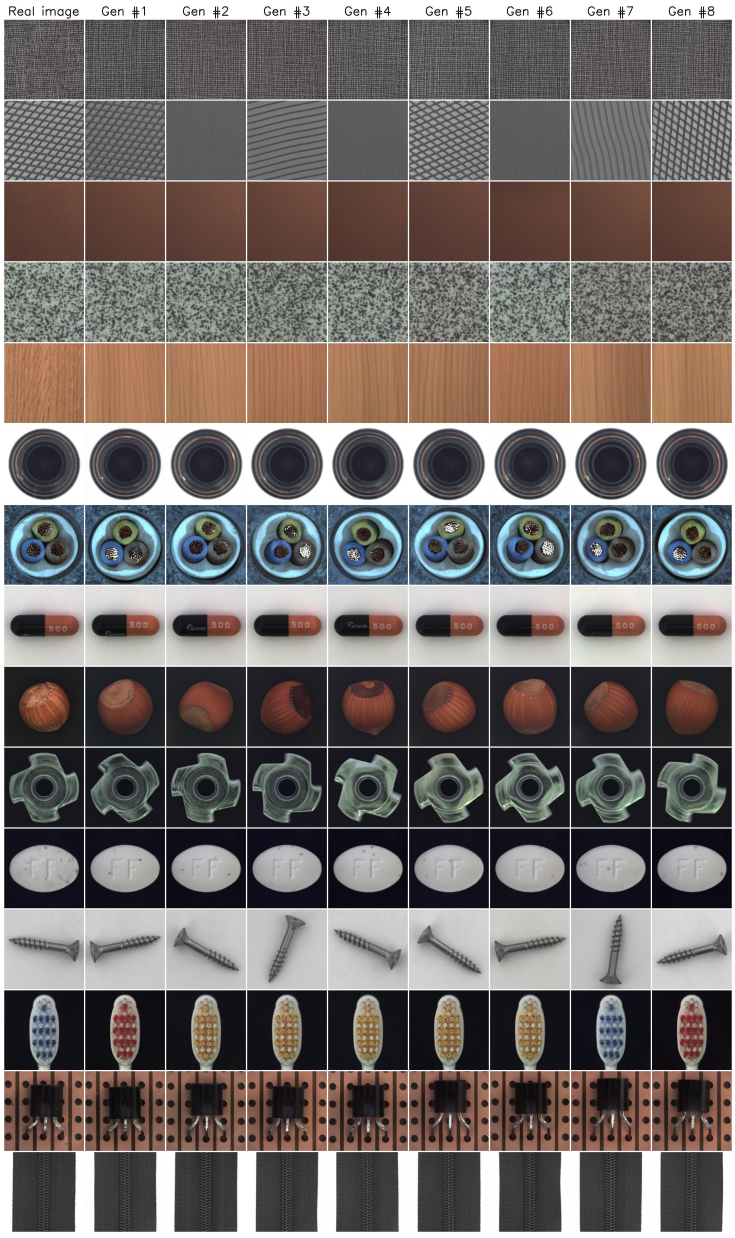


Figure 7.3: Visual assessment of the generative capabilities of the trained score-based models for all categories of the MVTec AD dataset, with each category represented in a separate row. Each row begins with an original image from the training dataset as a reference, followed by eight different generated samples. Generation quality is consistently excellent across the categories, although there is room for improvement in some cases, such as with the grid category.

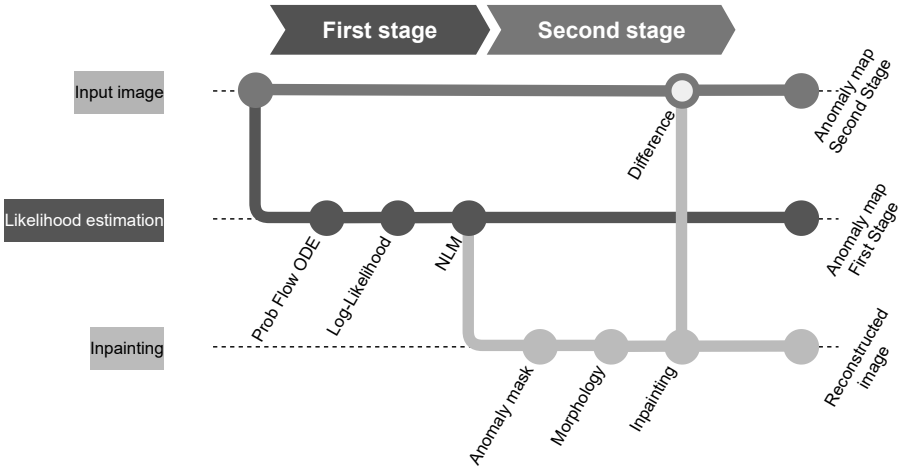


Figure 7.4: DAD diagram. The method has two main stages: likelihood estimation and inpainting, represented as branches in the diagram. In each of these branches, some specific tasks are performed. We start by finding the probability flow ODE, which is then used to estimate log-likelihood. Then, denoising with Non-Local Means (NLM) is applied, and a mask is obtained from it. In the inpainting branch, post-processing based on morphology is applied to perform inpainting in the selected region. A reconstructed "normal-like" image is generated, finally used to evaluate the difference with the original image and generate the final anomaly map.

Model using only normal samples and, in test time, sequentially applying the following six steps, grouped in two stages:

Stage 1

1. Log-likelihood estimation for each pixel (i, j) using the probability Flow ODE.

2. Refinement of the estimation for reducing the noise variance.

Stage 2

3. Creation of a binary mask indicating where the anomalous parts of the images are.

4. Morphological operations to ensure all parts of anomalies are selected.

5. Inpainting only on the regions with low log-likelihood, according to the previously created mask.

6. Estimation of the reconstruction error between the inpainted and original images.

An overview of the method can be found in the diagram of Figure 7.4.

The following explains each step in detail, showing the intermediate results.

7.6.1 Log-likelihood estimation

As explained in Section 7.3, and following (7.15) and (7.14), to compute the log-likelihood we first need to solve the probability flow ODE:

$$d\mathbf{x} = \underbrace{\left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\right]}_{\tilde{\mathbf{f}}_\theta(\mathbf{x}, t)} dt. \tag{7.21}$$

This differential equation is solved using the solver provided in the `scipy` library: `scipy.integrate.solve_ivp`. For all tests, the Runge-Kutta method of order 5(4) `RK45` solver is used. The solution of this ODE is a list of time-stamps t^*, and for each one of them, its corresponding $\mathbf{x}^*$. It can be seen as the whole trajectory of $\mathbf{x}^*(t^*)$, similar to the trajectories shown in white color in Figure 7.1.

Finally, using the solution $\mathbf{x}^*(\mathbf{t}^*)$ to evaluate (7.16), we can compute the log-likelihood as:

$$\log p_0(\mathbf{x}^*(0)) = \log p_T(\mathbf{x}^*(T)) + \int_0^T \nabla \cdot \tilde{\mathbf{f}}_\theta(\mathbf{x}^*(t^*), t^*) dt^*. \tag{7.22}$$

Again, the Skilling-Hutchinson trace estimator is used to compute the divergence in the last equation.

In the previous equation, the log-likelihood in $t = 0$ is estimated using the prior distribution in $t = T$ of $\mathbf{z} := \mathbf{x}^*(T)$, which is actually the embedding of the image in this Diffusion Model, and a correction term involving the integral of the divergence of $\tilde{\mathbf{f}}_\theta$. In our experiments, the latter proved to be always small and, in practice, irrelevant. Therefore, in the following, we use $\log p_T(\mathbf{x}^*(T))$ as our log-likelihood estimation. This choice has a key advantage: we have a model for normality (the Gaussian distribution), and we can analytically and very easily compute the log-likelihood given the embedding $\mathbf{z}$. Note that this embedding would follow the prior Gaussian distribution if the input sample is normal, i.e., follows the training distribution, and would present low likelihood according to the Gaussian distribution if the sample is anomalous.

Note that although this method uses Diffusion Models, it is computationally efficient for anomaly detection in the first stage. Rather than executing T discrete diffusion steps as in traditional DDPMs, it only needs to solve the Probability Flow ODE, significantly reducing computational overhead.

Some preliminary results of this likelihood estimation are illustrated in Figure 7.5, showcasing four images from different categories. The top row displays the original images with their respective defects highlighted, while the bottom row presents the corresponding log-likelihood estimations. Note that the log-likelihood is significantly lower (on average) in the anomaly regions, as expected. Additionally, the precision of anomaly localization is impressive, with the likelihood maps accurately delineating the anomaly areas. However, there

is a notable drawback of this estimation: it is too noisy. As discussed, this estimation is biased, but presents high variance and could be one of the main sources of error.

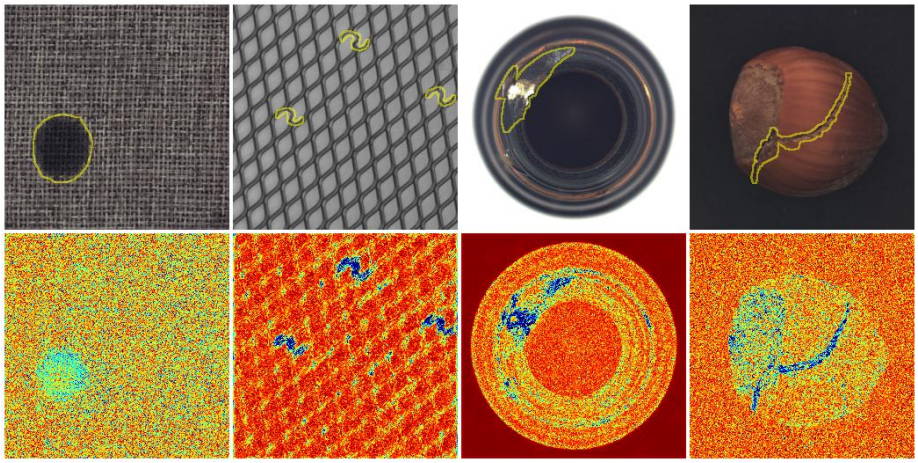


Figure 7.5: Example images of four different MVTec AD categories, with their corresponding log-likelihoods estimations, using (7.22). The top row shows the original images with the anomaly segmentation overimposed. The bottom row is the log-likelihood estimation. Note that although the estimation is extremely noisy, the anomaly structures can be easily identified, even with a very accurate localization precision.

This work is built on the basis of two key papers that introduced and improved the score-based Diffusion Models [138, 140]. But recently, two new important papers have proposed enhancements for likelihood estimation in these Diffusion Models. While these improvements are promising, the question of whether they are sufficiently effective for this task remains largely unanswered. First, VDM [80] proposes to optimize the noise schedule jointly with other model parameters, claiming to obtain more efficient and stable training by tailoring the noise levels to the data distribution more effectively. VDM also utilizes a variational inference approach to parameterize the noise process, allowing the noise schedule to adapt dynamically and enhancing the flexibility and performance of the model. By leveraging variational inference, VDM can better utilize the data, reducing the mismatch between the true data distribution and the model distribution, thus improving density estimation fidelity. On the other hand, i-DODE [177] also introduces improvements to enhance likelihood estimation. One of the primary innovations is that instead of training a score predicor $\mathbf{s}_{\boldsymbol{\theta}}$, the authors propose to parameterize the network using the velocity, defined as $\mathrm{d}\mathbf{x}_t/\mathrm{d}t$, which is claimed to enhance training efficiency by reducing complexity and speeding up convergence. Additionally, the paper employs variance reduction methods, which further improve the speed and efficiency of model convergence. Another significant contribution is the introduction of error-bounded high-order flow matching, a new objective function that fine-tunes models, and it is supposed to obtain better ODE likelihood estimation and smoother sam-

ple trajectories. Finally, the paper proposes truncated-normal dequantization, an evaluation technique that addresses the training-evaluation gap in diffusion ODEs, enabling, in theory, a more accurate likelihood estimation.

In all cases, existing work primarily focuses on using likelihood as a metric to evaluate how well a model can generate data that resembles the observed data, and they typically compute an image-level log-likelihood. In this work, however, we aim to estimate the log-likelihood for each pixel using Diffusion Models, which, to the best of our knowledge, has not been previously explored by the community. As a result, the noisy observations shown in Figure 7.5 have not been the focus of prior research, and existing methods are not tailored for this scenario. Most previous techniques were tested in our work, but they did not significantly improve log-likelihood estimation. Nevertheless, a deeper study and experimentation on these techniques are still needed but are left for future work. Instead, we adopt a different approach to enhance likelihood estimations, which is described in the following section.

7.6.2 Log-likelihood refinement

As explained in the previous section, we can obtain a likelihood estimation with score-based Diffusion Models that looks very accurate in terms of anomaly segmentation, but it is too noisy. For reliable anomaly detection, this noisiness needs to be addressed. Most of the techniques presented in [80] and [177] were tested, with no significant improvement, as the estimations remain too noisy.

In this section, we propose a different and more simple approach for improving these estimations: using a Non-Local Means (NLM) [19] denoising. NLM is a perfect match for this situation. In a normal region of the image, NLM would average different patches, greatly reducing the variance while keeping the mean unaltered, which is exactly what we need in this case, where the estimations are unbiased. The result of this refinement stage are shown in Figure 7.6, and, as it can be easily seen, NLM does a great job in reducing the variance, and the "denoised" anomaly maps clearly separates the anomalous parts of the images.

The impact of NLM in reducing variance is evident in Figure 7.7, which displays histograms before and after NLM denoising for the carpet sample in Figure 7.6 (first column). The histogram of the raw log-likelihood estimation, shown in blue, exhibits very high variance. In contrast, the histogram of the NLM-denoised image, shown in red, clearly demonstrates a significant reduction in noise variance. Additionally, this process effectively separates the anomaly from the normal parts of the image, making the anomaly easily identifiable in the histogram.

The proposed method could conclude at this point by using the denoised negative log-likelihood as an anomaly score. However, this estimation is sometimes poor, as illustrated in the third column of Figure 7.6, where the defect on the bottle is not fully detected, leaving some areas missing. Furthermore, in some cases, the denoising has a negative effect, eliminating the anomaly hints present in the original likelihood estimation. We propose enhancing the method

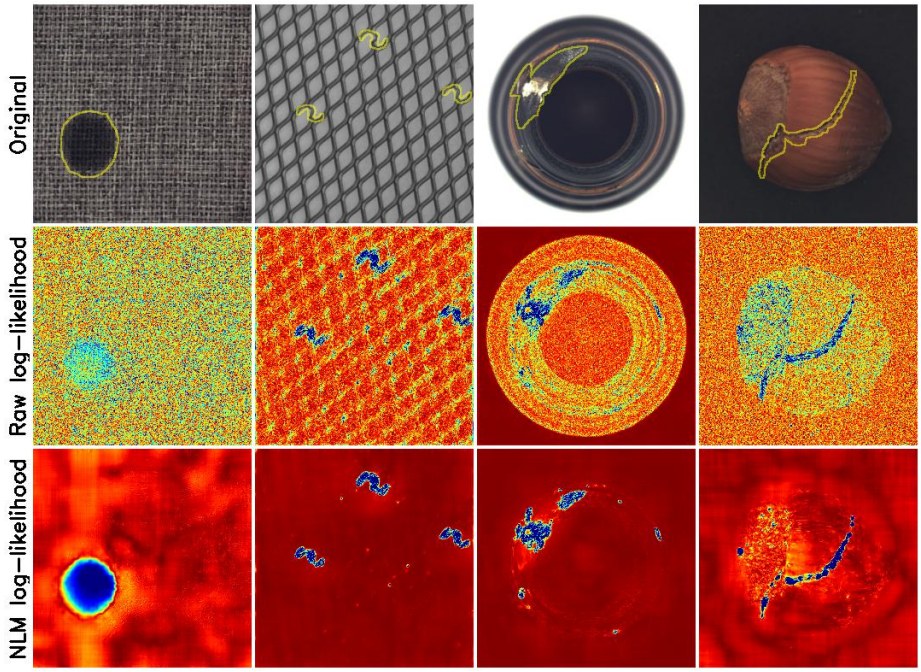


Figure 7.6: Four example images from MVTec-AD dataset. Top row: original images with the anomaly segmentation overimposed. Middle row: preliminary log-likelihood estimation. Bottom row: denoised log-likelihood using Non-Local Means algorithm. The denoised versions of the log-likelihood maps greatly reduce the noise variance and clearly enhance the result. For the bottle image (third column) there are some anomaly regions that are not properly detected.

by adding one more stage, based on inpainting the anomaly regions to address this. The following sections explain how to identify the region for inpainting, perform the inpainting process, and measure the distance between the inpainted and original images.

7.7 Likelihood-guided mask for inpainting

The goal of the second stage of the method is to inpaint the regions with low likelihood. This stage starts by computing a likelihood-guided mask to define the regions to inpaint.

Recalling that the embedding of the input sample, $\mathbf{z}$, would follow a Gaussian distribution if the sample is normal (i.e., adheres to the training distribution), and would deviate from this Gaussian behavior if the sample (or a region of the sample) is anomalous, we can construct the mask using the *a contrario* methodology, as was done all along this thesis. As seen before, this likelihood estimation is unbiased but very noisy. Therefore, we base the detection algorithm on the denoised version of $\mathbf{z}$: $\mathbf{z}_{nlm} = \mathrm{NLM}(\mathbf{z})$. However, in this case, the background model needed for the *a contrario* theory is not as easy as it was before. In

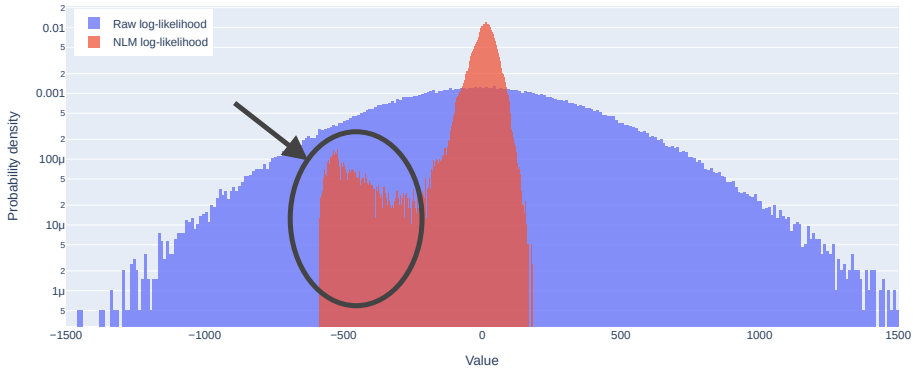


Figure 7.7: Histograms of the first image of Figure 7.6 (carpet category), corresponding to the log-likelihood estimations with and without the NLM processing. It can be easily seen that the noise variance is greatly reduced by NLM. This noise variance reduction makes it easy to identify the anomaly in the histogram values, indicated by the circle and the arrow in this figure.

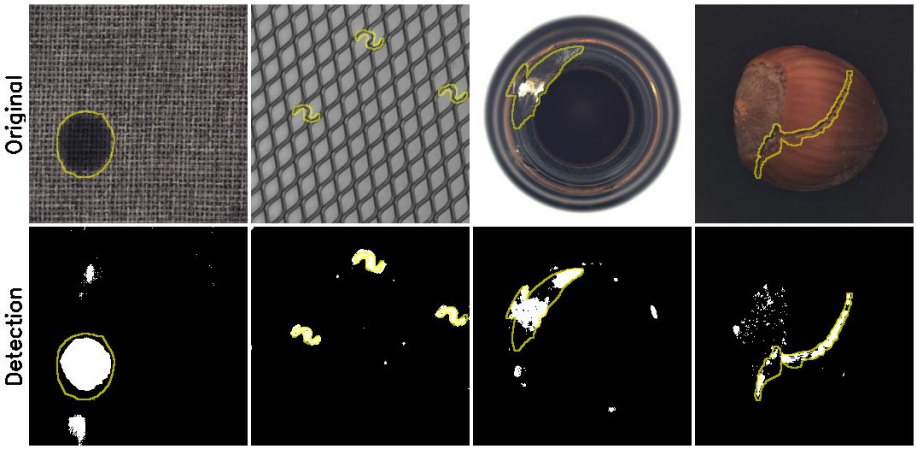


Figure 7.8: Likelihood-guided inpainting mask, using the *a contrario* approach. The top row shows the original images, and the bottom row the obtained masks. In all cases, the ground truth is the solid yellow line.

the normal parts of the image, $\mathbf{z}$ is expected to follow a Gaussian distribution with high variance. After applying the NLM algorithm, this variance is greatly reduced, and this reduction is intuitively related to the amount of averaged patches. According to [20], we still have a Gaussian distribution. On the other hand, in [175], it is argued that this residual follows a Laplacian distribution. To simplify the detection process, we follow the idea of [42], where the authors apply several non-linear transformations, fit a Laplacian and a Gaussian distribution, and keep the one that fits the best. Finally, the residual is transformed and rescaled to fit a standard Normal distribution using the best-fitting distribution and the applied non-linearity. By doing so, we can just apply the detection with the *a contrario* framework using $\mathcal{N}(0,1)$ as the background model for each

pixel position. Example results of this process are shown in the bottom row of Figure 7.8.

7.8 Inpainting and anomaly score

At this stage, we have computed a mask that ideally indicates the anomaly's location. However, this mask may not be perfect; otherwise, there would be no need for inpainting. The mask could have false positives (FPs), where normal image regions are detected as anomalies, and false negatives (FNs), where anomalous parts of the image are not detected. Among these two types of errors, false positives are less concerning for this stage because the reconstruction from the inpainting process is expected to be very similar to the input image in normal regions. Given this, we clearly prefer false positives over false negatives. Therefore, the threshold used for the number of false alarms is somehow permissive, allowing, on average, 1000 false alarms per image. Moreover, before inpainting, we apply a rough dilatation of the mask that was computed in the previous step. Note that if we leave some part of the anomaly unmasked (FNs), the inpainting process could propagate the anomaly, which is something we definitely would like to avoid.

Inpainting results and their corresponding anomaly maps are shown in Figure 7.9. As can be seen, the inpainting quality is excellent. Note that, depending on the application, this intermediate output could also be of great interest. The difference between the original and inpainted images is computed using the LPIPS metric. This metric is generally very effective in avoiding some pixel value differences that are perceptually irrelevant, but sometimes overlooks highlighting certain differences, such as in some defective parts of the bottle example (third column of Figure 7.9).

The whole method is summarized in Figure 7.10, showing the intermediate results of all steps.

7.9 Preliminary results

This thesis presents a preliminary end-to-end implementation of our proposed method. Each step was initially implemented to validate the overall concept. However, each step requires further refinement and enhancement to achieve optimal performance. The goal of this chapter is to explain the theory behind the method and introduce the concept, rather than present a finalized method. Future work will focus on these enhancements.

For the first stage, more advanced techniques need to be considered to improve the likelihood estimation, such as incorporating the improvements proposed by VDM [80] and i-DODE [177]. Additionally, instead of basing the anomaly score on the embedding $\mathbf{z}$, we could consider using the actual likelihood at time $t = 0$, $p_0(\mathbf{x}(0))$.

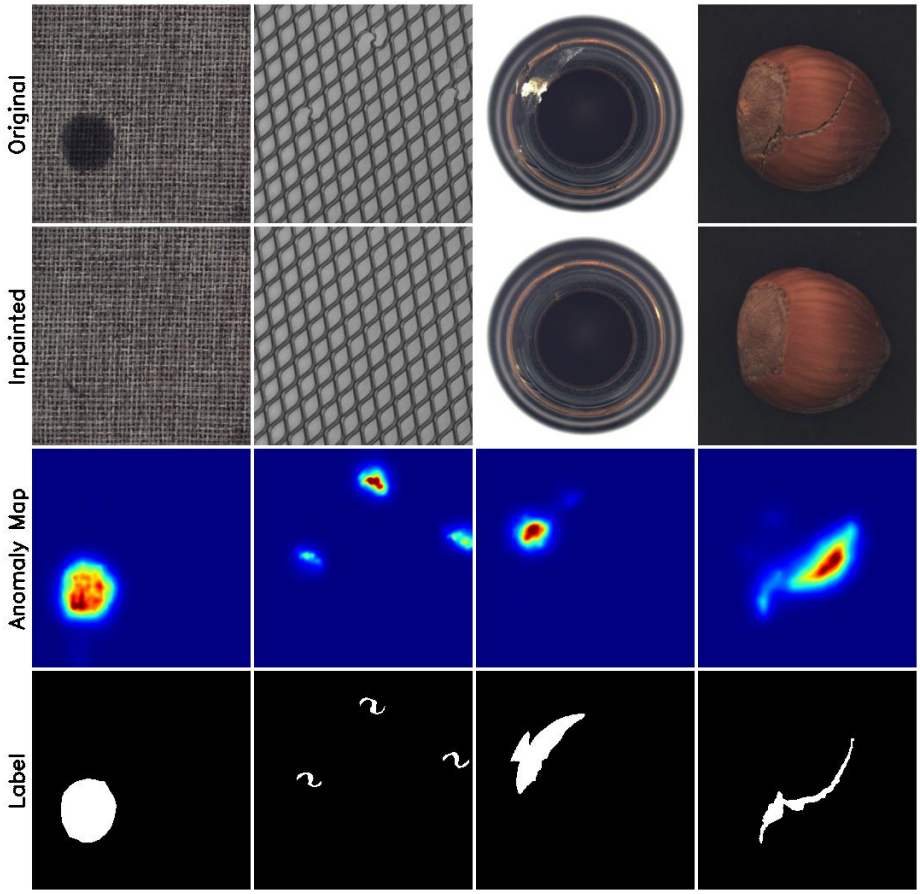


Figure 7.9: Inpainting results with their corresponding anomaly maps. The original images (first row) are used to perform a first raw estimation of the log-likelihood. Then, this estimation is refined with NLM, and a mask delineating the anomalies is created using the *a contrario* methodology. This mask is postprocessed (dilation) and used for inpainting the potentially anomalous regions. The results of this inpainting are shown in the second row of this figure. The difference between the original and inpainted images is shown in the third row, using LPIPS [176] as a metric. Finally, the ground truth is included in the last row for reference.

Regarding the NLM denoising, it is planned to modify the algorithm to use a fixed number of patches for averaging. This modification would allow us to derive a model for the resulting map for normal samples. With sufficient improvements in the first stage, the second stage might not be needed at all. However, the inpainted image, which is a sort of corrected image, could be useful depending on the application, providing a good reason to retain both stages. Another potential enhancement is to use the likelihood map itself as a weighting factor for solving the inverse problem, rather than relying on a binary mask for inpainting.

Considering all these potential improvements is left for future work. We

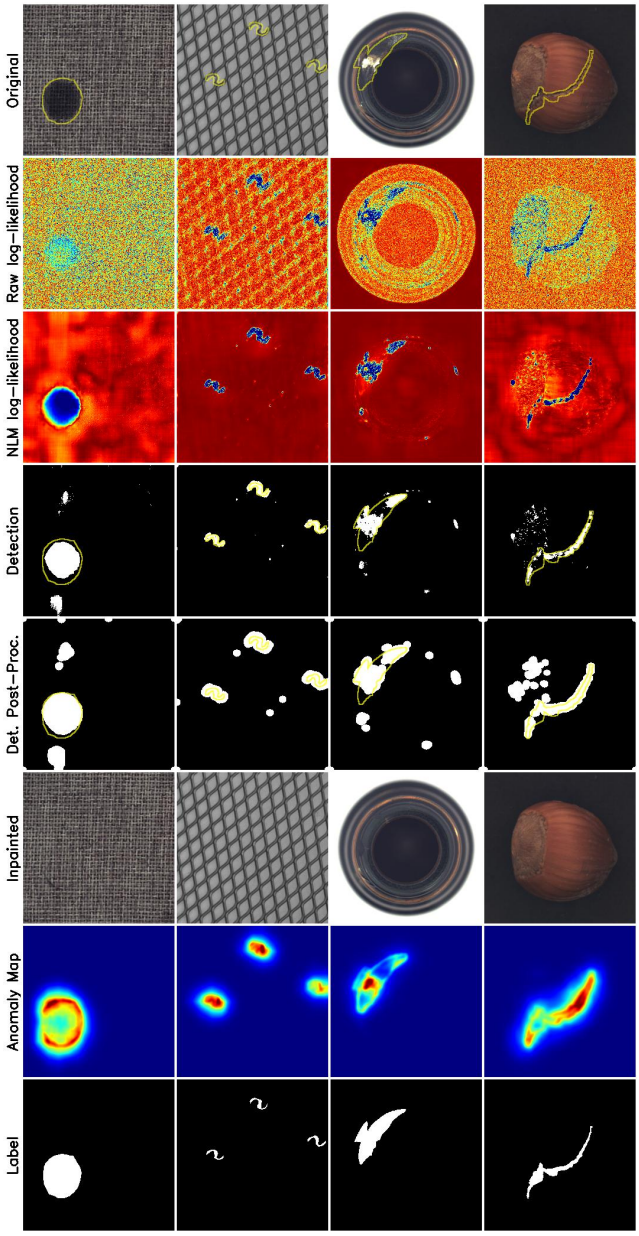


Figure 7.10: Summary of the DAD method with all intermediate results. All rows from top to bottom: original image with overimposed ground truth, raw log-likelihood estimation using the embedding $\mathbf{z}$, log-likelihood enhancement with NLM, log-likelihood segmentation using *a contrario* methodology, log-likelihood post-processing to cover better the anomalous regions, inpainted result in the masked region, LPIPS difference between the inpainted and original images, and ground truth.

	U-Flow (Reference)	DAD (first stage)	DAD (second stage)
Carpet	99.42	84.25	98.40
Grid	98.49	98.96	99.06
Leather	99.59	96.86	98.90
Tile	99.64	86.11	91.37
Wood	97.49	80.90	91.63
Av. texture	**98.51**	**89.42**	**95.96**
Bottle	98.65	92.26	83.18
Cable	98.61	91.15	83.48
Capsule	99.02	88.32	91.15
Hazelnut	99.46	94.85	94.18
Metal Mut	98.82	86.21	91.89
Pill	99.35	96.21	98.49
Screw	94.39	74.59	76.22
Toothbrush	98.79	94.59	96.89
Transistor	97.87	73.18	74.05
Zipper	99.49	94.86	98.03
Av. objects	**98.85**	**90.02**	**89.15**
Av. total	**98.74**	**89.82**	**91.42**

Table 7.1: Anomaly map AUROC. Comparison of U-Flow as a reference, with the two output anomaly maps from the DAD method: using only the first stage (log-likelihood estimation) and the final output of the second stage (reconstruction error after inpainting).

present in Table 7.1 the results of this preliminary implementation, including U-Flow [145] results for comparison. Despite the preliminary nature of this implementation, the results are very promising. With further refinement, we believe that we can achieve excellent results. Notably, while the second stage greatly improves results for some categories, it is less impactful for others and even deteriorates the performance for some categories, suggesting that we might simplify the method to rely solely on the first stage.

7.10 RIFA: Random Inpainting For Anomaly detection

The method presented in this section is a completely unsupervised variant of the previous one. It uses a pre-trained Diffusion Model, and the masking process is random instead of likelihood-guided.

It is inspired by MAE [69], which introduces a self-supervised learning framework for vision tasks, where a significant portion of input image patches is masked and then reconstructed using an auto-encoder. The model employs an asymmetric architecture with a Vision Transformer (ViT) encoder that processes only the visible patches, making it efficient, and a lightweight decoder that reconstructs the full image. This approach is inspired by masked language

modeling in NLP and leverages the sparse nature of visual data to learn robust representations without labeled data.

In this section, we adapt the concept of MAEs to Diffusion Models. Instead of training the model to reconstruct the missing parts of an image, as done in MAE, we leverage the inpainting capabilities of Diffusion Models. This adaptation aims to develop a fully unsupervised method, allowing the model to complete and refine images and obtaining a very versatile method.

The core idea of the RIFA method involves randomly masking an image multiple times and performing inpainting on these masked regions. When an anomaly is masked, the model is expected to reconstruct a normal version of that region since normal features have a higher occurrence probability than anomalies. Conversely, if a normal part is masked, the model should reconstruct the region similarly to the original. Repeating this process several times, masking different regions randomly, is crucial to ensure the anomalies are masked in some instances.

The method, illustrated in the diagram of Figure 7.11, involves two stages that both follow a procedure of masking, inpainting, and computing an anomaly map based on the difference between the inpainted and original images. The first stage aims to generate an initial approximation of anomaly locations by randomly masking large patches of the image and performing inpainting on these masked areas. The purpose of using large masks is to conceal anomalies as much as possible while retaining some parts of the original image to ensure coherent reconstruction.

In the second stage, the anomaly map generated from the first stage is used to create a new mask for inpainting, this time covering much smaller regions. This stage acts as a refinement step. With smaller masked regions, the inpainting will more closely match the original image, resulting in a more precise anomaly map derived from the difference between the inpainted and original images.

In summary, both stages involve masking the image, inpainting the masked regions, and computing an anomaly map from the difference. The key difference is that in the first stage, masking is done with random patches of a specified size, while in the second stage, masking is guided by the anomaly map from the first stage. This process is repeated multiple times with different random initial masks. The final anomaly map is calculated as the average error across all iterations. The LPIPS measure is used to compute the difference in all cases.

Although several inpainting models were evaluated, all experiments presented in this section were conducted using the Stable Diffusion model [119]. This decision was made due to Stable Diffusion's outstanding sampling quality and its rapid inference speed, making it the most convenient model for our purposes.

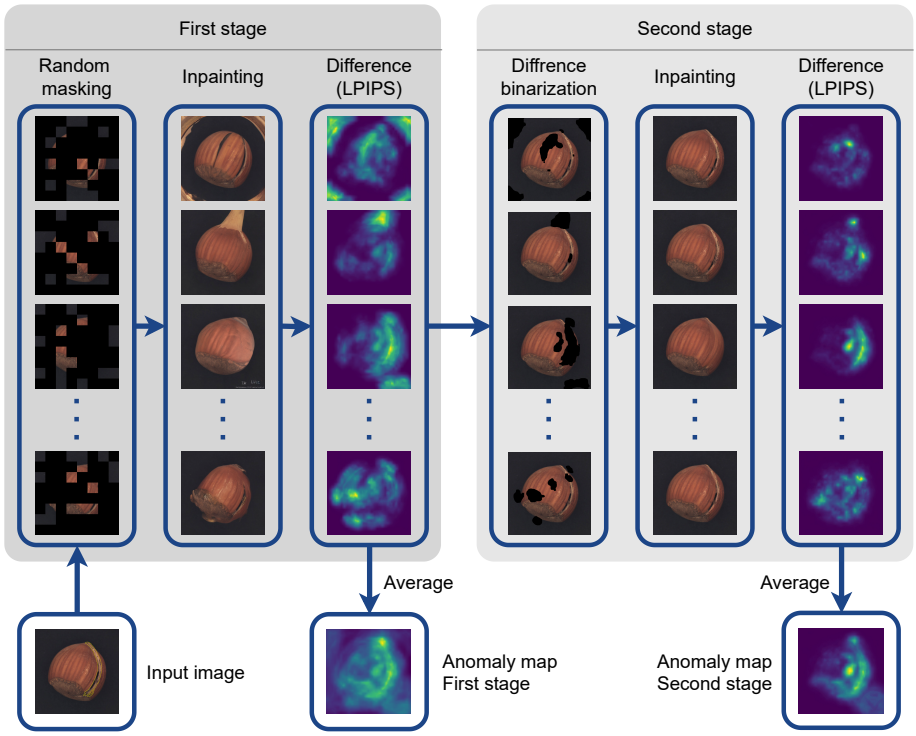


Figure 7.11: RIFA Overview. The method comprises two stages. In the first stage, the input image is randomly masked, the masked region is inpainted, and the anomaly map is computed as the LPIPS difference between the inpainted image and the original image. In the second stage, the anomaly map from the first stage is binarized using a threshold set at the midpoint between the minimum and maximum values, creating a new mask. This new mask is used to generate a second inpainted image, and a new anomaly score is computed in the same manner as before. This entire procedure is repeated multiple times with different random masks, and the final anomaly map is obtained by averaging the anomaly maps from all instances.

7.10.1 Hallucination

Using a pre-trained "general purpose" model comes with certain drawbacks. First, we need to deal with the well-known problem of hallucinations.

Models learn to generate data following the training distribution. When a model is trained on domain-specific data, it focuses solely on that type of data. Conversely, using a model trained on diverse natural images without domain-specific tailoring means the model can generate a wide variety of images. This makes the hallucination problem more prominent. The larger the masked part of the image, the more freedom the model has to hallucinate. An example is shown in Figure 7.12, where 75% of an image from the bottle category of the MVTec AD dataset is masked and reconstructed in six different samples. It is evident that the model reconstructs the missing parts (inpainting) with content that significantly deviates from the bottle's distribution.

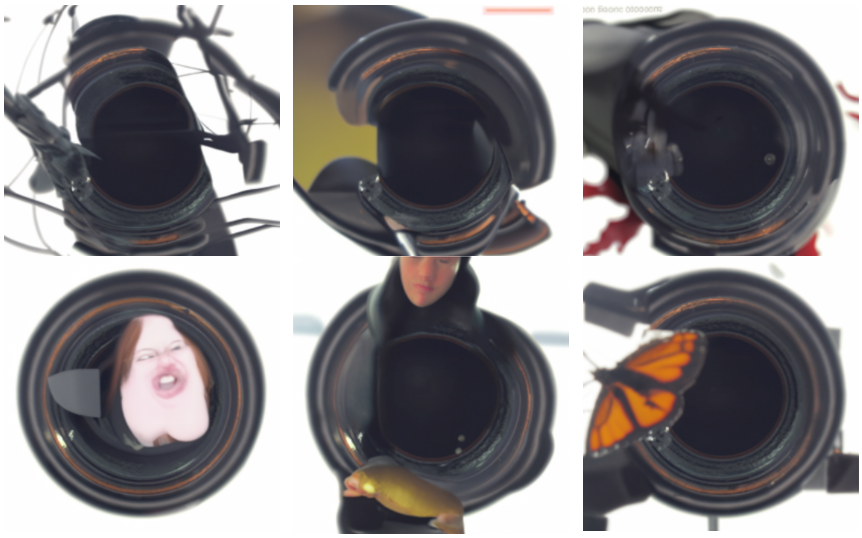
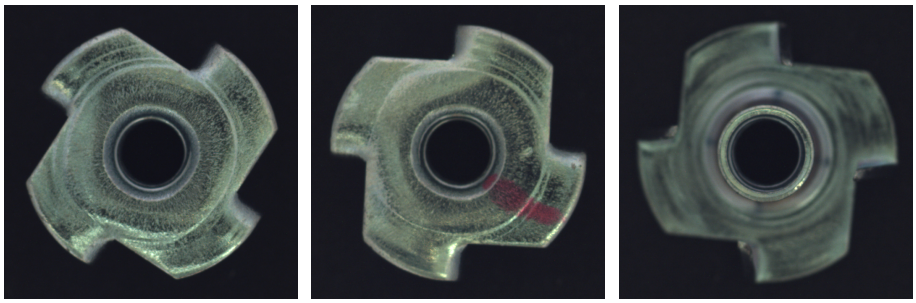


Figure 7.12: Inpainting hallucination examples. This figure demonstrates six different inpainting realizations applied to an image from the bottle category of the MVTec AD dataset. Each image has been randomly masked, with 80% of the image patches masked.

The two-stage approach of RIFA helps to mitigate the issue of hallucination inherent to Diffusion Models by refining the anomaly detection progressively. As mentioned, the first stage uses large masks and aims to conceal as many of the potential anomalies as possible. However, due to this extensive masking, there is significant freedom for the model to hallucinate and generate unrealistic content. The second stage addresses this by using a more targeted mask, which focuses on smaller, specific regions identified as potentially anomalous. By reducing the size of the masked regions, the inpainting process has less freedom to hallucinate, leading to reconstructions that are closer to the original image content. This stage refines the anomaly map and ideally removes the hallucinations created in the first stage. Note that hallucination would most certainly present a high difference with the original image, and thus high values in the anomaly map. At the end of the first stage, high values in the anomaly map can arise for two primary reasons: the model either removed an actual anomaly or generated a hallucination. By applying this process to multiple samples, we expect the hallucinations to appear in different locations each time, given their random nature. In contrast, when an actual anomaly is masked, it is expected to be consistently reconstructed in a manner similar to normal images. Consequently, when averaging the anomaly maps from all these instances, the consistent reconstructions of the true anomalies stand out, while the random hallucinations tend to cancel each other out. This averaging process thus helps to diminish the impact of hallucinations, resulting in a more accurate and reliable final anomaly map.



(a) Normal image

(b) Anomalous image, with no prior knowledge needed

(c) Anomalous image, only detectable with prior knowledge

Figure 7.13: Examples of a normal and two anomalous images from the Metal Nut category of the MVTec AD dataset, showcasing different types of anomalies. (a) shows an anomaly-free image. In (b) is shown an anomaly that could be detected with no extra knowledge beside the image itself. On the other hand the anomaly in (c), which correspond to a flip in the metal nut, could only be detected as anomalous with a prior knowledge of the Normal distribution.

7.10.2 Limitations

Another inherent limitation of this method also stems from using pre-trained models. Since the goal is to create a completely unsupervised algorithm for detecting anomalies, we cannot leverage any prior knowledge about the distribution of normal images. Consequently, we acknowledge that certain types of anomalies will inevitably go undetected. An example of this limitation is shown in Figure 7.13. In Figure 7.13(a), a normal image is displayed as a reference. Figure 7.13(b) shows an anomaly that the method can detect because it is apparent as an anomaly from the image itself. However, Figure 7.13(c) illustrates a type of anomaly that the method cannot detect. This anomaly involves a flipped image, which cannot be identified without prior knowledge of the normal images. This limitation highlights that some anomalies, which require contextual understanding beyond the image itself, remain undetectable with this method.

7.10.3 Preliminary results

Similar to the DAD method, this work is still an ongoing development, and each step could certainly be improved. However, the first implementation already achieved an interesting performance that is worth mentioning.

As discussed above, detecting some defect types is beyond the capabilities of any unsupervised method. Those cases correspond to anomalies that can only be interpreted as such with prior knowledge about the image or the set of normal images. Table 7.2 presents the whole set of defect types for each category of the MVTec AD dataset, indicating with a tick those that are expected to be detected by this method and, with a cross, those that are beyond the possibilities.

A quantitative evaluation of the method's performance is presented in Table 7.3. This table compares the results obtained using only the first stage of the method versus using both stages. For each scenario, we provide metrics derived from the entire test dataset (labeled as "full") and metrics calculated after excluding defect categories requiring prior detection knowledge. The comparison is done with MAEDAY [129], a recent work that leverages the idea of Masked Auto-Encoders (MAEs) [69] for anomaly detection. Notably, our proposed method could be seen as an adaptation of MAEDAY using Diffusion Models. Our approach significantly outperforms MAEDAY across nearly all MVTec AD categories by a large margin.

Table 7.3 shows that, while not all categories benefit from the second stage, performance improves in the majority of them, specifically in 10 out of 15 categories. However, the average AUROC values tend to decrease due to significant drops in some categories, such as Capsule, where the second stage performs no better than random. When compared to a similar approach using Masked Auto-Encoders (MAEDAY), our method significantly outperforms it in almost all categories, achieving better results in 14 out of 15 cases.

To conclude, Figure 7.14 includes some visual results. Figure 7.14(a) shows the original image, and Figure 7.14(b) the intermediate outputs of the method, including

- The randomly masked image
- Its corresponding inpainted result from the first stage
- The initial anomaly map, calculated as the LPIPS difference between the first-stage inpainted image and the original image
- A mask generated by thresholding the initial anomaly map at its midpoint
- A new inpainted image based on the previous mask, representing the output of the method's second stage
- The final anomaly map, computed by comparing the second-stage inpainted image to the original image, in the same manner as before, using LPIPS.

As stated before, the second stage of the process acts as a hallucination corrector. For instance, in the third row of Figure 7.14(b), a significant hallucination occurs in the background surrounding the hazelnut, resulting in an anomaly map filled with false detections. When the second stage is applied, it uses a much smaller mask, greatly reducing the hallucinations. Although this two-stage approach is not infallible (it can be seen in this example that a new hallucination appears in the second stage, affecting the second-stage anomaly map), it effectively minimizes the overall hallucination issue. The fourth row illustrates a similar scenario where the second stage successfully eliminates all false detections caused by the first stage's hallucination.

Bottle	
broken_large	✓
broken_small	✓
contamination	✓
Cable	
bent_wire	✓
cable_swap	×
combined	✓
cut_inner_insulation	✓
cut_outer_insulation	✓
missing_cable	×
missing_wire	×
poke_insulation	✓
Capsule	
crack	✓
faulty_imprint	×
poke	✓
scratch	✓
squeeze	✓
Carpet	
color	✓
cut	✓
hole	✓
metal_contamination	✓
thread	✓
Grid	
bent	✓
broken	✓
glue	✓
metal_contamination	✓
thread	✓
Hazelnut	
crack	✓
cut	✓
hole	✓
print	✓
Leather	
color	✓
cut	✓
fold	✓
glue	✓
poke	✓

Metal Nut	
bent	✓
color	✓
flip	×
scratch	✓
Pill	
color	✓
combined	✓
contamination	✓
crack	✓
faulty_imprint	✓
pill_type	×
scratch	✓
Screw	
manipulated_front	✓
scratch_head	✓
scratch_neck	✓
thread_side	✓
thread_top	✓
Tile	
crack	✓
glue_strip	✓
gray_stroke	✓
oil	✓
rough	✓
Toothbrush	
defective	✓
Transistor	
bent_lead	×
cut_lead	×
damaged_case	✓
misplaced	×
Wood	
color	✓
combined	✓
hole	✓
liquid	✓
scratch	✓
Zipper	
broken_teeth	✓
combined	✓
fabric_border	✓
fabric_interior	✓
rough	✓
split_teeth	✓
squeezed_teeth	✓

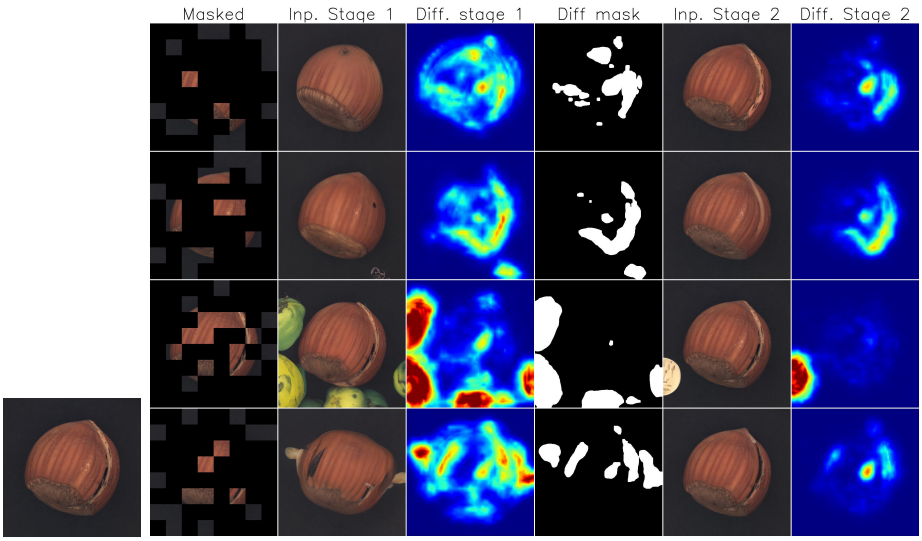
Table 7.2: Different types of defects for each category in the MVTec AD dataset. Each defect type it is indicated with a tick or a cross indicating whether it is detectable with the RIFA method. As explained in Section 7.10.2, the defects that could only be detected with prior knowledge about the image, or only by looking at some normal images, are not expected to be detected with this method, and therefore indicated with a cross in this table.

	MAEDAY	RIFA (1st stage)	RIFA (2nd stage)
Carpet	76.20	**90.09**	85.04
Grid	95.40	**97.58**	93.28
Leather	94.60	94.72	**95.84**
Tile	30.90	73.44	**75.29**
Wood	78.80	88.91	**90.71**
Av. Textures	75.18	**88.95**	88.03
Bottle	50.70	85.75	**87.27**
Cable (full)	55.60	76.05	**76.66**
Cable (fair)	-	85.79	87.32
Capsule (full)	48.10	**72.93**	51.08
Capsule (fair)	-	75.40	53.55
Hazelnut	94.10	95.05	**95.67**
Metal-nut (full)	36.90	**68.39**	64.41
Metal-nut (fair)	-	83.21	84.39
Pill (full)	61.50	78.01	**80.97**
Pill (fair)	-	83.06	83.49
Screw	**96.90**	84.91	82.35
Toothbrush (full)	72.30	91.84	**92.74**
Transistor (full)	59.70	64.66	**64.76**
Transistor (fair)	-	91.90	91.14
Zipper	76.20	80.39	**85.76**
Av. Objects (full)	66.46	**79.80**	78.17
Av. Objects (fair)	-	85.73	84.37
Av. Total (full)	69.37	**82.85**	81.45
Av. Total (fair)	-	86.80	85.59

Table 7.3: Pixel AUROC results for the proposed RIFA method, for the anomaly maps generated at both stages using the LPIPS distance between the inpainted and original images. The comparison is done with MAEDAY, which follows the same idea but using Masked Auto-Encoders.

We used 16 generated samples for all tests. Increasing the number of samples is likely to improve results, as the masking and also the hallucinations are random, tending to occur in different regions of each sample. Therefore, averaging multiple samples helps to mitigate these hallucinations.

The choice of the patch size and the masking percentage is also crucial. Larger patches are more likely to occlude the entire anomaly, whereas smaller patches might leave parts of the anomaly visible, leading the inpainting to reconstruct the anomaly (or some abnormal structure). On the other hand, using smaller patches helps to reduce hallucination, as it is less probable to occlude a big contiguous part of the image. Thus, finding a balance in patch size is essential for optimizing the inpainting process and reducing hallucinations. For all the shown tests, the masking percentage is 75%.



(a) Original (b) Intermediate outputs of RIFA

Figure 7.14: Intermediate outputs of the proposed RIFA method. In (a): the original image. In (b) are shown four different realizations in each row. The first column is the randomly masked image. The second column shows the inpainted result of the first stage, which uses the previous mask. The third column shows the anomaly score produced in the first stage, as the LPIPS difference between the original and the inpainted images. A mask from this anomaly score is created and shown in the fourth column. This mask is used for a new inpaiting, corresponding to the second stage, shown in the fifth column. The last column shows the LPIPS difference between the last inpainted image (second stage) and the original image.

As already commented, the presented results are preliminary, as this is an ongoing work. Every stage of this method needs to be revisited and improved, and extensive experimentation with varying parameters is also left for future work. All that said, we still consider that the obtained results are very good for a completely unsupervised method and could be improved with more work.

Chapter 8

Anomaly detection in two industrial problems

This chapter briefly presents results obtained on the two industrial problems considered in this thesis. As explained in Section 1.2, during the execution of this thesis, two of the projects addressed in Digital Sense were related to anomaly detection:

- **The "Bader" project**. This project served as the initial inspiration for this thesis. It was undertaken prior to and during the early stages of this research. The project aimed to automate defect detection for quality control in leather samples used in the upholstery industry. The goal was to identify defects in leather to ensure the high quality required for luxury car interiors.
- **The "Sienz" project**. Similar to the objective of the Bader project, this project focused on detecting defects in various types of fruits. Initially, the project targeted oranges, and we are currently starting to work with lemons, with plans to expand to other fruits. Sienz uses this defect information to sort and classify fruits based on their condition. This project was executed towards the end of this thesis, allowing us to apply some of the methods developed during this research.

In this chapter, only U-Flow (Chapter 5) is used to evaluate the results for these applications, as it is the best-performing method developed so far in this thesis.

In both cases, the approach that is followed is the same as before: training with normal samples to model normality and using this model to detect anomalies as anything that departs from normality.

8.1 Bader

The dataset built for this test has 720 normal images for training and 345 images for testing (295 defective and 50 normal).

Although the MVTec AD dataset already has a leather category, this specific problem is significantly more challenging due to the subtle distinction between normal and abnormal structures in the leather samples. Figure 8.1 shows example detections on this dataset. The results, with an $AUROC = 94.17$, demonstrate high accuracy in detecting even subtle defects.

8.2 Sienz

The dataset built for this case has only 73 normal images for training and 258 for testing (250 of which are defective images and 8 normal). The results, in this case, were also excellent, with an $AUROC = 90.17$ computed in the test dataset, enabling the use of U-Flow in practice to solve this real-world industrial problem. Figure 8.2 shows results over this dataset.

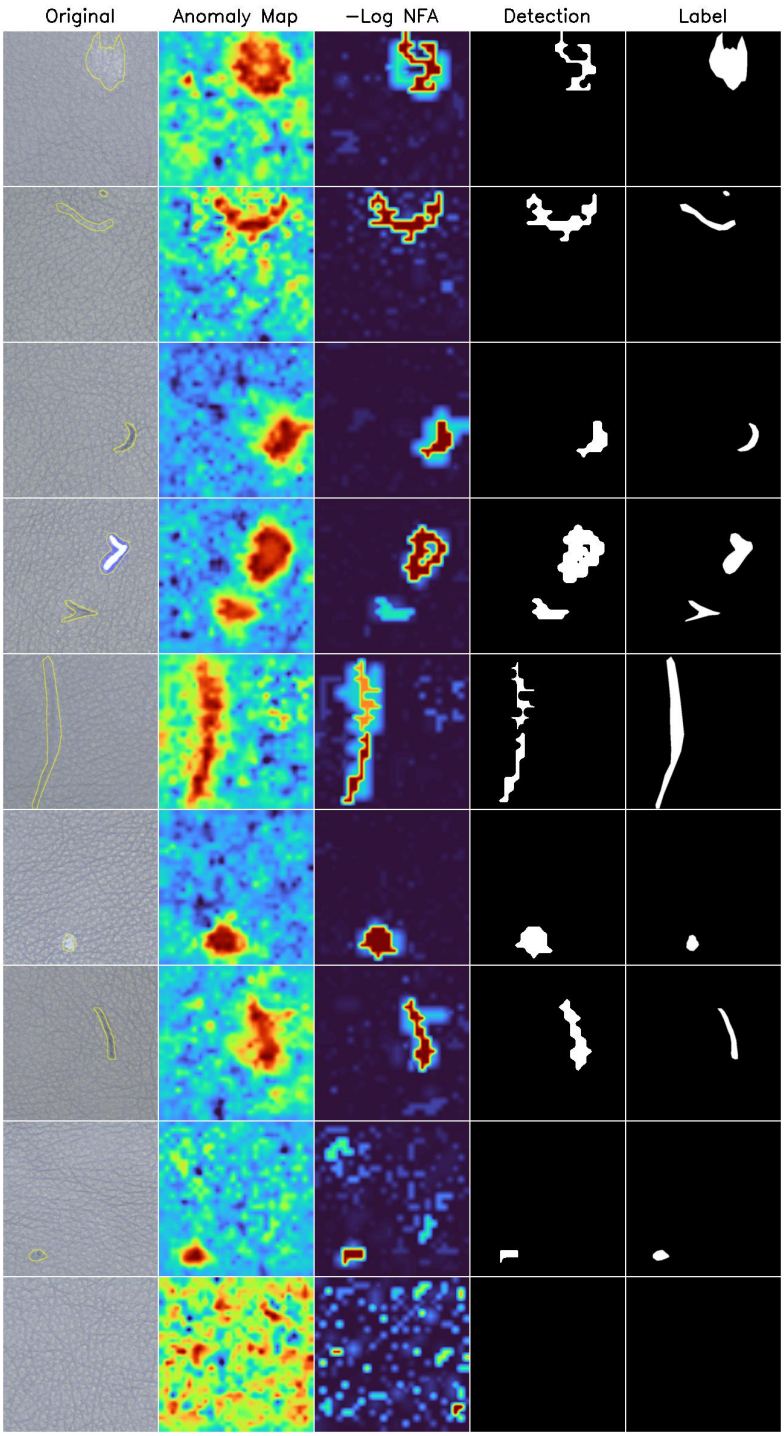


Figure 8.1: Bader results.

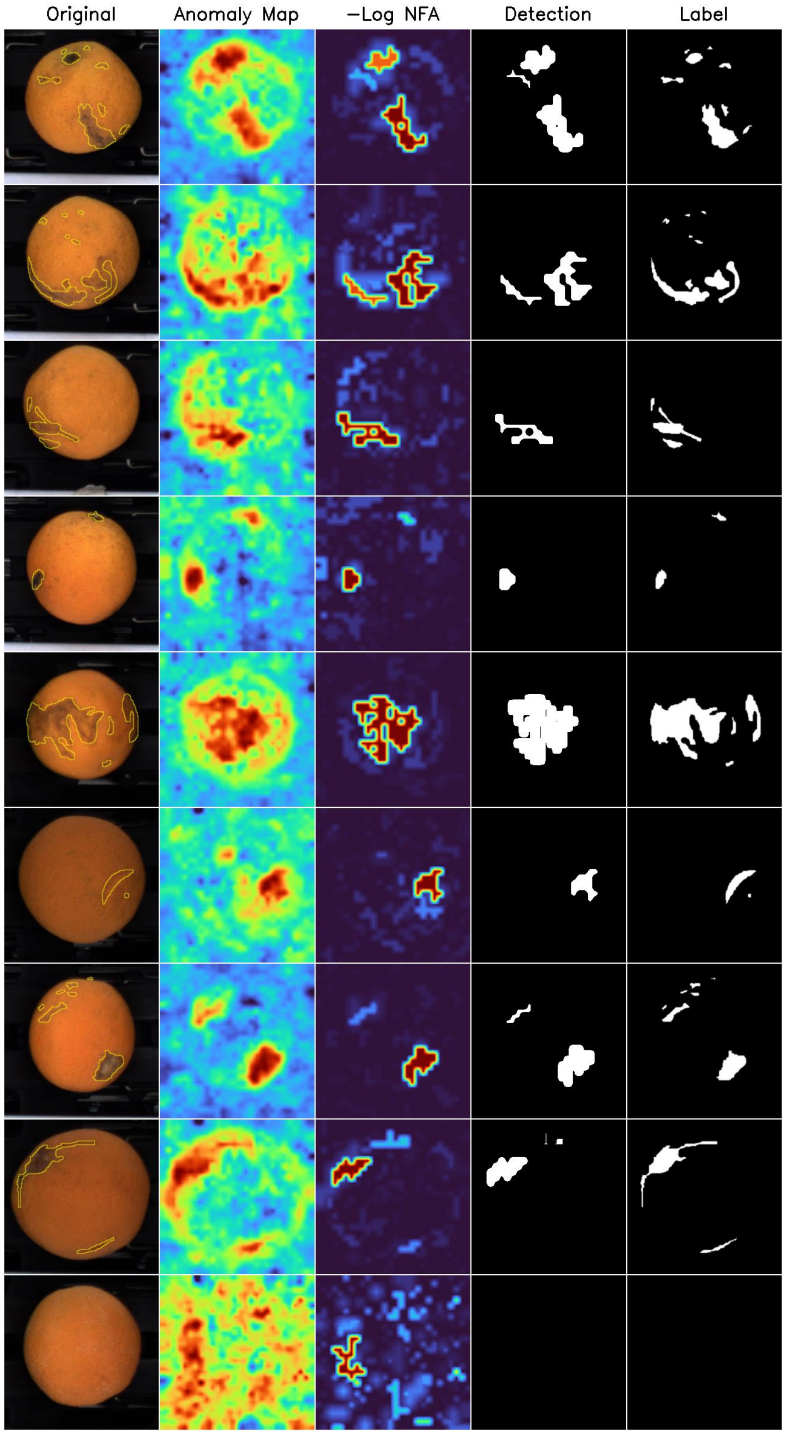


Figure 8.2: Sienz results.

Chapter 9

Conclusion and Future Work

This thesis has focused on advancing anomaly detection methodologies specifically tailored for industrial environments, combining classical image processing techniques with generative modeling approaches, particularly Normalizing Flows and Diffusion Models.

Four distinct methods were developed during this thesis, adopting unsupervised and self-supervised strategies, as they best match this problem due to the rareness and variability of anomalies. Self-supervised one-class methods rely solely on normal (anomaly-free) samples for training, enabling adaptability to different anomaly types without needing explicitly labeled data. Extensive evaluations were conducted on several benchmark datasets, and the approaches were successfully applied to real-world industrial scenarios (such as the Bader and Sienz projects at Digital Sense). To further validate the robustness and generalization of the methods, tests were extended to other domains like medical imaging and surveillance video analysis, highlighting the versatility and applicability beyond conventional industrial use cases.

Specifically, the first method, described in Chapter 3, leverages classical image processing techniques to extract and transform features from the data. The anomaly detection is performed using a multiple-hypothesis testing approach, based on the *a contrario* framework, assuming that the data follows a Gaussian distribution. While this might initially seem like a standard density estimation problem, the challenge lies in operating within a high-dimensional space. Traditional methods struggle due to their limitations in representing such distributions accurately in these large spaces. In the context of anomaly detection, where anomalies are typically located in regions of low probability, it becomes especially difficult to perform the detection in this environment, where all likelihoods are inherently low. At this point, generative modeling plays a crucial role. Specifically, Normalizing Flows are an effective solution to the low densities problem in anomaly detection because they provide a way to model complex, high-dimensional probability distributions while maintaining exact likelihood computation. We used this advantage for proposing U-Flow, the method introduced in Chapter 5. In addition, using Normalizing Flows and optimizing the whole network as a unique graph provides the theoretical guar-

antees needed by the last stage, the multiple-hypothesis testing for segmenting anomalies. This method also capitalizes on feature extraction using pre-trained backbones, such as vision transformers, which capture multi-scale features from the input data. At the final stage, the statistical testing method for anomaly segmentation was significantly enhanced by introducing a novel method based on level sets organized into a tree structure. This approach allows for the virtual testing of all possible regions while only testing a small subset in practice, thus keeping computational costs low. Additionally, we found that this method performs exceptionally well in few-shot learning scenarios, maintaining strong performance even when trained with only a limited number of images, with minimal performance degradation. U-Flow introduces a multi-scale architecture by integrating Normalizing Flows across various scales, operating on features extracted by pre-trained networks. However, since these feature maps are small in spatial resolution, accurately detecting very small defects becomes challenging. While the method achieves excellent performance—often surpassing the state of the art—and successfully detects defects, it struggles with precisely segmenting smaller anomalies. To address this issue, we shifted focus to pixel-level processing, aiming for more detailed detection. This led to the development of the DAD method, presented in Chapter 7, which uses score-based Diffusion Models trained exclusively on anomaly-free images. DAD works in two directions: the inference direction for likelihood estimation and the reverse direction for inpainting regions with low likelihood. It offers two anomaly scores: one derived from the likelihood itself and another based on the reconstruction error post-inpainting. To the best of our knowledge, DAD is the first method to use pixel-level likelihood estimation for anomaly detection, delivering superior precision, particularly for small defects. Finally, in Section 7.10, we introduce RIFA, a fully unsupervised method that performs inpainting over randomly masked regions instead of using likelihood-guided masks.

All the proposed methods in this thesis can be understood as variations of the same pipeline, introduced in Section 1.3, which consists of three main stages: feature extraction, feature transformation, and statistical testing. These stages were progressively refined throughout the development of this work.

- The first method, for example, had two key limitations: one practical and the other theoretical. Practically, it could only be applied to textured images, as it relied on self-similarity, a limitation that was later addressed in subsequent methods, which can handle any type of images. The theoretical limitation was related to the assumption that the data followed a Gaussian distribution after feature transformation without having a formal guarantee. In the later methods, the statistical testing was applied to data distributions that were guaranteed to follow a known distribution.

- Additionally, in the first method, the features were extracted using Gabor filters. However, in U-Flow (Chapter 5), we improved this by using transformers at different scales to generate a more comprehensive hierarchical feature set.

- The statistical testing aspect was also enhanced. The initial *a contrario*

	Type	Textures	Objects	Total
Classic Image Processing (3)	Unsupervised	94.00	-	-
U-Flow (5)	One-Class	98.51	98.85	98.74
DAD (7)	One-Class	95.96	89.15	91.42
RIFA (7.10)	Unsupervised	88.95	79.80	82.85

Table 9.1: Performance comparison in terms of *AUROC* on MVTec AD dataset for the different methods presented.

method operated solely at the pixel level, while later, a more advanced approach based on level sets was proposed. This method organizes connected components in a tree structure, allowing us to efficiently test regions of varying shapes and sizes.

As a final comparison, Table 9.1 provides an overview of the performance of the four anomaly detection methods evaluated on the MVTec AD dataset. U-Flow stands out as the top performer, delivering high accuracy across both textures and objects. DAD follows closely in performance. It's worth noting that U-Flow's soft anomaly maps may provide an advantage for the chosen metric, whereas DAD produces sharper borders around detected anomalies, which can be less favorable for certain metrics like the area under the curve (AUC). However, DAD is still under development, and with further refinements, it can improve and potentially outperform in this or other evaluation metrics.

Finally, to provide a visual comparison of all the methods, we present Figure 9.1, which illustrates examples of textured images, and Figure 9.2, showcasing examples of object images. As is clearly visible, the DAD method succeeds in delivering a far more precise and detailed outline of the anomalies.

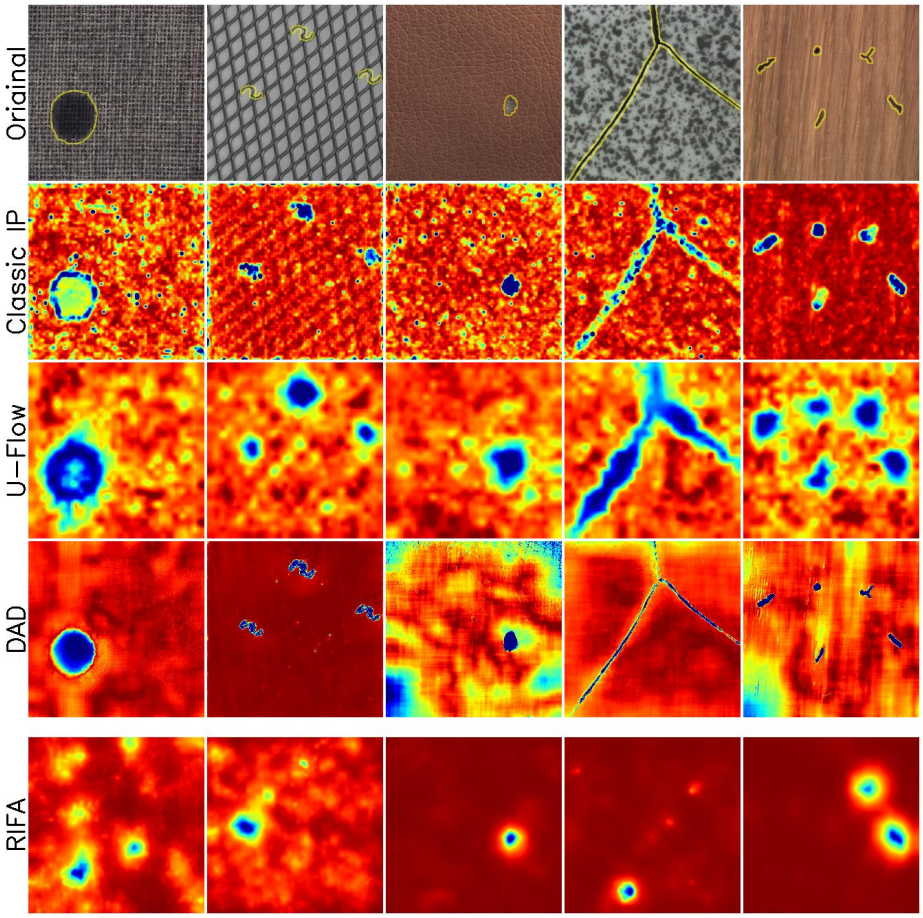


Figure 9.1: Example texture images and corresponding anomaly maps from the four developed methods. "Classic IP" corresponds to the method presented in Chapter 3, U-Flow is the method in Chapter 5, DAD is presented in Chapter 7, and RIFA in Section 7.10. While somewhat noisy, the anomaly maps generated by the first method still yield reasonably good results. These were subsequently refined in the U-Flow and DAD methods. Notably, DAD produces more precise and detailed anomaly maps, isolating the low-likelihood regions strictly over the anomalies without the spread seen in U-Flow's results. Although the anomaly maps from RIFA are less accurate, it is important to note that RIFA is a fully unsupervised method, operating without any prior information about the images, making this comparison somewhat unfair.

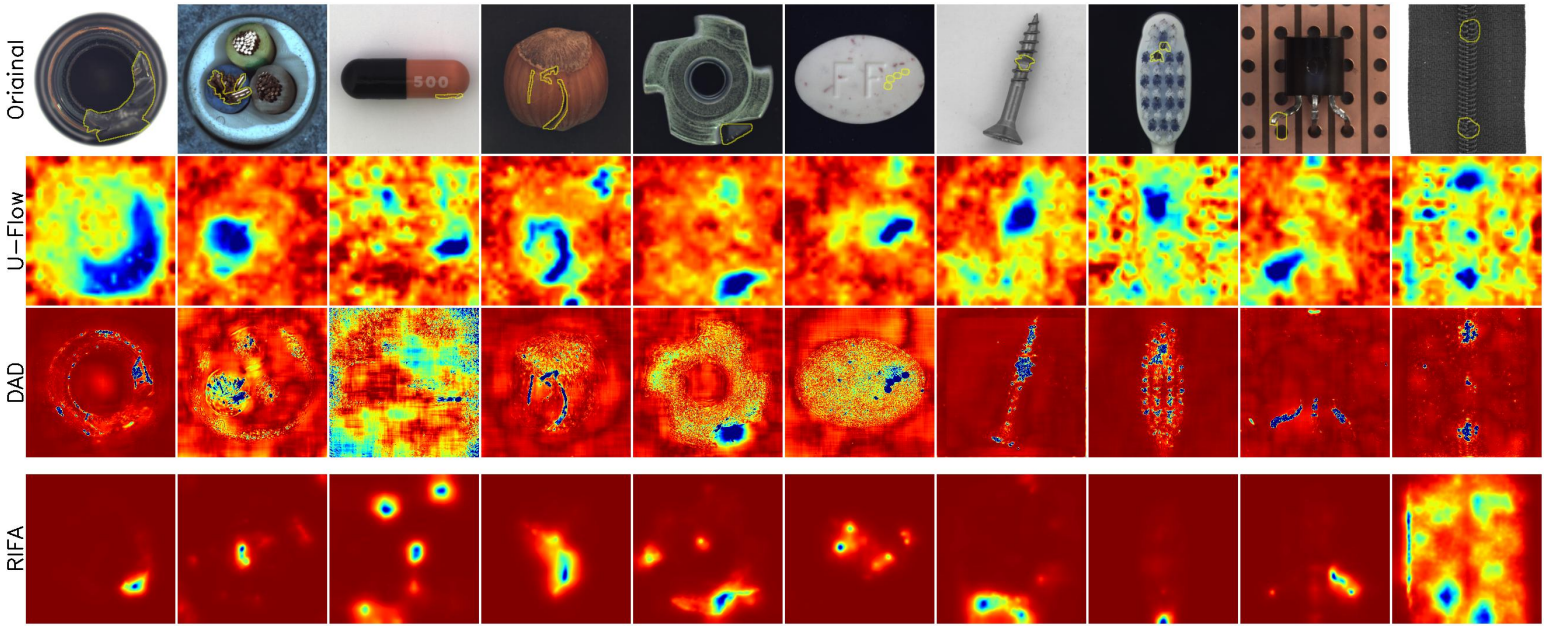


Figure 9.2: Example object images and their corresponding anomaly maps from the four developed methods. Similar to the texture examples in Figure 9.1, DAD provides a significantly more precise delineation of anomalies compared to U-Flow. While the RIFA method does not perform as well, it still manages to detect certain anomalies effectively despite being fully unsupervised.

9.1 Final remarks

This thesis was envisioned as a hybrid endeavor, bridging the gap between academic research and industrial application. Over the past three years, a major effort was put into theoretical understanding and demonstrating practical applications, culminating in state-of-the-art performance and novel solutions for real-world problems. It is particularly gratifying that the methods developed in this work were successfully applied to two distinct real-world problems, yielding excellent results. These applications were aligned with actual company projects, as detailed in Chapter 8.

Beyond industrial anomaly detection, the methods were also tested in different scenarios, such as surveillance and medical fields (Chapter 5). Moreover, taking advantage of the deep study carried out related to Diffusion Models, this thesis explores their utilization in a very different area, such as Counter Forensics, successfully developing an interesting method, with the corresponding paper included in Appendix A.

Recognizing the challenges posed by limited labeled data in many industrial contexts, this research emphasized unsupervised and one-class learning techniques. These methods proved effective in leveraging intrinsic patterns in normal data to detect anomalies.

Traditional image processing methods were explored and refined, providing a solid foundation for anomaly detection. These techniques, known for their reliability and interpretability, were pivotal in developing initial models and benchmarks. Later, extensive investigation into generative modeling techniques, particularly Normalizing Flows and Diffusion Models, enabled the creation of highly effective anomaly detection systems capable of capturing complex data distributions and generating high-quality samples. A significant emphasis was placed on the mathematical underpinnings of the proposed methods, ensuring that the practical implementations were both sound and innovative.

Several codes developed during this research were open-sourced, contributing to the broader research community and facilitating further advancements in the field. The research findings were disseminated through the publication of four papers, with more in the pipeline. These publications have contributed to the academic discourse, and we hope they can provide valuable insights for future research.

There is significant potential to build upon and enhance the methodologies presented in this thesis, and we have many paths in mind. Particularly, we aim to further develop the techniques introduced in Chapter 7, with the main focus on improving likelihood estimation. But other parts of the methods should also be revisited, such as improving the *a contrario* null hypothesis with a better normality model, using different reconstruction methods, and reducing variance with, for example, importance sampling or similar approaches. Furthermore, it would be much better if we could re-implement this score-based model, but

inside a latent space, such as Stable Diffusion [119] does, creating a new "latent score-based" model, that adds the benefits of the latent diffusion, such as its inference velocity, to the score-based likelihood estimation.

We hope the insights and methodologies developed in this work will serve as a foundation for continued innovation and drive progress toward smarter, more autonomous industrial quality control processes.

This page was intentionally left blank.

Appendix A

Diffusion models meet image counter-forensics

From its acquisition in the camera sensors to its storage, different operations are performed to generate the final image. This pipeline imprints specific traces into the image to form a natural watermark. Tampering with an image disturbs these traces; these disruptions are clues that are used by most methods to detect and locate forgeries. In this work, we assess the capabilities of Diffusion Models to erase the traces left by forgers and, therefore, deceive forensics methods. Such an approach has been recently introduced for adversarial purification, achieving significant performance. We show that diffusion purification methods are well suited for counter-forensics tasks. Such approaches outperform already existing counter-forensics techniques both in deceiving forensics methods and in preserving the natural look of the purified images. The source code is publicly available at `https://github.com/mtailanian/diff-cf`.

A.1 Introduction

Image forgeries are present everywhere [58], from fake news on social media [117] to scientific misconduct. Indeed, many image processing tools are available to create visually realistic image alterations. Yet, these modifications leave traces on the image that are tampering cues. Image forensics aims at detecting these alterations by finding local inconsistencies [58]. Image counter-forensics emerged as the research field that challenges forensics methods and explores their limitations [17].

Adversarial attacks share some common properties with image forgeries in the sense that both techniques introduce subtle modifications to the images that, though imperceptible to the naked eye, disrupt the image's traces. The goal of adversarial attacks is to deceive a model into making incorrect predictions. Adversarial purification can be, therefore, linked to counter-forensics since it

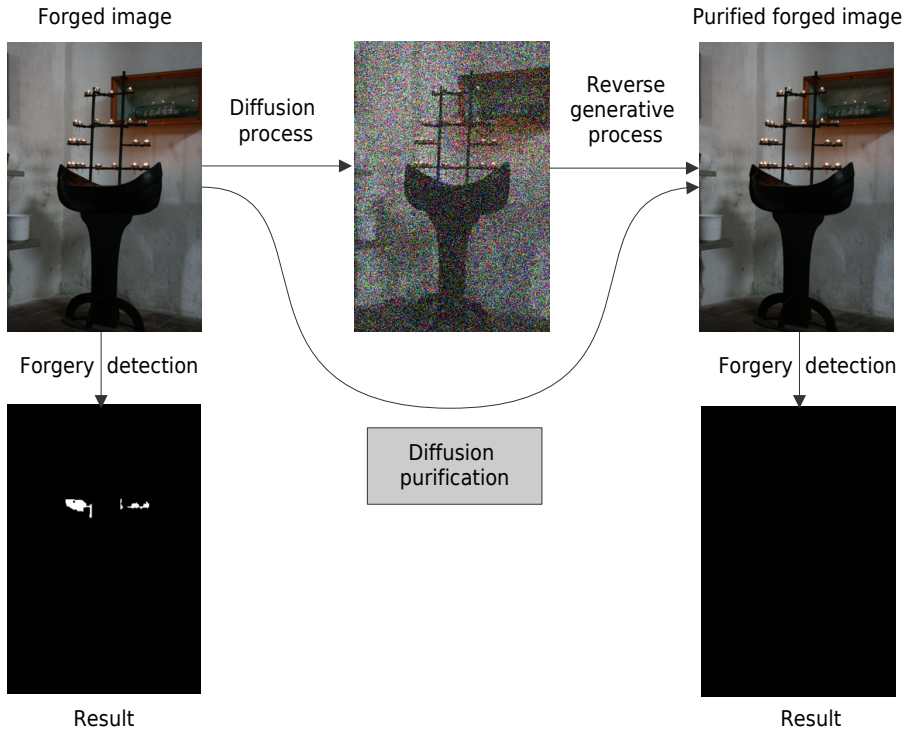


Figure A.1: Illustration of using Diffusion Models as a counter-forensic technique. A forged image from FAU dataset [33], correctly detected by ZERO [114], produces no detection after diffusion purification.

aims at preprocessing the input data to remove these adversarial perturbations. Generally, these purification methods are based on generative models [126].

In recent years, Diffusion Models have emerged as highly effective generative models [71, 140]. These models have showcased impressive capabilities in generating high-quality samples, outperforming traditional Generative Adversarial Networks (GANs) in the realm of image generation. The advancements in Diffusion Models have led to significant improvements in the fidelity and realism of synthesized images, highlighting their potential as state-of-the-art models in the field.

In this work, we evaluate, for the first time, the efficiency of diffusion purification methods, currently used for adversarial purification [113, 154], as counter-forensics methods. The rationale behind the use of Diffusion Models for adversarial purification is that these models learn the distribution of clean data. Hence, by diffusing an adversarial example and then applying the reverse generative process, the Diffusion Model gradually removes the adversarial perturbations and reconstructs the underlying clean sample.

The same rationale can be applied to hide the forensic traces caused by tampering. Indeed, since Diffusion Models are trained on pristine images, diffusion purification methods applied to forged images should recover purified images without any inconsistency in the camera traces. Once such disruptions on the

camera processing chain are erased, purified images should be able to deceive any forgery detection method relying on them. Fig. A.1 shows an example of the aforementioned approach: while ZERO [114] correctly detects the original forgery, once diffusion purification is applied, the method is no longer able to detect it.

A.2 Related work

A.2.1 Image counter-forensics

Counter-forensics attacks can be classified into two categories: the first one corresponds to those that focus on a specific trace or method, while the second category corresponds to generic attacks that aim at erasing all the forgery traces and should, therefore, be able to deceive any forensic method. Among methods of the first category, Fan et al. [57] and Comesana et al. [36] propose attacks against histogram-based methods, mainly used to detect JPEG compression traces. Kirchner et al. [83] propose hiding resampling traces by removing the periodic variations in the residual signal in the spatial domain. Do et al. [49] design SIFT-specific attacks that are able to deceive copy-move forgery detectors based on such local descriptors.

With the advent of learning-based forgery detectors, counter-forensic attacks specifically designed for such methods have also been proposed. Marra et al. [104] design a counter-forensic scheme on the feature space. Their goal is to restore the features of the pristine image and, by doing so, to cross the decision boundary of the target detector. In the case of perfect knowledge of the target method, this counter-forensic method delivers great results. However, when the target detector is unknown, the results degrade tremendously. Other methods countering specific learning-based detectors with an optimum attack, which relies on gradient descent solutions, have also been explored [29, 64].

With limited knowledge of forensic models, counter-forensics attackers focus more on erasing the traces by generic tools [11]. The median filter is a technique commonly used as an anti-forensics attack [166], in deep convolutional neural network versions [78] or even variational formulations [133]. Though this method can be effective on several traces, it leaves a distinctive streaking artifact that can be retrieved [84, 174]. To compensate for this, techniques to remove such artifacts have been proposed [60, 133].

More recently, Chen et al. [25] proposed erasing camera traces, trying not to damage the signal content by adopting a Siamese-based neural network. Cozzolino et al. [39] and Wu et al. [162] use generative adversarial approaches. Baracchi et al. [10] exploit a real camera firmware to perform the manipulation while reproducing the image statistics. This approach can be most efficient at creating real camera traces, and can easily fool camera identification methods into thinking the image was taken with this camera. However, this method is difficult to use, since it requires disassembling a camera to hack its input field.

A.3 Forensic methods

In this section, we provide a brief description of the forensic methods used to analyze the effectiveness of the counter-forensic approaches.

Choi et al. [31] aims to detect inconsistencies in the mosaic pattern with which the raw image was captured. To do so, they use the fact that sampled pixels were more likely to take extreme values. Also aiming at demosaicing inconsistencies, Shin et al. [131] use the fact that sampled pixels have a higher variance to detect forged regions. Bammey et al. [8] combined the translation invariance of convolutional neural networks with the periodicity of the mosaic pattern to train a self-supervised network into implicitly detecting demosaicing artefacts.

Splicebuster [38] uses the co-occurrences of noise residuals as local features revealing tampered image regions. Noiseprint [40] extends on Splicebuster and uses a Siamese network trained on authentic images to extract the noise residual of an image, which is then analyzed for inconsistencies. TruFor [67] also relies on a noise-sensitive fingerprint that is used with the RGB image to detect deviations from the expected regular pattern that characterizes each pristine image.

Zero [114] targets JPEG artifacts. This method counts the number of null DCT coefficients in all blocks and deduces the grid origin. By doing this locally, this method can detect regions having an inconsistent grid origin. Comprint [103] combines the use of a compression fingerprint with the noise fingerprint in [40]. Comprint is an end-to-end fully convolutional neural network including RGB and DCT streams, aiming at learning compression artifacts on RGB and DCT domains jointly.

ManTraNet [165] is a bipartite end-to-end network, trained to detect image-level manipulations with one part, while the second part is trained on synthetic forgery datasets to detect and localize forgeries in the image.

A.3.1 Diffusion-based adversarial purification

Nie et al. [113] were the first ones to propose the use of the forward and reverse processes of a pre-trained Diffusion Model for image adversarial purification. Their method –DiffPure– first diffuses adversarial examples with a small amount of noise. Then, the clean image is recovered through the reverse generative process. A very similar idea was developed at the same time by Blau et al. [16]. The theoretical fundamentals justifying the performance of such diffusion-based adversarial purification methods are derived in [167].

Wang et al. [154] face the difficult trade-off between choosing a long diffusion time, which guarantees the removal of the adversarial perturbation, and choosing a small one, which guarantees the similarity between the input image and the purified one. They propose to guide the reverse process by the adversarial image. By doing so, the purified image is forced to stay close to the input image.

Wu et al. [163] also guide the reverse process by the adversarial image. However, they propose to sample the initial input from pure Gaussian noise and gradually denoise it. The rationale of their approach is that the diffused image still carries corrupted structures, and the reverse process is likely to get stuck in those corrupted structures.

As the field evolves, several applications of these approaches have been developed. In [5], the authors analyze the performance of DiffPure [113] to purify adversarial attacks on the classification of metastatic tissue. In [141], the authors apply the same principle as in [113] but using an extension of Diffusion Models to the 3D space [102]. Similarly, [164] also shares the grounds of DiffPure [113] but using a waveform-based Diffusion Model [85] for adversarial audio purification.

Diffusion purification methods have rapidly gained attention in the field. This interest has even led to questioning the evaluation practices of such techniques [94].

A.4 Background

In this section, we provide a brief overview of Denoising Diffusion Models [71, 135,140] that will be used as a basis for the next section. Recently, denoising Diffusion Models, alternatively called score-based generative models, have emerged as a powerful approach amongst generative methods. Denoising Diffusion Models consist of two processes: a forward diffusion process that progressively adds noise to the input and a reverse generative process that learns to generate data by denoising.

Forward diffusion process. The diffusion process is a Markov process that gradually adds noise to the clean input data. Let T be the number of steps of the diffusion process, $\mathbf{x}_0$ an input image, and $\mathbf{x}_t$ the forward image until step t ($0 \leq t \leq T$). The diffusion process from clean data $\mathbf{x}_0$ to $\mathbf{x}_T$ is defined as

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{A.1}$$

$$\text{with } q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{A.2}$$

where the variances $\beta_1, \ldots, \beta_T$ are predefined small values.

A notable characteristic of the forward process is that there is a closed-form to generate $\mathbf{x_t}$ at any given time step t directly from $\mathbf{x_0}$ [71]. Indeed, let $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$, then we can directly sample $\mathbf{x}_t$ as

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha})}\,\epsilon, \text{ where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{A.3}$$

Reverse denoising process. The reverse generative process is a Markov process that gradually eliminates the noise added in the forward process. The reverse process from $\mathbf{x}_T$ to $\mathbf{x}_0$ is given by

$$p_\theta(\mathbf{x}_{0:T-1}|\mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{A.4}$$

$$\text{with } p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 I), \tag{A.5}$$

where the mean $\mu_\theta(\mathbf{x}_t, t)$ is a trainable network and the variances $\sigma_0^2, \ldots, \sigma_T^2$ can either be fixed or learned using a neural network.

A.5 Proposed method

Our goal is to introduce subtle modifications to a forged image to erase the traces left by the tampering process while, at the same time, preserving the semantic content. Our proposed approach is based on diffusion purification methods [113, 154, 163]. It consists of two steps: first, we add noise up to a certain time-step $t = t^*$ in the forward diffusion process, and then we gradually remove it following the reverse diffusion process, up to $t = 0$. We refer to this method as *Diffusion Counter-Forensics*, or shortly *Diff-CF*.

The intuition behind this idea is that the probability distributions of the forged and its corresponding clean image are separated in $t = 0$, but by adding noise in the forward process, the boundaries between the distributions get fuzzier, and they begin to overlap, more so the higher the value of t^*. Then, starting from a noisy sample that can belong to either probability distribution, the reverse diffusion process, which was trained on pristine images only, generates a purified version of the image with no forgery traces. See, for instance, Fig. 2 in [105].

The value t^* plays a fundamental role. Intuitively, t^* has to be large enough so that the noise added hides the forgery traces, but small enough so that we can preserve the image semantics and structure. If we set the value of t^* too high, the resulting image would deviate too much from the original one. On the other hand, if the value of t^* is too small, we might be unable to erase the forgery traces correctly. This trade-off is studied more in-depth in Sec. A.6.3.

With the purpose of being able to use larger values of t^* without deviating too much from the input image, we also analyze the introduction of guidance in the reverse diffusion process. We refer to this variant as *Counter-Forensics Guided Diffusion*, or *Diff-CFG*. More precisely, we propose to guide the reverse process using the forged image itself, as in [154]. In this way, we encourage the network to produce a clean image as close as possible to the forged one, under the assumption that the forgery traces are subtle enough that they are not reconstructed. In the normal reverse diffusion process, at each time step, a new image is sampled following Eq. A.5. Instead, for this variant, we propose to sample from

$$\begin{aligned} &p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \\ &\quad = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t) - s_t \Sigma \nabla_{x_t} \mathcal{D}(x_t, x_{in}), \sigma_t^2 I), \end{aligned} \tag{A.6}$$

where Σ is the variance of x_t, $\mathcal{D}(x_t, x_{in})$ is some similarity measure between x_t and the input image (forged image) x_{in}, and s_t is a scale factor that depends on the time step t. For high values of t, the forgery traces are completely hidden by the added noise, so we can afford to use large values for s_t, without the risk of guiding the process to reconstruct the forged traces. On the other hand, for small values of t, the forgery traces are more retained, and therefore we should use smaller values for s_t. Similar to what is proposed in [154, 163], we define s_t to be proportional to the added noise, as

$$s_t = s \, \frac{\sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}, \tag{A.7}$$

where s is a hyper-parameter.

For all experiments, we used the following values: $t^* = 40$, $s = 10^6$, and $\mathcal{D} =$ – SSIM [155] as the guidance metric. A detailed discussion on the influence of the hyper-parameters is presented in Sec. A.6.3. In all cases, the images are divided into patches of 256×256 pixels before running the diffusion process. As for the Diffusion Model, we used a pre-trained class unconditional checkpoint[1].

A.6 Experiments

To assess the performance of the proposed approaches, we compared both the non-guided (*Diff-CF*) and the guided (*Diff-CFG*) variants with the Camera Trace Erasing technique (CamTE) [25] and with BM3D [41,92]. While comparison with a plain denoiser is not a common practice in the field, we believe it should be included. Indeed, camera traces are a sort of noise in the sense they produce variations in the pixel's values that are not related to the captured scene. On the other hand, we excluded from the comparison the median filtering, which is a widespread technique in counter-forensics, since it was shown to be outperformed by CamTE [25].

We ran our comparisons in four image forgery detection benchmark datasets: Korus [86,87], FAU [33], COVERAGE [157] and DSO-1 [43].

The goal of counter-forensics methods is to erase all the traces left by the tampering process while preserving the image structures and their semantic content. Therefore, we evaluate two aspects of the counter-forensics techniques under analysis. First, how effectively they hide the forgeries (Sec. A.6.1) and second, the quality of the purified images (Sec. A.6.2).

A.6.1 Forgery traces removal

The first point to evaluate is how well the proposed approaches remove the forgery traces. To do so, we ran several state-of-the-art forgery detection methods on the original datasets as well as in their counter-forensics versions (images purified using different techniques). To evaluate their capability of deceiving the forensics methods, we look at the difference between the detection performance before and after purification. The forensics methods that were used are: ZERO [114], Noiseprint [40], Splicebuster [38], ManTraNet [165], Choi [9, 31], Bammey [8], Shin [131], Comprint [103], CAT-Net [88, 89] and TruFor [67]. Section A.3 introduces a brief description of each method.

To measure detections, we provide scores with the Intersection over Union (IoU), the Matthews Correlation Coefficient (MCC), and the F1 score. In terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), the IoU is the ratio between the number of pixels in the intersection of detected samples and of ground-truth-positive samples and the number of pixels in the union of these sets:

$$IoU = \frac{TP}{TP + FN + FP}. \tag{A.8}$$

[1] `https://github.com/openai/guided-diffusion`

On the other hand, the MCC represents the correlation between the ground truth and detections:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{A.9}$$

The scores were computed for each image and then averaged over each dataset. As most surveyed methods do not provide a binary output but a heat map, to adapt the metrics to the continuous setting, we used their weighted version. We regard the value of a heat map H at each pixel u as the probability of forgery of the pixel. Therefore, given the ground truth mask M, we define the *weighted* TP, *weighted* FP, *weighted* TN and *weighted* FN as:

$$TP_w = \sum_u H(u) \cdot M(u), \tag{A.10}$$

$$FP_w = \sum_x (1 - H(u)) \cdot M(u), \tag{A.11}$$

$$TN_w = \sum_x (1 - H(u)) \cdot (1 - M(u)), \tag{A.12}$$

$$FN_w = \sum_x H(u) \cdot (1 - M(u)). \tag{A.13}$$

IoU and MCC results for all datasets and all methods are presented in Tabs. A.1 and A.2. F1 scores are left out of these tables to ease the readability and are left for Section A.8.

For each dataset, we present in the first row the performance of the forgery detectors over the original images. Then, in the following rows, we show the performance of the same detectors over the considered counter-forensic versions of the images and the difference to the original performance ($\text{metric}_{purified} - \text{metric}_{orig}$). The lower this difference is, the better the counter-forensic method erased the forgery traces.

		CatNet	Choi	Comprint	MantraNet	Noiseprint	Shin	SpliceBuster	TruFor	ZERO	Avg_w
Korus	Original	0.0790	0.1971	0.0534	0.1261	0.0988	0.0221	0.1405	0.3428	0.0050	-
		0.0433	0.1261	0.0461	0.0982	0.0792	0.0568	0.1012	0.2575	0.0028	-
	CamTE	**0.0468** (-0.0322)	0.0597 (-0.1374)	0.0356 (-0.0179)	0.0646 (-0.0614)	0.0420 (-0.0569)	0.0305 (0.0084)	0.0817 (-0.0588)	**0.1961** (-0.1467)	0.0000 (-0.0050)	-0.1024
		0.0278 (-0.0155)	0.0400 (-0.0860)	0.0389 (-0.0072)	0.0644 (-0.0338)	0.0545 (-0.0247)	0.0578 (0.0011)	0.0729 (-0.0284)	**0.1489** (-0.1086)	0.0000 (-0.0028)	-0.0479
	BM3D	0.0997 (0.0207)	0.0352 (-0.1619)	0.0278 (-0.0256)	0.0652 (-0.0609)	0.0420 (-0.0569)	0.0155 (-0.0066)	0.0860 (-0.0545)	0.2579 (-0.0850)	0.0000 (-0.0050)	-0.0819
		0.0646 (0.0213)	**0.0227** (-0.1033)	0.0346 (-0.0115)	0.0746 (-0.0237)	0.0514 (-0.0278)	0.0540 (-0.0028)	0.0744 (-0.0268)	0.1964 (-0.0611)	0.0000 (-0.0028)	-0.0358
	Diff-CF	**0.0418** (-0.0371)	**0.0147** (-0.1825)	**0.0024** (-0.0510)	**0.0255** (-0.1005)	**0.0190** (-0.0798)	0.0027 (-0.0194)	**0.0350** (-0.1055)	**0.1454** (-0.1974)	0.0045 (-0.0005)	**-0.1451**
		0.0204 (-0.0229)	0.0246 (-0.1014)	**0.0215** (-0.0246)	**0.0416** (-0.0566)	**0.0360** (-0.0432)	0.0487 (-0.0081)	**0.0401** (-0.0611)	**0.1131** (-0.1445)	0.0027 (-0.0001)	**-0.0677**
	Diff-CFG	0.0852 (0.0063)	**0.0044** (-0.1927)	**0.0125** (-0.0409)	**0.0442** (-0.0818)	**0.0267** (-0.0722)	0.0040 (-0.0181)	**0.0456** (-0.0950)	0.2064 (-0.1364)	0.0005 (-0.0045)	**-0.1177**
		0.0527 (0.0095)	**0.0043** (-0.1217)	**0.0281** (-0.0180)	**0.0552** (-0.0430)	**0.0446** (-0.0346)	0.0491 (-0.0076)	**0.0488** (-0.0525)	0.1601 (-0.0975)	0.0011 (-0.0017)	**-0.0536**
FAU	Original	0.3228	0.3045	0.0393	0.0203	0.0358	0.1134	0.0074	0.4039	0.5855	-
		0.2329	0.2670	0.0305	0.0336	0.0482	0.1289	0.0251	0.3373	0.5003	-
	CamTE	**0.0141** (-0.3087)	0.1426 (-0.1620)	0.0092 (-0.0302)	0.0154 (-0.0049)	0.0242 (-0.0116)	0.0826 (-0.0308)	0.0045 (-0.0029)	0.0553 (-0.3486)	0.0441 (-0.5414)	-0.6120
		0.0085 (-0.2244)	0.1173 (-0.1497)	0.0206 (-0.0099)	0.0427 (0.0091)	0.0434 (-0.0049)	0.1046 (-0.0243)	0.0277 (0.0026)	**0.0572** (-0.2801)	0.0288 (-0.4715)	-0.4259
	BM3D	0.0757 (-0.2471)	0.0679 (-0.2367)	**-0.0017** (-0.0410)	-0.0268 (-0.0470)	**-0.0014** (-0.0372)	0.0411 (-0.0723)	0.0011 (-0.0064)	0.0802 (-0.3237)	**0.0393** (-0.5462)	-0.6145
		0.0517 (-0.1812)	0.0559 (-0.2111)	**0.0126** (-0.0179)	0.0377 (0.0041)	0.0331 (-0.0152)	0.0803 (-0.0486)	0.0243 (-0.0008)	0.0799 (-0.2574)	0.0266 (-0.4737)	-0.4298
	Diff-CF	**0.0070** (-0.3157)	**0.0242** (-0.2803)	0.0001 (-0.0392)	0.0057 (-0.0146)	**-0.0018** (-0.0376)	**0.0128** (-0.1006)	-0.0050 (-0.0124)	**0.0399** (-0.3640)	-0.0007 (-0.5862)	**-0.6922**
		0.0056 (-0.2273)	**0.0458** (-0.2213)	0.0159 (-0.0146)	0.0355 (0.0019)	**0.0199** (-0.0283)	**0.0602** (-0.0687)	0.0123 (-0.0128)	**0.0520** (-0.2853)	**0.0015** (-0.4988)	**-0.4687**
	Diff-CFG	0.0241 (-0.2986)	**0.0137** (-0.2908)	**-0.0059** (-0.0452)	0.0128 (-0.0075)	0.0002 (-0.0355)	**0.0202** (-0.0933)	0.0127 (0.0053)	**0.0470** (-0.3569)	**-0.0043** (-0.5898)	**-0.6882**
		0.0184 (-0.2145)	**0.0220** (-0.2451)	**0.0143** (-0.0162)	0.0339 (0.0003)	**0.0287** (-0.0196)	**0.0646** (-0.0643)	0.0246 (-0.0005)	0.0592 (-0.2781)	**0.0008** (-0.4995)	**-0.4688**

Table A.1: IoU and MCC results for Korus [86, 87], and FAU [33] datasets and all methods, except for Bammey *et al.* [8]. For each dataset, we present in the first row the performance of the forgery detectors over the original images. Then, in the following rows, we show the performance of the same detectors over the considered counter-forensic versions of the images, and the difference to the original performance ($\text{metric}_{CF} - \text{metric}_{orig}$). The lower this difference is, the better the counter-forensic method erased the forgery traces. The best two scores are shown in bold and underlined for each database. The last column (Avg_w), is the average of the differences $\text{metric}_{CF} - \text{metric}_{orig}$, weighted by the performance in the original dataset.

		CatNet	Choi	Comprint	MantraNet	Noiseprint	Shin	SpliceBuster	TruFor	ZERO	Avg_w
COVERAGE	Original	0.2747	0.0075	0.0230	0.2617	0.0062	0.0615	-0.0571	0.4442	0.0082	-
		0.2199	0.0109	0.0856	0.1856	0.0858	0.1106	0.0423	0.3752	0.0070	-
	CamTE	**0.1480** (-0.1267)	0.0056 (-0.0020)	-0.0015 (-0.0245)	0.0790 (-0.1827)	-0.0230 (-0.0292)	**0.0489** (-0.0127)	-0.0722 (-0.0151)	**0.2614** (-0.1828)	0.0000 (-0.0082)	**-0.1646**
		0.1162 (-0.1038)	0.0079 (-0.0030)	0.0711 (-0.0145)	0.0719 (-0.1137)	0.0770 (-0.0089)	**0.1043** (-0.0063)	0.0361 (-0.0062)	**0.2212** (-0.1541)	0.0000 (-0.0070)	**-0.1048**
	BM3D	0.2666 (-0.0081)	0.0051 (-0.0024)	-0.0281 (-0.0511)	**0.0371** (-0.2246)	-0.0145 (-0.0207)	0.0515 (-0.0100)	-0.0771 (-0.0200)	0.3267 (-0.1175)	0.0000 (-0.0082)	-0.1141
		0.2151 (-0.0049)	0.0036 (-0.0072)	0.0617 (-0.0240)	0.0841 (-0.1015)	0.0773 (-0.0085)	0.1055 (-0.0051)	0.0336 (-0.0087)	0.2863 (-0.0889)	0.0000 (-0.0070)	-0.0571
	Diff-CF	**0.1598** (-0.1149)	0.0011 (-0.0064)	-0.0065 (-0.0295)	**0.0483** (-0.2133)	-0.0115 (-0.0176)	0.0514 (-0.0101)	-0.0602 (-0.0031)	0.2849 (-0.1594)	0.0000 (-0.0082)	**-0.1595**
		0.1278 (-0.0922)	0.0059 (-0.0050)	0.0687 (-0.0169)	**0.0537** (-0.1320)	0.0790 (-0.0068)	0.1055 (-0.0051)	0.0383 (-0.0040)	0.2427 (-0.1325)	0.0000 (-0.0070)	**-0.0974**
	Diff-CFG	0.2003 (-0.0745)	-0.0004 (-0.0079)	-0.0124 (-0.0354)	0.0680 (-0.1937)	0.0024 (-0.0038)	**0.0475** (-0.0140)	-0.0717 (-0.0146)	**0.2738** (-0.1704)	0.0000 (-0.0082)	-0.1478
		0.1607 (-0.0592)	0.0010 (-0.0099)	0.0630 (-0.0226)	**0.0693** (-0.1163)	0.0858 (0.0000)	**0.1051** (-0.0055)	0.0334 (-0.0090)	**0.2386** (-0.1367)	0.0000 (-0.0070)	-0.0890

Table A.2: IoU and MCC results for COVERAGE [157] dataset and all methods, except for Bammey *et al.* [8]. For each dataset, we present in the first row the performance of the forgery detectors over the original images. Then, in the following rows, we show the performance of the same detectors over the considered counter-forensic versions of the images, and the difference to the original performance ($\text{metric}_{CF} - \text{metric}_{orig}$). The lower this difference is, the better the counter-forensic method erased the forgery traces. The best two scores are shown in bold and underlined for each database. For the sake of readability, methods that are not able to obtain a reasonable performance over the original dataset (MCC < 0.03) are grayed out. Bammey *et al.* [8] is excluded from this table, as it was not able to obtain an acceptable performance over any of the considered datasets. The last column (Avg_w), is the average of the differences $\text{metric}_{CF} - \text{metric}_{orig}$, weighted by the performance in the original dataset.

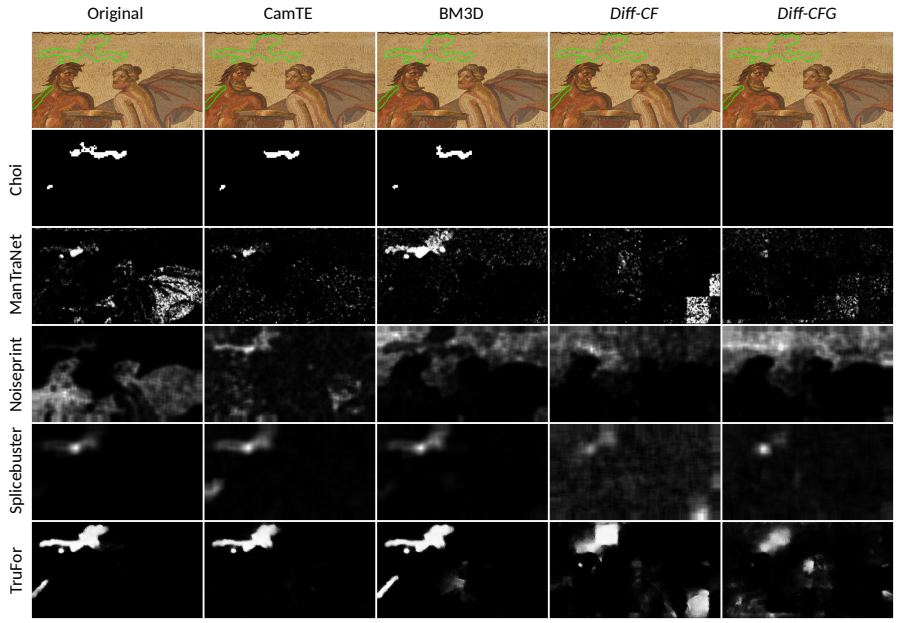


Figure A.2: Results obtained by different forensics methods on the different versions of image `r7710a7fat` from the Korus dataset [86, 87]. We observe that Choi [31], ManTraNet [165], and Noiseprint [40] feature no detection when *Diff-CF* or *Diff-CFG* are applied. For Splicebuster [38] and TruFor [67], even if counter-forensics techniques are not completely able to deceive them, the proposed approaches degrade their detections the most. More examples are included in Section A.9.

The results in Tabs. A.1 and A.2 show that the proposed counter-forensic methods based on Diffusion Models outperform other counter-forensic techniques in most cases. Indeed, except for the COVERAGE dataset [157] where our methods rank second and third (after CamTE), in all the rest of the datasets *Diff-CF* and *Diff-CFG* achieve the best score reductions. When comparing *Diff-CF* to *Diff-CFG*, we observe that the non-guided version delivers, in most cases, the best results as a counter-forensic method. This can be explained by the fact that when we do not condition the method, the reverse generative process is able to get closer to the distribution of the clean training data.

Regarding the forensic methods individually, we observe that TruFor [67] outperforms the rest of the methods in most cases. Furthermore, it is the only method that still performs acceptably after applying counter-forensics attacks, except on the FAU dataset [33]. Indeed, in this case, once counter-forensic methods are applied, the method delivers highly deteriorated results.

Fig. A.2 shows an example of the results obtained by different forensics methods on the different versions of the same forged image. We observe that Choi delivers nearly the same result as in the original forgery when CamTE or BM3D are applied. However, it features no detection when *Diff-CF* or *Diff-CFG* are used as counter-forensics techniques. Noiseprint and ManTraNet provide better detections when CamTE or BM3D are applied, respectively. However, no detection is made when using the proposed approaches. On the other hand, none of the counter-forensics methods is able to deceive Splicebuster and Trufor

completely. However, we can observe that their results degrade the most when *Diff-CF* and *Diff-CFG* are applied [2].

A.6.2 Image Quality Assessment

Another important point to evaluate the pertinence of counter-forensic methods is their resulting image quality. We evaluate this quality in two senses. Firstly, we are interested in how natural the purified images are. To evaluate this, we use the reference-free image quality assessment techniques NIQE [108] and BRISQE [107]. Secondly, it is also important to measure the similarity between the input image and the one obtained after the counter-forensic attack. We, of course, want these two images to be perceptually similar. To evaluate this aspect, we use the full reference image quality assessment methods LPIPS [176], SSIM [155], and PSNR. For all the metrics, we use the implementations provided by the PyIQA library [26].

Results are presented in Tab. A.3. For all reference-free metrics, the proposed diffusion-based counter-forensics methods achieve the best performance. For the full-reference metrics, we also obtained the best performance for LPIPS, but BM3D and CamTE get better performance in terms of PSNR and SSIM.

Among the proposed methods, the guided variant always achieves better performance in terms of PSNR and SSIM, as expected. Indeed, the guidance explicitly encourages the purified image to be close to the input image. Still, the results are not so conclusive when evaluating the LPIPS score, where the non-guided version shows a slightly better performance on the Korus dataset.

It is important to mention that even if *Diff-CFG* uses SSIM as the guidance distance, this does not imply that the obtained scores on that metric should be perfect. In Eq. A.6, the guidance can be interpreted as a sort of gradient descent towards the minimum of $\mathcal{D}(\cdot, x_{in})$. To achieve this minimum, the guidance scale s_t plays a crucial role. Using a non-optimum (in terms of the optimization problem), guidance scale causes the final SSIM score not to be optimal. But this "optimum" guidance scale could not be the best to effectively erase the forgery traces. Sec. A.6.3 studied this trade-off more in-depth.

Regarding the reference-free image quality assessment metrics, *Diff-CF* consistently achieves better results than *Diff-CFG*. This can be explained by the fact that the unconstrained generative process gets closer to the distribution of the images with which it was trained. Therefore, these images look more natural.

Fig. A.3 shows a qualitative example of the different purified images. We observe that both *Diff-CF* and *Diff-CFG* are good at preserving the fine textures and edges of the image, while CamTE and BM3D blur all these fine structures. For instance, the details highlighted in the green patch show that the granularity in the cherubs' cheeks is blurred out by BM3D and CamTE, while it is preserved by the diffusion-based models. This is also visible in the cherubs' chin, highlighted in the yellow patch. As for the edges, the sharpness of the nose (green patch) and the lips (yellow patch) are also better preserved by the proposed approaches.

[2] An analysis of the robustness of these forensic methods is out of the scope of this work and will be addressed in the future.

		NIQE (▼)	BRISQUE (▼)	LPIPS (▼)	PSNR (▲)	SSIM (▲)
Korus	Original	5.7271	13.7602	0.0000	80.0000	1.0000
	CamTE	5.5442	34.5632	0.1684	**38.2833**	**0.9433**
	BM3D	5.1004	38.0418	0.0835	**43.1409**	**0.9802**
	Diff-CF	**3.8693**	**23.1161**	**0.0733**	32.9680	0.8769
	Diff-CFG	**4.1070**	**28.3290**	**0.0771**	34.3391	0.9126
FAU	Original	4.7392	20.5726	0.0000	80.0000	1.0000
	CamTE	5.8360	40.1577	0.2098	**37.8765**	**0.9460**
	BM3D	5.4875	42.7470	0.1045	**41.2625**	**0.9797**
	Diff-CF	**3.8896**	**19.8268**	**0.0985**	33.0308	0.8792
	Diff-CFG	**4.2440**	**29.9920**	**0.0952**	34.4725	0.9159
COVERAGE	Original	4.5529	19.0256	0.0000	80.0000	1.0000
	CamTE	5.4513	30.3558	0.0631	**35.7974**	**0.9648**
	BM3D	5.8792	35.9560	**0.0237**	**44.1417**	**0.9888**
	Diff-CF	**4.3343**	**17.1298**	0.0281	33.4959	0.9275
	Diff-CFG	**5.0359**	**27.8903**	**0.0276**	34.6969	0.9487
DSO-1	Original	3.9174	16.6183	0.0000	80.0000	1.0000
	CamTE	5.2180	40.2867	0.2022	**38.5459**	**0.9446**
	BM3D	5.1870	39.8485	0.1239	**41.9057**	**0.9750**
	Diff-CFG	**3.3907**	**9.2614**	**0.0950**	34.1204	0.8862
	Diff-CFG	**3.6686**	**19.3601**	**0.0899**	35.3473	0.9154

Table A.3: Image quality assessment results of the evaluated counter-forensics techniques. The ▼ indicates that the lower the score the better while the ▲ indicates that the higher the score the better. The best two scores are shown in bold for each database. For the no-reference metrics NIQE and BRISQE, the proposed diffusion-based counter-forensics methods achieve the best performance.

A.6.3 Influence of the parameters

The goal of this work is to provide a first study on the use of Diffusion Models as counter-forensics techniques. As such, it is important to evaluate how the results vary along with the parameters. The non-guided approach *Diff-CF* has only one parameter: the time step t^*, while *Diff-CFG* has two: the time step t^* and the guidance scale s. In this experiment, we focus mainly on *Diff-CFG* since we think the interaction of both parameters is way more complex than analyzing a single one. The experiments in this section are carried out on Korus dataset [86, 87]. We evaluate both the forgery traces removal capabilities and the image quality of the purified images. For the first, we compute the performance drop for the best-performing methods over the original dataset: Choi, MantraNet, Noiseprint, Splicebuster, and TruFor. For the second, we use all the image quality assessment metrics presented in Sec. A.6.2.

Figure A.3: Image quality comparison for all considered counter-forensics methods. We observe that both *Diff-CF* and *Diff-CFG* are good at preserving the fine textures and edges of the image while CamTE and BM3D blur all these fine structures.

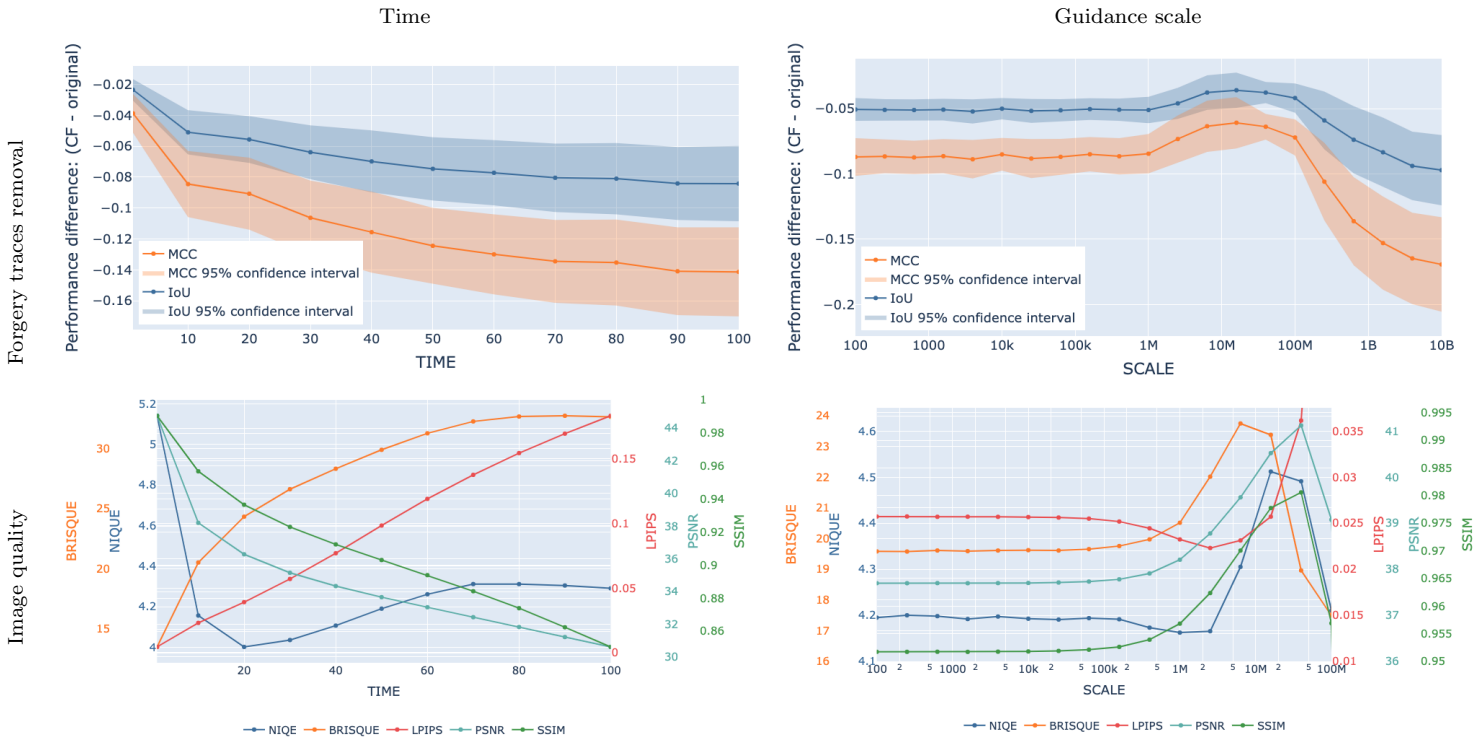


Figure A.4: Study of the impact of the time-step t^* (left-hand side), and guidance scale s (right-hand side). For each parameter, we evaluate its influence on the forgery traces removal task (top) and on the purified image quality (bottom). For the forgery traces removal task, we plot the average difference between the performance before and after purification for the best-performing methods in the original dataset as a function of the parameters' value. The colored background area represents the 95% confidence interval. For the Image quality assessment, all five metrics presented in Sec. A.6.2 are plotted as a function of the parameters' value, each with a different axis, for better visualization. This figure is best viewed in color.

Diffusion time-step. The results of the impact of the time-step t^* are presented on the left-hand side of Fig. A.4. The analysis is pretty straightforward: the larger the value of t^*, the forgery traces removal performance improves (gets lower). On the other hand, the image quality metrics improve the smaller the value of t^*. There is a clear trade-off in the selection of this parameter, that is simple to understand: with higher values of t^*, we add more noise to the original image in the forward diffusion process, which makes it easier to hide the forgery traces. On the other hand, starting the reverse process too far away from the original image leads to larger deviations between the original image and the purified one.

In addition, it is interesting to note that all the full-reference metrics keep strongly degrading as we increase t^*, but the reference-free metrics seem to follow a more asymptotic behavior. This evidence can be explained due to the fact that, even if the generated images are more apart from the original one, the diffusion process, following the learned distribution, is still generating natural images.

Guidance scale The guidance scale ensures that the purified image remains close to the manipulated image, thus not modifying its semantic content. However, it is crucial that the chosen guidance scale is not excessively large since it would cause the purified image to match the adulterated image, potentially retaining the manipulation traces [163].

We conducted a series of experiments to study the scale influence, varying the scale value (s in Eq. A.7), while keeping a fixed time-step $t^* = 10$. As can be seen in the right-hand side of Fig. A.4, the performance difference has small variations for about the first half of the scale range studied, then shows a slight increase, and finally, a great drop. The best point we could choose would be with the lowest value, so at first, one could be tempted to use the highest value for the scale. But if we add the image quality assessment to the analysis, we observe that for those scale values, the quality of the images is highly degraded. Therefore, an intermediate point should be chosen. Note that the optimal point for the removal of forgery traces is not the optimal point in terms of image quality. As mentioned in Sec. A.6.2, this could explain why, in our experiments, we do not obtain the best performance in terms of SSIM, even though we are guiding the diffusion process with this metric.

A.7 Conclusions and Future Work

In this work, we presented a first study on the use of Diffusion Models for counter-forensics tasks. We showed that such an approach can deliver better results than the existing techniques for both forgery trace removal and image quality. Of course, there is a risk that the shown approaches would be used by people wanting to create forgeries and make them look authentic. The simplicity of this method increases this risk. However, it is also because of its simplicity that the method should be made public: It is important to expose the shortcomings of current methods so that one can know how much trust can be put into an image and so that alternative ways of authentication are developed.

In this direction, future work includes analyzing the traces left by the diffusion purification process [37] to check whether the use of such a counter-forensic approach can be detected or not. Also, it would be interesting to analyze the robustness of the different methods to such kind of counter-forensic methods.

A.8 Extra: F1 scores for assessing the traces removal

The F1 scores obtained by all the tested methods are presented in Table A.4 for the Korus dataset, Table A.5 for the FAU dataset, Table A.6 for the COVERAGE dataset, and Table A.7 for the DSO-1 dataset.

A.9 Extra: Visual results

Figures A.5-A.10 present supplementary examples of the results obtained by the different forensics methods on the different versions of the very same image. The image used in each figure is specified in the caption. For all figures, we present the result of all considered forensic methods, even if they do not perform a good detection.

	CatNet	Bammey *et al.*	Choi	Comprint	MantraNet	Noiseprint	Shin	Splicebuster	TruFor	Zero
Original	0.0657	0.0825	0.1717	0.0817	0.1607	0.1389	0.1035	0.1682	0.3473	0.0043
CamTE	0.0420 (-0.0237)	0.0833 (0.0009)	0.0562 (-0.1155)	0.0710 (-0.0107)	0.1147 (-0.0460)	0.0987 (-0.0402)	0.1059 (0.0024)	0.1261 (-0.0422)	0.2246 (-0.1227)	0.0000 (-0.0043)
BM3D	0.0937 (0.0280)	0.0877 (0.0053)	0.0314 (-0.1403)	0.0634 (-0.0184)	0.1278 (-0.0329)	0.0939 (-0.0450)	0.0987 (-0.0048)	0.1274 (-0.0409)	0.2803 (-0.0670)	0.0000 (-0.0043)
Diff-CF	0.0324 (-0.0333)	0.0931 (0.0106)	0.0440 (-0.1278)	0.0398 (-0.0419)	0.0769 (-0.0838)	0.0669 (-0.0720)	0.0903 (-0.0131)	0.0720 (-0.0963)	0.1841 (-0.1631)	0.0046 (0.0003)
Diff-CFG	0.0775 (0.0118)	0.0912 (0.0087)	0.0077 (-0.1640)	0.0520 (-0.0298)	0.0980 (-0.0627)	0.0816 (-0.0573)	0.0910 (-0.0125)	0.0852 (-0.0831)	0.2398 (-0.1074)	0.0019 (-0.0024)

Table A.4: F1 scores obtained for all the tested methods on the Korus dataset [86, 87].

	CatNet	Bammey *et al.*	Choi	Comprint	MantraNet	Noiseprint	Shin	Splicebuster	TruFor	Zero
Original	0.3187	0.0827	0.3122	0.0566	0.0630	0.0888	0.1843	0.0471	0.4203	0.2958
CamTE	0.0159 (-0.3027)	0.0909 (0.0082)	0.1496 (-0.1626)	0.0376 (-0.0190)	0.0787 (0.0157)	0.0806 (-0.0082)	0.1639 (-0.0204)	0.0517 (0.0045)	0.1008 (-0.3196)	0.1089 (-0.1870)
BM3D	0.0741 (-0.2446)	0.0911 (0.0084)	0.0722 (-0.2399)	0.0237 (-0.0328)	0.0690 (0.0060)	0.0606 (-0.0282)	0.1298 (-0.0545)	0.0460 (-0.0012)	0.1287 (-0.2917)	0.1012 (-0.1946)
Diff-CF	0.0099 (-0.3088)	0.0973 (0.0146)	0.0781 (-0.2341)	0.0294 (-0.0271)	0.0643 (0.0013)	0.0383 (-0.0505)	0.1061 (-0.0781)	0.0238 (-0.0233)	0.0935 (-0.3269)	0.0027 (-0.2931)
Diff-CFG	0.0301 (-0.2886)	0.0953 (0.0126)	0.0318 (-0.2804)	0.0275 (-0.0291)	0.0630 (-0.0000)	0.0544 (-0.0344)	0.1109 (-0.0733)	0.0450 (-0.0021)	0.1051 (-0.3153)	0.0016 (-0.2942)

Table A.5: F1 scores obtained for all the tested methods on FAU dataset [33].

	CatNet	Bammey *et al.*	Choi	Comprint	MantraNet	Noiseprint	Shin	Splicebuster	TruFor	Zero
Original	0.2971	0.1585	0.0184	0.1508	0.2948	0.1512	0.1948	0.0759	0.4864	0.1890
CameraTE	0.1651 (-0.1320)	0.1593 (0.0008)	0.0128 (-0.0056)	0.1263 (-0.0245)	0.1287 (-0.1661)	0.1360 (-0.0152)	0.1846 (-0.0102)	0.0661 (-0.0098)	0.3220 (-0.1644)	0.1830 (-0.0060)
BM3D	0.2934 (-0.0037)	0.1602 (0.0018)	0.0065 (-0.0120)	0.1118 (-0.0390)	0.1485 (-0.1463)	0.1379 (-0.0133)	0.1870 (-0.0079)	0.0613 (-0.0145)	0.3879 (-0.0985)	0.1830 (-0.0060)
Diff-CF	0.1797 (-0.1173)	0.1652 (0.0067)	0.0103 (-0.0082)	0.1222 (-0.0286)	0.0961 (-0.1987)	0.1404 (-0.0109)	0.1867 (-0.0082)	0.0692 (-0.0067)	0.3377 (-0.1488)	0.0000 (-0.1890)
Diff-CFG	0.2253 (-0.0717)	0.1571 (-0.0014)	0.0019 (-0.0166)	0.1136 (-0.0373)	0.1248 (-0.1700)	0.1510 (-0.0002)	0.1859 (-0.0090)	0.0609 (-0.0149)	0.3374 (-0.1490)	0.0000 (-0.1890)

Table A.6: F1 scores for all tested methods on the COVERAGE dataset [157].

	CatNet	Bammey *et al.*	Choi	Comprint	MantraNet	Noiseprint	Shin	Splicebuster	TruFor	Zero
Original	0.0300	0.6876	0.0638	0.0341	0.0853	0.0802	0.4881	0.0600	0.0655	0.0028
CameraTE	0.0069 (-0.0231)	0.6604 (-0.0272)	0.0490 (-0.0148)	0.0522 (0.0181)	0.2487 (0.1634)	0.1317 (0.0515)	0.5070 (0.0189)	0.0868 (0.0269)	0.2536 (0.1881)	0.0009 (-0.0018)
BM3D	0.0114 (-0.0186)	0.6264 (-0.0612)	0.0156 (-0.0482)	0.0575 (0.0234)	0.4571 (0.3718)	0.1312 (0.0509)	0.5161 (0.0280)	0.0909 (0.0309)	0.3338 (0.2683)	0.0020 (-0.0007)
Diff-CF	0.0035 (-0.0265)	0.5874 (-0.1002)	0.3084 (0.2446)	0.0500 (0.0159)	0.1431 (0.0577)	0.0898 (0.0095)	0.5182 (0.0301)	0.0604 (0.0004)	0.3661 (0.3006)	0.0025 (-0.0003)
Diff-CFG	0.0081 (-0.0219)	0.5908 (-0.0969)	0.1298 (0.0660)	0.0652 (0.0312)	0.1785 (0.0931)	0.1177 (0.0375)	0.5180 (0.0299)	0.0681 (0.0081)	0.4155 (0.3500)	0.0043 (0.0015)

Table A.7: F1 scores for all tested methods on the DSO-1 dataset [43].

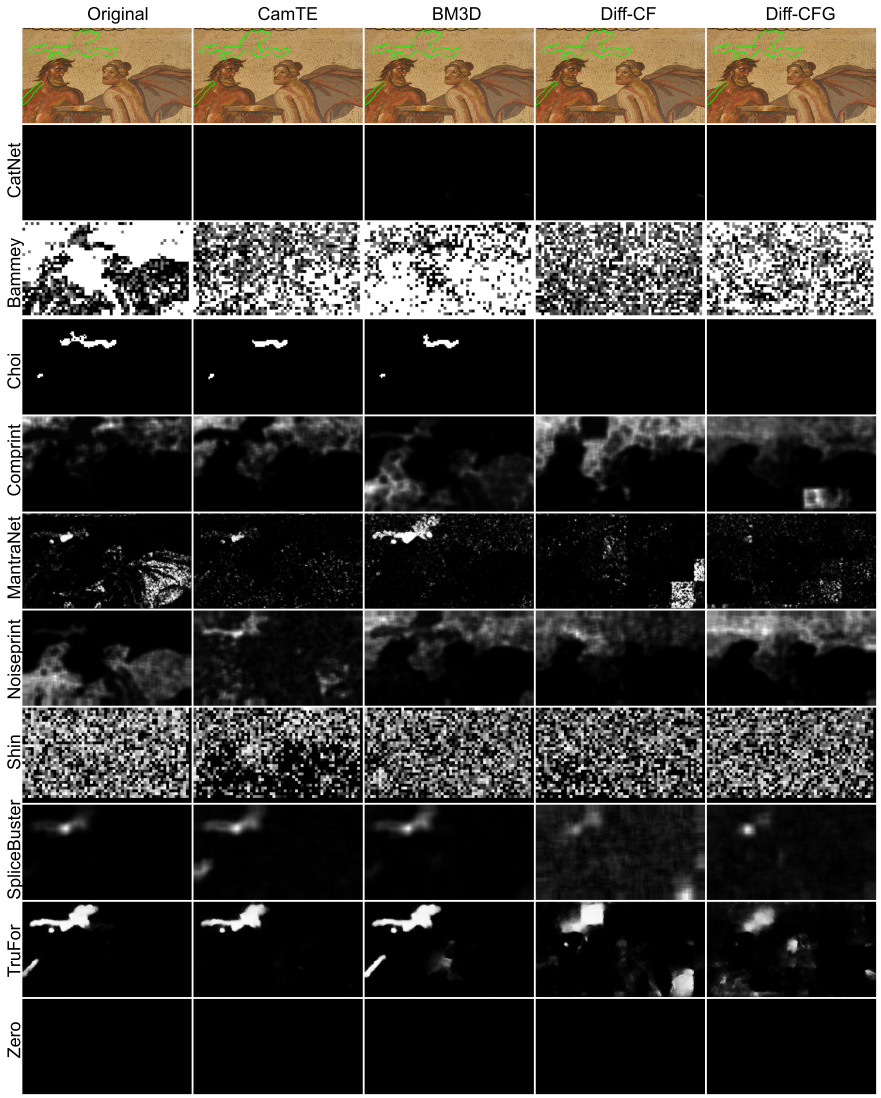


Figure A.5: Results obtained by all considered forensics methods on the different versions of image `r7710a7fat` from the Korus dataset [86, 87]. This is the same image shown in Figure 1 in the paper, but with the results for all methods, even if they do not detect anything in the original image.

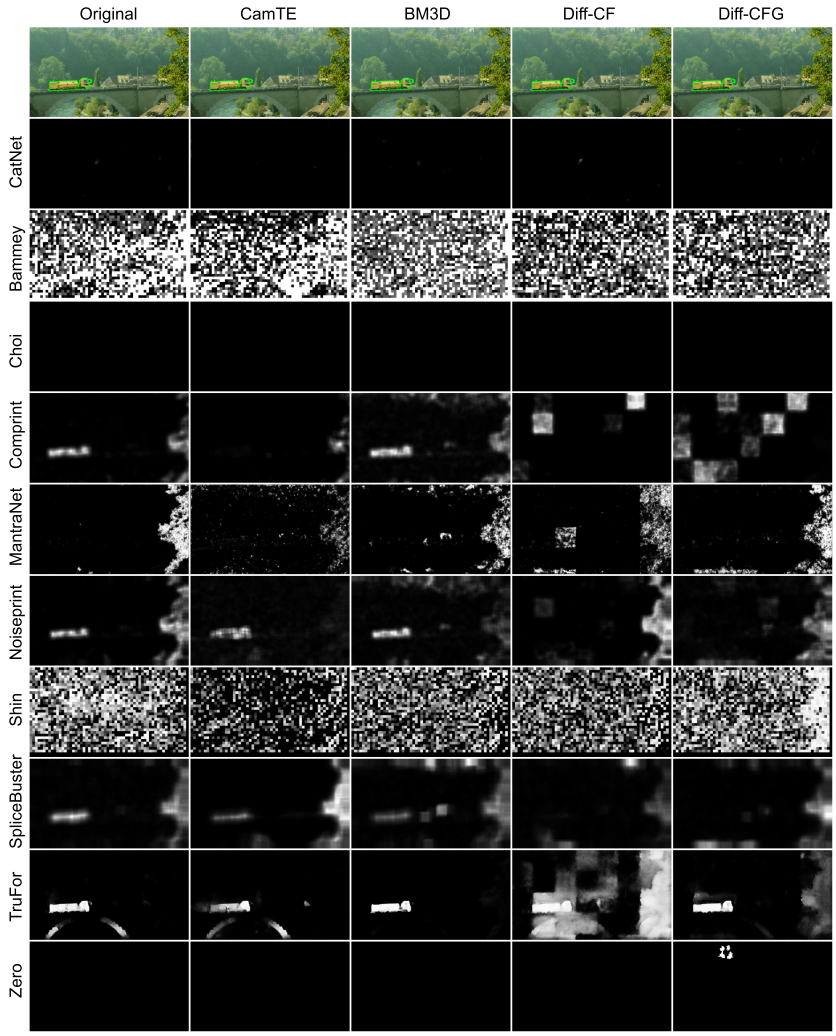


Figure A.6: Results obtained by different forensics methods on the different versions of image `r02354285t` from the Korus dataset [86,87]. In this image we observe that Noiseprintt, Splicebuster and TruFor give fairly correct detections in the original forged image. Splicebuster and Noiseprint present degraded detections once BM3D or CamTE are applied. However, the forgery is not even highlighted when *Diff-CF* or *Diff-CFG* are used as counter-forensics attacks. TruFor is more robust to such attacks. Still, their results degrade after *Diff-CF*

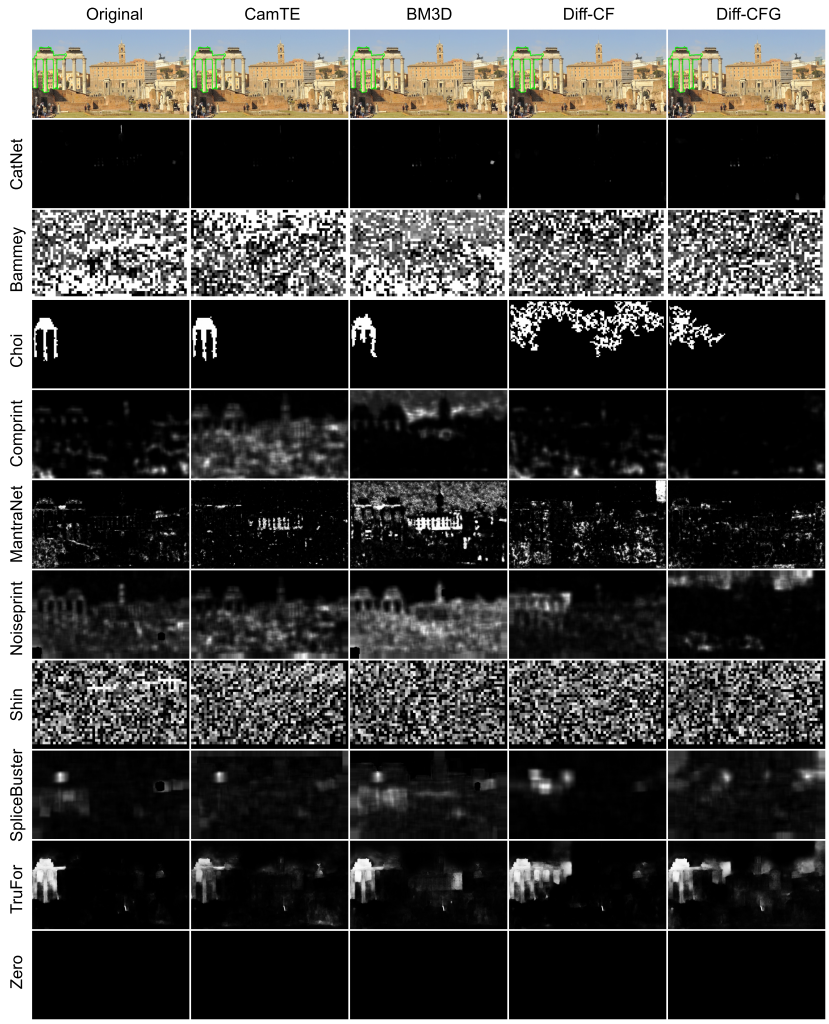


Figure A.7: Results obtained by different forensics methods on the different versions of image `rbc87504ct` from the Korus dataset [86, 87]. In this image, we observe that only Choi and TruFor detect the forgery in the original image. Choi still detects the forgery once BM3D and CamTE are applied, but is unable to detect it once *Diff-CF* or *Diff-CFG*

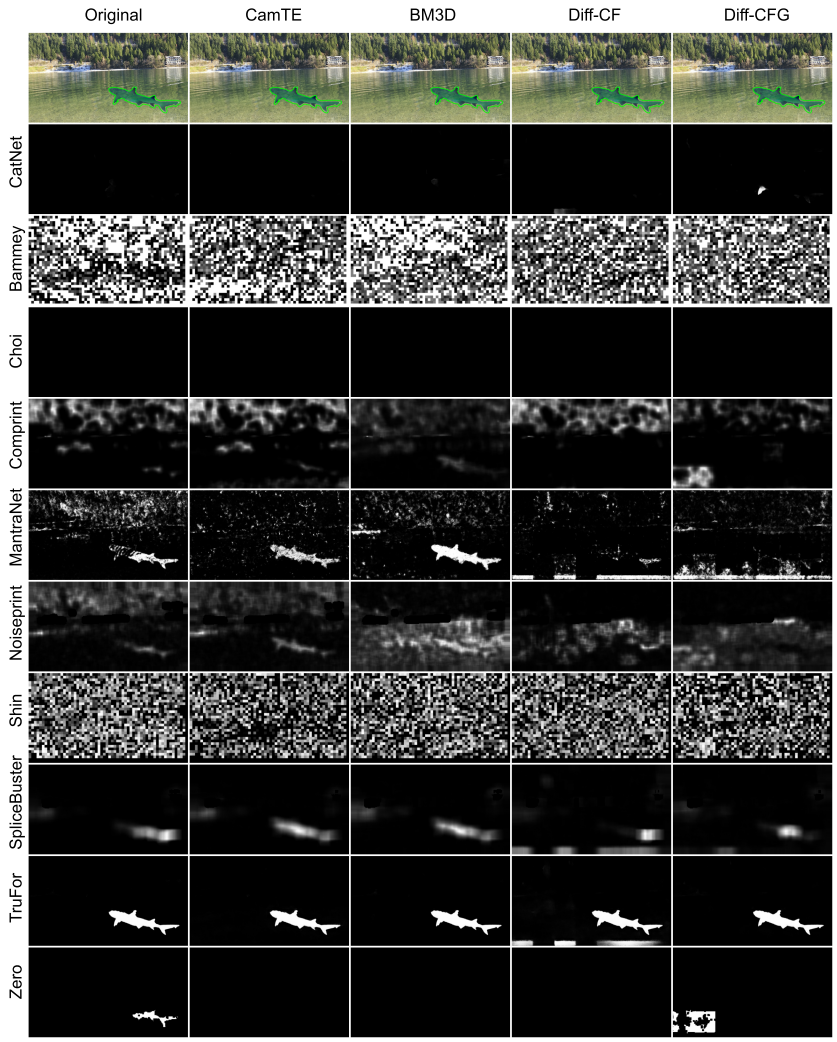


Figure A.8: Results obtained by different forensics methods on the different versions of image `re6d1dc1et` from the Korus dataset [86, 87].

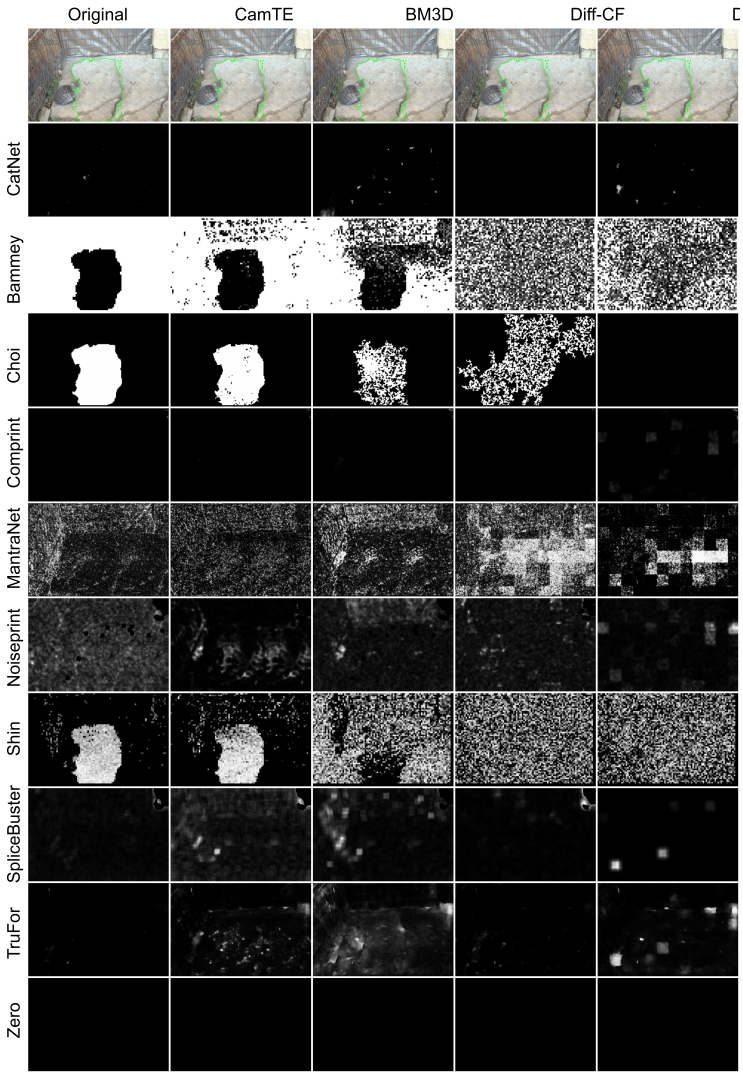


Figure A.9: Results obtained by different forensics methods on the different versions of image `lone_cat_copy` from the FAU dataset [33].

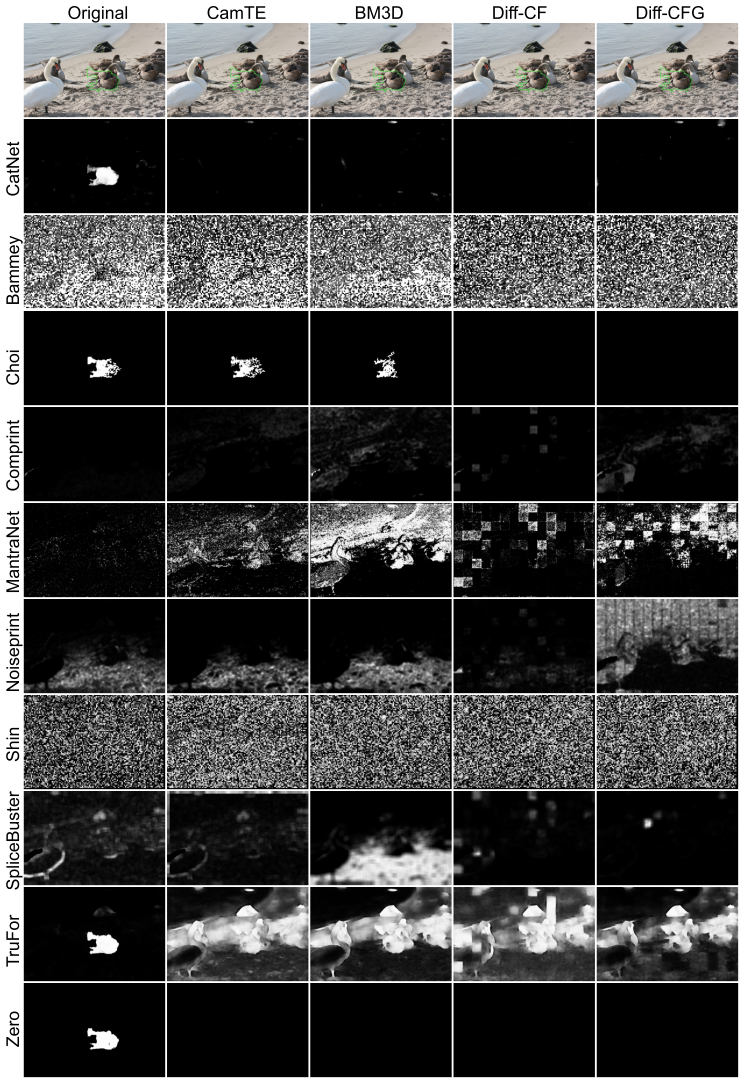


Figure A.10: Results obtained by different forensics methods on the different versions of image `swan_copy` from the FAU dataset [33].

This page was intentionally left blank.

Appendix B

Datasets and Metrics

B.1 Datasets

This section presents the three primary anomaly detection datasets utilized in this thesis: MVTec-AD [13], VisA [180], and Bean Tech (BTAD) [106]. Additionally, in certain instances, datasets from other domains are employed to showcase the generalization capabilities and robustness of the proposed methods. These supplementary datasets will be introduced as they become relevant throughout the discussion.

B.1.1 MVTec-AD dataset

The MVTec Anomaly Detection (MVTec-AD) [13] dataset is a comprehensive benchmark designed for evaluating anomaly detection algorithms in industrial applications. Widely recognized and extensively used in the anomaly detection community, it serves as a standard dataset for evaluation and comparison. The dataset includes 15 different categories, each representing a distinct type of object or material commonly found in industrial settings, such as screws, hazelnuts, and bottles, as well as materials like wood and textiles. Each category contains images of both normal (defect-free) and anomalous (defective) samples.

The anomalies in the MVTec-AD dataset are designed to reflect real-world defects that might occur in manufacturing processes. These defects vary in nature and complexity, including scratches, dents, contaminations, and structural deformations, ensuring that the dataset covers a broad spectrum of potential issues encountered in practice. The images in the dataset are high-resolution, providing detailed visual information crucial for accurate defect detection, allowing for the precise identification and localization of small and subtle anomalies.

Each image is accompanied by ground truth annotations indicating the presence and location of anomalies. These annotations are essential for evaluating the performance of anomaly detection algorithms, providing a benchmark for comparison. The dataset is organized in a structured format, where each category is divided into separate folders for normal and anomalous images, making it easy to use for training and evaluating machine learning models.

MVTec-AD serves as a standard benchmark for the research community,

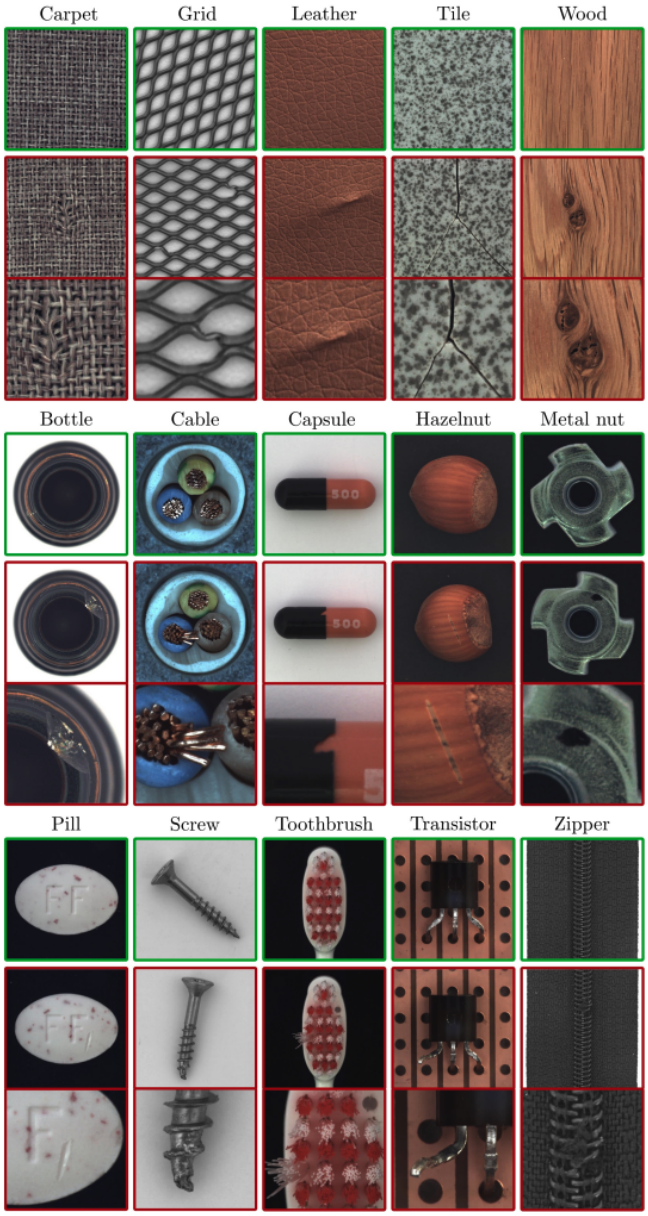


Figure B.1: Image adapted from [13]. Example images for all five textures and ten object categories of the MVTec AD dataset. For each category, the top row shows an anomaly-free image. The middle row shows an anomalous example for which, in the bottom row, a close-up view that highlights the anomalous region is given.

allowing for the comparison of different anomaly detection approaches under consistent conditions. The dataset is primarily used to develop and test anomaly detection, defect localization, and quality control algorithms.

Example images from this dataset are shown in Figure B.1.

B.1.2 VisA dataset

The VisA dataset [180] is a comprehensive dataset also designed specifically for anomaly detection and segmentation tasks in industrial settings. It offers a diverse collection of images that includes a broader variety of anomalies, such as scratches, dents, and other defects commonly found in industrial products. This diversity ensures that the dataset accurately represents the wide range of conditions encountered in real-world industrial scenarios.

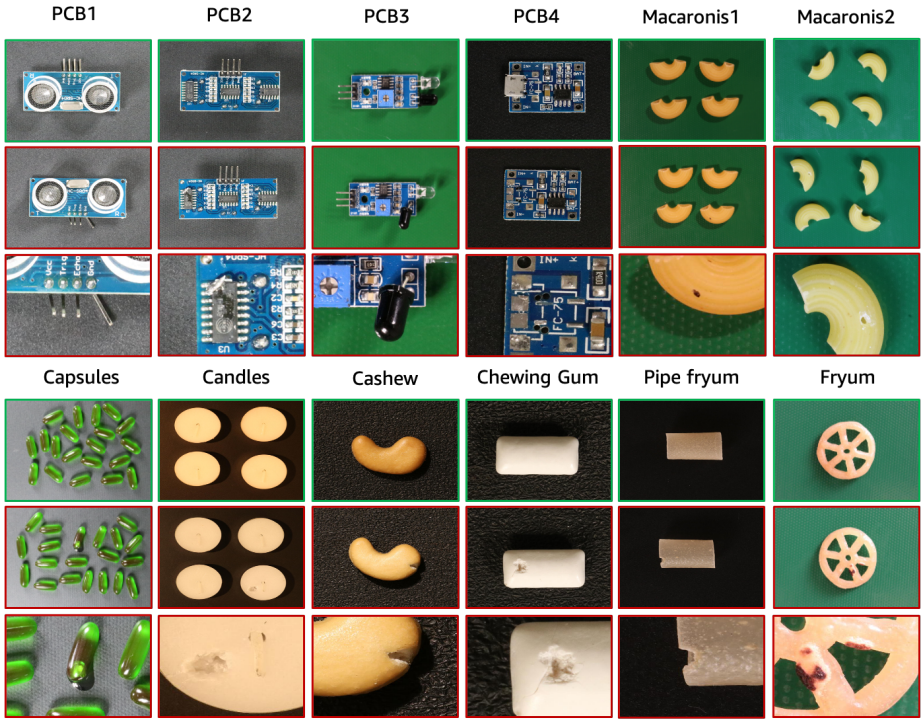


Figure B.2: Image taken from [180]. Samples of VisA datasets. First row: normal images; Second row: anomalous images; Third row: anomalies viewed by zooming in.

The images in the VisA dataset are high-resolution, providing detailed visual information essential for detecting subtle anomalies. The VisA dataset is organized into 12 categories based on different industrial products, each containing images of normal (defect-free) and anomalous items. Examples images are shown in Figure B.2.

An important difference compared to the MVTec-AD dataset is that some VisA categories contain more than one object (of the same type) in a single

image.

Each image in the VisA dataset is also meticulously annotated with the locations and types of anomalies, serving as ground truth for evaluating the performance of anomaly detection algorithms. The VisA dataset more recent than MVTec, and it is becoming a prominent benchmark for evaluating anomaly detection models.

B.1.3 Bean Tech dataset

The Bean Tech anomaly dataset [106] focused on providing a rich collection of images specifically curated to challenge and refine anomaly detection algorithms. These images are sourced from various stages of industrial processes and encompass anomalies, including surface defects, structural irregularities, and other inconsistencies that are critical for maintaining quality control in industrial manufacturing. It contains three classes corresponding to different products. Products 1, 2, and 3 have 400, 1000, and 399 train images, respectively, and image resolutions vary from 600 to 1600 pixels. Example images are shown in Figure B.3.

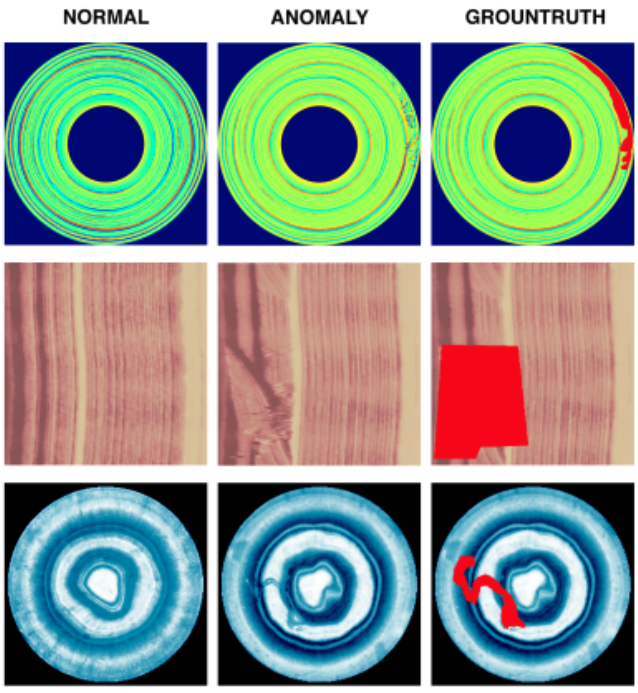


Figure B.3: Image adapted from [106]. The three products of BTAD dataset. First column shows an example of normal images, second column shows anomalous images, third column shows the anomalous image with pixel-level ground truth labels.

B.2 Metrics

This section presents the main metrics used in this thesis.

B.2.1 AUROC

The Area Under the Receiver Operating Characteristic Curve (AUROC) is a performance metric used to evaluate the effectiveness of binary classification models. It measures the ability of a model to distinguish between two classes, typically referred to as the positive class and the negative class.

The ROC curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve plots two quantities:

True Positive Rate (TPR) or Sensitivity or Recall: The ratio of correctly predicted positive observations to all actual positives,

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{B.1}$$

False Positive Rate (FPR): The ratio of incorrectly predicted positive observations to all actual negatives,

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}. \tag{B.2}$$

To construct the ROC curve, the model's predicted probabilities for the positive class are sorted in descending order. Starting with the highest probability, the threshold is adjusted downward, and for each threshold value, the TPR and FPR are calculated. These pairs of (FPR, TPR) values are plotted to form the ROC curve.

The AUROC is the integral of the ROC curve, providing a single scalar value to summarize the overall performance of the classifier. The AUROC value ranges from 0 to 1. AUROC=1 indicates a perfect model that perfectly distinguishes between the positive and negative classes. AUROC = 0.5 indicates a model with no discriminative ability, equivalent to random guessing. And AUROC $<$ 0.5 suggests that the model performs worse than random guessing.

This metric evaluates a classifier's performance across all possible classification thresholds, making it useful for comparing models without committing to a specific decision threshold. Moreover, the AUROC interpretation is intuitive and easy to understand: it can be seen as the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance.

However, as it considers all pixels of all images in the dataset at once, it can suffer from the class imbalance problem. For instance, if a large anomaly in one image is correctly detected while a small anomaly in another image is completely missed, the AUROC might still return a very high value. This occurs because the false negative from the small anomaly has minimal weight compared to the correctly detected large anomaly, thereby masking the model's poor performance on the smaller yet critical anomaly. This happens because

AUROC implicitly gives equal importance to the true positive rate and false positive rate. In some applications, like anomaly detection, these rates are not equally important. In addition, AUROC can sometimes give an overly optimistic view of model performance, particularly when the model performs well on easy-to-classify instances but poorly on more difficult ones. The AUROC might be high even if the model fails to identify critical instances accurately. These disadvantages encouraged us to also consider a different metric, AUPRO, which is explained in the following section. Nevertheless, it is important to evaluate this metric because it is widely used and accepted in the field, providing a standard benchmark for comparing the performance of different models.

B.2.2 AUPRO

The Area Under the Per-Region Overlap ($AUPRO$) provides a more granular assessment compared to traditional metrics, by focusing on the overlap of predicted and ground truth regions on a per-region basis. $AUPRO$ measures how well a segmentation algorithm identifies and overlaps with regions of interest (e.g., anomalies or objects) within an image.

Similarly to the $AUROC$ computation, this metric also reports the area under a curve—in this case, the PRO curve. This curve is created by evaluating the performance and varying the threshold. The main difference with the ROC curve is that instead of computing the metric's value based on the value of all pixels, the computation is done per region. Therefore, it assigns the same weight to different-sized regions.

B.2.3 mIoU

As mentioned before, this thesis not only focuses on generating anomaly maps, but also present several methods for computing an automatic segmentation of the anomalies. Therefore, a metric for assessing the anomaly masks generated is needed. For such purpose, the $mIoU$ metric is proposed.

Before presenting the *mean Intersection over Union (mIoU)* metric, let's start by presenting IoU. Intersection over Union (IoU), also known as the Jaccard Index, is a fundamental metric for evaluating the accuracy of object detection and segmentation models. It calculates the overlap between a single class's predicted and ground truth regions. It is computed as

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}, \tag{B.3}$$

where "Area of Overlap" is the area where the predicted mask and the ground truth mask intersect, and "Area of Union" is the total area covered by both the predicted mask and the ground truth mask.

On the other hand, $mIoU$ computes the average value of the $IoUs$ for all images:

$$mIoU = \frac{1}{N}\sum_{i=1}^{N} IoU_i \tag{B.4}$$

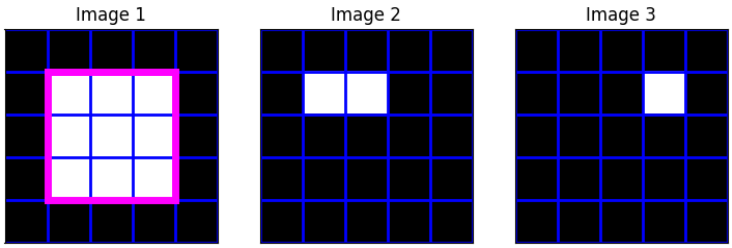
where N is the total number of images, and IoU_i is the IoU computed on image i.

The key advantage of $mIoU$ with respect to IoU is that, by averaging the per-image IoU values, $mIoU$ assigns the same weight to the detection performance of all anomalies, regardless of their size. This ensures that the detection of smaller anomalies contributes equally to the overall metric. In this way, $mIoU$ better reflects the model's ability to detect all anomalies, not just the prominent ones. This is crucial in applications like anomaly detection, where identifying every anomaly, regardless of size, is important for overall performance.

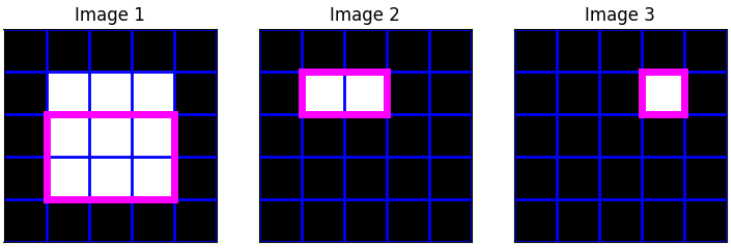
Consider, for example, the case of an image dataset with both large and small anomalies. Suppose a model perfectly detects a large anomaly (resulting in a high IoU for that instance) but misses several small anomalies (resulting in low $IoUs$ for those instances). In that case, the overall IoU might still be high because the large anomaly's IoU dominates the metric. However, the $mIoU$ would average these IoU values, reflecting the poor performance on the small anomalies and providing a lower overall score, which more accurately represents the model's inability to detect all anomalies. This example is depicted in Figure B.4, where a dataset compound of three images is presented. The underlying image represents the ground truth, where the white color is the anomaly. The first image has a big anomaly, and the other two images have one small anomaly each. Figure B.4(a) and B.4(b) show in magenta color, example outputs of some models ("Model 1", and "Model 2"). It is clear that "Model 2" behaves much better than "Model 1", as it is able to detect all three anomalies, although the borders in the first image are not great. In this case, both models present the same IoU: $IoU = 9 \ / \ 12 = 0.75$. But when computing the $mIoU$ we obtain the following results:

- Model 1: $mIoU = \frac{1}{3}(1 + 0 + 0) = 0.33$
- Model 2: $mIoU = \frac{1}{3}(0.66 + 1 + 1) = 0.89$,

making evident that "Model 2" is far better than "Model 1".



(a) Example output of "Model 1" over three images.



(b) Example output of "Model 2" over three images.

Figure B.4: Toy dataset with three images. Actual anomalies are represented in white, and normal parts of the image in black. The magenta rectangles indicate the detections made by two different example models. This example shows "Model 1" perfectly detecting a big anomaly but completely failing with the other two small anomalies, and "Model 2" detecting all three anomalies, although not getting an exact segmentation for the first one.

Bibliography

[1] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomalib: A deep learning library for anomaly detection, 2022.

[2] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 622–637. Springer, 2019.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[4] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[5] Lars Lien Ankile, Anna Midgley, and Sebastian Weisshaar. Denoising diffusion probabilistic models as a defense against adversarial attacks. *ArXiv*, abs/2301.06871:null, 2023.

[6] Lynton Ardizzone, Till Bungert, Felix Draxler, Ullrich Köthe, Jakob Kruse, Robert Schmier, and Peter Sorrenson. Framework for Easily Invertible Architectures (FrEIA), 2018-2022.

[7] Coloma Ballester, Vicent Caselles, and Pascal Monasse. The tree of shapes of an image. *ESAIM: Control, Optimisation and Calculus of Variations*, 9:1–18, 2003.

[8] Quentin Bammey, Rafael Grompone von Gioi, and Jean-Michel Morel. An adaptive neural network for unsupervised mosaic consistency analysis in image forensics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[9] Quentin Bammey, Rafael Grompone von Gioi, and Jean-Michel Morel. Image Forgeries Detection through Mosaic Analysis: the Intermediate Values Algorithm. *Image Processing On Line*, 11:317–343, 2021.

[10] Daniele Baracchi, Dasara Shullani, Massimo Iuliani, Damiano Giani, and Alessandro Piva. Camera obscura: Exploiting in-camera processing for image counter forensics. *Forensic Science International: Digital Investigation*, 38:301213, 2021.

[11] Mauro Barni, Matthew C Stamm, and Benedetta Tondi. Adversarial multimedia forensics: Overview and challenges ahead. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 962–966. IEEE, 2018.

[12] Cosmin I Bercea, Benedikt Wiestler, Daniel Rueckert, and Julia A Schnabel. Towards universal unsupervised anomaly detection in medical imaging. *arXiv preprint arXiv:2401.10637*, 2024.

[13] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9592–9600, 2019.

[14] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *arXiv preprint arXiv:1807.02011*, 2018.

[15] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018.

[16] Tsachi Blau, Roy Ganz, Bahjat Kawar, Alex Bronstein, and Michael Elad. Threat model-agnostic adversarial defense using diffusion models, 2022.

[17] Rainer Böhme and Matthias Kirchner. Counter-forensics: Attacking image forensics. In *Digital image forensics*, pages 327–366. Springer, 2013.

[18] Tobias Böttger and Markus Ulrich. Real-time texture error detection on textured surfaces with compressed sensing. *Pattern Recognition and Image Analysis*, 26(1):88–94, 2016.

[19] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 60–65, Washington, DC, USA, 2005. IEEE Computer Society.

[20] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Nonlocal image and movie denoising. *International journal of computer vision*, 76:123–139, 2008.

[21] Mateusz Buda, Ashirbani Saha, and Maciej A Mazurowski. Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm. *Computers in biology and medicine*, 109:218–225, 2019.

[22] F. Cao, J.L. Lisani, J.-M. Morel, P. Musé, and F. Sur. *A theory of shape identification*, volume 1948 of *Lecture Notes in Mathematics*. Springer, 2008.

[23] Frédéric Cao, Julie Delon, Agnès Desolneux, Pablo Musé, and Frédéric Sur. A unified framework for detecting groups and application to shape recognition. *Journal of Mathematical Imaging and Vision*, 27:91–119, 2007.

[24] Frédéric Cao, Pablo Musé, and Frédéric Sur. Extracting meaningful curves from images. *Journal of Mathematical Imaging and Vision*, 22:159–181, 2005.

[25] Chang Chen, Zhiwei Xiong, Xiaoming Liu, and Feng Wu. Camera trace erasing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[26] Chaofeng Chen and Jiadi Mo. IQA-PyTorch: Pytorch toolbox for image quality assessment. [Online]. Available: `https://github.com/chaofengc/IQA-PyTorch`, 2022.

[27] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[28] Yuanhong Chen, Yu Tian, Guansong Pang, and Gustavo Carneiro. Unsupervised anomaly detection with multi-scale interpolated gaussian descriptors. *arXiv preprint arXiv:2101.10043*, 2021.

[29] Zhipeng Chen, Benedetta Tondi, Xiaolong Li, Rongrong Ni, Yao Zhao, and Mauro Barni. A gradient-based pixel-domain attack against svm detection of global image manipulations. In *2017 IEEE workshop on information forensics and security (WIFS)*, pages 1–6. IEEE, 2017.

[30] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[31] Chang-Hee Choi, Jung-Ho Choi, and Heung-Kyu Lee. Cfa pattern identification of digital cameras using intermediate value counting. In *Proceedings of the Thirteenth ACM Multimedia Workshop on Multimedia and Security*, MM&Sec '11, page 21–26, New York, NY, USA, 2011. Association for Computing Machinery.

[32] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11472–11481, 2022.

[33] Vincent Christlein, Christian Riess, Johannes Jordan, Corinna Riess, and Elli Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on Information Forensics and Security*, 7(6):1841–1854, 2012.

[34] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020.

[35] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020.

[36] Pedro Comesana and Fernando Perez-Gonzalez. The optimal attack to histogram-based forensic detectors is simple (x). In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 137–142. IEEE, 2014.

[37] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models, 2022.

[38] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2015.

[39] Davide Cozzolino, Justus Thies, Andreas Rossler, Matthias Niessner, and Luisa Verdoliva. Spoc: Spoofing camera fingerprints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 990–1000, June 2021.

[40] Davide Cozzolino and Luisa Verdoliva. Noiseprint: A cnn-based camera model fingerprint. *IEEE Transactions on Information Forensics and Security*, 15:144–159, 2020.

[41] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[42] Axel Davy, Thibaud Ehret, Jean-Michel Morel, and Mauricio Delbracio. Reducing anomaly detection in images to detection in noise. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1058–1062. IEEE, 2018.

[43] Tiago José de Carvalho, Christian Riess, Elli Angelopoulou, Hélio Pedrini, and Anderson de Rezende Rocha. Exposing digital image forgeries by illumination color classification. *IEEE Transactions on Information Forensics and Security*, 8(7):1182–1194, 2013.

[44] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*, pages 475–489. Springer, 2021.

[45] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. Edge detection by Helmholtz principle. *Journal of mathematical imaging and vision*, 14(3):271–284, 2001.

[46] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. *From gestalt theory to image analysis: a probabilistic approach*, volume 34. Springer Science & Business Media, 2007.

[47] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[48] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

[49] Thanh-Toan Do, Ewa Kijak, Teddy Furon, and Laurent Amsaleg. Deluding image recognition in sift-based cbir systems. In *Proceedings of the 2nd ACM Workshop on Multimedia in forensics, Security and Intelligence*, pages 7–12, 2010.

[50] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[51] Bo Du and Liangpei Zhang. Random-selection-based anomaly detector for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 49(5):1578–1589, 2010.

[52] Thibaud Ehret, Axel Davy, Mauricio Delbracio, and Jean-Michel Morel. How to reduce anomaly detection in images to anomaly detection in noise. *Image Processing On Line*, 9:391–412, 2019.

[53] Thibaud Ehret, Axel Davy, Jean-Michel Morel, and Mauricio Delbracio. Image anomalies: A review and synthesis of detection methods. *Journal of Mathematical Imaging and Vision*, 61:710–743, 2019.

[54] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[55] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019.

[56] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021.

[57] Wei Fan, Kai Wang, François Cayre, and Zhang Xiong. Jpeg anti-forensics using non-parametric dct quantization noise estimation and natural image statistics. In *Proceedings of the first ACM workshop on Information hiding and multimedia security*, pages 117–122, 2013.

[58] Hany Farid. *Photo Forensics*. The MIT Press, 2016.

[59] William Feller. On the theory of stochastic processes, with particular reference to applications. In *Proceedings of the [First] Berkeley Symposium on Mathematical Statistics and Probability*, 1949.

[60] Marco Fontani and Mauro Barni. Hiding traces of median filtering in digital images. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1239–1243. IEEE, 2012.

[61] Marina Gardella, Pablo Musé, Jean-Michel Morel, and Miguel Colom. Noisesniffer: a fully automatic image forgery detector based on noise analysis. In *2021 IEEE International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6. IEEE, 2021.

[62] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.

[63] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[64] Diego Gragnaniello, Francesco Marra, Giovanni Poggi, and Luisa Verdoliva. Analysis of adversarial attacks against cnn-based image forgery detectors. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 967–971. IEEE, 2018.

[65] Bénédicte Grosjean and Lionel Moisan. A-contrario detectability of spots in textured backgrounds. *Journal of Mathematical Imaging and Vision*, 33:313–337, 2009.

[66] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 98–107, 2022.

[67] Fabrizio Guillaro, Davide Cozzolino, Avneesh Sud, Nicholas Dufour, and Luisa Verdoliva. Trufor: Leveraging all-round clues for trustworthy image forgery detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20606–20615, June 2023.

[68] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. , 2007.

[69] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

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

[71] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[72] Xun Huang, Chengyao Shen, Xavier Boix, and Qi Zhao. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 262–270, 2015.

[73] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[74] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[75] Eric Jang. Normalizing flows tutorial, part 1: Distributions and determinants. `https://blog.evjang.com/2018/01/nf1.html`. Created: 2018-01-17.

[76] Iwan Jensen and Anthony J Guttmann. Statistics of lattice animals (polyominoes) and polygons. *Journal of Physics A: Mathematical and General*, 33(29):L257, 2000.

[77] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2083–2090, 2013.

[78] Dongkyu Kim, Han-Ul Jang, Seung-Min Mun, Sunghee Choi, and Heung-Kyu Lee. Median filtered image restoration and anti-forensics using adversarial networks. *IEEE Signal Processing Letters*, 25(2):278–282, 2017.

[79] Jin-Hwa Kim, Do-Hyeong Kim, Saehoon Yi, and Taehoon Lee. Semi-orthogonal embedding for efficient unsupervised anomaly segmentation. *arXiv preprint arXiv:2105.14737*, 2021.

[80] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[81] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations*, 2014.

[82] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[83] Matthias Kirchner and Rainer Bohme. Hiding traces of resampling in digital images. *IEEE Transactions on Information Forensics and Security*, 3(4):582–592, 2008.

[84] Matthias Kirchner and Jessica Fridrich. On detection of median filtering in digital images. In Nasir D. Memon, Jana Dittmann, Adnan M. Alattar, and Edward J. Delp III, editors, *Media Forensics and Security II*, volume 7541, page 754110. International Society for Optics and Photonics, SPIE, 2010.

[85] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

[86] P. Korus and J. Huang. Evaluation of random field models in multi-modal unsupervised tampering localization. In *Proc. of IEEE Int. Workshop on Inf. Forensics and Security*, 2016.

[87] P. Korus and J. Huang. Multi-scale analysis strategies in prnu-based tampering localization. *IEEE Trans. on Information Forensics & Security*, 2017.

[88] Myung-Joon Kwon, Seung-Hun Nam, In-Jae Yu, Heung-Kyu Lee, and Changick Kim. Learning jpeg compression artifacts for image manipulation detection and localization. *International Journal of Computer Vision*, 130(8):1875–1895, August 2022.

[89] Myung-Joon Kwon, In-Jae Yu, Seung-Hun Nam, and Heung-Kyu Lee. Cat-net: Compression artifact tracing network for detection and localization of image splicing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 375–384, 2021.

[90] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

[91] Sylvie Le Hégarat-Mascle, Emanuel Aldea, and Jennifer Vandoni. Efficient evaluation of the number of false alarm criterion. *EURASIP Journal on Image and Video Processing*, 2019(1):1–15, 2019.

[92] Marc Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2:175–213, 2012. `https://doi.org/10.5201/ipol.2012.l-bm3d`.

[93] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[94] M. Lee and Dongwoo Kim. Robust evaluation of diffusion-based adversarial purification. *ArXiv*, abs/2303.09051:null, 2023.

[95] Sungwook Lee, Seunghyun Lee, and Byung Cheol Song. Cfa: Coupled-hypersphere-based feature adaptation for target-oriented anomaly localization. *arXiv preprint arXiv:2206.04325*, 2022.

[96] José Lezama, Jean-Michel Morel, Gregory Randall, and Rafael Grompone Von Gioi. A contrario 2d point alignment detection. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):499–512, 2014.

[97] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9664–9674, 2021.

[98] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Improved multiscale vision transformers for classification and detection. *arXiv preprint arXiv:2112.01526*, 2021.

[99] W. Liu, D. Lian W. Luo, and S. Gao. Future frame prediction for anomaly detection – a new baseline. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[100] Philipp Liznerski, Lukas Ruff, Robert A Vandermeulen, Billy Joe Franks, Marius Kloft, and Klaus-Robert Müller. Explainable deep one-class classification. *arXiv preprint arXiv:2007.01760*, 2020.

[101] David Lowe. *Perceptual organization and visual recognition*, volume 5. Springer Science & Business Media, 2012.

[102] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845, June 2021.

[103] Hannes Mareen, Dante Vanden Bussche, Fabrizio Guillaro, Davide Cozzolino, Glenn Van Wallendael, Peter Lambert, and Luisa Verdoliva. Comprint: Image forgery detection and localization using compression fingerprints. *arXiv preprint arXiv:2210.02227*, 2022.

[104] Francesco Marra, Giovanni Poggi, Fabio Roli, Carlo Sansone, and Luisa Verdoliva. Counter-forensics in machine learning based forgery detection. In *Media Watermarking, Security, and Forensics 2015*, volume 9409, page 94090L. International Society for Optics and Photonics, 2015.

[105] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

[106] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, pages 01–06, 2021.

[107] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708, 2012.

[108] Anish Mittal, Rajiv Soundararajan, and Alan Conrad Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20:209–212, 2013.

[109] Pascal Monasse and Frederic Guichard. Fast computation of a contrast-invariant image representation. *IEEE Transactions on Image Processing*, 9(5):860–872, 2000.

[110] Mary M Moya, Mark W Koch, and Larry D Hostetler. One-class classifier networks for target recognition applications. *NASA STI/Recon Technical Report N*, 93:24043, 1993.

[111] Pablo Musé, Frédéric Sur, Frédéric Cao, Yann Gousseau, and Jean-Michel Morel. An a contrario decision method for shape element recognition. *International Journal of Computer Vision*, 69:295–315, 2006.

[112] Paolo Napoletano, Flavio Piccoli, and Raimondo Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors*, 18(1):209, 2018.

[113] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning (ICML)*, 2022.

[114] Tina Nikoukhah, Jérémy Anger, Thibaud Ehret, Miguel Colom, Jean-Michel Morel, and Rafael Grompone von Gioi. Jpeg grid detection based on the number of dct zeros and its application to automatic and localized forgery detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 110–118, 2019.

[115] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[116] Jonathan Pirnay and Keng Chai. Inpainting transformer for anomaly detection. *arXiv preprint arXiv:2104.13897*, 2021.

[117] M Ali Qureshi and M Deriche. A review on copy move image forgery detection techniques. In *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*, pages 1–5. IEEE, 2014.

[118] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538, 2015.

[119] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[120] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

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

[122] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1907–1916, 2021.

[123] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Fully convolutional cross-scale-flows for image-based defect detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1088–1097, 2022.

[124] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[125] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[126] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[127] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, 54:30–44, 2019.

[128] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017.

[129] Eli Schwartz, Assaf Arbelle, Leonid Karlinsky, Sivan Harary, Florian Scheidegger, Sivan Doveh, and Raja Giryes. Maeday: Mae for few-and zero-shot anomaly-detection. *Computer Vision and Image Understanding*, 241:103958, 2024.

[130] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, Inc., USA, 1983.

[131] Hyun Jun Shin, Jong Ju Jeon, and Il Kyu Eom. Color filter array pattern identification using variance of color difference image. *Journal of Electronic Imaging*, 26(4):043015, 2017.

[132] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[133] Kulbir Singh, Ankush Kansal, and Gurinder Singh. An improved median filtering anti-forensics with better image quality and forensic undetectability. *Multidimensional Systems and Signal Processing*, 30(4):1951–1974, 2019.

[134] John Skilling. The eigenvalues of mega-dimensional matrices. *Maximum Entropy and Bayesian Methods: Cambridge, England, 1988*, pages 455–466, 1989.

[135] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[136] Kihyuk Sohn, Chun-Liang Li, Jinsung Yoon, Minho Jin, and Tomas Pfister. Learning and evaluating representations for deep one-class classification. *arXiv preprint arXiv:2011.02578*, 2020.

[137] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[138] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[139] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[140] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[141] Jiachen Sun, Weili Nie, Zhiding Yu, Z. Mao, and Chaowei Xiao. Pointdp: Diffusion-driven purification against adversarial attacks on 3d point cloud recognition. *ArXiv*, abs/2208.09801:null, 2022.

[142] Matías Tailanian, Marina Gardella, Álvaro Pardo, and Pablo Musé. Diffusion models meet image counter-forensics. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3925–3935, 2024.

[143] Matías Tailanián, Pablo Musé, and Álvaro Pardo. A multi-scale a contrario method for unsupervised image anomaly detection. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 179–184. IEEE, 2021.

[144] Matías Tailanian, Pablo Musé, and Álvaro Pardo. A contrario multi-scale anomaly detection method for industrial quality inspection. In *Deep Learning Applications, Volume 4*, pages 193–216. Springer, 2022.

[145] Matías Tailanian, Álvaro Pardo, and Pablo Musé. U-flow: A u-shaped normalizing flow for anomaly detection with unsupervised threshold. *Journal of Mathematical Imaging and Vision*, May 2024.

[146] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

[147] Chin-Chia Tsai, Tsung-Hsuan Wu, and Shang-Hong Lai. Multi-scale patch-based representation learning for image anomaly detection and segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3992–4000, 2022.

[148] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[149] Rafael Grompone von Gioi, Charles Hessel, Tristan Dagobert, Jean-Michel Morel, and Carlo de Franchis. Ground visibility in satellite optical time series based on a contrario local image matching. *Image Processing On Line*, 11:212–233, 2021.

[150] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2008.

[151] Rafael Grompone Von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. On straight line segment detection. *Journal of Mathematical Imaging and Vision*, 32(3):313–347, 2008.

[152] Qian Wan, Yunkang Cao, Liang Gao, Weiming Shen, and Xinyu Li. Position encoding enhanced feature mapping for image anomaly detection. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 876–881, 2022.

[153] Guodong Wang, Shumin Han, Errui Ding, and Di Huang. Student-teacher feature pyramid matching for unsupervised anomaly detection. *arXiv preprint arXiv:2103.04257*, 2021.

[154] Jinyi Wang, Zhaoyang Lyu, Dahua Lin, Bo Dai, and Hongfei Fu. Guided diffusion model for adversarial purification. *ArXiv*, abs/2205.14969:null, 2022.

[155] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[156] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*, 2023.

[157] Bihan Wen, Ye Zhu, Ramanathan Subramanian, Tian-Tsong Ng, Xuanjing Shen, and Stefan Winkler. Coverage – a novel database for copy-move forgery detection. In *IEEE International Conference on Image processing (ICIP)*, pages 161–165, 2016.

[158] Lilian Weng. Flow-based deep generative models. *lilianweng.github.io*, 2018.

[159] Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021.

[160] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[161] Andrew P Witkin and Jay M Tenenbaum. On the role of structure in vision. In *Human and machine vision*, pages 481–543. Elsevier, 1983.

[162] Jianyuan Wu, Zheng Wang, Hui Zeng, and Xiangui Kang. Multiple-operation image anti-forensics with wgan-gp framework. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1303–1307. IEEE, 2019.

[163] Quanlin Wu, Hang Ye, and Yuntian Gu. Guided diffusion model for adversarial purification from random noise. *ArXiv*, abs/2206.10875:null, 2022.

[164] Shutong Wu, Jiong Wang, Wei Ping, Weili Nie, and Chaowei Xiao. Defending against adversarial audio via diffusion model. *ArXiv*, abs/2303.01507:null, 2023.

[165] Yue Wu, Wael AbdAlmageed, and Premkumar Natarajan. Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[166] Zhung-Han Wu, Matthew C Stamm, and KJ Ray Liu. Anti-forensics of median filtering. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3043–3047. IEEE, 2013.

[167] Chaowei Xiao, Zhongzhu Chen, Kun Jin, Jiongxiao Wang, Weili Nie, Mingyan Liu, Anima Anandkumar, Bo Li, and Dawn Song. Densepure: Understanding diffusion models towards adversarial robustness. *arXiv preprint arXiv:2211.00322*, 2022.

[168] Yongchao Xu, Thierry Géraud, and Laurent Najman. Context-based energy estimator: Application to object segmentation on the tree of shapes. In *2012 19th IEEE International Conference on Image Processing*, pages 1577–1580, 2012.

[169] Shinji Yamada, Satoshi Kamiya, and Kazuhiro Hotta. Reconstructed student-teacher and discriminative networks for anomaly detection. *arXiv preprint arXiv:2210.07548*, 2022.

[170] Jie Yang, Yong Shi, and Zhiquan Qi. Dfr: Deep feature reconstruction for unsupervised anomaly segmentation. *arXiv preprint arXiv:2012.07122*, 2020.

[171] Jie Yang, Yong Shi, and Zhiquan Qi. Dfr: Deep feature reconstruction for unsupervised anomaly segmentation. *arXiv preprint arXiv:2012.07122*, 2020.

[172] Jihun Yi and Sungroh Yoon. Patch svdd: Patch-level svdd for anomaly detection and segmentation. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[173] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows. *arXiv preprint arXiv:2111.07677*, 2021.

[174] Hai-Dong Yuan. Blind forensics of median filtering in digital images. *IEEE Transactions on Information Forensics and Security*, 6:1335–1345, 12 2011.

[175] Zhiyuan Zha, Xin Liu, Ziheng Zhou, Xiaohua Huang, Jingang Shi, Zhenhong Shang, Lan Tang, Yechao Bai, Qiong Wang, and Xinggan Zhang. Image denoising via group sparsity residual constraint. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1787–1791. IEEE, 2017.

[176] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[177] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Improved techniques for maximum likelihood estimation for diffusion odes. In *International Conference on Machine Learning*, pages 42363–42389. PMLR, 2023.

[178] Ye Zheng, Xiang Wang, Rui Deng, Tianpeng Bao, Rui Zhao, and Liwei Wu. Focus your distribution: Coarse-to-fine non-contrastive learning for anomaly detection and localization. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2022.

[179] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

[180] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In *European Conference on Computer Vision*, pages 392–408. Springer, 2022.

List of Tables

List of Figures

This is the last page.
Thursday 7^{th} November, 2024.