Industrial Anomaly Detection with Normalizing Flows

Von der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte

Dissertation

von

Marco Rudolph, M.Sc.

geboren am 1. Juli 1996 in Hannover

2024

Hauptreferent: Prof. Dr.-Ing. B. Rosenhahn
Korreferent: Prof. Dr.-Ing. Bastian Wandt
Vorsitzender: Prof. Dr.-Ing. J. Ostermann

Tag der Promotion: 5. April 2024

CONTENTS

1 Introduction 1
  1.1 Anomaly Detection in the Industrial Context 3
  1.2 Addressed Applications 4
  1.3 Structure of the Thesis 5
  1.4 List of Publications 7
    1.4.1 Anomaly Detection 7
    1.4.2 Other Publications 10
2 Fundamentals 13
  2.1 Anomaly Detection 13
    2.1.1 Semi-Supervised Anomaly Detection 13
    2.1.2 Taxonomy 14
  2.2 Related Work 15
    2.2.1 Overview 16
    2.2.2 Traditional Approaches 17
    2.2.3 Generative Models 19
    2.2.4 Student-Teacher Networks 20
    2.2.5 Density Estimation 21
    2.2.6 Synthetic Data 22
    2.2.7 Methods for Multivariate Machine Data 22
3 Datasets and Metrics 24
  3.1 MVTec AD 24
  3.2 MVTec 3D-AD 26
  3.3 Magnetic Tile Defects 28
  3.4 Metrics 29
4 DifferNet 31
  4.1 Normalizing Flows 33
    4.1.1 Introduction 33
    4.1.2 Related Work 35
    4.1.3 Real-NVP 36
  4.2 Method 37
    4.2.1 Architecture 38
    4.2.2 Training 39
    4.2.3 Scoring Function 40
    4.2.4 Localization 40
  4.3 Experiments 41
    4.3.1 Implementation Details 42
    4.3.2 Detection 43
    4.3.3 Localization 44

4.3.4 Ablation Studies 45
4.4 Conclusion 48
4.5 Open Problems 48
5 Cross-Scale-Flow 50
5.1 Method 51
5.1.1 Cross-Scale Flow 52
5.1.2 Learning Objective 54
5.1.3 Localization 55
5.2 Experiments 55
5.2.1 Implementation Details 55
5.2.2 Detection 57
5.2.3 Localization 58
5.2.4 Ablation Studies 59
5.3 Conclusion 61
5.4 Open Problems 61
6 Asymmetric Student-Teacher Networks 63
6.1 Method 65
6.1.1 Teacher 66
6.1.2 Student 67
6.1.3 3D Preprocessing 68
6.2 Experiments 70
6.2.1 Implementation Details 70
6.2.2 Results 71
6.2.3 Ablation Studies 75
6.3 Open Problems 79
6.4 Comparison with DifferNet and CS-Flow 79
6.5 Conclusion 81
7 Voraus-AD dataset 82
7.1 Overview 84
7.2 Setup 85
7.3 Robot and Signals 86
7.4 Pick-And-Place Operation 88
7.5 Anomalies 88
7.5.1 Process Errors 88
7.5.2 Gripping Errors 91
7.5.3 Robot axis wear 91
7.6 Analysis 91
7.6.1 Normal Data 91
7.6.2 Anomalies 92
7.6.3 Comparison to AURSAD 94
7.6.4 Evaluation Protocol 95
7.7 Conclusion 95

8 MTS-Flow 96
- 8.1 Method 96
  - 8.1.1 Architecture 97
  - 8.1.2 Training 98
  - 8.1.3 Temporal Analysis 99
- 8.2 Experiments 99
  - 8.2.1 Implementation Details 100
  - 8.2.2 Results 102
  - 8.2.3 Ablation Studies 105
  - 8.2.4 Temporal Analysis 107
- 8.3 Conclusion 107

9 Conclusion 110

Bibliography 115

ACRONYMS

AD	Anomaly Detection
AE	Autoencoder
AOI	Automatic Optical Inspection
AST	Asymmetric Student-Teacher
AUROC	Area Under the Receiver Operating Characteristic Curve
CAE	Convolutional Autoencoder
CS-Flow	Cross-Scale Flow
GAN	Generative Adversarial Network
HMM	Hidden Markov Model
LSTM	Long Short-term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
MTD	Magnetic Tile Defects Dataset
MVT	Multivariate Time Series
ND	Novelty Detection
NF	Normalizing Flow
NN	Neural Network
OCSVM	One-Class Support Vector Machine
OOD	out-of-distribution
OSR	Open Set Recognition
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RGB	red, green, blue (image channels)
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SSAD	Semi-supervised Anomaly Detection
SVM	Support Vector Machine
VAE	Variational Autoencoder

NOTATIONS

Numbers and Arrays

$\boldsymbol{a}$	Vector
$\boldsymbol{a}^T$	Transpose of vector $\boldsymbol{a}$
$\boldsymbol{A}$	Matrix
$\mathcal{A}$	Space
$\boldsymbol{A}^{-1}$	Inverse of quadratic matrix $\boldsymbol{A}$
$\|\boldsymbol{a}\|_2$	ℓ_2-norm of vector $\boldsymbol{a}$
$\boldsymbol{I}_n$	Identity matrix of dimension $n \times n$
$\mathbf{0}$	Vector of all zeros
$\odot$	Hadamard-Product
$\det(\boldsymbol{A})$	Determinant of matrix A
$\dim(\boldsymbol{x})$	number of dimensions of $\boldsymbol{x}$
∇_x	Gradient of x

Symbols

x	Input data (usually image or multivariate signal)
μ	Mean
$\boldsymbol{C}$	Covariance Matrix
θ	Decision Threshold
α	Clamping Parameter
η	Learning rate
w	Image width
h	Image height
c	Condition

$p_X(x)$	Density of the sample x regarding the variable X
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal distribution with mean $\boldsymbol{\mu}$ and covariance Σ
$s_i(\cdot)$	Internal network for scaling coefficients
$t_i(\cdot)$	Internal network for translation coefficients
$f_{\downarrow}(\cdot)$	Image downscaling operator
G	Gaussian kernel
S	Number of signals
T	Number of time steps
C	Number of (image) channels
$\mathcal{X}$	Data space
$\mathcal{Y}$	Feature space
$\mathcal{Z}$	Latent space
$\mathcal{R}$	Real numbers
$\mathcal{L}$	Loss function

ABSTRACT

This thesis addresses deep learning-based methods for automatic anomaly detection in an industrial context. It involves image- or sensor-based detection of defects in the production process that can affect the quality of products. Automating this task provides a reliable and cost-effective alternative to humans, who perform this task manually by sighting. Since this setup has special requirements such as detecting previously unknown defects that traditional approaches cannot fulfill, this paper presents anomaly detection methods that learn without any examples of anomalies and include only normal data in the training process.

Most of our proposed methods address the problem from a statistical perspective. Based on a deep-learning-based density estimation of the normal data, it is assumed that anomalies are considered unlikely according to the modeled distribution. The density estimation is performed by so-called *Normalizing Flows*, which, in contrast to conventional neural networks, can model a formally valid probability distribution due to their bijective mapping. Moreover, due to their flexibility, Normalizing Flows allow modeling of more complex distributions in contrast to traditional methods, which usually use strong simplifications about the distribution. The first chapters focus on anomaly detection on RGB images, which is the standard case for most optical-based scenarios. We show that density estimation based on feature vectors of a pre-trained neural network is an effective solution. The likelihood of different augmentations of the image are combined to form a final prediction. Thereby, by considering the gradients of the input image, in addition to the instance-based detection, a localization of the anomaly can be given. In the further course, this method is optimized with respect to runtime and detection performance. Instead of vectors, fully resolved feature maps of different sizes are combined and processed by an adapted Normalizing Flow architecture. Another proposed method shows how the Normalizing Flow can be integrated into a student-teacher approach, which is motivated by the fact that a student network fails to mimic the teacher on anomalous data. Here, a Normalizing Flow is used as the teacher for density estimation and a conventional convolutional network is used as the student. We can show that this asymmetry between the networks favors anomaly detection by further hindering the student to predict the behavior of the teacher for anomalies. Besides RGB data, this can also be demonstrated on 3D scans and their combination.

In addition to image-based detection, the application to machine data time series from a robot is addressed towards the end of this work. Since there has been a lack of datasets in this area, we first present our own dataset, which includes 130 signals from a robot performing a pick-and-place task. We induced 12 different types of anomalies for the test set, which are divided into process errors, wear, and gripping errors. The previously proposed concept of density estimation with Normalizing Flows is then transferred to this data domain.

keywords – Automation, anomaly detection, density estimation, Normalizing Flows.

KURZFASSUNG

Diese Arbeit thematisiert deep-learning-basierte Methoden zur automatischen Anomaliedetektion im industriellen Kontext. Dabei ist das Ziel, bild- oder sensorbasiert Fehler im Produktionsprozess zu erkennen, welche die Qualität von Produkten beeinträchtigen. Die Automatisierung dieser Aufgabe stellt eine zuverlässige und kostengünstige Alternative zu Menschen dar, die diese Sichtung manuell durchführen. Da die Detektion spezielle Anforderungen wie das Erkennen vorher unbekannter Fehler aufweist, die mit klassischen Verfahren nicht erfüllt werden können, werden in dieser Arbeit Verfahren der Anomaliedetektion vorgestellt, welche ohne jegliche Beispiele von Anomalien lernen und ausschließlich Normaldaten ins Training einbeziehen.

Die hier vorgestellten Verfahren betrachten das Problem zum Großteil aus einer statistischen Perspektive. Dabei wird eine mittels deep-learning eine Dichte der Normaldaten modelliert und angenommen, dass Anomalien gemäß der modellierten Verteilung als unwahrscheinlich gelten. Eine solche Dichteschätzung wird durch sogenannte *Normalizing Flows* durchgeführt, welche im Gegensatz zu konventionellen neuronalen Netzen aufgrund ihrer besonderen Eigenschaft der Bijektivität eine formal gültige Wahrscheinlichkeitsverteilung modellieren können. Außerdem ermöglichen diese durch ihre Flexibiltät der komplexere Verteilungen abzubilden, wohingegen klassischen Methoden meist starke Annahmen zur Verteilung treffen.

Die ersten Kapitel konzentrieren sich auf die Fehlererkennung auf RGB-Bildern als Standardfall für die meisten optisch-basierten Szenarien. Wir zeigen, dass eine Dichteschätzung auf Grundlage von Merkmalsvektoren eine effektive Lösung darstellt. Dabei werden die Likelihood verschiedener Augmentierungen des Bildes zu einer Prädiktion zusammengefasst. Es kann durch Betrachtung der Gradienten des Eingangsbildes neben der instanzbasierten Detektion zusätzlich auch das örtliche Auftreten der Anomalie prädiziert werden. Im weiteren Verlauf wird dieses Verfahren in Bezug auf Laufzeit und Güte optimiert. Anstatt von Vektoren werden vollaufgelöste Merkmalskarten verschiedener Skalen zusammengefasst und mittels einer speziell angepasste Normalizing-Flow-Architektur verarbeitet. In einem weiteren Verfahren wird der Normalizing Flow in einen Student-Teacher-Ansatz integriert, wobei ausgenutzt wird, dass ein Student die Ausgabe eines Teacher-Netzwerk auf Anomalien nicht nachbilden kann. Als Teacher wird ein Normalizing Flow zur Dichte-

schätzung und als Student ein klassisches Faltungsnetz verwendet. Wir zeigen, dass diese Asymmetrie zwischen den Netzwerken die Anomaliedetektion verbessert, da es dem Student-Netzwerk erschwert wird, die Ausgaben des Teacher-Netzwerkes auf Anomalien zu rekonstruieren. Dies wird neben RGB-Daten auch auf 3D-Scans und deren Kombination demonstriert.
Neben der bildbasierten Detektion wird gegen Ende der Arbeit auch die Anwendung auf Zeitreihen von Maschinendaten eines Roboters thematisiert. Da es in diesem Bereich bisher einen Mangel an Datensätzes gab, stellen wir zunächst einen eigenen Datensatz vor. Dieser beinhaltet den zeitlichen Verlauf von 130 Signalen eines Roboters, welcher eine Pick-and-Place-Operation durchführt. Das Testset enthält 12 verschiedene Anomalietypen, welche sich in Prozessfehler, Verschleiß und Greiffehler gliedern. Anschließend wird das eingangs vorgestellte Konzept der Dichteschätzung mit Normalizing Flows auf diese Datendomäne erfolgreich übertragen und evaluiert.

Stichworte – Automatisierung, Anomaliedetektion, Dichteschätzung, Normalizing Flows.

1

INTRODUCTION

An anomaly, which is derived from the Greek word $\alpha\nu\omega\mu\alpha\lambda\iota\alpha$ (anomalía) meaning "deviation from the rule", is an unusual event that is unexpected from the known behavior. For example, a hurricane in Central Europe, the outbreak of a global disease or an uncommon coloration of a species (as in Figure 1.1) would be seen as anomalies. The nature of anomalies can be diverse: They may include phenomena that have never been observed before in any form, as known structures with unusual properties in detail. In general, anomalies can be defined by being "out-of-distribution" while assuming that the normal behavior is "in-distribution".

This work addresses the automatic detection of such anomalies using machine learning. Usually, this cannot be implemented with traditional supervised methods, since these can only reliably detect familiar concepts that have been learned in training. Unfortunately, the data for all possible types of anomalies cannot be collected. Instead, the problem is formulated itself as a task named anomaly detection (AD). In the semi-supervised anomaly detection investigated in this work, only normal (i. e. anomaly-free) data is made available to the system during training. After training, the task is to output whether or to what extent a given observation fits the already known normality or shows unusual characteristics. Figure 1.2 shows a 2D toy example of a dataset showing normal and anomalous samples. It can be observed that an anomaly can

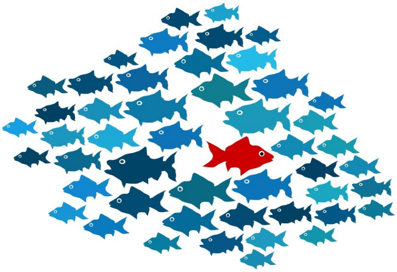


Figure 1.1: Illustration of anomaly detection by Isura Nimalasiri. The color and the orientation make the central fish appear anomalous.

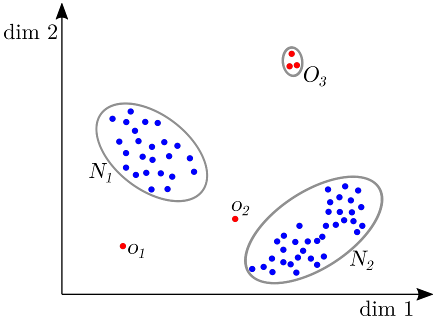


Figure 1.2: Examples of anomalies in a 2D data set. The data shows two normal regions N_1 and N_2 in blue, where the majority of observations are concentrated. However, points that are significantly distant from these regions, such as o_1 and o_2, as well as points in region O_3, are considered anomalies (in red).

deviate from the distribution of normal samples to varying degrees, as for example o_3 is far more distant to the normal clusters compared to o_2. Although this deviation could be differentiated in principle and existing methods provide such an indication with a so-called *anomaly score*, the research community considers a binary distinction between normal and anomalous for simplification. Details on the anomaly score as well as a more detailed and technical definition of anomaly detection is given in Section 2.1.

Anomaly detection is a critical technique used in various real-world applications, to identify unusual or abnormal behavior in data, systems, or processes. One of the significant applications of anomaly detection is in cybersecurity, where it is used to detect potential security breaches, network intrusions, and anomalous user activities that may indicate cyber attacks or data breaches [1]. Anomaly detection is also utilized in financial fraud detection, where it helps identify suspicious transactions, unusual trading patterns, or fraudulent activities that may indicate fraudulent behavior in banking and financial systems [2]. In healthcare, it is used to detect anomalies in patient health data, identify potential health risks, or detect abnormal behaviors in medical devices [3]. There are also applications in *Internet of Things* (IoT) systems to detect unusual behavior in sensor data from connected devices, such as abnormal energy consumption or unexpected behavior in smart home devices [4]. Furthermore, as a focus of this work, anomaly detection is highly relevant in industrial manufacturing to detect production failures on images, unusual patterns

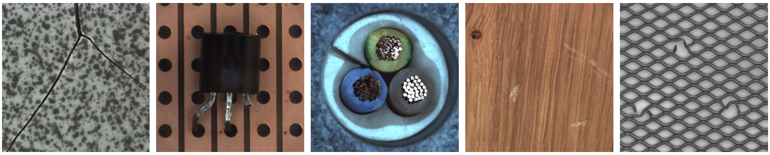


Figure 1.3: Defective examples from the MVTec AD dataset [7]. Possible errors include for example cracks, missing components, cuts, scratches and dents.

in sensor data, or deviations in production processes that may lead to quality issues [5], [6] as shown in Figure 1.3.

1.1 ANOMALY DETECTION IN THE INDUSTRIAL CONTEXT

This thesis focuses on the design of anomaly detection methods for applications in the industrial context of manufacturing. Manufacturing represents one of the most important economic sectors worldwide, accounting for 17% of global GDP in 2021[1], or about $16 trillion, and providing the basis for any goods. This makes the automation and optimization of processes in this sector, which the public refers to as *Industry 4.0*, of great interest.

In the context of industrial production, anomaly detection is an essential tool for identifying and preventing equipment failures, process deviations, and other irregularities that can cause significant downtime, product defects, and safety hazards. Traditionally, this is made by humans which is costly and error-prone in practice [8] since operators may suffer from fatigue and give different assessments. Anomaly detection algorithms can analyze large volumes of sensor data in real-time to detect any deviations from expected patterns or behaviors. By detecting anomalies early, proactive steps can be taken to ensure safety, prevent further equipment failures and reduce follow-up costs. In the case of defect detection, a model identifying an anomaly triggers either a fully automatic rejection of the product or a semi-automatic inspection of the operator.

Although defects or irregularities could in principle be detected with a traditional classification, semi-supervised anomaly detection is of particular interest in the industry [7]. Firstly, there are only normal examples at the beginning of production with no anomalies at all, since these rarely occur in most scenarios. Secondly, it is impossible to ensure that examples are collected for all defect scenarios [9]. An incomplete training

---

[1] source: World Bank

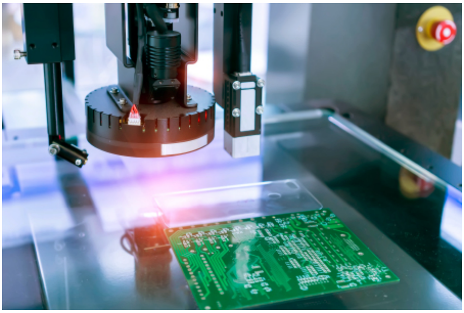


Figure 1.4: Example for a setup of automatic optical inspection. A camera is placed over the object of interest. Image is taken from *geospacemfg.com*.

set in a supervised setting often results in the model being insensitive to unknown defects. In contrast, AD is considered to be more robust against any deviations by estimating the normality directly.

1.2 ADDRESSED APPLICATIONS

This work addresses anomaly detection on image data as part of a manufacturing pipeline, and machine data from a robot. Most of the methods and experiments focus on RGB images which are typically used in the context of an automatic optical inspection (AOI). In practice, this is implemented by placing a camera at a specific position on the production line as in Figure 1.4 to predict, usually in real-time, whether a visually determinable defect is present [8]. However, not all defects are necessarily visually determinable as in the case of small surface changes [9]. Furthermore, RGB images are sensitive to various illuminations and shadows, which in practice are sometimes difficult to keep constant over time. Therefore, we also conduct experiments on mostly illumination-invariant 3D scans of a stereo camera which are used instead or as an add-on for the inspection of surface anomalies. This work focuses its evaluation on image-level detection (*Does this instance contain any defect?*), since in practice a decision is usually made on this level [8]. As a secondary task, we investigate the defect segmentation [10] which can be used to interpret at which location the system has detected a defect to offer explainability to the user (*Where is the defect?*).

In addition to the inspection of the product, it is also of interest to examine the machines involved in production for defects, as these are often the source of defects that subsequently occur in the product. Their behavior can be precisely observed by the machine data, which are usually recorded for their control anyway. We present a data set for AD in the context of a robot application which is described in Chapter 7. In addition, we introduce an AD method adapted to the data domain of high-dimensional machine data in Chapter 8.

1.3 STRUCTURE OF THE THESIS

This thesis is organized into the following parts and visualized in Figure 1.5:

Chapter 2: Fundamentals
The problem of anomaly detection is first formally defined and differentiated from related problems. Furthermore, an overview of existing work is given, presenting both general classical methods alongside those in the context of industrial applications.
Chapter 3: Datasets and Metrics
The image data sets used to evaluate the methods of this thesis are presented. These are MVTec AD [7], MVTec 3D-AD [9] and the Magnetic Tile dataset [11] which include various objects such as electrical components, food or drugs, and textures such as textiles and building materials. Besides, the evaluation metric AUROC is presented.
Chapter 4: DifferNet
This chapter introduces *DifferNet*, a method for image-based anomaly detection. It employs a pretrained neural network to extract image features across multiple scales. Based on these feature vectors, DifferNet estimates feature density from normal data and uses the likelihood of an observation as an anomaly score. The density estimation is performed by a Normalizing Flow, which maps the feature distribution to a well-defined distribution to quantify individual likelihoods. DifferNet also identifies anomalous regions within the image by backpropagating the anomaly score and highlighting pixels with a high gradient magnitude.
Chapter 5: Cross-Scale-Flow
Based on DifferNet, an extension of the method is given by introducing a new architecture called *Cross-Scale-Flow* (CS-Flow). CS-Flow enables a density estimation on a set of multi-scale feature maps, instead of feature vectors, allowing for the usage of fine-grained image information. The architecture is designed for joint processing of these multi-scale feature maps to enable the utilization of correlations between different-sized feature maps.
Chapter 6: Asymmetric Student-Teacher Networks
In this chapter, we introduce a *Asymmetric Student-Teacher Network* that combines concepts from CS-Flow with student-teacher networks. It uti-

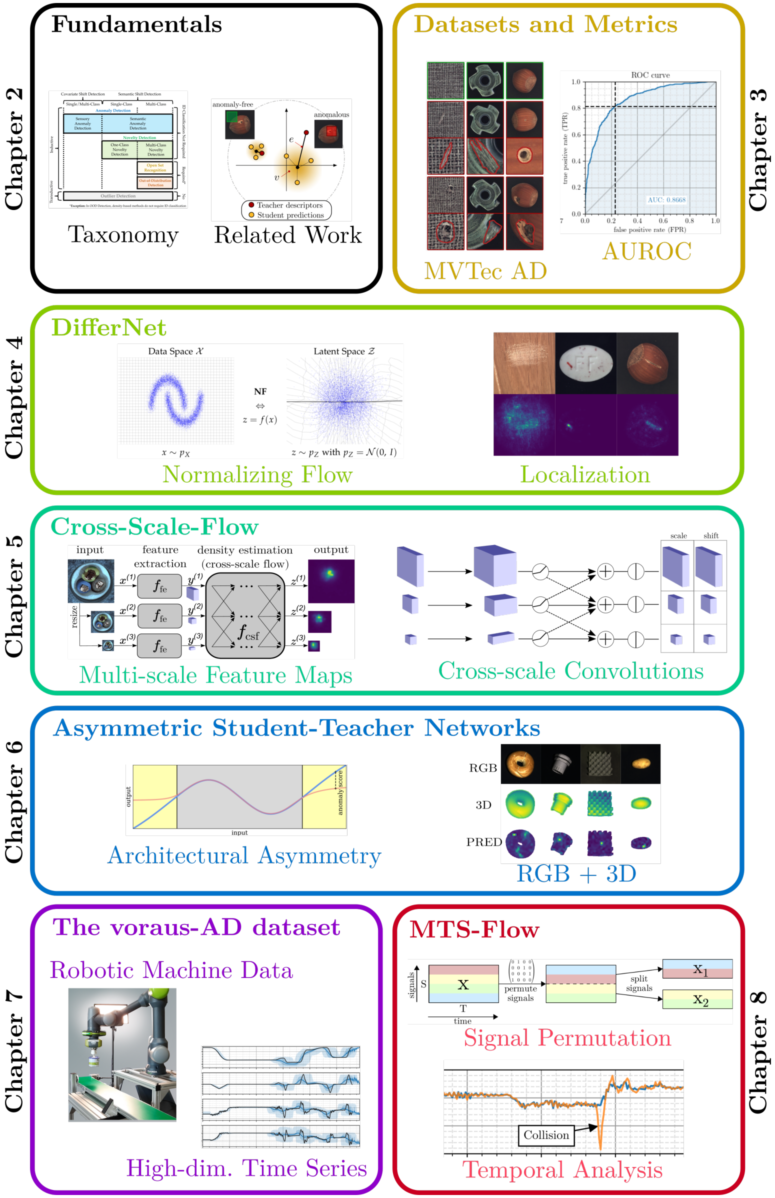


Figure 1.5: Thesis overview.

lizes the distance from an NF-based teacher and a standard convolutional network as student, mimicking the teacher, to indicate anomalies. This addresses both the issue of undesired generalization of symmetric student-teacher pairs in previous work and the limitations of Normalizing-Flow-based density estimation.

Chapter 7: voraus-AD Dataset
This chapter introduces our *voraus-AD* dataset, which contains machine data of a robot in a non-deterministic pick-and-place task, from which various anomalies are to be detected. *voraus-AD* represents the first anomaly detection dataset in robotics with such a wide range of anomalies and signals.

Chapter 8: MVT-Flow
We used the concepts of Chapters 4 and 5 to transfer NF-based density estimation for AD to multivariate time series as in voraus-AD. This chapter demonstrates that our proposed *MVT-Flow* outperforms existing methods on voraus-AD by a large margin.

Chapter 9: Conclusion
The thesis is concluded and some options for future work are given.

1.4 LIST OF PUBLICATIONS

In this section, an overview of all the publications during my work is given. Subsection 1.4.1 focuses on publications related to the present thesis, which serve as the basis for Chapter 4-8 of the thesis. In Subsection 1.4.2, other publications in the fields of dimensionality reduction, human pose estimation and interpretable machine learning are listed.

1.4.1 *Anomaly Detection*

[12] **Marco Rudolph**, Bastian Wandt, 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*, 2021.

The detection of manufacturing errors is crucial in fabrication processes to ensure product quality and safety standards. Since many defects occur very rarely and their characteristics are mostly unknown a priori, their detection is still an open research question. To this end, we propose DifferNet: It leverages the descriptiveness of features extracted by convolutional neural networks to estimate their density using Normalizing Flows. Normalizing flows are well-suited to deal with low-dimensional data distributions. However, they struggle with the high dimensionality of images. Therefore, we employ a multi-scale feature extractor which enables the Normalizing Flow to assign meaningful likelihoods to the images. Based on these likelihoods we develop a scoring function that indicates defects. Moreover, propagating the score back to the image enables pixel-wise localization. To achieve a high robustness and performance we exploit multiple transformations in training and evaluation. In contrast to most other methods, ours does not require a large number of training samples and performs well with as low as 16 images. We demonstrate superior performance over existing approaches on the challenging and newly proposed MVTec AD and Magnetic Tile Defects datasets.

[13] **Marco Rudolph**, Tom Wehrbein, Bastian Wandt, Bodo Rosenhahn. Fully Convolutional Cross-scale-flows for Image-based Defect Detection. *In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

In industrial manufacturing processes, errors frequently occur at unpredictable times and in unknown manifestations. We tackle this problem, known as automatic defect detection, without requiring any image samples of defective parts. Recent works model the distribution of defect-free image data, using either strong statistical priors or overly simplified data representations. In contrast, our approach handles fine-grained representations incorporating the global and local image context while estimating flexibly the density. To this end, we propose a novel fully convolutional cross-scale Normalizing Flow (CS-Flow) that jointly processes multiple feature maps of different scales. Using Normalizing Flows to assign meaningful likelihoods to input samples allows for efficient defect detection on image-level. Moreover, due to the preserved spatial arrangement the latent space of the Normalizing Flow is interpretable, ie it is applicable to localize defective regions in the image. Our work sets a new state-of-the-art in image-level defect detection on the benchmark datasets Magnetic Tile Defects and MVTec AD showing a 100% AUROC on 4 out of 15 classes.

[14] **Marco Rudolph**, Tom Wehrbein, Bastian Wandt, Bodo Rosenhahn. Asymmetric Student-Teacher Networks for Industrial Anomaly Detection. *In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023.

Industrial defect detection is commonly addressed with anomaly detection (AD) methods where no or only incomplete data of potentially occurring defects is available. This work discovers previously unknown problems of student-teacher approaches for AD and proposes a solution, where two neural networks are trained to produce the same output for the defect-free training examples. The core assumption of student-teacher networks is that the distance between the outputs of both networks is larger for anomalies since they are absent in training. However, previous methods suffer from the similarity of student and teacher architecture, such that the distance is undesirably small for anomalies. For this reason, we propose asymmetric student-teacher networks (AST). We train a Normalizing Flow for density estimation as a teacher and a conventional feed-forward network as a student to trigger large distances for anomalies: The bijectivity of the Normalizing Flow enforces a divergence of teacher outputs for anomalies compared to normal data. Outside the training distribution, the student cannot imitate this divergence due to its fundamentally different architecture. Our AST network compensates for wrongly estimated likelihoods by a Normalizing Flow, which was alternatively used for anomaly detection in previous work. We show that our method produces state-of-the-art results on the two currently most relevant defect detection datasets MVTec AD and MVTec 3D-AD regarding image-level anomaly detection on RGB and 3D data.

[15] **Jan Thieß Brockmann**, **Marco Rudolph**, Bodo Rosenhahn, Bastian Wandt. The voraus-AD Dataset for Anomaly Detection in Robot Applications. *Transactions on Robotics*, 2023.

During the operation of industrial robots, unusual events may endanger the safety of humans and the quality of production. When collecting data to detect such cases, it is not ensured that data from all potentially occurring errors is included as unforeseeable events may happen over time. Therefore, anomaly detection (AD) delivers a practical solution, using only normal data to learn to detect unusual events. We introduce a dataset that allows training and benchmarking of anomaly detection methods for robotic applications based on machine data which will be made publicly available to the research community. As a typical robot task, the dataset includes a pick-and-place application which involves movement, actions of the end effector and interactions with the objects of the environment. Since several of the contained anomalies are not task-specific but general, our dataset offers value for other robotics applications as well. Additionally, we present MVT-Flow (multivariate time-series flow) as a new baseline method for anomaly detection: It relies on deep-learning-based density estimation with Normalizing Flows, tailored to the data domain by taking its structure into account.

1.4.2 *Other Publications*

[16] **Marco Rudolph**, Bastian Wandt, Bodo Rosenhahn. Structuring Autoencoders. *In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

In this paper we propose Structuring AutoEncoders (SAE). SAEs are neural networks which learn a low-dimensional representation of data and are additionally enriched with a desired structure in this low-dimensional space. While traditional Autoencoders have proven to structure data naturally they fail to discover semantic structure that is hard to recognize in the raw data. The SAE solves the problem by enhancing a traditional Autoencoder using weak supervision to form a structured latent space. In the experiments we demonstrate, that the structured latent space allows for a much more efficient data representation for further tasks such as classification for sparsely labeled data, an efficient choice of data to label, and morphing between classes. To demonstrate the general applicability of our method, we show experiments on the benchmark image datasets MNIST, Fashion-MNIST, DeepFashion2 and on a dataset of 3D human shapes.

[17] **Bastian Wandt**, Marco Rudolph, Petrissa Zell, Helge Rhodin, Bodo Rosenhahn. CanonPose: Self-Supervised Monocular 3D Human Pose Estimation in the Wild . *In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Human pose estimation from single images is a challenging problem in computer vision that requires large amounts of labeled training data to be solved accurately. Unfortunately, for many human activities (e.g. outdoor sports) such training data does not exist and is hard or even impossible to acquire with traditional motion capture systems. We propose a self-supervised approach that learns a single image 3D pose estimator from unlabeled multi-view data. To this end, we exploit multi-view consistency constraints to disentangle the observed 2D pose into the underlying 3D pose and camera rotation. In contrast to most existing methods, we do not require calibrated cameras and can therefore learn from moving cameras. Nevertheless, in the case of a static camera setup, we present an optional extension to include constant relative camera rotations over multiple views into our framework. Key to the success are new, unbiased reconstruction objectives that mix information across views and training samples. The proposed approach is evaluated on two benchmark datasets (Human3.6M and MPII-INF-3DHP) and on the in-the-wild SkiPose dataset.

[18] **Tom Wehrbein**, Marco Rudolph, Bastian Wandt, Bodo Rosenhahn. Probabilistic Monocular 3D Human Pose Estimation with Normalizing Flows. *In: Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

3D human pose estimation from monocular images is a highly ill-posed problem due to depth ambiguities and occlusions. Nonetheless, most existing works ignore these ambiguities and only estimate a single solution. In contrast, we generate a diverse set of hypotheses that represents the full posterior distribution of feasible 3D poses. To this end, we propose a normalizing-flow-based method that exploits the deterministic 3D-to-2D mapping to solve the ambiguous inverse 2D-to-3D problem. Additionally, uncertain detections and occlusions are effectively modeled by incorporating uncertainty information of the 2D detector as a condition. Further keys to success are a learned 3D pose prior and a generalization of the best-of-M loss. We evaluate our approach on the two benchmark datasets Human3. 6M and MPI-INF-3DHP, outperforming all comparable methods in most metrics. The implementation is available on GitHub.

[19] **Thomas Norrenbrock**, Marco Rudolph, Bodo Rosenhahn. Take 5: Interpretable Image Classification with a Handful of Features. *In: Progress and Challenges in Building Trustworthy Embodied AI (NeurIPS Workshop)*, 2022.

Deep Neural Networks use thousands of mostly incomprehensible features to identify a single class, a decision no human can follow. We propose an interpretable sparse and low-dimensional final decision layer in a deep neural network with measurable aspects of interpretability and demonstrate it on fine-grained image classification. We argue that a human can only understand the decision of a machine learning model if the features are interpretable and only very few of them are used for a single decision. For that matter, the final layer has to be sparse and – to make interpreting the features feasible – low-dimensional. We call a model with a Sparse Low-Dimensional Decision “SLDD-Model”. We show that a SLDD-Model is easier to interpret locally and globally than a dense high-dimensional decision layer while being able to maintain competitive accuracy. Additionally, we propose a loss function that improves a model's feature diversity and accuracy. Our more interpretable SLDD-Model only uses 5 out of just 50 features per class, while maintaining 97% to 100% of the accuracy on four common benchmark datasets compared to the baseline model with 2048 features.

2

FUNDAMENTALS

This chapter describes the problem of anomaly detection in Section 2.1 and gives an overview of existing work in this area in Section 2.2. In addition to generalized AD, we address the challenges and the associated methods in the industrial context of production. We assume a basic knowledge of deep learning, computer vision and stochastics. For a more detailed introduction, we refer the reader to standard literature [20]–[23].

2.1 ANOMALY DETECTION

Anomaly detection describes the problem of finding patterns that deviate from expected behavior [24]. In other words, we want to identify the samples that differ from the majority of observations [25] which is often referred to as *normal*. Formally, this is a binary classification $\mathcal{A} : \mathcal{X} \rightarrow \{0,1\}$ where an input $x \in \mathcal{X}$ is to be assigned to one of the classes *anomaly* ($\mathcal{A}(x) = 1$) or *normal* ($\mathcal{A}(x) = 0$). There are many sub-problems and related problems in the literature, all of which pursue this goal, but assume different conditions regarding existing training data or types of anomalies. In the following, we specify the setting addressed in this paper and compare it to related problems. Note that the research area in AD shows parallel existing terminologies and taxonomies. We take the definitions of [26] and [27].

2.1.1 *Semi-Supervised Anomaly Detection*

This work considers the following setting of semi-supervised anomaly detection (SSAD): Given a training set X_{tr} with all $x \in X_{\text{tr}}$ known to be normal, a model should decide whether unseen examples are normal or anomalous. In practice, models do not directly output this binary decision since a decision boundary is hardly determinable without any anomalous examples. Instead, the model usually represents a function $f : \mathcal{X} \rightarrow \mathcal{R}$ that maps from the input space $\mathcal{X}$ to a scalar *anomaly score* $a \in \mathcal{R}$ that indicates an anomalous input. The higher the score, the more is the input considered to be anomalous. In practice, the binary decision is taken by thresholding:

$$\mathcal{A}(x) = \begin{cases} 1 & \text{for } a \geq \theta \\ 0 & \text{for } a < \theta \end{cases}. \tag{2.1}$$

The threshold parameter θ can be calibrated with a hold-out validation set. A particular setting depends strongly on the use case and the associated requirement at the trade-off between sensitivity and specificity. More details for the evaluation are given in Section 3.4.

2.1.2 *Taxonomy*

Anomaly detection is seen as a part of *generalized out-of-distribution detection* which includes many related tasks which aim to identify unusual observations regarding a given dataset. For example, the definition of SSAD simultaneously describes novelty detection (ND) and out-of-distribution detection (OOD detection) and are sometimes used as synonyms, although they are often motivated differently: While anomaly detection primarily should detect erroneous or malicious examples that deviate from the "good" or normal observations, ND aims to complement the model with anomalous examples ([25], [28]) and OOD detection aims to reject anomalous examples as the model cannot perform a reliable prediction [27]. Open Set Recognition (OSR) shares the same motivation as ND but is specialized to multi-class problems and is additionally supported with "known unknown class" samples in training which can be interpreted as adding labeled anomalies to the training set for AD which is called supervised anomaly detection. In contrast, for unsupervised anomaly detection there are no labels given at all, although there may be a small ratio of anomalies in the training set. Outlier detection also operates on a contaminated data set, whereby no train-test-split is assumed here. Instead, outliers or anomalies are to be identified directly on the given data which makes OD transductive rather than inductive.

All of the mentioned problems deal with a shift from the known majority in training. The type of this shift can be divided into covariate shifts and semantic shifts [25]: Covariate shifts are considered to be changes in the sensory space which includes adversarial examples and style or domain changes. In contrast, semantic shifts deal with the occurrence of new concepts or classes. Anomaly detection can in principle be used for the detection of both shifts: It can be divided into *sensory AD* for identifying covariate shifts and *semantic AD* for semantic shifts. However,

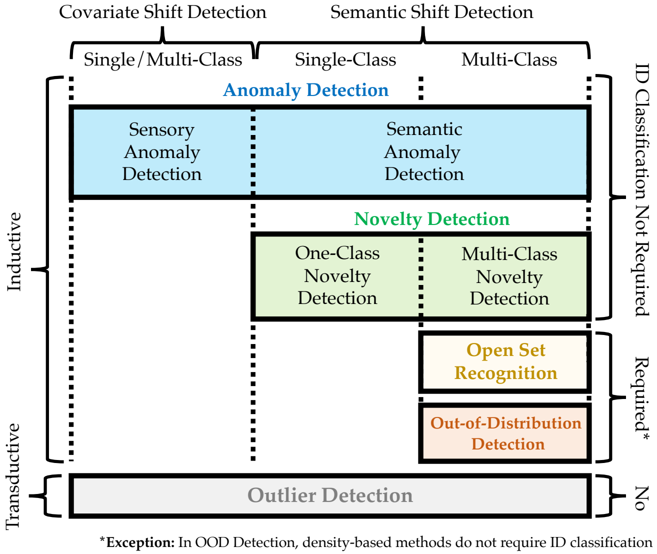

Covariate Shift Detection
Semantic Shift Detection
Single/Multi-Class
Single-Class
Multi-Class
Anomaly Detection
Sensory Anomaly Detection
Semantic Anomaly Detection
Novelty Detection
One-Class Novelty Detection
Multi-Class Novelty Detection
Open Set Recognition
Out-of-Distribution Detection
Outlier Detection
Inductive
Transductive
ID Classification Not Required
Required*
No
*Exception: In OOD Detection, density-based methods do not require ID classification

Figure 2.1: Overview of the taxonomy of tasks in generalized out-of-distribution detection, grouped in shift type, class modalities and learning characteristics. This work focuses the sensory anomaly detection. Image is taken from [25].

the defect and event detection discussed in this work is a sensory AD task. The problems ND, OOD detection and OSR deal with semantic shifts. A visual overview of the taxonomy is given in Figure 2.1.

2.2 RELATED WORK

This section gives an overview of the literature related to anomaly detection. First, we provide a summary of image-based anomaly detection concepts in Subsection 2.2.1. Subsequently, Subsection 2.2.2 introduces traditional machine learning approaches for general AD. Contrary, Subsection 2.2.3 to 2.2.6 are related to more recent work dealing with image-based AD for defect detection and are mostly based on deep learning. Finally, we review the literature that focuses AD on multivariate machine data in Subsection 2.2.7. For a general overview of recent anomaly detection methods and their usage in other data domains, we refer the reader to [29]–[31].

Group	Method
Traditional approaches	OCSVM [32]
	Isolation Forest [33]
	Local Outlier Factor [34]
	Nearest Neighbor [35], [36]
Generative Models	ℓ_2-Autoencoder [37], [38]
	SSIM-based Autoencoder [39]
	MemAE [40]
	GANomaly [41]
	ARNet [42]
Student-Teacher Networks	Uninformed Students [43]
	3D-ST [44]
	DTSNE [45]
	STFPM [46]
Density Estimation	Rippel et al. [47]
	PaDiM [48]
Synthetic Data	CutPaste [49]
	DRÆM [50]
	NSA [51]
	Anoseg [52]

Table 2.1: Overview of common AD methods for image-based defect detection grouped in concepts.

2.2.1 *Overview*

In the following, we give an overview of the different concepts of state-of-the-art methods regarding image-based anomaly detection. Classical approaches often adapt existing concepts from supervised learning like decision trees, nearest-neighbor classification or support vector machines so that they are applicable for anomaly detection. Approaches based on generative models attempt to reconstruct a given example and use the resulting error as an anomaly indicator. Whereas, density estimation methods directly measure how likely the occurrence of the observation is under the assumption that a sample would be normal. Student-teacher networks exploit the mechanism that the student overfits to the teacher causing that the teacher cannot be imitated as precisely on anomalies. In approaches based on synthetic data, the problem is transformed into a supervised problem by generating artificial anomalies. Table 2.1 lists the most common image-based AD methods grouped by the aforementioned concepts. In the following, we review the mentioned concepts in more detail.

2.2.2 *Traditional Approaches*

This subsection summarizes selected classical approaches for anomaly detection with most of them being published before the deep learning era.

2.2.2.1 *One-Class Support Vector Machines*

A One-class Support Vector Machines (OCSVM) [32] is a max-margin method which is an adaption of the classical support vector machine (SVM [53]) for AD. One-Class SVMs aim to learn a boundary around the normal data points so that data points that fall outside the boundary are classified as anomalies.

The OCSVM uses a kernel ϕ to map the data into a high-dimensional feature space and then finds a hyperplane $\boldsymbol{w}$ that maximally separates the n data points from the origin with distance ρ. The hyperplane is constructed such that it contains as many data points as possible, while still having a maximum distance to the nearest data point. The points that fall outside this hyperplane are considered anomalies. This can be formulated as the following minimization problem

$$\begin{aligned} &\min_{w,\xi,\rho} \frac{1}{2}||w||^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad &\langle \boldsymbol{w}, \phi(x_i) \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned} \tag{2.2}$$

with ξ_i being the slack variables of the data point x_i, which allow training errors, and ν as the upper bound of the fraction of training errors and lower bound of the fraction of support vectors. The OCSVM heavily depends on the choice of the kernel function and the ν parameter. Since this technique cannot deal with the high dimensionality of images, Andrews et al. [54] propose to apply it on image features obtained by a pretrained neural network.

2.2.2.2 *Isolation Forests*

An Isolation Forest (IF) [33] is a type of tree-based AD algorithm that is inspired by common decision trees [55]. In inference, they recursively partition the data into subsets, using random feature subsampling to create an ensemble of decision trees that aim to isolate one data point from all others. The anomaly score for each data point is calculated as the average path length of these *isolation trees* required to isolate the point. IFs are especially efficient for small datasets as they do not tend to overfit heavily. However, the performance of isolation forests is often sensitive to the choice of hyperparameters such as the subsample size per tree and the maximum tree depth.

2.2.2.3 *Local Outlier Factor*

The Local Outlier Factor (LOF) [34] algorithm is a density-based anomaly detection method that measures the local deviation of a data point with respect to its neighbors. LOF computes the density of each data point by considering the number and distance of its k-nearest neighbors and compares this density to the densities of its neighbors. Data points with low densities relative to their neighbors are considered anomalies. LOF shows to be effective in detecting anomalies in datasets with complex structures and varying densities. The drawbacks of LOFs are the sensitivity to hyperparameters and the application to high-dimensional datasets. Note that this method does not directly compute the density in statistical terms as in 2.2.5.

2.2.2.4 *Nearest-Neighbor Approaches*

Inspired by the k-Nearest-Neighbor algorithm [56] for supervised classification, Amer and Goldstein [57] propose to use the distance to the nearest neighbor among normal samples as an anomaly score. As the distance between images is not meaningful in pixel space, Nazare et al. [35] computes the distance in a PCA-reduced feature space obtained by a pretrained neural network. The PCA [58] helps for reducing the runtime and space complexity, being linear dependent from the number of dimensions d, while maintaining the essential information of the features.

The nearest-neighbor approach was revisited and enhanced with *PatchCore* by Roth et al. [36]. They extend the approach by replacing the feature vectors with the original full-sized feature maps. Nearest neighbors are obtained for vectors of all feature map positions where the candidate set of neighbors is reduced to a patch around the original location. The PCA is replaced with a random projection to d dimensions for complexity reduction. However, the inference time also depends on the size of the memory bank size $|\mathcal{M}|$, which includes candidates for the nearest neighbors search. For reducing the size of the memory bank, a training stage that performs a *minimax facility location coreset selection* [59], [60] is integrated. This greedy algorithm adds samples iteratively from the original memory bank to the coreset $\mathcal{M}_C$ if this sample has the maximal distance to its nearest neighbor among other candidates in the $\mathcal{M}_C$. Using this selection, the performance is nearly maintained with 1% of the original data in the memory bank.

2.2.3 *Generative Models*

Generative models are a class of unsupervised machine learning models that learn to generate new data that is similar to a given dataset by capturing the underlying structure of it. One popular example is an Autoencoder [61] containing an encoder that compresses the input to a low-dimensional latent space from which a decoder aims to reconstruct the input again by applying any reconstruction loss in training. Another common technique is a Generative Adversarial Network (GAN) [62] which is able to create synthetic data examples from random noise by its generator model. During training, the generator is trained alternating in an adversarial manner with another model, the discriminator, which is optimized to correctly identify real examples from the generated ones, while the generator aims to trick the discriminator with its generated samples.

Many anomaly detection methods are based on generative models as Autoencoders and GANs. These techniques exploit the limitation of the generative model after being trained on normal data only, preventing it from reconstructing anomalies. In the simplest case, the input and the reconstruction of an autoencoder is compared [37], [38]. A high reconstruction error is interpreted as an indicator of an anomaly. Bergmann et al. [63] replace the common l_1 or l_2 error with the structural similarity index measure (SSIM) [39] to have a more powerful metric for visual similarity. However, in some cases, autoencoder-based methods fail because they generalize too strongly, i. e.anomalies can be reconstructed as well as normal samples. Gong et al. [40] use memory modules in the latent space to prevent the autoencoder from generalizing to anomalous data. Zhai et al. [64] combine energy-based models and regularized autoencoders to model the data distribution. Denoising autoencoders are used by Huang et al. [42] by letting autoencoders learn to restore transformed images.

Similar to the decoding part of autoencoders, generators of GANs are utilized for anomaly detection. Schlegl et al. [65] propose to learn an inverse generator after training a GAN, utilizing both together for reconstruction. A combination of autoencoders and GANs is proposed by Akcay et al. [41]. They apply the autoencoder directly as the GAN's generator to ensure the generation of normal data only.

Although autoencoders in particular are used for many data domains and applications for AD, autoencoders and GANs perform moderate on defect detection tasks, as later seen in the experimental sections. Since different types of anomalies with individual size, shape and structure have inconsistent characteristics regarding reconstruction errors, they are not widely applicable for defect detection. For example, structures with

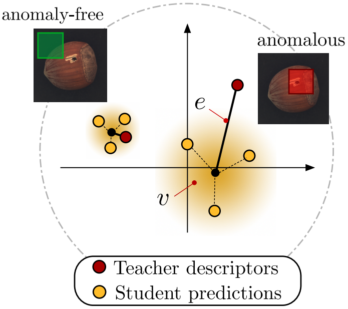


Figure 2.2: Visualization of the output space of student-teacher networks for AD as in [43]: The regression error e between the mean of student predictions (black dots) and the respective teacher outputs (red dots) is taken together with the variance v of the student ensemble (yellow dots) as an anomaly score. As shown by the examples, both terms should be higher for anomalies as student and teacher were trained only on normal samples. Image is taken from [43].

high frequency cannot be represented and reconstructed accurately in general and small defect areas cause smaller errors.

Most of the methods introduced in this thesis are based on Normalizing Flows [66] which are also generative models. Unlike the methods described above, this model is not used to generate images that resemble the input. Instead, we measure how likely a given image would be generated by the model.

2.2.4 *Student-Teacher Networks*

In student-teacher-based approaches, two neural networks are trained consecutively such that the student network imitates the teacher network. First, the teacher is trained on a certain task. In the second step, the student is trained to match the output of the teacher by minimizing the distance between their outputs. Originally, the motivation for having a student network is to distill knowledge and save model parameters [67]–[69]. For AD, student and teacher are both used in inference, using the distance between its outputs as an indicator of an anomaly at test time. This is motivated by the fact that the student is able to match the outputs of the teacher better on normal data as it is optimized exclusively on these, which is why the distance is assumed to be larger for defective examples compared to defect-free examples. As the teacher cannot make use of any labels for semi-supervised AD, it is trained on any pretext task to create an embedding that carries semantics being sensitive to anomalies. This is for example a self-supervised task or any pretraining with other datasets on any task as classification [43].

Bergmann et al. [43] propose to train an ensemble of students regressing the output of a teacher. This teacher is either a distilled version of a pretrained network or trained via metric learning on the given data. The anomaly score is composed of the student uncertainty, measured by the variance of the ensemble, and the regression error as visualized in Figure 2.2. In another work, Bergmann and Sattlegger [44] adapt the student-teacher concept to 3D point clouds. Local geometric descriptors are extracted to train the teacher by reconstructing a set of neighbored points. Wang et al. [46] extend the student task by regressing a feature pyramid obtained by an ImageNet-pretrained feature extractor. Xiao et al. [45] let teachers learn to classify applied image transformations. The regression error and the class score entropy of an ensemble of students are linearly combined to obtain an anomaly score. All of the existing work is based on identical and conventional (non-injective) networks for student and teacher, which causes undesired generalization of the student. This motivates our asymmetric student-teacher network in Chapter 6.

2.2.5 *Density Estimation*

AD can be viewed from a statistical perspective: Estimating the density of normal samples, a model identifies anomalies by their low likelihood. The concept of density estimation for AD is naively realized by assuming a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ having the covariance $\boldsymbol{\Sigma}$ of normal samples $x \sim \boldsymbol{X}$ with $\dim(x) = d$ to approximate the likelihood as

$$p_{\mathbf{X}}(\boldsymbol{x}) \approx \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{2.3}$$

For example, the Mahalanobis distance [70] of pre-extracted features can be applied as an anomaly score [47], [48] which is practically equivalent to computing the negative log likelihood of a multivariate Gaussian as in Eq. 2.3. However, this method is inflexible to most real training distributions, since the assumption of a Gaussian distribution is a strong simplification. To this end, most of our proposed methods are based on a more flexible density estimation with a Normalizing Flow [66] which is introduced in Chapter 4 and further used in Chapters 5, 6 and 8.

2.2.6 *Synthetic Data*

Some work reformulates semi-supervised anomaly detection as a self-supervised problem by synthetically generating anomalies. Either parts of training images [49], [51], [52] or random images [50] are patched into normal images dynamically during the training process. The shape of the pasted patches are mostly simple rectangles with random aspect ratio [49], [51], [52] or come from a noise generator as proposed by Zavrtani et al. [50]. While [51], [52] uses the corresponding synthetic masks to optimize a segmentation map which is postprocessed for image-level detection in inference, [50] and [49] additionally or optionally optimize an image-level classification. While these approaches often provide a clear segmentation map and perform solidly on most benchmarks, they are not considered robust as the distributional shift of a real occurring anomaly type to synthetic examples severely affects the detection.

2.2.7 *Methods for Multivariate Machine Data*

In the following, we provide an overview of past work regarding semi-supervised AD methods in the context of robotic applications. Note that we focus on machine data where no external sensors are used.

Chen et al. [71] and Sölch et al. [72] introduce adapted versions of variational autoencoders [73] to detect anomalies using the reconstruction error which resemble the image-based approaches described in Subsection 2.2.3. While [71] is based on convolutional layers, [72] integrates residual neural networks (RNNs [74]). Park et al. [75] makes also use of RNN-based VAEs by integrating LSTMs, however, it utilizes the evidence lower bound (ELBO) as the anomaly score. Learning a progress-based latent space that can be partitioned by only a small set of nominal observations but is also built with unlabeled examples is proposed by Azzalini et al. [76]. Also using the reconstruction error as an anomaly indicator, Hornung et al. [77] identify anomalies by performing a principal component analysis (PCA [58]). It is assumed that anomalies cause high reconstruction errors due to existing variance on the non-principal components not modeled by the PCA. However, anomalies are not detected if their deviations are within the principal components.

Addressing the problem from a statistical perspective, [78] and [79] try to model the distribution of normal samples similar to the approaches in Subsection 2.2.5. Khalastchi et al. [78] assume the data to be Gaussian distributed and apply the Mahalanobis distance [70] within a sliding window as an anomaly score. Romeres et al. [79] evaluate Gaussian process models and several extensions of these. This necessitates identifying suitable kernels to ensure the validity of the Gaussian process assumption. In contrast, our baseline models the distribution of normal samples while being capable of handling any data distribution without any hard assumptions or manual kernel search.

Zhang et al. [80] analyze the time series with an autoregressive model that predicts future torques. The error between this prediction and the actual signals should identify anomalies for real-time collision detection. Again, more complex mechanisms cannot be modeled as it relies on linear models.

Park et al.[81] employ Hidden Markov Models (HMMs[82]) by leveraging the observation probability to detect unusual events subsequent to fitting an HMM to normal data, a process further enhanced through Gaussian process regression [83]. Azzalini et al.[84] propose two alternative approaches for HMMs: The online approach utilizes the Hellinger distance[85] to compute the emission distribution between normal data and test data in the current state. In an offline version, a distinct HMM is fitted on test data, and the states are aligned with those of the normal data, prior to applying the Hellinger distance between these models as an anomaly score.

3

DATASETS AND METRICS

This chapter summarizes the data on which the methods of this thesis are evaluated. In Sections 3.1 to 3.3, we introduce the three image datasets *MVTec AD* [7], *MVTec 3D-AD* [9] and *Magnetic Tile Defects* [11] which are all obtained from the literature. In Chapter 7, we present our own dataset voraus-AD, including time series of robot machine data. At the end of this chapter in Section 3.4, we give detailed information about the evaluation metrics used to assess the performance of the anomaly detection models.

3.1 MVTEC AD

The MVTec Anomaly Detection (MVTec AD) dataset [7], introduced by Bergmann et al. in 2019, serves as the most common image dataset for AD in the industrial context. It addresses the challenge of detecting defects in product images having only normal images in training without any prior knowledge about the defects in the test set. MVTec AD contains in total of 5354 high-resolution RGB images from 10 object and 5 texture categories. The categories are from various industries, including electronics (*cable*, *transistor*), medicine (*capsule*, *pill*), consumables (*hazelnut*, *bottle*), materials (*wood*, *tile*, *grid*) and textiles (*zipper*, *carpet*). All defect types, namely 1 to 7 per category, were manually induced. The defects include deviations in color, orientation and shape, as well as common errors such as scratches, cracks, dents and bends. In addition to the labels *normal* vs. *anomaly*, the test set also provides the defect type and a segmentation mask of the defect. The anomalous images may also have multiple defect types or defect regions per instance. Figure 3.1 visualizes some anomalies and their corresponding segmentation mask. Table 3.1 shows detailed information about the dataset statistics, broken down into the different categories. The number of training samples per category ranges from 60 to 320, which makes AD challenging as this number is comparatively low for machine learning purposes. The dataset simulates a controlled environment of a production floor, meaning that the images were taken from the same angle, camera and lighting.

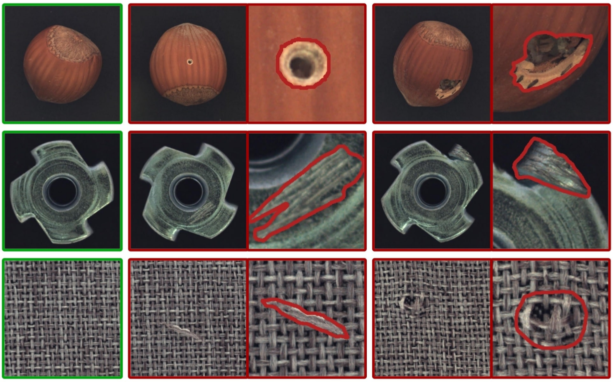


Figure 3.1: Samples of defect-free (green framed) and defective images (red framed) from the categories *hazelnut*, *metal nut* and *carpet* in MVTec AD. Image is taken from [7].

Type	Category	# Train	# Test (good)	# Test (defective)	# Defect groups	# Defect regions	Image side length
Textures	Carpet	280	28	89	5	97	1024
	Grid	264	21	57	5	170	1024
	Leather	245	32	92	5	99	1024
	Tile	230	33	84	5	86	840
	Wood	247	19	60	5	168	1024
Objects	Bottle	209	20	63	3	68	900
	Cable	224	58	92	8	151	1024
	Capsule	219	23	109	5	114	1000
	Hazelnut	391	40	70	4	136	1024
	Metal Nut	220	22	93	4	132	700
	Pill	267	26	141	7	245	800
	Screw	320	41	119	5	135	1024
	Toothbrush	60	12	30	1	66	1024
	Transistor	213	60	40	4	44	1024
	Zipper	240	32	119	7	177	1024
	Total	3629	467	1258	73	1888	-

Table 3.1: Statistical overview of the MVTec AD dataset. For each category, the number of images in training and test set is given together with additional details about the defects in the test set. The table is taken from [7].

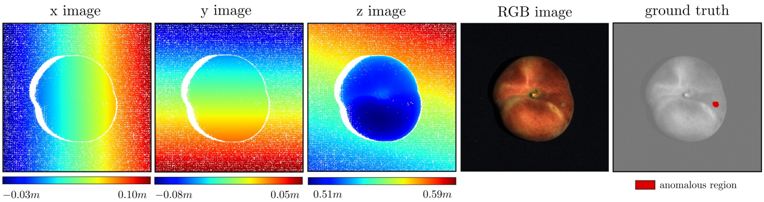


Figure 3.2: Visualization of the spatial channels (x, y, z) of the 3D scans in MVTec 3D-AD (left) which are aligned with the RGB image and the ground truth annotation (right). White pixels in the scans denote areas that were missed by the sensor. Image is taken from [9].

3.2 MVTEC 3D-AD

The MVTec 3D-AD dataset, similar to MVTec AD [7] and again published by Bergmann et al., is also designed for the detection of previously unknown defects on various products. In addition to RGB images, the dataset also provides a 3D scan of the object for each sample. This is motivated by the fact that some defects, such as surface irregularities, are often more difficult to detect on RGB images. Furthermore, 3D scans are more independent from the illumination. The high-resolution 3D scans were obtained using an industrial sensor that utilizes structured light which is projected on the object lying on a background plane that is tilted relative to the camera. The sensor captures the data in the form of a three-channel image corresponding to the x, y and z coordinates, which are measured with respect to the local camera coordinate frame and visualized in Figure 3.2. The (x, y, z) values in the image can be directly associated with the corresponding point cloud, thus providing a one-to-one mapping between the two representations. Note that some pixels are missing in 3D as the sensor may find no correspondence for a given pixel.

The dataset contains 10 object categories as shown in Figure 3.3, which include deformable and non-deformable objects and 5 cases of natural variation, which are considered more challenging. There are anomalies that are exclusively apparent in one of the two data domains, which motivates the use of models that can handle multimodal data. Table 3.2 shows a statistical overview of the dataset. The number of training samples (approx. 270 per category), test samples (approx. 120 per category) and defect types (3 to 5 per category) is similar to MVTec AD. Likewise, MVTec 3D-AD provides segmentations of the defects next to the image-level labels (normal or anomaly) in the test set.

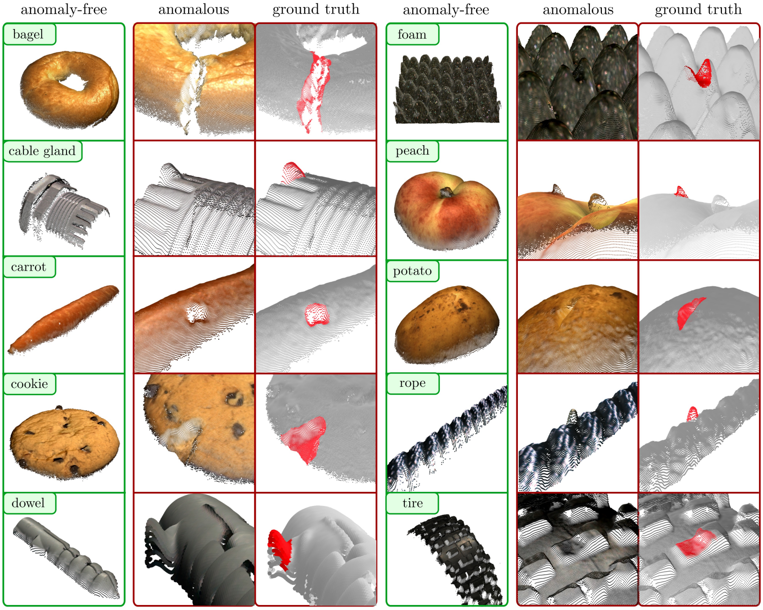


Figure 3.3: Visualization of examples of all classes in MVTec 3D-AD [9] by projecting the RGB values on the 3D point clouds. The green-frames left column shows an anomaly-free example of each category. To the right, a close-up of a defective example is shown along with the respective annotation of it. The background plane has been removed for better visibility. Image is taken from [9].

Category	# Train	# Val	# Test (good)	# Test (anomalous)	# Defect types	# Annotated regions	Image size (width × height)
bagel	244	22	22	88	4	112	800 × 800
cable gland	223	23	21	87	4	90	400 × 400
carrot	286	29	27	132	5	159	800 × 800
cookie	210	22	28	103	4	128	500 × 500
dowel	288	34	26	104	4	131	400 × 400
foam	236	27	20	80	4	115	900 × 900
peach	361	42	26	106	5	131	600 × 600
potato	300	33	22	92	4	115	800 × 800
rope	298	33	32	69	3	72	900 × 400
tire	210	29	25	87	4	95	600 × 800
total	2656	294	249	948	41	1148	

Table 3.2: Statistical overview of the MVTec 3D-AD dataset. The columns show the number of training, validation, and test images for each category. Additionally, the number of different defect types and annotated regions, along with the image resolution is given. The table is taken from [9].

3.3 MAGNETIC TILE DEFECTS

Magnetic Tile Defects [11] (MTD) comprises grayscale images of magnetic tiles under different illuminations with and without defects. Magnetic tiles should provide a constant magnetic potential in engines. If the magnetic field is not consistent or stable, the motion of the conductor and therefore the overall engine performance can be affected. This can lead to reduced efficiency, increased wear and tear on the engine components, and potential damage to the engine itself.

The dataset was originally released for a standard supervised defect classification and converted to an anomaly detection dataset [12]. We split the 952 defect-free images randomly into a train and a test set, where the test set contains 20% of the defect-free data. All 392 defect images are used for testing. These show uneven or frayed regions, cracks, breaks and blowholes as anomalies, as seen in Figure 3.4. MTD offers a large variance within the defect-free examples due to the differences in illumination and non-uniform textures which is a more challenging setup compared to MVTec AD [7].

In addition to the image datasets presented in this chapter, we also created our own dataset *vorausAD* with time series of robot machine data, which is presented in Chapter 7.

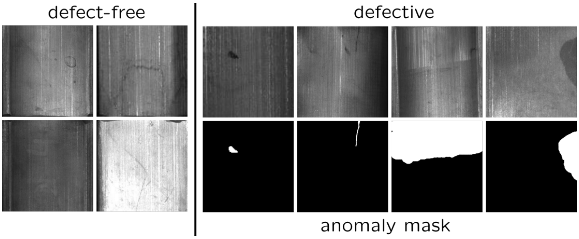


Figure 3.4: Samples of defect-free and defective images together with the corresponding anomaly masks from Magnetic Tile Defects [11].

3.4 METRICS

We measure and compare the performance of the proposed methods with the area under the receiver operating characteristic curve (AUROC) which is the most common evaluation metric for anomaly detection. While this work mainly focuses on image-level detection of anomalies, the pixel-level AUROC is also reported. The ROC curve reflects the trade-off between the true positive rate (TPR) and the false positive rate (FPR) for varying classification thresholds.

The TPR

$$TPR = \frac{TP}{P} \tag{3.1}$$

is the proportion of actual positive samples (P) that are correctly identified as positive (TP) by the model, while the FPR

$$FPR = \frac{FP}{N} \tag{3.2}$$

is the proportion of negative samples (N) that are incorrectly identified as positive (FP) by the model. In other words, TPR is the sensitivity or recall of the model, and FPR is the probability of a false alarm. The ROC curve is obtained by plotting TPR against FPR for different classification thresholds $\theta \in [-\infty, \infty]$ on the anomaly score, with each point on the curve representing a different threshold.

The AUROC is a scalar metric that quantifies the performance of a binary classification by computing the area under the ROC curve:

$$AUROC = \int_0^1 TPR(FPR^{-1}(u))\mathrm{d}u \tag{3.3}$$

where $TPR(FPR^{-1}(u))$ is the TPR at the threshold where the FPR is equal to u. The AUROC ranges between 0 and 1, where a higher AU-

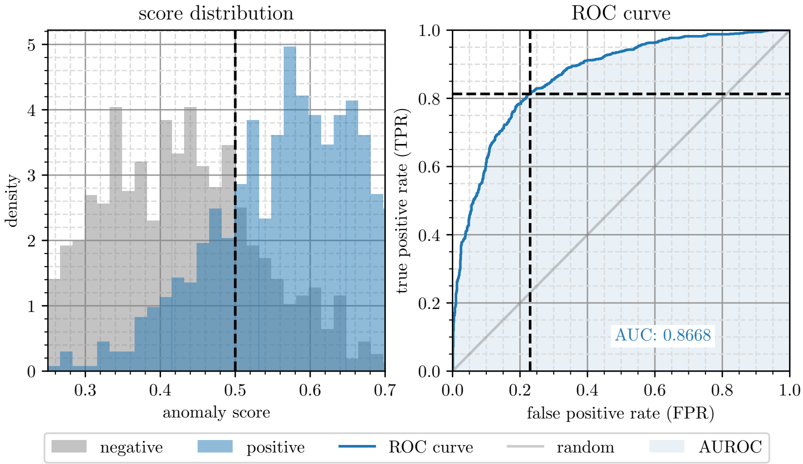


Figure 3.5: Distribution of anomaly scores (left) with their corresponding ROC curve (right). The points of the ROC curve are obtained by varying the decision threshold value, exemplarily marked by the dashed line.

ROC indicates better performance: A value of 0.5 indicates a random classifier, and a value of 1 indicates a perfect classifier. The AUROC is insensitive to the ratio of anomalies and the threshold which are both highly application-dependent. Figure 3.5 shows an example of a score distribution and the corresponding AUROC.

In the following chapters, we introduce the methodological contributions of this thesis. These are presented in chronological order of publication. The next chapter presents *DifferNet*, which provides a basis for the following chapters with the application of Normalizing Flows for image-based AD.

4

DIFFERNET

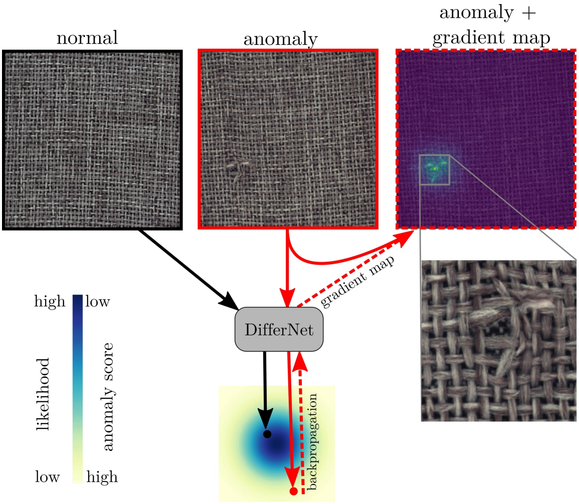


Figure 4.1: DifferNet estimates likelihoods of data being normal in order to detect anomalies. In contrast to the top left image being defect-free, the top middle image is estimated as unlikely to be intact due to the defect (see enlarged patch on the right side), resulting in a high anomaly score. Additionally, DifferNet identifies the defective region by propagating the likelihood back to the input which gives a gradient map (top right image).

This chapter is adapted from the work "Same Same But DifferNet: Semi-Supervised Defect Detection with Normalizing Flows" [12]. *DifferNet* is a method for image-based anomaly detection, which is based on estimating the density of features from normal data. It exploits the likelihood of an observation as an anomaly score having the intuition that anomalies should be unlikely according to the estimated density as illustrated in Figure 4.1. Image features are extracted by a pretrained neural network, which processes multiple scales of the image. The density is estimated by a Normalizing Flow, which is a bijective neural network that is optimized to map the distribution of features to a well-defined

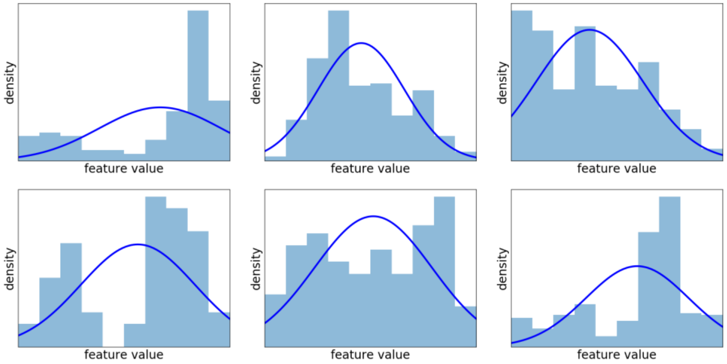


Figure 4.2: Histogram of single features from MVTec AD images extracted with EfficientNet [86]. Assuming a Gaussian, which is shown by the blue line, appears to be insufficient to capture the real data distribution.

distribution, making the likelihoods measurable. In addition to detecting anomalous instances, DifferNet also identifies anomalous regions within the image by propagating the anomaly score back to the input image and highlighting pixels with a high gradient magnitude. The implementation is publicly available at GitHub[1]github.com/marco-rudolph/differnet.

DifferNet makes use of Normalizing Flows for density estimation as they allow for approximating arbitrary distributions, unlike classical density estimators explained in Subsection 2.2.5, without making any assumptions about the given data distribution. This is an important property since a given data distribution may be in practice multimodal and/or not follow a particular parametrization which is commonly a multivariate Gaussian distribution [47], [48]. Figure 4.2 demonstrates that the Gaussian assumption already struggles to model the distribution of single features of a given dataset.

In addition to being used as density estimators, Normalizing Flows are also considered as generative models. As reviewed in Subsection 2.2.3, there are several approaches based on generative models such as autoencoders and GANs. These usually attempt to recreate the given input image and use the error between the input and the reconstruction as an anomaly indicator [63], [65]. However, the reconstruction error depends on many factors that are independent of anomalies. For example, high-frequency patterns are harder to reconstruct in general. Thus, the reconstruction error is only an indirect measure of how well the model

---

[1] github.com/marco-rudolph/differnet

could generate the given sample. In contrast, we directly measure how likely the given input is to be generated according to the model. Variational autoencoders [73] are also able to map to a given distribution, but unlike Normalizing Flows, they do not ensure the modelling of a valid distribution. While the model may map data from the training distribution to a desired distribution properly, out-of-distribution data is mapped arbitrarily and thus possibly to high-likelihood regions, as no injective mapping is ensured such that multiple inputs can be mapped to the same output. In contrast, a Normalizing Flow learns to maximize the probability mass of the training data, leaving low-probability regions to remain for out-of-distribution samples as they are constrained to be mapped elsewhere due to bijectivity.

DifferNet estimates the density of universal high-level features of a pretrained deep neural network, as these already carry significantly more semantics than the raw RGB data, on which a Normalizing Flow would learn local pixel correlations rather than consider the visual concepts [87]. However, these are not scale invariant [21], which is why we concatenate the features vectors of the image from different scales as shown in Figure 4.6, allowing us to cover a wider range of potentially useful features. Furthermore, as common convolutional-based feature extractors are generally not rotation invariant [21], [88], multiple rotations of the image during optimization and evaluation are processed, making our detection much more robust. Taking into account the feature sensitivity in terms of brightness and contrast, these two factors are partially varied as well.

4.1 NORMALIZING FLOWS

This section introduces Normalizing Flows on which the density estimation of our proposed methodology is based. First, a general overview of the concept of NFs and related work is given. Then, the real-NVP architecture used for DifferNet is explained.

4.1.1 *Introduction*

A Normalizing Flow (NF [66]) is a generative model that learns transformations between data distributions and well-defined densities as visualized in Figure 4.3. Unlike conventional neural networks, Normalizing Flows map bijectively. With this property, inference can be done in both directions: First, they are capable of assigning a likelihood to a given sample. Second, they enable data generation by sampling from the

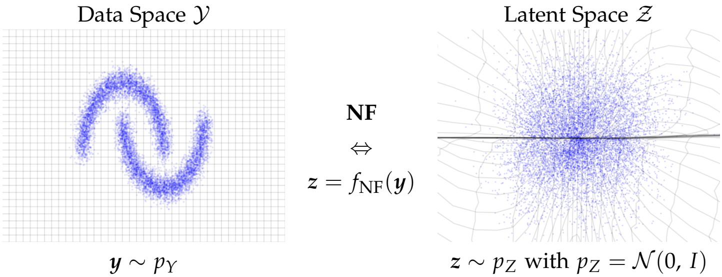


Figure 4.3: A Normalizing Flow learns a transformation between an unknown data distribution to a well-defined distribution, which is typically a Gaussian. Image is adapted from [89].

modeled distribution. The bijectivity is ensured by implementing a series of K invertible transformations f_i with $i = 1, ..., K$ on the input $\boldsymbol{y}$ as

$$z = f_{\mathrm{NF}}(\boldsymbol{y}) = f_K \circ f_{K-1} \circ \ldots \circ f_1(\boldsymbol{y}) \tag{4.1}$$

with $z \in \mathcal{Z}$ as the latent output and $\circ$ as the function composition operator. The inverse of this mapping is given by passing a z in reverse order through the inverse of these transformations

$$\boldsymbol{y} = f_{\mathrm{NF}}^{-1}(z) = f_1^{-1} \circ f_2^{-1} \circ \ldots \circ f_K^{-1}(z). \tag{4.2}$$

The bijectivity of the transformations ensures that there is a one-to-one mapping between the input $\boldsymbol{y}$ and the latent output z. This property, along with the requirement for differentiability of each transformation, allows us to compute the likelihood of a given sample by evaluating the density of the latent space distribution, which is typically assumed to be a simple distribution (e.g., a standard Gaussian), and the Jacobian of f_{NF}. Based on the change-of-variable formula

$$p_Y(\boldsymbol{y}) = p_Z(z) \left| \det\left(\frac{\partial z}{\partial \boldsymbol{y}} \right) \right| \tag{4.3}$$

we can factorize the Jacobian determinant $\det\left(\frac{\partial z}{\partial y}\right)$ [89], defining the output after the i-th transformation h_i as

$$h_i = f_i \circ f_{i-1} \circ \ldots \circ f_1(\boldsymbol{y}) \tag{4.4}$$

to formulate the likelihood of a data sample $\boldsymbol{y}$ as

$$p_Y(\boldsymbol{y}) = p_Z(\boldsymbol{z}) \left| \prod_{i=1}^{K} \det\left(\frac{\partial h_i}{\partial h_{i-1}}\right) \right| \tag{4.5}$$

with $\det\left(\frac{\partial h_i}{\partial h_{i-1}}\right)$ being the Jacobian determinant of the transformation f_i. Due to this relation, the Jacobian determinant of each f_i should be efficiently computable.

4.1.2 *Related Work*

A common class of NFs are masked autoregressive flows (MAFs) as *MADE* (Germain et al. [90]) which makes use of the Bayesian chain rule to decompose the density. So far, masked autoregressive flows have been used to learn distributions of large datasets containing mostly small images. In contrast, we capture the distribution of a comparably small number of images at a high resolution. Masked autoregressive flows compute likelihoods fast, but are slow at sampling. Inverse autoregressive flows, proposed by Kingma et al. [91], show the exact opposite behavior. Real-NVP [89] can be seen as a special inverse autoregressive flow which is simplified such that both forward and backward pass is processed quickly. It utilizes affine transformations as f_i which enables a fast computation of the Jacobian which is needed for the likelihood computation in Eq. 4.3.

Ardizzone et al. [92] propose to use NFs for inverse problems where non-deterministic backward processes of deterministic forward processes are to be predicted. They explicitly model the deterministic and non-deterministic variables in the mapping and refer to their Real-NVP-based network as *Invertible Neural Network* (INN). Due to the invertibility, the stochastic backward process is indirectly learned by training the fixed forward process. In this way, inverse kinematics and other real-world problems in medicine and astronomy are modeled. Ardizzone et al. propose a conditional variant of this approach in a follow-up work [93]. As an application, the ambiguous problem of image coloring is addressed by extending the INN concept with the grayscale image as a condition. Fig. 4.4 shows some results obtained by this method. Wehrbein et al. [18] utilized the conditional INN approach for the ambigious problem of 3D Pose estimation from 2D keypoint detections. It is shown that the uncertainty in prediction can be obtained from the set of the sampled 3D hypotheses.

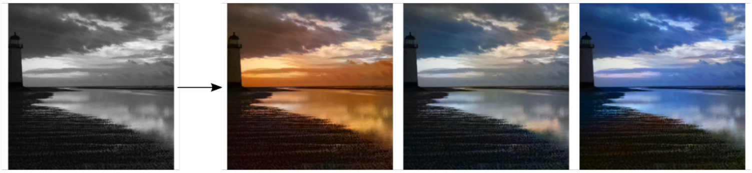


Figure 4.4: Diverse image colorization results obtained by a conditional INN [93].

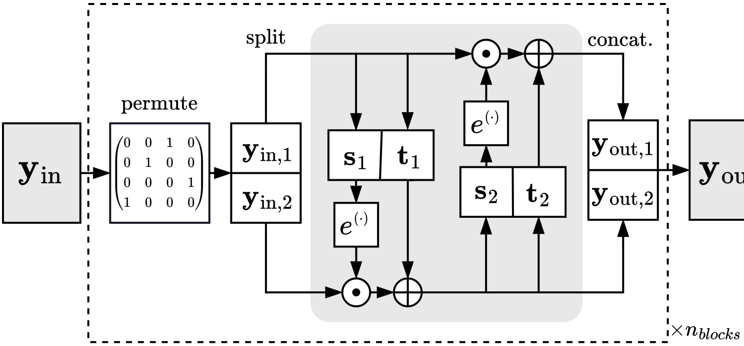

split
concat.
permute
$\mathbf{y}_{\text{in}}$
$\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{pmatrix}$
$\mathbf{y}_{\text{in},1}$
$\mathbf{y}_{\text{in},2}$
$\mathbf{s}_1$
$\mathbf{t}_1$
$e^{(\cdot)}$
$e^{(\cdot)}$
$\mathbf{s}_2$
$\mathbf{t}_2$
$\mathbf{y}_{\text{out},1}$
$\mathbf{y}_{\text{out},2}$
$\mathbf{y}_{\text{out}}$
$\times n_{blocks}$

Figure 4.5: Architecture of one block in the Real-NVP architecture: After a fixed random permutation, the input is split into two parts that regress scale and shift parameters to transform their respective counterpart. Symbols $\odot$ and $\oplus$ denote element-wise multiplication and addition, respectively.

The property of Normalizing Flows as an adequate estimator of probability densities to detect anomalies has not raised much attention in previous work to this point, although some works present promising results using Real-NVP and MADE [94]–[96]. However, none of the works deal with visual data.

4.1.3 *Real-NVP*

Real-NVP [89] is a specific NF architecture that we used to realize the bijective transformation for density estimation in DifferNet. This architecture is composed of so-called *coupling blocks* to implement the series of invertible transformations $f_1, f_2, ... f_n$ as in Eq. 4.1. The detailed structure of one block is shown in Figure 4.7. The entire mapping of the flow f_{NF} is a chain of such blocks.

Inside the block, the input is first randomly permuted in a fixed manner which is defined individually for each block before training. After that, the input y_{in} is split into $y_{\text{in},1}$ and $y_{\text{in},2}$ which are usually of the same size. These parts are used to manipulate each other by calculating element-wise scale and shift coefficients in the subnetworks s and t; these manipulations are applied to their respective counterpart successively. The scale and shift operations are described by

$$\begin{aligned} y_{\text{out},2} &= y_{\text{in},2} \odot e^{s_1(y_{\text{in},1})} + t_1(y_{\text{in},1}) \\ y_{\text{out},1} &= y_{\text{in},1} \odot e^{s_2(y_{\text{out},2})} + t_2(y_{\text{out},2}), \end{aligned} \tag{4.6}$$

with $\odot$ as the element-wise product. It can be shown that this computation is invertible by turning additions into subtractions and multiplications into divisions (or by using the negative exponent) as

$$\begin{aligned} y_{\text{in},1} &= (y_{\text{out},1} - t_2(y_{\text{out},2})) \odot e^{-s_2(y_{\text{out},2})} \\ y_{\text{in},2} &= (y_{\text{out},2} - t_1(y_{\text{in},1})) \odot e^{-s_1(y_{\text{in},1})}. \end{aligned} \tag{4.7}$$

Using an exponential function before scaling preserves the invertibility by ensuring non-zero coefficients. The internal functions s and t can be realized by any differentiable function and are usually learned with a neural network.

Each block first performs a predefined random permutation on the features to allow each dimension to affect all other dimensions at some point. The output of one coupling block is given by the concatenation of $y_{\text{out},1}$ and $y_{\text{out},2}$. Note that the input, which may be n-dimensional, could be permuted, split and concatenated along any axis and may be reshaped arbitrarily between blocks.

4.2 METHOD

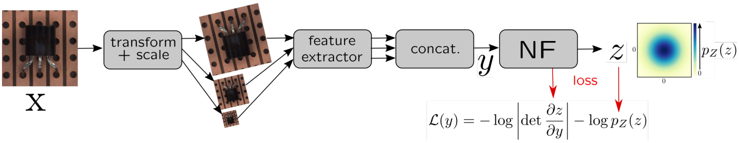

X
transform + scale
feature extractor
concat.
y
NF
z
loss
$p_Z(z)$
0
0
0
$\mathcal{L}(y) = -\log \left|\det \frac{\partial z}{\partial y}\right| - \log p_Z(z)$

Figure 4.6: Overview of our pipeline: Multiple scales of a transformed input image are fed into a feature extractor. The distribution of its concatenated outputs is captured by transforming it via a Normalizing Flow (NF) into a normal distribution by maximum likelihood training.

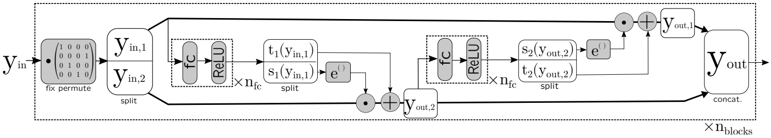


Figure 4.7: Realization of a Real-NVP block in DifferNet. The internal networks s and t are implemented via a single MLP whose output is split. Symbols $\odot$ and $\oplus$ denote element-wise multiplication and addition, respectively. Numerical operations are symbolized by grey blocks. White blocks contain variable names.

Figure 4.6 shows an overview of our pipeline. Our method is based on density estimation of image features $\boldsymbol{y} \in \mathcal{Y}$ from the anomaly-free training images $\boldsymbol{x} \in \mathcal{X}$ with $\boldsymbol{x} \sim X$ and $\boldsymbol{y} \sim Y$. Let $f_{\text{ex}} : \mathcal{X} \to \mathcal{Y}$ be the mapping of a pretrained feature extractor which is not further optimized. The estimation of $p_Y(\boldsymbol{y})$, provided by $f_{\text{ex}}(\boldsymbol{x})$, is achieved by mapping from Y to a latent space Z – with a well-defined distribution $p_Z(\boldsymbol{z})$ – by applying a Normalizing Flow $f_{\text{NF}} : Y \to Z$. Likelihoods for image samples are directly calculated from $p_Z(\boldsymbol{z})$. Features of anomalous samples should be out of distribution and hence have lower likelihoods than normal images. Likelihoods of multiple transforms on the image are maximized in training and used in inference for a robust prediction of the anomaly score. To capture structures at different scales and thus have a more descriptive representation in $\boldsymbol{y}$, we define the output of f_{ex} as the concatenation of features at 3 scales.

4.2.1 *Architecture*

We use the Real-NVP architecture for the NF which is described in subsection 4.1.3. The network is fed with vectors, which are the features for the first block and are randomly permuted and equally split at the beginning of each block. The internal networks s_i and t_i are implemented as a single fully connected network with ReLU nonlinearities. This network provides both the scaling and translation coefficients which are obtained by splitting the output (see Figure 4.7).

Similar to Ardizzone et al. [93], we apply soft-clamping to the values of s to preserve model stability which is crucial in our case for better convergence. This is achieved by using the activation

$$\sigma_\alpha(s) = \frac{2\alpha}{\pi} \arctan \frac{s}{\alpha} \tag{4.8}$$

as the last layer of s. This prevents large scaling components by restricting them to the interval $(-\alpha, \alpha)$.

4.2.2 *Training*

The goal during training is to find parameters for f_{NF} that maximize likelihoods for extracted features $\boldsymbol{y} \sim Y$ which are quantifiable in Z. With the mapping $\boldsymbol{z} = f_{\text{NF}}(\boldsymbol{y})$ and according to the change-of-variables formula Eq. 4.3, we describe the optimization problem as maximizing

$$p_Y(\boldsymbol{y}) = p_Z(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \right|. \tag{4.9}$$

This is equivalent to maximizing the log-likelihood, which is more convenient by simplifying $\log p_Z(\boldsymbol{z})$ in our case of a Gaussian Z to

$$\begin{aligned} \log p_Z(\boldsymbol{z}) &= \log \left(\frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{1}{2} \|\boldsymbol{z}\|_2^2} \right) \\ &= C - \frac{\|\boldsymbol{z}\|_2^2}{2} \end{aligned} \tag{4.10}$$

with d as $\dim(\boldsymbol{z})$ and $C = -\frac{d}{2} \log 2\pi$ as a constant which can be ignored for optimization. We use the negative log-likelihood loss $\mathcal{L}(\boldsymbol{y})$ to obtain a minimization problem:

$$\begin{aligned} \mathcal{L}(\boldsymbol{y}) &= -\log p_Y(\boldsymbol{y}) = -\log p_Z(\boldsymbol{z}) - \log \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \right| \\ \mathcal{L}(\boldsymbol{y}) &= \frac{\|\boldsymbol{z}\|_2^2}{2} - \log \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \right|. \end{aligned} \tag{4.11}$$

Intuitively, the flow f_{NF} is optimized to map all $\boldsymbol{y}$ as close as possible to $\boldsymbol{z} = \boldsymbol{0}$ while penalizing trivial solutions with scaling coefficients close to zero[2]. The latter is ensured by the negative log determinant of the Jacobian $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}}$ in $\mathcal{L}(\boldsymbol{y})$. In our case, the log determinant of the Jacobian is the sum of scaling coefficients before exponentiation.

During Training, $\mathcal{L}(\boldsymbol{y})$ is optimized for features $\boldsymbol{y}$ which are extracted from various transformations of the training images for a fixed epoch length. Subsection 4.3.1 describes the training in more detail.

---

[2] The exponentiation inhibits the coefficients from being zero.

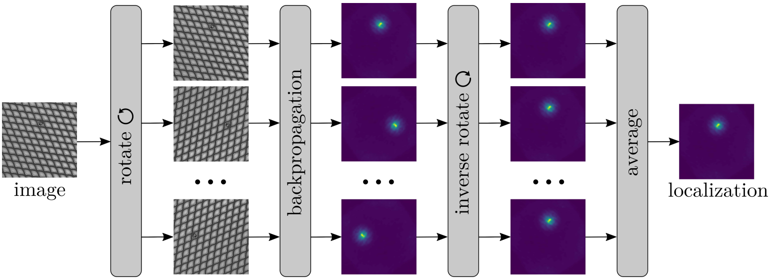


Figure 4.8: Visualization of the procedure for localization: First, a backpropagation of several rotations of the image is performed up to the input. The gradients are translated into a map by summing the absolute gradient of each input channel per pixel. These maps are rotated back and aggregated by averaging.

4.2.3 *Scoring Function*

We use the calculated likelihoods as a criterion to classify a sample as anomalous or normal. To get a robust anomaly score $\tau(\boldsymbol{x})$, the negative log-likelihoods using multiple transformations $T_i(\boldsymbol{x}) \in \mathcal{T}$ of an image $\boldsymbol{x}$ are averaged:

$$\tau(\boldsymbol{x}) = \mathbb{E}_{T_i \in \mathcal{T}}[-\log p_Z(f_{\text{NF}}(f_{\text{ex}}(T_i(\boldsymbol{x}))))]. \tag{4.12}$$

As $\mathcal{T}$ we choose rotations and manipulations of brightness and contrast. An image is classified as anomalous if the anomaly score $\tau(\boldsymbol{x})$ is above the threshold value θ. Thus, the decision can be expressed as

$$\mathcal{A}(\boldsymbol{x}) = \begin{cases} 1 & \text{for } \tau(\boldsymbol{x}) \geq \theta \\ 0 & \text{for } \tau(\boldsymbol{x}) < \theta \end{cases}, \tag{4.13}$$

where $\mathcal{A}(\boldsymbol{x}) = 1$ indicates an anomaly. In Section 4.3, we vary θ to calculate the *Receiver Operating Characteristic* (ROC).

4.2.4 *Localization*

In contrast to several other approaches, DifferNet is not optimized for localizing the defects on the image. Nevertheless, it localizes areas where anomalous features occur as it allows for propagating the negative log-likelihood $\mathcal{L}$ back to the input image $\boldsymbol{x}$. The gradient $\nabla \boldsymbol{x}_c$ of each input channel $\boldsymbol{x}_c$ is a value indicating how much the pixels influence the error which relates to an anomaly. For better visibility, we blur these gradients

	Method	GeoTrans [97]	GANomaly [41]	DSEBM [64]	OCSVM* [54]	1-NN* [35]	**DifferNet (ours)**	DifferNet (16 shots)
Textures	Grid	61.9	70.8	71.7	41.0	55.7	**84.0**	65.8
	Leather	84.1	84.2	41.6	88.0	90.3	**97.1**	92.9
	Tile	41.7	79.4	69.0	87.6	96.9	**99.4**	98.9
	Carpet	43.7	69.9	41.3	62.7	81.1	**92.9**	77.0
	Wood	61.1	83.4	95.2	95.3	93.4	**99.8**	99.2
Objects	Bottle	74.4	89.2	81.8	**99.0**	98.7	**99,0**	98.5
	Capsule	67.0	73.2	59.4	54.4	71.1	**86.9**	61.4
	Pill	63.0	74.3	80.6	72.9	83.7	**88.8**	65.1
	Transistor	86.9	79.2	74.1	56.7	75.6	**91.1**	76.6
	Zipper	82.0	74.5	58.4	51.7	88.6	**95.1**	88.3
	Cable	78.3	75.7	68.5	80.3	88.5	**95.9**	86.4
	Hazelnut	35.9	78.5	76.2	91.1	97.9	**99.3**	97.3
	Metal Nut	81.3	70.0	67.9	61.1	76.7	**96.1**	77.7
	Screw	50.0	74.6	**99.9**	74.7	67.0	96.3	75.9
	Toothbrush	97.2	65.3	78.1	61.9	91.9	**98.6**	92.3
	Average	67.2	76.2	70.9	71.9	83.9	**94.7**	87.3

Table 4.1: Area under ROC in % for detected anomalies of all categories of MVTec AD [7] by the time of the submission in 2020. DifferNet could set a new state of the art for most categories and on average. Best results are in bold, second best underlined. OCSVM and 1-NN are performed on the same features as DifferNet. *16 shots* denotes a model trained on only 16 images.

with a Gaussian kernel $\boldsymbol{G}$ and sum the absolute values over the channels C according to

$$g_x = \sum_{c \in C} |\boldsymbol{G} * \nabla \boldsymbol{x}_c|, \tag{4.14}$$

with $*$ as 2D convolution and $|\cdot|$ as the element-wise absolute value, which results in the gradient map g_x. Averaging the maps of multiple rotations of one single image – after rotating back the obtained maps – gives a robust localization. This procedure is visualized in Figure 4.8.

4.3 EXPERIMENTS

Within this subsection, we detail the experimental methodology and outcomes of the proposed approach. This includes implementation details, a quantitative evaluation of detection performance, a qualitative analysis of localization, and ablation studies to analyze the characteristics of DifferNet.

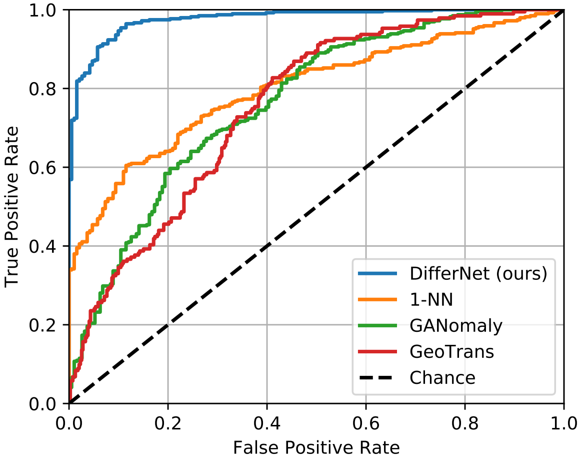


Figure 4.9: ROC-Curve for different methods for detecting defects in MTD by the time of the submission in 2020. DifferNet is significantly more accurate in detecting the defects compared to other approaches.

4.3.1 *Implementation Details*

For all experiments, the convolutional part of AlexNet [98] serves as the feature extractor and global average pooling is applied on the feature maps at each scale. We tested more complex topologies, for instance, ResNet [99] and VGG [100], but did not observe better performance. The feature extractor is pretrained on ImageNet [101] and remains fixed during training. We use features at 3 scales with input image sizes of 448×448, 224×224 and 112×112 pixels - resulting in $3 \cdot 256 = 768$ features. The Normalizing Flow consists of 8 coupling blocks with fully connected networks as internal functions s and t. These include 3 hidden dense layers with a size of 2048 neurons and ReLU activations. The clamping parameter is set to $\alpha = 3$. For training, we use the Adam Optimizer [102] with the author-suggested β- parameters and a learning rate of $2 \cdot 10^{-4}$. As transformations $\mathcal{T}$, random rotations, which are uniformly distributed in the interval $[0, 2\pi]$, are applied. In addition, we manipulated the contrast and brightness of the magnetic tiles with a uniformly distributed random factor in the interval $[0.85, 1.15]$. In each case of training and inference, the same transformations are applied. The models are trained for 192 epochs with a batch size of 96.

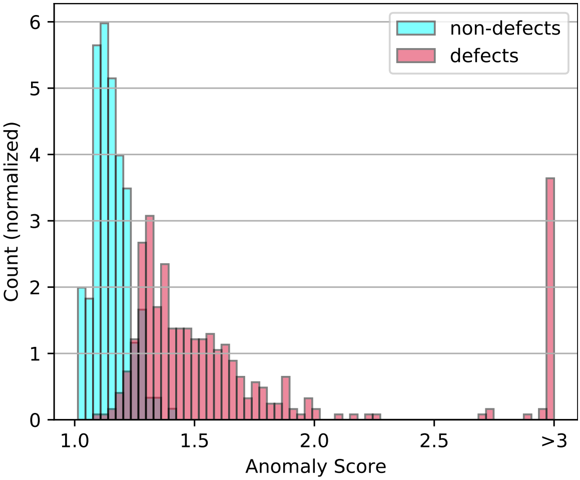


Figure 4.10: Normalized histogram of DifferNet's anomaly scores for the test images of MTD. As can be seen, the score is a reliable indicator for defects except for a narrow range of borderline cases. Note that the rightmost bar summarizes all scores above 3.

4.3.2 *Detection*

For reporting the performance of our method regarding the detection of anomalies, we follow [41] and compute the Area Under Receiver Operator Characteristics (AUROC) depending on the scoring function described in Section 3.4. It measures the area under the true positive rate as a function of the false positive rate. The AUROC metric is not sensitive to any threshold or the percentage of anomalies in the test set. Besides other anomaly detection methods, we compare our method to the baselines one-class SVM (OCSVM) [54] and the distance to the nearest neighbor (1-NN) after PCA reduction to 64 dimensions and z-score normalization [35]. Note that both methods OCSVM and 1-NN are adapted to our setting: We used every technique of our pipeline (see Figure 4.1) but replaced the Normalizing Flow with them. Consequently, the mean score over several transformations is evaluated. Apart from these approaches and some state-of-the-art models, we compare our method with GeoTrans [97] which computes an anomaly score based on the classification error of applied transformations. Table 4.1 shows the results for MVTec AD. Compared to other approaches, our method outperforms existing methods in almost every category, up to a large margin of 15%. In all of the 15 categories, our method achieves an AUROC

Method	AUROC [%]
GeoTrans[97]	75.5
GANomaly[41]	76.6
DSEBM [64]	57.2
ADGAN [103]	46.4
OCSVM [54]	58.7
1-NN [35]	80.0
DifferNet (ours)	**97.7**

Table 4.2: Area under ROC in % for detecting anomalies on MTD

of at minimum 84%, which shows that our approach is not limited to a specific set of defects or features. The fact that 1-NN outperforms other competitors except us, demonstrates that our feature extraction and evaluation is well-suited for the problem.

We can observe similar characteristics on MTD, seen in Table 4.2. The ROC-Curve in Figure 4.9 shows that our method provides a much higher true positive rate for any false positive rate. DifferNet achieves a recall of about 50% without any false positives among 191 defect-free test images. The histogram of anomaly scores is visualized in Figure 4.10. There is a large subset of defective samples whose scores differ significantly from all scores of non-defective samples. The assignment of extremely high scores without any false positives is a characteristic of our method and can be similarly observed for other evaluated product categories.

4.3.3 *Localization*

Results of the localization procedure described in Subsection 4.2.4 are shown in Figure 4.11. The localizations are accurate for many types, sizes and shapes of anomalies; despite the average pooling of feature maps before being processed by the Normalizing Flow. Our architecture produces meaningful gradients which can be explained by the model's architecture: First, AlexNet is relatively shallow such that noisy or vanishing gradients are prevented. Second, the bijectivity of the Normalizing Flow causes a direct relation between all image features y and all values of z with non-zero gradients. The gradients tend to appear speckled for larger anomalous regions. We conject that the reason is that pixels, whose features influencing the anomaly score, are usually not located evenly distributed in the corresponding region. For some cases, the localization does not sharply segment the error as in Fig. 4.12. However, our method

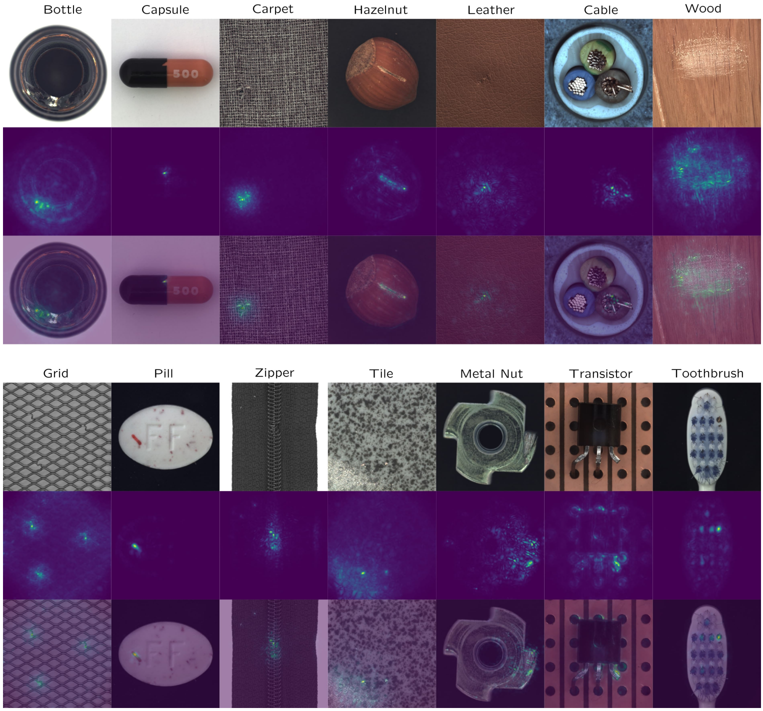


Figure 4.11: Localization of anomalous regions of different categories in MVTec AD. The upper rows show the original anomaly images, the mid rows the localizations provided by DifferNet and the lower rows the superimposition of both. They were generated by backpropagating the negative log-likelihood loss to the input image.

enables the human to perceive the defective region in most cases and interpret which areas influenced the network's decision to what extent.

4.3.4 *Ablation Studies*

To quantify the effects of individual strategies used in our work, we performed an ablation study by comparing the performance on MVTec AD [7] when modifying the strategies. In addition, the model's behavior for different characteristics of the training set is analyzed.

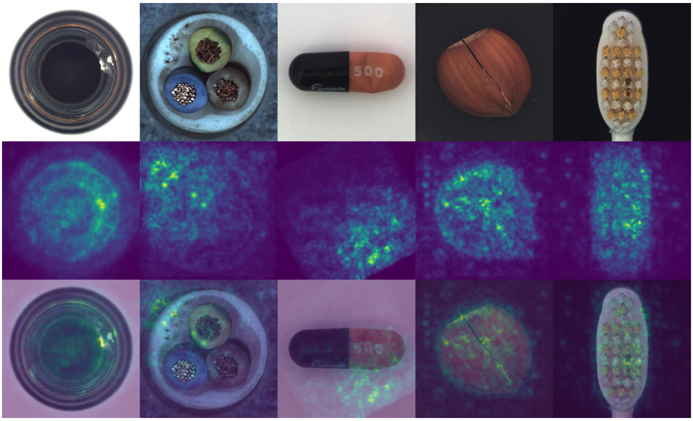


Figure 4.12: Failure cases of the localization. Although the error is highlighted, there is a noise pattern with high gradients in places in the surrounding area.

Config.	A	B	C	D	E	F	G
multi-scale	✗	✗	✓	✓	✓	✓	✓
train transf.	✗	✓	✗	✓	✓	✓	✓
# test transf.	1	64	1	1	4	16	64
AUROC [%]	84.4	90.2	86.6	86.5	91.6	94.1	**94.7**

Table 4.3: Average detection performance for all categories of MVTec AD when modifying our proposed training and evaluation strategy. The columns show parameter configurations named from A to F. Parameters that differ from our proposed configuration are underlined.

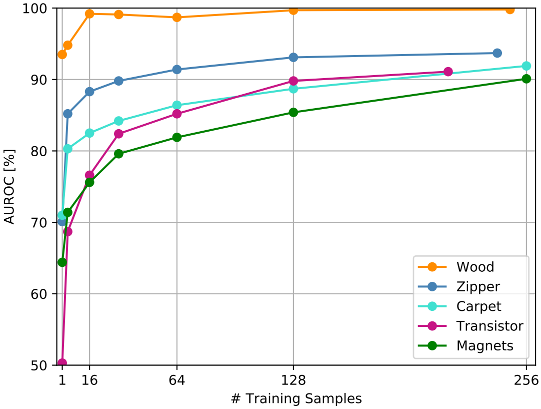


Figure 4.13: Detection performance of DifferNet, measured by AUROC, depending on the training set size of MTD and of some categories of MVTec AD.

PREPROCESSING PIPELINE AND EVALUATION

Table 4.3 compares the detection performance on MVTec AD for different configurations regarding multi-scaling, the usage of transformations in training and the number of used transformations T_i for evaluation. Having one test transformation means that only the original image was used for evaluation. Note that we outperform existing methods even without the proposed transformations and multi-scale strategy. Since relevant features could appear at any scale, it is beneficial to include features at multiple scales which is shown by an AUROC improvement of 4.7%. Having transformed samples in training is crucial as it enables multi-transform evaluation and helps with generalization and data augmentation. The similar performances of configurations C and D reflect that applying transformations in training is only useful if they are performed in inference as well. The more of these transformations are then used, the more meaningful the score is, as the rising performance of configurations D to G shows.

NUMBER OF TRAINING SAMPLES

We investigate the effect of the training set size on the detection performance as shown on Figure 4.13 and on the right of Table 4.1. The reported results are the average over three runs with different random subsets per training set size. It can be seen that our model and training procedure allow for a stable training even on small training sets. This can be explained by the usage of multiple transformations and the averaging of feature maps. Our model profits from this strategy which is a mixture of augmentation and compression. DifferNet requires only 16 training samples to outperform existing approaches that use the whole training set with 369 samples per class on average. For some classes, the subsets cannot represent the feature variation of normal samples.

MULTIMODALITY

The feature distributions of the evaluated categories are unimodal. We also investigated the performance on multimodal distributions. Therefore, a model was trained on all 15 categories of MVTec as training data. To capture this more complex distribution, we increased the number of coupling blocks to 12. This experiment led to a mean AUROC of 90.2% which shows that our method is able to handle multimodal distributions well. The regressing sub-blocks inside the NF appear to capture the modes and switch between them depending on their input.

4.4 CONCLUSION

This chapter presents *DifferNet* which detects defects on images by utilizing a normalizing-flow-based density estimation of image features at multiple scales. Likelihoods of several transformations of a single image are used to compute a robust anomaly score. Therefore, there is no need for a large amount of training samples. The design and scoring function is chosen such that image gradients can be exploited to localize defects. As shown, the method also scales to multimodal distributions which resemble many real-world settings.

4.5 OPEN PROBLEMS

Even though the approach seems effective via its theoretical foundation and some measure of robustness, several problems still remain: The average pooling of features into vectors makes the representation much more compact and facilitates the optimization and processing of the data, but introduces a massive loss of important local information. First, the averaging suppresses small-scale anomalies, and second, the local context of features cannot be included in the decision due to the information loss regarding neighborhoods. For example, a feature may in principle occur in non-anomalous images, but only in the neighborhood of certain other features. Thus, processing the entire feature maps would in principle allow for much more precise anomaly detection.

Furthermore, the method strongly benefits from the fact that different augmentations of the image are processed during the evaluation in order to detect or localize defects. While this can be solved by parallelization, it would be costly in many real-world industrial applications, where CPU-based embedded devices are today rather used than high-tech GPUs. Therefore, evaluation with only one network pass would be desirable.

In principle, localization enables a useful highlighting of many fault types, but struggles in some cases as it produces noisy maps in which the error is hardly identifiable as in Figure 4.12. Besides, backpropagation requires a potentially critical time and memory overhead. Ideally, regular inference should already bring a local interpretation of the anomaly score.

These issues motivate several changes in the methodology, which we implement in the following chapter with the extension *CS-Flow*.

5

CROSS-SCALE-FLOW

In the last chapter, we introduced DifferNet to show how NFs can be used to identify anomalous examples on image data by performing density estimation on previously extracted feature vectors. This chapter is largely based on the publication "Fully Convolutional Cross-Scale-Flows for Image-based Defect Detection" [13] which retains the principle from DifferNet of density estimation on image features while addressing the open problems listed in Section 4.5 by introducing a new flow architecture. The implementation is freely accessible at GitHub[1]github.com/marco-rudolph/cs-flow.

Instead of performing density estimation on feature vectors as in DifferNet, which causes a massive information loss due to the average pooling of feature maps, density estimation in CS-Flow is based on a set of multi-scale feature maps as visualized in Fig. 5.1. Estimation on the composite of different-sized feature maps is explicitly targeted since chapter 4.3.4 and subsequent experiments of this chapter show that multi-scale approaches have benefits for detection performance due to broader coverage of detectable features. We aim for interacting feature maps that enables the utilization of correlations of different-sized feature maps which is why the architecture is designed for parallel processing of multi-scale feature maps. The size of the respective maps should be maintained to allow for interactions inside the internal networks between fine-grained local features (on the larger maps) as well as more global features (on the smaller maps).

In CS-Flow, the inputs and outputs consist of a pyramid of multi-scale feature maps. On each scale, these are split along the channel dimension within the coupling blocks as in DifferNet, also resulting in feature pyramids. The internal networks receive these pyramids as input and output pyramids for scaling and shift coefficients. We keep the local structure of the feature maps to consider the local neighborhood for the occurrence of features. The fully connected networks from DifferNet are replaced with small fully convolutional subnetworks with parallel paths that act across scales: The addition of feature maps after performing resizing operations within the internal networks enables the interactions

---

[1] github.com/marco-rudolph/cs-flow

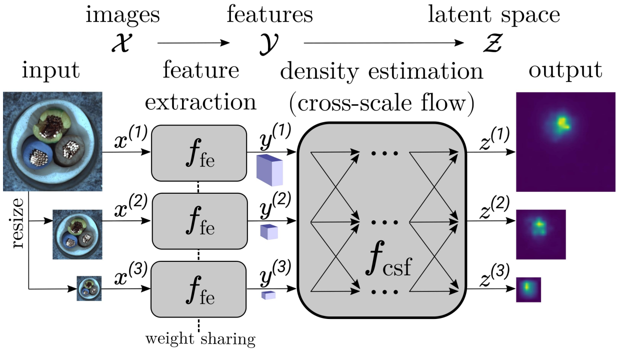


Figure 5.1: CS-Flow detects and localizes defects based on the density estimation of feature maps from the differently sized input images. We process the multi-scale feature maps jointly, using a fully convolutional Normalizing Flow with cross-connections between scales.

between the scales. By using fully convolutional internal networks in combination with shuffling, splitting, and recombining the channels within the coupling blocks, the local structure of the feature maps is maintained. In this way, the output maps give a direct indication of the occurrence of anomalies as they implicitly represent a likelihood for the respective image region. This is because all image regions as a composite are optimized to a Gaussian distribution for normal images. Feeding the flow with significantly more data, as feature maps have far more dimensions compared to feature vectors, and preserving more information while not having a massive increase in model parameters by using convolutions causes the network to be more robust such that the inference of multiple augmentations as for DifferNet is not necessary here. This saves a lot of computation time with better detection performance. The following chapter describes the method formally and in detail.

5.1 METHOD

To detect defects in images, we first learn a statistical model of features $\boldsymbol{y} \in \mathcal{Y}$ of defect-free images $\boldsymbol{x} \in \mathcal{X}$ similar to DifferNet in the previous chapter. During inference, we assign a likelihood to the input image $\boldsymbol{x}$ by using a density estimation on image features $\boldsymbol{y}$, assuming a low likelihood is an indicator of a defect. The density estimation is learned via a bijective mapping of the unknown distribution p_Y of the feature space $\mathcal{Y}$ to a latent space $\mathcal{Z}$ with a Gaussian distribution p_Z. Thus, as shown in Figure 5.1, our method is divided into the steps feature extraction $f_{\text{fe}} : \mathcal{X} \rightarrow \mathcal{Y}$ and density estimation $f_{\text{csf}} : \mathcal{Y} \rightarrow \mathcal{Z}$.

From the input image $\boldsymbol{x}$ we extract the features $\boldsymbol{y}$ by using a pretrained neural network $f_{\text{fe}}(\boldsymbol{x}) = \boldsymbol{y}$ which will remain unchanged during training. To have a more descriptive representation of $\boldsymbol{x}$, feature maps of different scales are included in $\boldsymbol{y}$ via extracting features from s different resolutions of the image. In contrast to DifferNet from Chapter 4, CS-Flow is able to perform density estimation on different scaled full-sized feature maps in parallel instead of concatenated feature vectors. Thus, important fine-grained positional and contextual information is better maintained. We define $\boldsymbol{y} = [y^{(1)}, ..., y^{(n_{\text{maps}})}]$ with $y^{(i)}$ as the 3D feature tensor of the image $x^{(i)}$ at scale $i \in \{1, ..., n_{\text{maps}}\}$. Our proposed cross-scale-flow f_{csf} transforms the feature tensors bijectively and in parallel to

$$f_{\text{csf}}(y^{(1)}, ..., y^{(n_{\text{maps}})}) = [z^{(1)}, ..., z^{(n_{\text{maps}})}] = \boldsymbol{z} \in \mathcal{Z} \tag{5.1}$$

with the same dimensionality[2] as $\boldsymbol{y}$. The likelihood $p_Z(z)$ is measured according to the target distribution which in our case is a multivariate standard normal distribution $\mathcal{N}(0, I)$.

We use the likelihood of $p_Z(\boldsymbol{z})$ to decide whether $\boldsymbol{x}$ is anomalous according to a threshold θ:

$$\mathcal{A}(\boldsymbol{x}) = \begin{cases} 1 & \text{for } p_Z(\boldsymbol{z}) < \theta \\ 0 & \textit{else} \end{cases}. \tag{5.2}$$

5.1.1 *Cross-Scale Flow*

The cross-scale flow is a chain of so-called *coupling blocks*, each performing affine transformations. As the base architecture of the coupling block we chose Real-NVP [89] as described in Subsection 4.1.3. In the following, we describe how we integrated different-sized feature tensors into this concept.

The detailed structure of one block with $s = 3$ is shown in Figure 5.2. Inside, each input tensor $y_{\text{in}}^{(i)}$ is first randomly permuted and evenly split across its channel dimension into the two parts $y_{\text{in},1}^{(i)}$ and $y_{\text{in},2}^{(i)}$. Both parts, each as a compound of the s scales, manipulate each other by regressing scale-wise and element-wise scale and shift parameters that are successively applied to their respective counterparts. Finally, we

---

[2] For better readability, in the following $\boldsymbol{z}$ without any index represents a vector which is the concatenation of the flattened tensors $[z^{(1)}, ..., z^{(n_{\text{maps}})}]$. The same applies for $\boldsymbol{y}$.

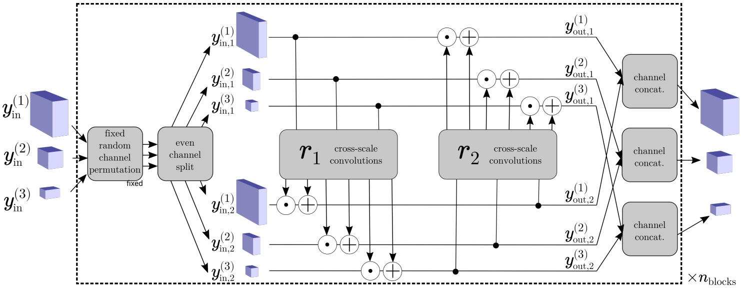


Figure 5.2: Architecture of one block inside the cross-scale flow: After a fixed random permutation, every input tensor is split into two parts across the channel dimension where each ensemble is used to estimate scale and shift parameters that transform the respective counterpart. Symbols $\odot$ and $\oplus$ denote element-wise multiplication and addition, respectively.

obtain the outputs $[y_{\text{out},1}^{(i)}, y_{\text{out},2}^{(i)}]$ which are concatenated for each scale to form the output of the block.

The scale and shift parameters are estimated by coupling block-individual subnetworks r_1 and r_2 whose output is split into $[s_1, t_1]$ and $[s_2, t_2]$ and is then used as follows:

$$\begin{aligned} y_{\text{out},2} &= y_{\text{in},2} \odot e^{\gamma_1 s_1(y_{\text{in},1})} + \gamma_1 t_1(y_{\text{in},1}) \\ y_{\text{out},1} &= y_{\text{in},1} \odot e^{\gamma_2 s_2(y_{\text{out},2})} + \gamma_2 t_2(y_{\text{out},2}), \end{aligned} \tag{5.3}$$

with $\odot$ as the element-wise product. To initialize the model stably, we introduce the learnable block-individual scalar coefficients γ_1 and γ_2. They are initialized to 0 and thus cause $y_{\text{out}} = y_{\text{in}}$. The affinity property is preserved by having non-zero scaling coefficients with the exponentiation in Equation 5.3. The internal networks r_1 and r_2 is implemented as a fully convolutional network that regresses both components by splitting the output (see Figure 5.3 for details of the architecture). Features are processed with one hidden layer per scale on which the number of channels is increased. Motivated by HRNet [104], we adjust the size of individual feature maps of different scales by bilinear upsampling or strode convolutions before aggregation by summation.

As for DifferNet in Chapter 4, we apply soft-clamping to the scale components s, as proposed by Ardizzone et al. [93], to preserve model stability in spite of the exponentiation. The clamping is applied as the last layer to the outputs s_1 and s_2 by the activation

$$\sigma_\alpha(s) = \frac{2\alpha}{\pi} \arctan \frac{s}{\alpha}. \tag{5.4}$$

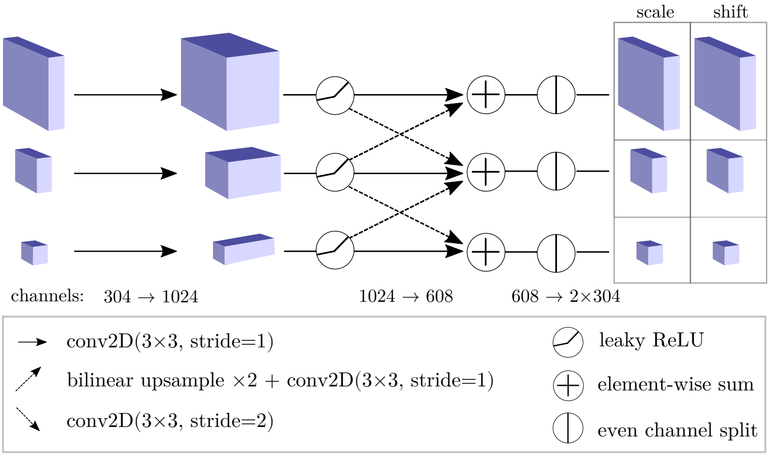


Figure 5.3: Architecture of the internal networks r inside the coupling blocks. Convolutions are performed at two levels, with cross-connections between scales at the second level. Feature map resizing is implemented by upsampling and strode convolutions. Aggregation is implemented by summation. The output is split across the channel dimension to obtain the scale and shift parameters.

This prevents extreme scaling components by restricting the values to the interval $(-\alpha, \alpha)$.

5.1.2 *Learning Objective*

During training, we want the cross-scale flow f_{csf} to maximize the likelihoods of feature tensors $p_Y(\boldsymbol{y})$ which we obtain by mapping them to the latent space $\mathcal{Z}$ where we model a well-defined density p_Z. As shown in Subsection 4.2.2, this results in minimizing

$$\mathcal{L}(\boldsymbol{y}) = -\log p_Y(\boldsymbol{y}) = \frac{\|z\|_2^2}{2} - \log \left| \det \frac{\partial z}{\partial \boldsymbol{y}} \right| \tag{5.5}$$

with $\left|\det \frac{\partial z}{\partial \boldsymbol{y}}\right|$ denoting the absolute determinant of the Jacobian. The logarithm of this term simplifies in our case to the sum of all values of s since the Jacobian of the element-wise product operator in Equation 5.3 is a diagonal matrix. The training is conducted over a fixed number of epochs. To stabilize it further, we limit the ℓ_2-norm of the gradients to 1. Subsection 5.2.1 describes the training in more detail.

5.1.3 *Localization*

In Chapter 4, the latent space of the Normalizing Flow has been used such that all entries of z are considered to produce a score at the image level. Since CS-Flow processes feature maps fully convolutionally, positional information is preserved. This allows for the interpretation of the output in terms of the likelihood of individual image regions, which in our application is the localization of the defect.

Analogous to the definition of the anomaly score of the entire image, we define an anomaly score for each local position (i, j) of the feature map $y^{(k)}$ by aggregating the values along the channel dimension with $\|z^k_{i,j}\|_2^2$. Thus, we can localize the defect by marking image regions with high norms in the output feature tensors z^s.

5.2 EXPERIMENTS

In this section, the experimental methodology and results of the proposed approach is presented. This comprises the implementation details, a quantitative evaluation of the detection performance, a qualitative analysis of the localization, and ablation studies to demonstrate the effectiveness of CS-Flow.

5.2.1 *Implementation Details*

We utilize the output of layer 36 of EfficientNet-B5 [86] as the feature extractor for all experiments as it provides feature maps having a good balance between feature semantic and spatial resolution. The feature extractor remains fixed during training after being trained on ImageNet [101]. For MVTec AD, we use features at $n_{\text{maps}} = 3$ scales with input image sizes of 768×768, 384×384 and 192×192 pixels - resulting in feature maps with spatial dimensions 24×24, 12×12 and 6×6 and each 304 channels. Due to the smaller original image size of MTD samples, we resized the images to 384×384, 192×192 and 96×96 pixels. We use $n_{\text{blocks}} = 4$ coupling blocks inside CS-Flow using 3×3 convolutional kernels in internal networks for the first 3 blocks and 5×5 kernels for the last block. The clamping parameter is set to $\alpha = 3$ and the negative slope of the leaky ReLU is set to 0.1. For optimization, we use Adam [102] with a learning rate of $2 \cdot 10^{-4}$, a weight decay of 10^{-5} and momentum values $\beta_1 = 0.5$ and $\beta_2 = 0.9$. We train our models with a batch size of 16 for a fixed number of 240 epochs for MVTec AD and 60 epochs for MTD, respectively, since there is no validation set to define a stopping criterion.

2020

	Category	GeoTrans [42]	GAN [97]	DSEBM [41]	1-NN [64]	DifferNet [47]	ARNet [35]	Rippel [12]	PaDiM [48]	**CS-Flow (ours)** (16 shots/full set)	
Textures	Grid	61.9	70.8	71.7	81.8	84.0	88.3	93.7	-	93.3	**99.0**
	Leather	84.1	84.2	41.6	**100**	97.1	86.2	**100**	-	**100**	**100**
	Tile	41.7	79.4	69.0	**100**	99.4	73.5	**100**	-	99.9	**100**
	Carpet	43.7	69.9	41.3	98.5	92.9	70.6	99.6	-	**100**	**100**
	Wood	61.1	83.4	95.2	95.8	99.8	92.3	99.3	-	99.5	**100**
	Avg. Text.	59.6	77.5	63.8	96.1	94.6	82.2	98.5	99.0	98.5	**99.8**
Objects	Bottle	74.4	89.2	81.8	99.6	99.0	94.1	99.0	-	**100**	99.8
	Capsule	67.0	73.2	59.4	89.4	86.9	68.1	96.3	-	83.1	**97.1**
	Pill	63.0	74.3	80.6	79.9	88.8	78.6	91.4	-	90.9	**98.6**
	Transistor	86.9	79.2	74.1	95.4	91.1	84.3	98.2	-	98.0	**99.3**
	Zipper	82.0	74.5	58.4	97.1	95.1	87.6	98.8	-	95.3	**99.7**
	Cable	78.3	75.7	68.5	95.1	95.9	83.2	**99.1**	-	94.4	**99.1**
	Hazelnut	35.9	78.5	76.2	98.2	99.3	85.5	**100**	-	97.9	99.6
	Metal Nut	81.3	70.0	67.9	91.1	96.1	66.7	97.4	-	**99.1**	**99.1**
	Screw	50.0	74.6	99.9	91.4	96.3	**100**	94.5	-	65.2	97.6
	Toothbrush	97.2	65.3	78.1	94.7	98.6	**100**	94.1	-	85.6	91.9
	Avg. Obj.	71.6	75.5	74.5	93.2	94.7	84.8	96.9	97.2	91.0	**98.2**
	Average	67.2	76.2	70.9	93.9	94.7	83.9	97.5	97.9	93.5	**98.7**

Table 5.1: Area under ROC in % for detecting defects of all categories of MVTec AD [7] by the time of the submission in 2021. CS-Flow could set a new state of the art for most categories and on average. Best results are in bold. *16 shots* denotes that a subset of only 16 random images per category was used in training. Besides the average value, detailed results of PaDiM [48] were not provided by the authors. Methods listed to the right of the central vertical line have been published after the submission of DifferNet.

Method	AUROC [%] ↑
GeoTrans [42]	75.5
GANomaly [41]	76.6
DSEBM [64]	57.2
1-NN [35]	97.8
DifferNet [12]	97.7
Rippel [47]	98.0
PaDiM [48]	98.7
CS-Flow (ours)	**99.3**

Table 5.2: Area under ROC in % for detecting anomalies on MTD.

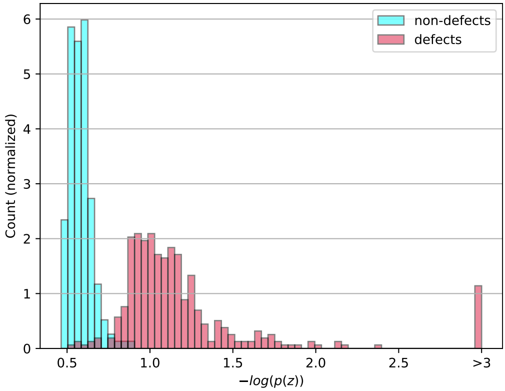


Figure 5.4: Distribution of negative log-likelihood for test images of MTD as a normalized histogram. By this criterion, the defective samples are almost completely separable from the non-defective samples. Note that for clarity, the rightmost bar summarizes all scores above 3.

A training run of one class of MVTec AD takes about 45 minutes on average using a NVIDIA RTX 2080 Ti.

5.2.2 *Detection*

In order to measure and compare the defect detection performance of our models, we follow [12] and calculate the area under ROC (AUROC) at image-level on the respective test sets as described in Section 3.4. The ROC (*Receiver Operating Characteristics*) curve relates the true positive rate to the false positive rate with respect to a parameter (in our case the threshold θ). Thus, it is invariant to the ratio of anomalies in the set and is therefore representative of realistic settings. Table 5.1 shows the defect detection performance of our method and other state-of-the-art works on the individual categories of MVTec AD. For a fair comparison, we evaluated [35] and [47] on the same multi-scale features as our method which improved their performance in every case. Here, we averaged the feature maps of different scales individually, resulting in a feature vector with $3 \cdot 304 = 912$ dimensions. Note that PaDiM [48] is originally based on EfficientNet-B5 [86]. Since the results of [12] dropped heavily with this backbone, we report the paper-given results with AlexNet [98]. We outperform or match the competitors on 12 of 15 categories with an average AUROC of 98.7%, which considerably closes the gap to the optimum of 100% compared to competitors. CS-Flow works reliably on a wide range of defects having an AUROC over 97% in 14 of 15 categories. Our method remains competitive when training on only 16 samples

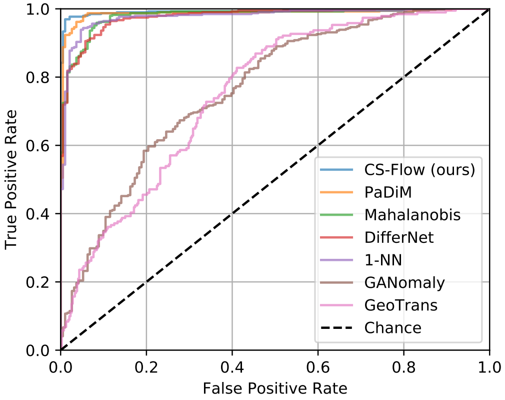


Figure 5.5: Comparison of defect detection performance of different methods on MTD by the time of the submission in 2020. The graphs are the ROC Curves of the individual methods. Best viewed in color.

per category, with even showing roughly the same performance on the texture categories.

We also set a new state of the art of 99.3% AUROC on MTD as shown in Table 5.2. As shown in Figure 5.4, the likelihood assigned by our model clearly distinguishes the defective from the non-defective parts, with only a few exceptions. Being just 0.7% AUROC close to an optimal ROC, we want to emphasize that in this metric a margin of a few percent compared to competitors is a relatively strong increase in performance as visualized in Figure 5.5.

5.2.3 *Localization*

Although the objective of our approach is to detect defects on image level, it can also be used to localize defective regions in images, due to its global and local feature-preserving nature. In this section, we study the localization, as described in Subsection 5.1.3. Our goal is to give quick visual feedback to an operator. Figure 5.6 shows the visualization of the highest scale outputs $z^{(1)}$. These were scaled up with a bilinear interpolation after summing up the squared values along the channel dimension. It can be seen that the magnitude of the output values is directly related to the occurrence of anomalous regions at the respective position. Therefore, our method localizes anomalies of various sizes with respect to color, pattern and shape. Except for dilations due to the convolutional receptive field, defective regions are determined properly. We do not aim to provide pixel-precise segmentations as the method

Method	AUROC [%] ↑
single scale NF (768 × 768)	97.8
single scale NF (384 × 384)	96.8
single scale NF (192 × 192)	96.1
separate multi-scale	98.2
concat multiscale	98.0
CS-Flow (ours)	**98.7**

Table 5.3: Ablation study on MVTec AD with varying strategies regarding the usage of scales.

n_{blocks}	1	2	3	4	5	6
AUROC [%] ↑	94.6	97.8	98.5	**98.7**	**98.7**	98.6

Table 5.4: Ablation study on MVTec AD for a different number of coupling blocks.

is not optimized for it and processes small-resolution feature maps. Nevertheless, this visualization helps in the interpretation of the output in practice to quickly find or assess the potential error.

5.2.4 *Ablation Studies*

To quantify the influence of the individual design decisions of our model, we report results obtained when varying the hyperparameters of our method. Table 5.3 shows the results of these experiments. We measure the impact of the multi-scale approach on the defect detection performance. To this end, we train models on feature maps from one of the three scales at a time (denoted as *single scale NF*). The results confirm that the features of a single scale are weaker with respect to the discriminability between defective and non-defective samples. Furthermore, we set another baseline by adding the log-likelihoods provided by the networks from every scale (denoted as *separate multi-scale*). The increase in AUROC compared to the individual performance for the single scale models demonstrates that the features of different scales complement each other well to obtain a more robust score. Nevertheless, this method is 0.5% AUROC below the performance of our joint training of the individual scales with CS-Flow. To test our architecture against a naive approach of joint training, we feed a single-scale NF the concatenation of differently sized feature maps along the channel dimension after upscaling each of them to the highest feature map size with bilinear interpolation, comparable to [48]. This setup (denoted as *concat mutiscale*) results in a performance drop of 0.7%, which justifies our cross-convolutional multi-scale procedure.

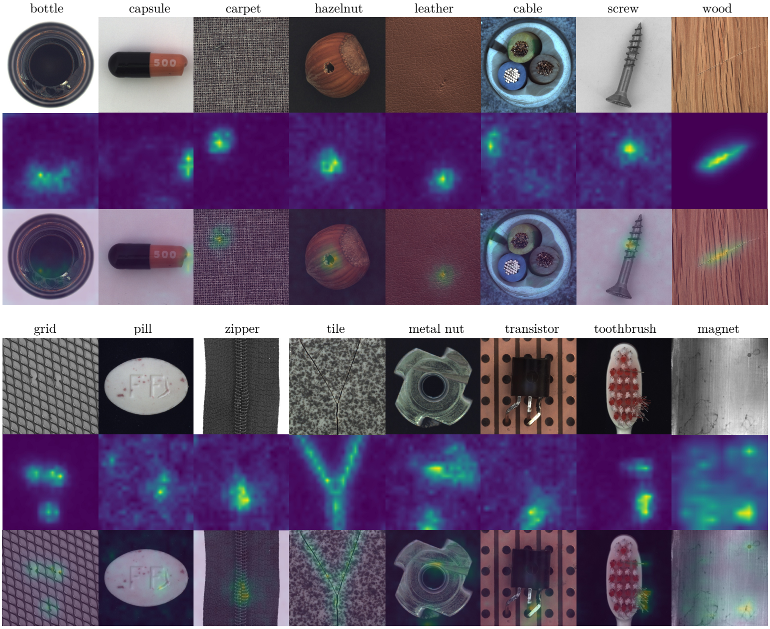


Figure 5.6: Defect localization of one defective example per category of MVTec AD and MTD. The rows each show the original image, the localization and the overlay of both images, from top to bottom. The localization maps show the sum of squares along the channel dimension of the network's output at the highest scale.

In another experiment, we studied the influence of the number of coupling blocks. The results in Table 5.4 show that the performance improves with increasing number of coupling blocks up to $n_{blocks} = 4$ and then saturates.

To test our model on a setting with more intra-class variance in the normal data, we additional experiment training simultaneously with all 15 classes of MVTec AD as normal data. The average detection AUROC is 98.2% which shows that our model can handle multi-modal distributions.

5.3 CONCLUSION

This chapter presents the semi-supervised method CS-Flow to effectively detect and localize defects on feature tensors of different scales using Normalizing Flows. We utilize the context within and between multi-scale feature maps by integrating cross-convolution blocks inside the Normalizing Flow to assign likelihoods and detect unlikely samples as defects. This addresses weaknesses of previous methods that struggle either due to restrictions of overly simplified data representations or limited distribution models and enables our method to set state-of-the-art performance on MVTec AD and MTD. CS-Flow is robust to few training samples, provides an image-level result with just one inference step, and provides localization directly at the same time.

5.4 OPEN PROBLEMS

Even though CS-Flow already shows near-perfect results on current benchmarks, the detection performance drops for more challenging scenarios when the intra-class variance of normal data is more complex as shown in the following chapter. This raises the question of what inherent weaknesses the usage of Normalizing Flow has, which in theory should provide near-ideal results in terms of density estimation. To this end, we hypothesize the following:

The architecture of the Normalizing Flow is susceptible to unstable training. The transformations within the blocks can trigger chain reactions of noise, which are amplified by the exponentiation of the scaling coefficients, resulting in a potential of exploding gradients, as discussed in [93]. Instability can, for example, lead to normal data being mapped outside the normal distribution, as the network is sensitive to small input changes which is further enhanced by rewarding a high Jacobian determinant in the training (see Eq. 5.5). Although the instability is partially mitigated by limiting the gradients and soft-clamping the scaling coefficients, the latter limits the flexibility of the transformation, creating a tradeoff between stability and flexibility.

In addition, density estimation optimization includes the background, which often takes up more than half of the image. Consequently, the NF will optimize to assign this portion of the probability mass to the background and model minimal changes in it, although the detection of it should be trivial and anomalies usually cannot be found in this area. As a result, the capacity of parameters and probability mass of the NF is not concentrated on the relevant part of the foreground. Ideally, a separate treatment of foreground and background would be desirable.

These concerns drive us to alternate the methodology in the subsequent chapter introducing *Asymmetric Student-Teacher Networks*.

6

ASYMMETRIC STUDENT-TEACHER NETWORKS

In the previous two chapters, we have shown how a Normalizing Flow can provide density estimation for effective anomaly detection on RGB images. However, there are also limitations of CS-Flow pointed out in chapter 5.4, which include the flexibility-stability tradeoff of NFs and the lack of missing foreground/background distinction. Based on these problems, we introduce in this chapter a method based on our work "Asymmetric Student-Teacher Networks for Industrial Anomaly Detection" [14]. Here, we adopt some techniques from CS-Flow and combine them with the concept of student-teacher networks which we want to briefly recap in the following.

As reviewed in detail in chapter 2.2.4, several AD methods are based on *student-teacher networks* [43]–[46], [105] where first the teacher is trained on a pretext task to learn a semantic embedding and in a second step, the student is trained to match the output of the teacher. The motivation is that the student can only match the outputs of the teacher on normal data since it is trained only on normal data. Thus, the distance between the outputs of student and teacher is used as an indicator of an anomaly at test time which is visualized in Figure 6.1. It is assumed that this distance is larger for defective examples compared to defect-free examples.

However, this is not necessarily the case in previous work, since we discovered that both teacher and student are conventional (i. e. non-

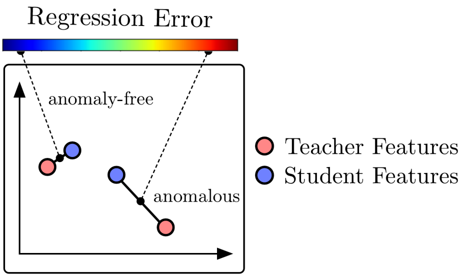


Figure 6.1: Concept of student-teacher approaches: Anomaly scores are determined by evaluating the regression error between the student and teacher network. Increased regression indicates anomalous observations. Image is taken from [44].

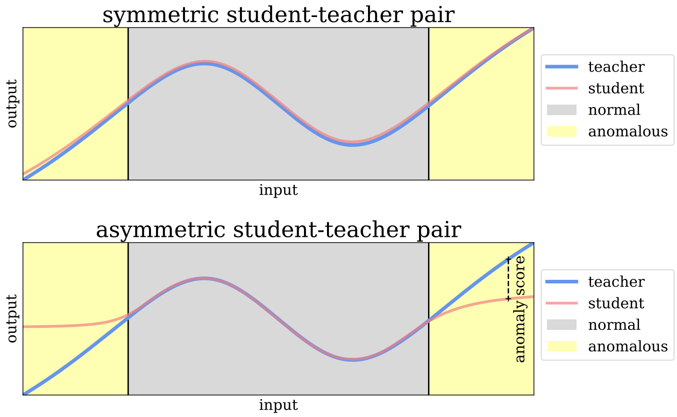


Figure 6.2: Toy example with mini-MLPs: The students were optimized to match the outputs in the grey area. While the symmetric student-teacher pair (top) generalizes unintentionally and maps anomalous data very similarly, the distance between student and teacher outputs can be used for anomaly detection in the asymmetric student-teacher pair (bottom).

injective) neural networks with identical architecture. A student with similar architecture tends to undesired generalization, such that it extrapolates similar outputs as the teacher for inputs that are out of the training distribution, which, in turn, gives an undesired low anomaly score. This effect is shown in Figure 6.2 using an explanatory experiment with one-dimensional data: If the same neural network with one hidden layer is used for student and teacher, the outputs are still similar for anomalous data in the yellow area of the upper plot. In contrast, the outputs for anomalies diverge if an MLP with 3 hidden layers is used as the student.

In general, it is not guaranteed that an out-of-distribution input will cause a sufficiently large change in both outputs due to the missing injectivity of common neural networks. In contrast to Normalizing Flows, conventional networks have no guarantee to provide out-of-distribution outputs for out-of-distribution inputs. These issues motivate us to use an asymmetric student-teacher pair (*AST*): A bijective Normalizing Flow [66] acts as a teacher while a conventional backbone model acts as a student. In this way, the teacher guarantees to be sensitive to changes in the input caused by anomalies. Furthermore, the usage of different architectures and thus of different sets of learnable functions enforces the effect of distant outputs for out-of-distribution samples.

As a pretext task for the teacher, we optimize to transform the distribution of image features and/or depth maps to a normal distribution via maximum likelihood training which is equivalent to a density estimation [89] as in chapters 4 and 5 where the likelihoods are directly used as anomaly score. We show that our student-teacher distance is a better measure for anomaly detection compared to the obtained likelihoods by the teacher. The advantage to using a Normalizing Flow itself for anomaly detection is that a possible misestimation in likelihood can be compensated for: If a low likelihood of being normal is incorrectly assigned to normal data, this output can be predicted by the student, thus still resulting in a small anomaly score. If a high likelihood of being normal is incorrectly assigned to anomalous data, this output cannot be predicted by the student, again resulting in a high anomaly score. In this way, we combine the benefits of student-teacher networks and density estimation with Normalizing Flows.

The method is not limited to RGB images and can also handle 3D scans or their combination by evaluating on the MVTec 3D-AD dataset presented in Section 3.2. We further use the depth maps to model foreground and background which helps us to focus the optimization and detection on the foreground. Furthermore, a positional encoding is used to consider the position as context to the features to detect misplacements. In the upcoming section, we present a thorough and formal description of the method. The implementation is publicly available at GitHub[1]github.com/marco-rudolph/ast.

6.1 METHOD

Our goal is to train two models, a student model f_s and a teacher model f_t, such that the student learns to regress the teacher outputs on defect-free image data only. The training process is divided into two phases: First, the teacher model is optimized to transform the training distribution p_X to a normal distribution $\mathcal{N}(0, I)$ bijectively with a Normalizing Flow. Second, the student is optimized to match the teacher outputs by minimizing the distance between $f_s(x)$ and $f_t(x)$ of training samples $x \in X$. We apply the distance for anomaly scoring at test time, which is further described in Subsection 6.1.2.

We follow [13], [43], [106] and use extracted features obtained by a pre-trained network on ImageNet [101] instead of RGB images as direct input for our models. Such networks have been shown in Chapters 4 and 5 to be universal feature extractors whose outputs carry relevant semantics for industrial anomaly detection.

---

[1] github.com/marco-rudolph/ast

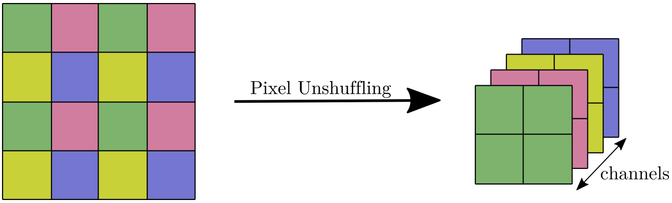


Figure 6.3: Visualization of pixel unshuffling with $d = 2$. The pixels within a $d \times d$ block are each divided into d^2 channels.

In addition to RGB data, our approach is easily extendable to multimodal inputs including 3D data. If 3D data is available, we concatenate depth maps to these features along the channels. Since the feature maps are reduced in height and width compared to the depth map resolution by a factor d, we apply pixel-unshuffling [107] by grouping a depth image patch of $d \times d$ pixels as one pixel with d^2 channels to match the dimensions of the feature maps. This procedure is visualized in Figure 6.3 for $d = 2$.

Any 3D data that may be present is used to extract the foreground. This is straightforward and reasonable whenever the background is static or planar, which is the case for almost all real applications. Pixels that are in the background are ignored when optimizing the teacher and student by masking the distance and negative log-likelihood loss, which are introduced in Subsections 6.1.1 and 6.1.2. If no 3D data is available, the whole image is considered as foreground. Details of the foreground extraction are given in Subsection 6.1.3.

Similar to [106], we use a sinusoidal positional encoding [108] for the spatial dimensions of the input maps as a condition for the Normalizing Flow f_t. In this way, the occurrence of a feature is related to its position to detect anomalies such as misplaced objects. An overview of our pipeline is given in Figure 6.4.

6.1.1 *Teacher*

Similar to DifferNet and CS-Flow in chapters 4 and 5, we train a Normalizing Flow based on Real-NVP [89] to transform the training distribution to a normal distribution $\mathcal{N}(0, I)$. In contrast to our previous work, we do not use the outputs to compute likelihoods and thereby obtain anomaly scores directly. Instead, we interpret this training as a pretext task to create targets for our student network. Furthermore, the NF is extended with a condition as in [93]. In the following, we recap the Real-NVP architecture described in detail in subsection 4.1.3, along with our use of it.

Our Normalizing Flow consists of multiple subsequent affine coupling blocks. Let the input $x \in \mathcal{R}^{w \times h \times n_{\text{feat}}}$ be feature maps with n_{feat} features of size $w \times h$. Within these blocks, the channels of the input x are split evenly along the channels into the parts x_1 and x_2 after randomly choosing a permutation that remains fixed. These parts are each concatenated with a positional encoding c as a static condition. Both are used to compute scaling and shift parameters for an affine transformation of the counterpart by having subnetworks s_i and t_i for each part:

$$\begin{aligned} y_2 &= x_2 \odot e^{s_1([x_1,c])} + t_1([x_1,c]) \\ y_1 &= x_1 \odot e^{s_2([y_2,c])} + t_2([y_2,c]), \end{aligned} \tag{6.1}$$

where $\odot$ is the element-wise product and $[\cdot,\cdot]$ denotes concatenation. The output of one coupling block is the concatenation of y_1 and y_2 along the channels. Thus, we use the same affine transformation as in CS-Flow from Chapter 5, extended with position as a condition and with just a single scale of feature maps.

To stabilize training, we adopt the alpha-clamping of scalar coefficients as in [93] and the gamma-trick as in Chapter 5 (CS-Flow). Using the change-of-variable formula with z as our final output

$$p_X(x) = p_Z(z) \left| \det \frac{\partial z}{\partial x} \right| \quad , \tag{6.2}$$

we minimize the negative log-likelihood with p_Z as the normal distribution $\mathcal{N}(0, I)$ by optimizing the mean of

$$\mathcal{L}^t_{ij} = -\log p_X(\boldsymbol{x}_{ij}) = \frac{\left\| \boldsymbol{z}_{ij} \right\|_2^2}{2} - \log \left| \det \frac{\partial \boldsymbol{z}_{ij}}{\partial \boldsymbol{x}_{ij}} \right| \tag{6.3}$$

over all (foreground) pixels at pixel position (i,j).

6.1.2 *Student*

As opposed to the teacher, the student is a conventional feed-forward network that does not map injectively or surjectively. We propose a simple fully convolutional network with residual blocks which is shown in Figure 6.5. Each residual block consists of two sequences of 3×3 convolutional layers, batch normalization [109] and leaky ReLU activations. We

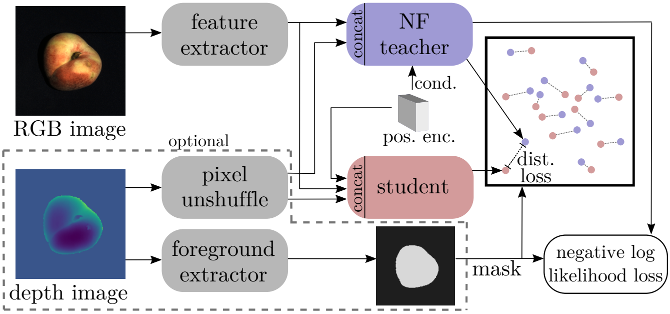


Figure 6.4: Overview of the pipeline of AST: Teacher and student receive image features and/or depth maps as input which is conditioned by a positional encoding. First, the teacher represented by a Normalizing Flow is optimized to reduce the negative log-likelihood loss that may be masked by a foreground map from 3D. Second, the student is trained to match the teacher's outputs by minimizing the (masked) distance between them.

add convolutions as the first and last layer to increase and decrease the feature dimensions.

Similarly to the teacher, the student takes image features as input which are concatenated with 3D data if available. In addition, the positional encoding c is concatenated. The output dimensions match the teacher to enable pixel-wise distance computation. We minimize the squared ℓ_2-distance between student outputs $f_s(x)$ and the teacher outputs $f_t(x)$ on training samples x at a pixel position (i, j) of the output:

$$\mathcal{L}^s_{ij} = \left\| f_s(x)_{ij} - f_t(x)_{ij} \right\|_2^2. \tag{6.4}$$

Averaging $\mathcal{L}^s_{ij}$ over all (foreground) pixels gives us the final loss. The distance $\mathcal{L}^s$ is also used in testing to obtain an anomaly score on image level: Ignoring the anomaly scores of background pixels, we aggregate the pixel distances of one sample by computing either the maximum or the mean over the pixels.

6.1.3 *3D Preprocessing*

The rasterized 3D point clouds are processed as follows: We discard the x and y coordinates due to the low informative content and use only the depth component z. As some pixels are missed by the sensor, their depth values are repeatedly filled by using the average value of valid pixels from an 8-connected neighborhood for 3 iterations. We model the background

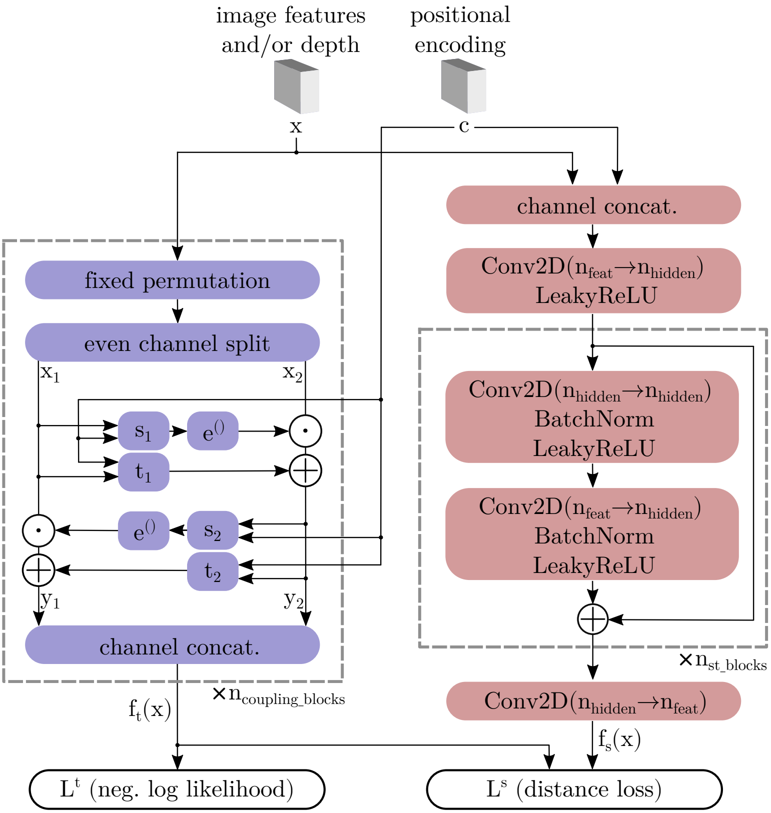


Figure 6.5: Model architecture of teacher (left side) and student (right side). While the teacher is a Real-NVP-based [89] conditional Normalizing Flow [93], the student is a conventional convolutional neural network.

as a 2D plane by interpolating the depth of the 4 corner pixels. A pixel is assumed as foreground if it is valid and its depth is further than $7mm$ distant from the background plane. As an input to our models, we first resize the masks to 192×192 pixels via bilinear downsampling and then perform pixel-unshuffling [107] with $d = 8$ as described in Section 6.1 to match the feature map resolution. In order to detect anomalies at the edge of the object and fill holes of missing values, the foreground mask is dilated using a square structural element of size 8. We subtract the mean foreground depth from each depth map and set its background pixels to 0. The binary foreground mask M with ones as foreground and zeros as background is downsampled to feature map resolution to mask the loss for student and teacher. This is done by a bilinear interpolation $f_\downarrow$ followed by a binarization where all entries greater than zero are assumed as foreground to mask the loss at position (i, j):

$$\mathcal{L}_{ij}^{\text{masked}} = \begin{cases} \mathcal{L}_{ij} & \text{if} \quad f_\downarrow(M)_{ij} > 0 \\ 0 & else \end{cases} . \tag{6.5}$$

6.2 EXPERIMENTS

Within this section, we detail the experimental methodology and outcomes of the proposed approach. This includes implementation details, an analysis of detection performance including localization, and ablation studies to validate the effectiveness of AST.

6.2.1 *Implementation Details*

6.2.1.1 *Image Preprocessing*

Following [13], [48], we use the layer 36 output of EfficientNet-B5 [86] pre-trained on ImageNet [101] as a feature extractor. This feature extractor is not trained during the training of the student and teacher networks. The images are resized to a resolution of 768×768 pixels resulting in feature maps of size 24×24 with 304 channels.

6.2.1.2 *Teacher*

For the Normalizing Flow architecture of the teacher, we use 4 coupling blocks which are conditioned on a positional encoding with 32 channels. Each pair of internal subnetworks s_i and t_i is designed as one shallow convolutional network r_i with one hidden layer whose output is split

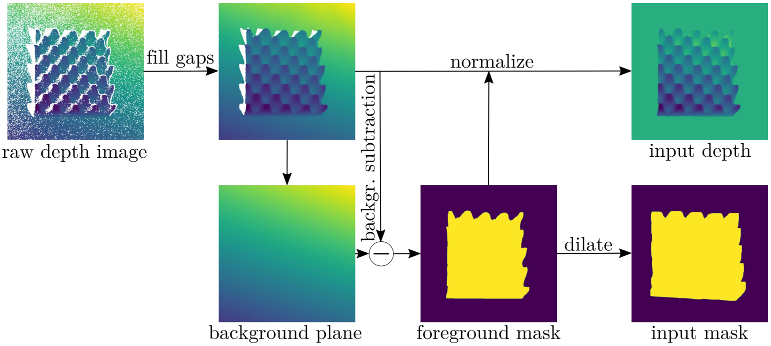


Figure 6.6: Preprocessing pipeline for 3D data. After imputing missing depth values, the foreground is extracted by assuming a background plane using the corner points. The foreground of the depth image is then normalized by subtracting the mean foreground depth while the background depth is set to 0. The foreground plane is further dilated to take the boundary regions into account. Note that resizing and pixel unshuffling is not part of this figure for visibility reasons.

into the scale and shift components. Inside r_i we use ReLU-Activations and a hidden channel size of 1024 for MVT2D and 64 for MVT3D. We choose the alpha-clamping parameter $\alpha = 3$ for MVT2D and $\alpha = 1.9$ for MVT3D. The teacher networks are trained for 240 epochs for MVT2D and 72 epochs for MVT3D, respectively, with the Adam optimizer [102], using author-given momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a learning rate of $2 \cdot 10^{-4}$ and a weight decay of 10^{-5}.

6.2.1.3 *Student*

For the student networks, we use $n_{\text{st\_blocks}} = 4$ residual convolutional blocks as described in Subsection 6.1.2. The Leaky-ReLU-activations have a slope of 0.2 for negative values. We choose a hidden channel size of $n_{hidden} = 1024$ for the residual block. Likewise, we take over the number of epochs and optimizer parameters from the teacher. The scores at feature map resolution are aggregated for evaluation at image level by the maximum distance if a foreground mask is available, and the average distance otherwise (RGB only).

6.2.2 *Results*

Table 6.1 shows the AUROC of our method and previous work for detecting anomalies on the 15 classes of MVT2D as well as the averages for

							2021				
	Category	Uninf. Stud. [43]	DifferNet [12]	Rippel [47]	PaDiM [48]	CS-Flow [13]	CFlow [106]	STFPM* [46]	DRÆM [50]	PatchCore [36]	**AST** (ours)
Textures	Grid	98.1	84.0	93.7	-	99.0	99.6	**100**	99.9	98.2	99.1
	Leather	94.7	97.1	**100**	-	99.9	**100**	**100**	100	**100**	**100**
	Tile	94.7	99.4	**100**	-	**100**	99.9	95.5	99.6	98.7	**100**
	Carpet	99.9	92.9	99.6	-	**100**	98.7	98.9	97.8	98.7	97.5
	Wood	99.1	99.8	99.2	-	**100**	99.1	99.2	99.1	98.8	**100**
	Avg. Text.	97.3	94.6	98.5	99.0	**99.8**	99.5	98.7	99.3	98.3	99.3
Objects	Bottle	99.0	99.0	99.0	-	99.8	**100**	**100**	99.2	**100**	**100**
	Capsule	92.5	86.9	96.3	-	97.1	97.7	88.0	98.5	98.1	**99.7**
	Pill	92.2	88.8	91.4	-	98.6	96.8	93.8	98.9	96.6	**99.1**
	Transistor	79.4	91.1	98.2	-	99.3	95.2	93.7	93.1	**100**	99.3
	Zipper	94.4	95.1	98.8	-	99.7	98.5	93.6	**100**	99.4	99.1
	Cable	78.7	95.9	99.1	-	99.1	97.6	92.3	91.8	**99.5**	98.5
	Hazelnut	99.1	99.3	**100**	-	99.6	**100**	**100**	**100**	**100**	**100**
	Metal Nut	89.1	96.1	97.4	-	99.1	99.3	**100**	98.7	**100**	98.5
	Screw	86.0	96.3	94.5	-	97.6	91.9	88.2	93.9	98.1	99.7
	Toothbrush	**100**	98.6	94.1	-	91.9	99.7	87.8	**100**	**100**	96.6
	Avg. Obj.	91.0	94.7	96.9	97.2	98.2	97.7	93.7	97.4	**99.2**	99.1
	Average	93.2	94.7	97.5	97.9	98.7	98.3	95.4	98.0	99.1	**99.2**

Table 6.1: AUROC in % for detecting defects of all categories of MVT2D [7] on image-level grouped into textures and objects. Best results are in bold. Besides the average value, detailed results of PaDiM [48] were not provided by the authors. The numbers of STFPM* [46] were obtained by a reimplementation. Methods listed to the right of the double vertical line have been published after the submission of CS-Flow.

textures, objects and all classes. We set a new state-of-the-art performance on the mean detection AUROC over all classes, improving it slightly to 99.2%. This is mainly due to the good performance on the more challenging objects, where we outperform previous work by a comparatively large margin of 0.9%, except for PatchCore [36]. The detection of anomalies on textures, which CS-Flow [13] has already almost solved with a mean AUROC of 99.8%, still works very reliably at 99.3%. Especially compared to the two student-teacher approaches [43], [46], a significant improvement of 6% and 3.6% respectively is achieved. Moreover, our student-teacher distances show to be a better indicator of anomalies compared to the likelihoods of current state-of-the-art density estimators [13], [106] which, like our teacher, are based on Normalizing Flows.

Even though MVT2D has established itself as a standard benchmark in the past, this dataset (especially the textures) is easily solvable for recent methods, and differences are mainly in the sub-percent range, which is only a minor difference in terms of the comparatively small size of the dataset. In the following, we focus on the newer, more challenging MVT3D dataset, introduced in Section 3.2, where the normal data shows more variance and anomalies only partly occur in one of the two data modalities, RGB and 3D.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
3D	Voxel GAN [9]	38.3	62.3	47.4	63.9	56.4	40.9	61.7	42.7	66.3	57.7	53.7
	Voxel AE [9]	69.3	42.5	51.5	79.0	49.4	55.8	53.7	48.4	63.9	58.3	57.1
	Voxel VM [9]	75.0	**74.7**	61.3	73.8	82.3	69.3	67.9	65.2	60.9	**69.0**	69.9
	Depth GAN [9]	53.0	37.6	60.7	60.3	49.7	48.4	59.5	48.9	53.6	52.1	52.3
	Depth AE [9]	46.8	73.1	49.7	67.3	53.4	41.7	48.5	54.9	56.4	54.6	54.6
	Depth VM [9]	51.0	54.2	46.9	57.6	60.9	69.9	45.0	41.9	66.8	52.0	54.6
	1-NN (FPFH) [110]	82.5	55.1	95.2	79.7	**88.3**	58.2	75.8	88.9	92.9	65.3	78.2
	3D-ST$_{128}$ [44]	86.2	48.4	83.2	89.4	84.8	66.3	76.3	68.7	95.8	48.6	74.8
	AST (ours)	**88.1**	57.6	**96.5**	**95.7**	67.9	**79.7**	**99.0**	**91.5**	**95.6**	61.1	**83.3**
RGB	PatchCore [36]	87.6	88.0	79.1	68.2	91.2	70.1	69.5	61.8	84.1	70.2	77.0
	DifferNet [12]	85.9	70.3	**64.3**	43.5	79.7	79.0	78.7	64.3	71.5	59.0	69.6
	PADiM [48]*	**97.5**	77.5	69.8	58.2	95.9	66.3	85.8	53.5	83.2	76.0	76.4
	CS-Flow [13]	94.1	**93.0**	82.7	79.5	**99.0**	88.6	73.1	47.1	98.6	74.5	83.0
	STFPM [46]*	93.0	84.7	**89.0**	57.5	94.7	76.6	71.0	59.8	96.5	70.1	79.3
	AST (ours)	94.7	92.8	85.1	**82.5**	98.1	**95.1**	**89.5**	61.3	**99.2**	**82.1**	**88.0**
3D + RGB	Voxel GAN [9]	68.0	32.4	56.5	39.9	49.7	48.2	56.6	57.9	60.1	48.2	51.7
	Voxel AE [9]	51.0	54.0	38.4	69.3	44.6	63.2	55.0	49.4	72.1	41.3	53.8
	Voxel VM [9]	55.3	77.2	48.4	70.1	75.1	57.8	48.0	46.6	68.9	61.1	60.9
	Depth GAN [9]	53.8	37.2	58.0	60.3	43.0	53.4	64.2	60.1	44.3	57.7	53.2
	Depth AE [9]	64.8	50.2	65.0	48.8	80.5	52.2	71.2	52.9	54.0	55.2	59.5
	Depth VM [9]	51.3	55.1	47.7	58.1	61.7	71.6	45.0	42.1	59.8	62.3	55.5
	PatchCore+FPFH [110]	91.8	74.8	96.7	88.3	**93.2**	58.2	89.6	91.2	92.1	**88.6**	86.5
	AST (ours)	**98.3**	**87.3**	**97.6**	**97.1**	**93.2**	**88.5**	**97.4**	**98.1**	**100**	79.7	**93.7**

Table 6.2: AUROC in % for detecting defects of all categories of MVT3D [9] on image-level for 3D data, RGB data and the combination of both. Best results per data domain are in bold. A * indicates that we used a reimplementation. The numbers from PatchCore are taken from [110].

The results for individual classes of MVT3D grouped by data modality are given in Table 6.2. We are able to outperform all previous methods for all data modalities regarding the average of all classes by a large margin of 5.1% for 3D, 5% for RGB and 7.2% for the combination. Facing the individual classes and data domains, we set a new state-of-the-art in 21 of 30 cases. Note that this data set is much more challenging when comparing the best results from previous work (99.1% for MVT2D vs. 86.5% AUROC for MVT3D). Nevertheless, we detect defects in 7 out of 10 cases for RGB+3D at an AUROC of at least 93%, which demonstrates the robustness of our method. In contrast, the nearest-neighbor approach PatchCore [36], which provides a comparable performance on MVT2D, struggles with the increased difficulty of the dataset and is outperformed by 11% on RGB. The same applies to the 3D extension [110] using FPFH [111] despite using a foreground mask as well. Figure 6.7 shows qualitative results for the RGB+3D case given both inputs and ground truth annotations. Despite the low resolution, the regions of the anomaly can still be localized well for practical purposes.

For the class *Cable Gland* in the RGB+3D setting, the top of Figure 6.8 compares the distribution of student-teacher distances for anomalous and

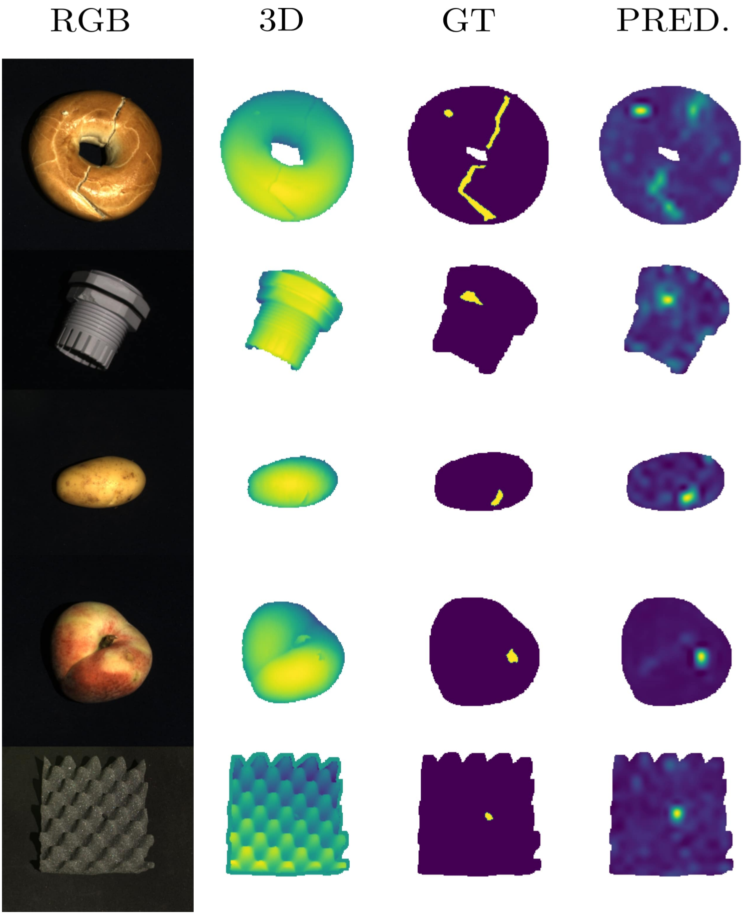


Figure 6.7: Qualitative results on MVTec 3D-AD [9]. The two left columns show the input, the third the ground truth and the fourth our anomaly detection. Images are masked by foreground extraction. Our method is able to successfully combine RGB and 3D data to detect defects even if only present in one data domain.

Method	3D	RGB	3D+RGB
Teacher only	82.2	69.8	90.9
NF student (symm.)	81.8	76.0	88.9
NF student (deeper)	81.8	76.7	92.7
AST (ours)	**83.3**	**88.0**	**93.7**

Table 6.3: Comparison of average detection performance in AUROC percentage on MVT3D of teacher and student-teacher in a symmetric and asymmetric setting. Our proposed asymmetric student-teacher pair outperforms all baselines in all cases.

normal regions. The distribution of anomalous samples shows a clear shift towards larger distances. At the bottom of Figure 6.8, the outputs of student and teacher as well as the distance of corresponding pairs representing our anomaly score are visualized by a random orthographic 2D projection. Note that visualizations made by techniques such as t-SNE [112] or PCA [58] are not meaningful here, since the teacher outputs (and therefore most of the student outputs) follow an isotropic standard normal distribution. Therefore, different random projections barely differ qualitatively.

6.2.3 *Ablation Studies*

We demonstrate the effectiveness of our contributions and design decisions with several ablation studies. Table 6.3 compares the performance of variants of students with the teacher, which can be used as a density estimator itself for anomaly detection by using its likelihoods, given by Eq. 6.2, as anomaly score. In comparison, a symmetric student-teacher pair worsens the results by 1 to 2%, except in the RGB case. However, the performance is already improved for RGB and 3D+RGB by creating the asymmetry with a deeper version of the student than the teacher by doubling the number of coupling blocks to 8. This effect is further enhanced if the architecture of the NF-teacher is replaced by a conventional feedforward network as proposed for AST. We also vary the depth of our student network and analyzed its relation to performance, model size and inference time in Table 6.4. With an increasing number of residual blocks $n_{\text{st\_blocks}}$, an increasing performance is observed which is almost saturated after 4 blocks. Since the remaining potential in detection performance is not in relation to the linearly increasing additional computational effort per block, we suggest choosing 4 blocks to have a good trade-off.

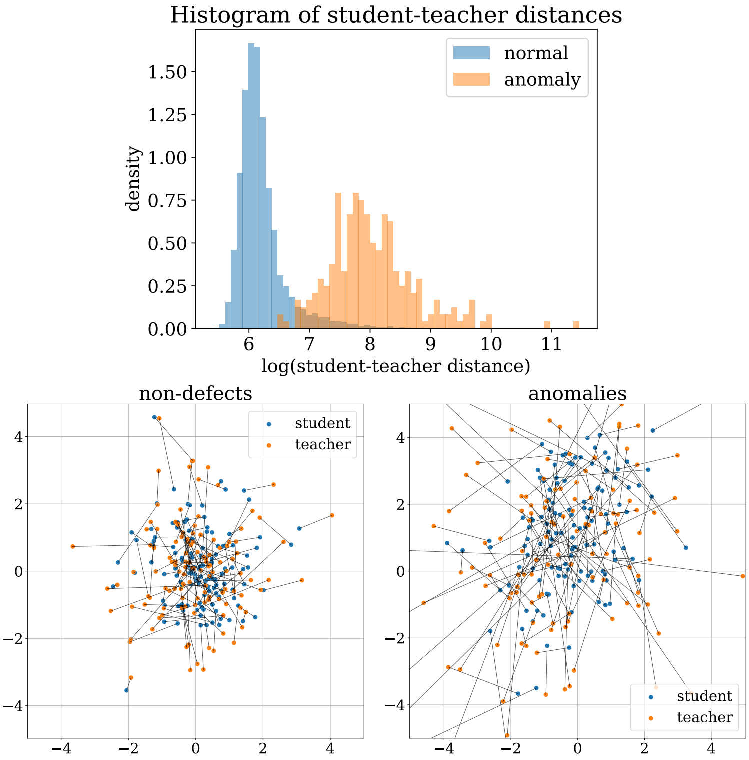


Figure 6.8: Top: Histogram of our AST distances for normal and anomalous regions of the class *rope* in MVT3D. Bottom: Random orthographic projections of student and teacher outputs grouped in non-defective (left plot) anomalous regions (right plot) for the class peach. The plotted student-teacher distance representing the anomaly score is clearly higher for anomalous regions since the student is not able to match the teacher outputs, as it was only trained on non-defective regions.

$n_{\text{st\_blocks}}$	AUROC [%] ↑	#Params. [M] ↓	inf. time [ms] ↓
1	92.8	26.0	0.184
2	93.3	44.8	0.229
4	93.7	82.6	0.352
8	93.7	151.1	0.609
12	93.8	233.6	0.821
teacher	90.9	3.8	1.46

Table 6.4: Tradeoff between performance and computational effort on 3D+RGB data of MVT3D. The inference time was measured with a *NVIDIA GeForce RTX 4090*.

input	pos. enc.	mask	teacher	**AST**
3D	✗	✓	78.4	81.9
	✓	✗	59.4	67.2
	✓	✓	82.2	**83.3**
RGB	✗	✗	69.3	87.8
	✓	✗	69.8	**88.0**
	✓	✓	n. a.	n. a.
3D+RGB	✗	✓	90.9	**93.8**
	✓	✗	66.2	84.0
	✓	✓	90.9	93.7

Table 6.5: Impact of the positional encoding and the foreground mask on the detection performance of student and teacher on MVT3D. Numbers are given in AUROC percentage. Since masks are obtained from 3D data, there is no mask for RGB.

In Table 6.5 we investigate the impact of the positional encoding and the foreground mask. For MVT3D, positional encoding improves the detection by 1.4% of our AST-pair when trained with 3D data as the only input. Even though the effect is not present when combining both data modalities, we consider it generally reasonable to use the positional encoding, considering that the integration with just 32 additional channels does not significantly increase the computational effort.

Foreground extraction in order to mask the loss for training and anomaly score for testing is also highly effective. Since the majority of the image area often consists of background, the teacher has to spend a large part of the distribution on the background. Masking allows the teacher and student to focus on the essential structures. Moreover, noisy background scores are eliminated.

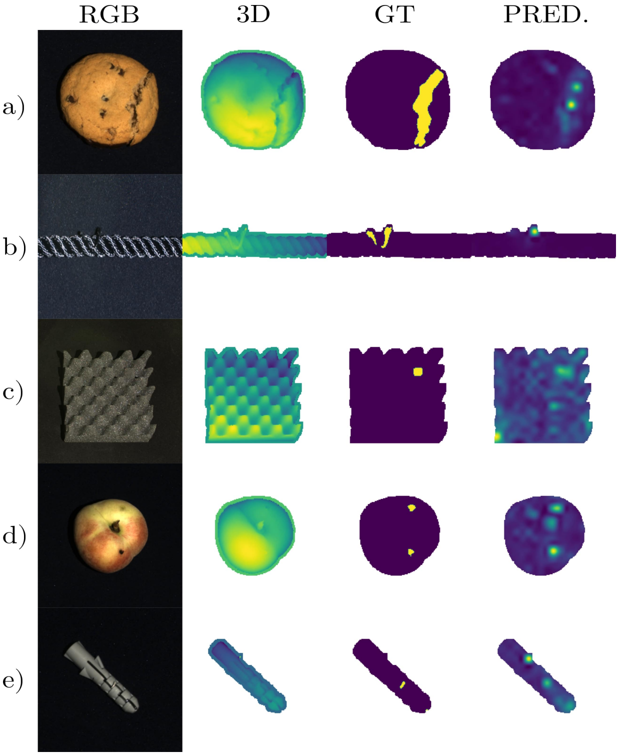


Figure 6.9: Failure cases of the localization.

6.3 OPEN PROBLEMS

As described in the previous section or Figure 6.7, AST locates anomalies by calculating the distance between student and teacher for each position in the outputs. This visualization may be sufficient for most use cases, but shows some flaws: First, the resolution is limited by the feature map (24×24 in this case), so very fine structures cannot be accurately represented. In addition, the maps are somewhat dilated by the receptive field of the convolutions in the feature extractor and the flow model. Figure 6.9 shows some failure cases of the localization showing the limitation of the method and the dataset. The detection errors can be roughly divided into 5 types, each attributed as false positives (FP) or false negatives (FN), which are visualized by one example each in Figure 6.9 a) - e):

- a) FN: The method does not capture the entire anomalous area but focuses on parts of it.
- b) FN: An anomalous area is completely overseen.
- c) FP: The method highlights normal areas beside the anomaly.
- d) FP: The method highlights rare but still normal structures.
- e) (FP:) The method interprets sensor noise as an anomaly.

Examples from categories c) and d) should be reduced by a sufficient amount of training data. The error type e) could be eliminated by a separate detection of sensor noise. The FN-related errors a) and b) would be minimized by a more suitable feature extractor on which the detection is based.

It can be shown that some recent approaches provide a more accurate localization. Table 6.6 compares the AUROC on pixel-level of our method, PatchCore [36] and the SSIM-based AE by Bergmann et al. [39]. While our approach still localizes noticeably better than AE-SSIM, the nearest-neighbor-based method PatchCore is significantly more accurate at segmentation. Future work could extend the method to increase both resolution and sharpness of localization.

6.4 COMPARISON WITH DIFFERNET AND CS-FLOW

In this subsection, we compare the 3 image-based anomaly detection approaches DifferNet, CS-Flow and AST which are presented in this thesis. In the following, we will examine the similarities and differences between the methods, as well as the impact of the latter.

Method	MVT2D	MVT3D (RGB+3D)
AE-SSIM [63]	87.0	-
PatchCore [36]	**98.4**	-
PatchCore+FPFH [110]	-	**99.2**
AST (ours)	95.0	97.6

Table 6.6: Anomaly segmentation results measured by the mean pixel-AUROC over all classes. Despite image-level detection being the focus of this work, our method is able to localize defects for practical purposes with an AUROC of 95% or 97.6%.

All 3 methods are based on a feature representation of a pretrained neural network. In addition, these approaches each use a Normalizing Flow that is trained to perform a density estimation on these features. Furthermore, each method is able to provide a map with anomaly localizations besides an instance-level anomaly score.

Table 6.7 compares the differences in characteristics of the methods in terms of anomaly score evaluation, underlying feature representation, localization procedure, detection performance and computation time measured with a *NVIDIA GeForce RTX 4090* GPU. A steady improvement in terms of runtime and detection performance can be observed throughout the approaches of these chapters. While the performance in the case of CS-Flow increases mainly due to the use of full-sized feature maps carrying more information compared to aggregated vectors, the runtime is reduced since these maps could be processed more efficiently and thus with fewer blocks within the network. Due to the direct computation of the localization map from the regular network output of CS-Flow, the localization of CS-Flow is nearly costless compared to the memory and time-consuming backpropagation in DifferNet.

This is why the localization procedure from CS-Flow for AST was kept in principle. The benefits of AST lie in the Student-Teacher procedure, which is performed after the NF-based density estimation and brings a decent benefit in improving the mean detection AUROC on MVT3D by 5% to 88%. Since MVT2D already shows near-perfect results for CS-Flow, this is observed at a weakened level (+0.5%) for this dataset. In addition, the computation time could be heavily reduced from 8.53 to 1.79ms, since the new methodology is very effective without the use of multi-scale feature maps which no longer provide a benefit for this approach.

Method		DifferNet	CS-Flow	AST
Anomaly Score		Likelihood	Likelihood	Stud.-Teacher dist.
Feature representation		Vectors	Maps	Maps
Localization procedure		Backprop.	Output aggreg.	Output aggreg.
Detection [AUROC %] ↑	MVT2D	94.7	98.7	**99.2**
	MVT3D	69.9	83.0	**88.0**
Inference time [ms] ↓		11.6	8.53	**1.79**

Table 6.7: Comparison of the image-based approaches proposed in chapters 4-6 of this thesis. The methods presented throughout these chapters each improved upon their predecessors in terms of detection performance and computational complexity. This is mainly due to the use of full-sized feature maps in the case of CS-Flow and the specialized student-teacher approach in the case of AST.

6.5 CONCLUSION

We discovered the generalization problem of previous student-teacher pairs for AD and introduced an alternative student-teacher method that prevents this issue by using a highly different architecture for student and teacher. We were able to compensate for skewed likelihoods of a Normalizing Flow-based teacher, which was used directly for detection in previous work, by the additional use of a student. This approach could be extended to more data domains and improved with a higher localization resolution.

Since we have now discussed in detail the methodology and application for detection on optical data, we would like to shift the focus to robotic applications. Robots play an important part in several manufacturing pipelines in the context of Industry 4.0 as they perform precise assembly and transportation tasks. In the course of the next two chapters, we address anomaly detection on time series of robot machine data, first introducing our own dataset *voraus-AD*.

7

VORAUS-AD DATASET

The voraus-AD dataset, created as part of this thesis, contains machine data of a robot in a pick-and-place task, from which various anomalies are to be detected. This and the following chapter are based on the article "The voraus-AD Dataset for Anomaly Detection in Robot Applications" [15]. The dataset is publicly available at GitHub[1]tnt.uni-hannover.de/vorausAD.

As a subfield of automation, the use of robotic arms is essential for industrial applications and processes [113]. Industrial robots can relieve humans of monotonous and dangerous tasks and perform them with consistent quality without interruption. With automated monitoring, robotic arms can operate continuously without the presence of humans. Detecting unusual behavior during automated execution of robotic applications is critical for ensuring process and product quality as well as for human safety [114]. With the increasing importance of collaboration (e.g., human-machine interactions as shown in Figure 7.1) and the growing use of artificial intelligence, the complexity of robotic applications is increasing

---

[1] https://tnt.uni-hannover.de/vorausAD

Figure 7.1: Example for human-robot interaction with a collaborative robot. Image source: voraus robotik

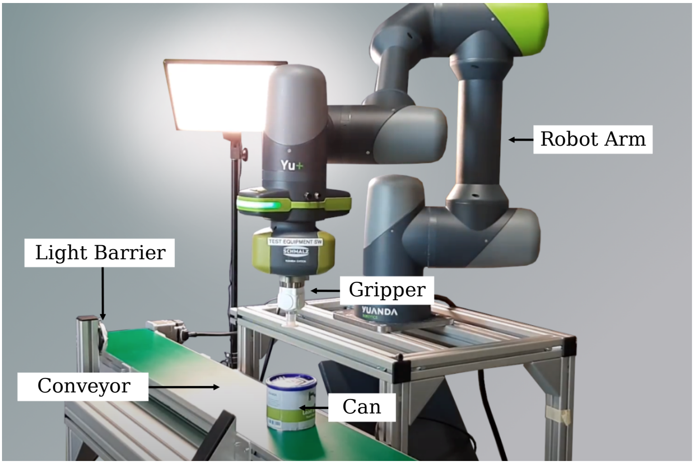


Figure 7.2: Setup of the pick-and-place application in voraus-AD: A robot arm grips a randomly positioned can and moves it to a fixed position.

as well compared to the more monotonous applications with traditional industrial robots [115]. For example, objects to be gripped no longer have to be in a fixed position, but are localized by visual object recognition. Since naive approaches are no longer feasible for the detection of unusual events, the design of more sophisticated machine learning methods is coming more and more into the interest of research [71], [72], [77]–[80]. As with industrial anomaly detection on images, the availability of anomaly samples is critical for robot data, motivating the use of semi-supervised anomaly detection. Again, it can never be ensured that all scenarios are represented in the collected data, as machine wear, unpredictable changes in the environment, or chain reactions over multiple process steps may occur in the future.

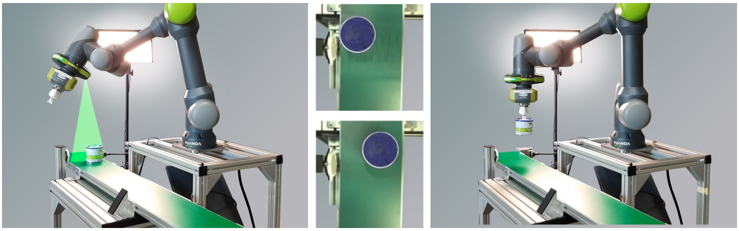
In order to ensure comparability of different methods w.r.t. different applications and to establish transparency, it is established to evaluate on public datasets. Unfortunately, this is hardly the case for AD in robot applications at the moment. The only public dataset *AURSAD* [116] is not widely used since it is very specific to a screwdriver application. Publications have so far been evaluated on private datasets [71], [72], [77]–[80] which makes the experiments irreproducible. This issue motivates us to present a publicly available benchmark based on machine data which enables an appropriate evaluation due to the included number, types, and diversity of anomalies. As an application, we choose a standard pick-and-place operation, whose setup is shown in Figure 7.2, that includes typical elements such as the movement of a robotic arm, actions of the end effector, and interaction with the environment. The dataset contains 1367 time series of anomaly-free data, in which a collaborative robot picks a randomly placed can with a vacuum gripper and places it at a predetermined position. For the test set, 755 additional samples, divided in 12 anomaly types, were intentionally induced, which include robot axis wear, gripping errors and process errors such as collisions, manipulation of the can or an unstable setup. In contrast to AURSAD, where the anomalies are specific to its screwdriver application, many of the anomalies in our dataset, such as axis wear or collisions, also occur in other applications. Thus, the evaluation on our dataset provides a more general validity of the respective method. We define an evaluation protocol that individually evaluates each anomaly type.

Since the usually integrated safety systems of collaborative robots interrupt the operation in case of large deviations from the norm, which can be detected easily, we especially chose subtle anomalies, which would not be recognized by the safety systems. For example, our dataset includes collisions with a free-hanging cable, which impacts the signals only marginally, although tangling of this cable may be a major safety hazard. In addition, different intensities of anomalies were simulated in each case, for example by varying the weight of the can in several steps.

As the machine data of the robot is usually recorded anyway, we model a practicable use case in that no additional sensing is required. The signals include a total of 78 mechanical signals as well as 52 electrical signals from a total of 6 axes at a sampling frequency of 500 Hz. Furthermore, we provide metadata which, for example, gives detailed information about the anomaly and the status of the robot.

7.1 OVERVIEW

Our dataset voraus-AD contains machine data of a collaborative robot, which moves a can by performing an industrial pick-and-place task. The samples $X = \{x_1, x_2, ..., x_{n_{rec}}\}$ consist of time series of machine data $x_i \in \mathbb{R}^{T \times S}$ with T as the number of time steps and S as the number of signals, each recorded over one pick-and-place operation. The data is split into a training set X_{tr} and a test set X_{te}. As usual in AD, the training set contains only normal data, which includes regular samples without anomalies. The test set contains both, normal data and anomalies, including 12 diverse anomaly types as explained in Section 7.5. In order to create a realistic scenario, we have divided the normal data into training and test data as follows: Up to a certain period of time, only training data including 948 samples was recorded. Subsequently, recordings of



(a) (b) (c)

Figure 7.3: Scanning (a) the conveyor with the robot's built-in camera drawn in green, detecting (b) and gripping (c) the can from different positions.

anomalies (755 samples) and normal data (419 samples) for the test set were taken alternately. This simulates a real application where training data would be recorded first in the same way to train the model before the test case occurs.

7.2 SETUP

The setup consists of a conveyor equipped with a light barrier, a robot arm with a vacuum gripper and the can to be moved as shown in Figure 7.2. We use the collaborative robot arm *Yu-Cobot* with 6 axes and an integrated camera, which is mounted on a platform made of aluminum profiles. Our robot arm is equipped with a vacuum gripper[2] for gripping the can. The conveyor is also mounted on the platform in the working space of the robot arm. The conveyor control is connected to the digital outputs of the robot arm so that the conveyor can be triggered by our robot control software. To detect the presence of a can at the end of the conveyor belt, the light barrier connected to the digital inputs of the robot arm can be used. A light source permanently ensures that the internal camera of the robot arm can also be used for visual object detection at night.

The machine data, which is further described in Section 7.3, is transmitted between the robot controller and the individual axes via a field bus as a network interface (EtherCAT). The software *tshark5*[3] is used to record all the data of this network interface.

---

[2] Schmalz vacuum generator ECBPi

[3] https://www.wireshark.org/docs/man-pages/tshark.html

No.	Action	Step	Record
1	Can detection (camera-aided)	1.1 Move to scan position 1.2 Detect can position	✓
2	Grip can	2.1 **Move down** 2.2 Close gripper 2.3 **Move up**	✓
3	Place can	3.1 Move to target position 3.2 Move down 3.3 Open gripper 3.4 Move up to init. position	✓
4	Create variance by random placement	4.1 Move down 4.2 Close gripper 4.3 Move up 4.4 **Move to random position** 4.5 **Move down** 4.6 Open gripper 4.7 Move up to init. position	✗

Table 7.1: Actions of the pick-and-place application: All actions included in the recording are highlighted in gray.

7.3 ROBOT AND SIGNALS

We recorded the machine data of the robot arm including all the 6 axes at a frequency of 500 Hz. The machine data includes 130 signals in total which are listed in Table 7.2 and can be divided into target, measured, estimated and auxiliary signals. The target signals are the states commanded by the robot controller, the measured signals comprise directly measured sensor signals from which further signals are estimated. The auxiliary values are metadata that have been added subsequently for investigations of the dataset with the pick-and-place application and may not be entirely available in real operations. They must not be used for machine learning approaches. Measured signals related to each axis include joint positions, velocities, torques (2 sensors per axis) and torque-forming current I_q. Equivalent to the measured values, the target signals for each axis contain the target position, the target velocity and the expected torque. In addition to the 21 signals that are given for all 6 axes, 9 general electrical signals such as the supply voltage and the total current of the robot arm, for example, are also part of our dataset.

	Feature	∀Axes	Signal type	unit	type
metadata	time	✗	auxiliary	s	float
	sample	✗	auxiliary	-	int
	anomaly	✗	auxiliary	-	bool
	category	✗	auxiliary	-	int
	setting	✗	auxiliary	-	int
	active	✗	auxiliary	-	bool
	variant	✗	auxiliary	-	int
mechanical	target_position	✓	target	rad	float
	target_velocity	✓	target	rad/s	float
	target_accel.	✓	target	rad/s	float
	target_torque	✓	target	Nm	float
	computed_inertia	✓	estimation	kg · m^2	float
	computed_torque	✓	estimation	Nm	float
	motor_position	✓	measurement	rad	float
	motor_velocity	✓	measurement	rad/s	float
	joint_position	✓	measurement	rad	float
	joint_velocity	✓	measurement	rad/s	float
	motor_torque	✓	estimation	Nm	float
	torque_sensor_a	✓	measurement	Nm	float
	torque_sensor_b	✓	measurement	Nm	float
	power_motor_mech	✓	estimation	W	float
	power_load_mech	✓	estimation	W	float
electrical	motor_iq	✓	estimation	A	float
	motor_id	✓	estimation	A	float
	power_motor_el	✓	estimation	W	float
	motor_voltage	✓	measurement	V	float
	supply_voltage	✓	measurement	V	float
	brake_voltage	✓	measurement	V	float
	robot_voltage	✗	measurement	V	float
	robot_current	✗	measurement	A	float
	io_current	✗	measurement	A	float
	system_current	✗	measurement	A	float

Table 7.2: List of signals from our dataset, grouped according to metadata, mechanical and electrical signals. ∀*Axis* denotes an axis-specific signal which is given for all 6 axes. The metadata is further described in Section 7.3.

The metadata includes the following information:

- *time*: elapsed time (in seconds) within the sample
- *sample*: ID of the sample the data point is related to

- *anomaly*: denotes if related sample contains an anomaly
- *category*: denotes the anomaly type (see Table 7.3)
- *variant*: variant within the anomaly type (see Table 7.3)
- *active*: denotes if robot is currently moving
- *action*: action ID within the sample (see Table 7.1)

7.4 PICK-AND-PLACE OPERATION

A recording cycle of the dataset proceeds as follows: The robot is moved to a fixed scanning position and first detects the position of a randomly placed can on the conveyor belt through its integrated camera and grips it (see Figure 7.3). The robot arm then moves the can to a fixed target position on the opposite side of the conveyor belt ending the recording cycle. Finally, the can is placed at a random position and then conveyed back to the starting area for the next recording cycle which is triggered by the light barrier.

The random placement of the can introduces variance into the set of normal data, which is to be expected in this form in real applications. Note that the random placement itself is not part of the recording since it only serves as an auxiliary task for the dataset design. The steps of the entire recording process are described in more detail in Table 7.1. As usual, most of the motions in path planning were joint angle optimized for time efficiency. Only the *move up* and *move down* commands were performed with linear motion to ensure smooth picking and placing of the can.

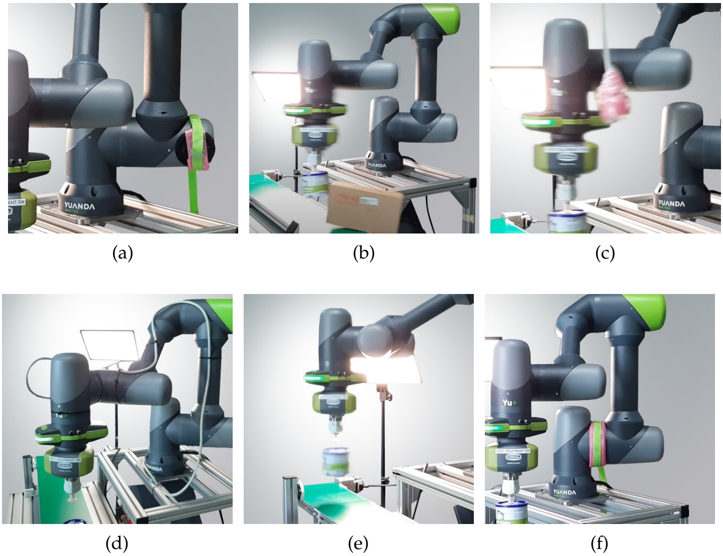
7.5 ANOMALIES

For the test set, we intentionally induced anomalies of different types, which cover most cases of anomalies occurring in reality. We divide the anomalies into process errors, gripping errors, and robot axis wear. An overview of all induced anomalies is given in Table 7.3.

7.5.1 *Process Errors*

Process errors concern errors in the environment around the robot, which arise from a faulty setup. Our dataset includes the following scenarios:

- Additional axis weight: This might be caused by components that have not been disassembled from the robot. A weight of 115g, 231g or 500g was attached to each of the 6 axes as shown in Figure 7.4 (a).



(a) (b) (c)

(d) (e) (f)

Figure 7.4: Examples of induced anomalies: Increased moment of inertia caused by additional axis weight (a), collisions with boxes (b) and a hanging cable (c), cable routed at robot (d), losing the can during movement (e) and increased friction of a single axis (f).

Anomaly type	**Cause**	**Num.**	**Variants**
Additional friction	axis wear	144	2 levels × 6 axes
Miscommutation	axis wear	89	6 axes
Misgrip of can	gripping err.	11	1
Losing the can	gripping err.	74	random position
Add. axis weight	process err.	156	3 weights × 6 axes
Collision w/ foam	process err.	72	2 sizes × rand. pos.
Collision w/ cables	process err.	48	2 types × rand. pos.
Collision w/ cardboard	process err.	22	random position
Varying can weight	process err.	80	6 weights
Cable routed at robot	process err.	10	1
Invalid gripping pos.	process err.	12	random position
Unstable platform	process err.	37	3 levels

Table 7.3: List of anomaly types grouped in their cause. The listed variants are labeled in the dataset.

This was done for each axis. The additional weight will affect all joint torques due to increased inertia and gravity.

- Collisions: Collisions of the robot with objects can occur due to process flow errors or human errors. Our dataset includes collisions with lightweight objects such as foam, hanging cables, or empty cartons induced, as these are difficult to detect. The cartons and various foam cubes are placed at different positions so that recordings of collisions exist at different times. In some cases the objects are hit head-on, in others they are merely touched. Note that collisions sometimes occur even when the robot is not moving, since the previously hit object may swing back. Figure 7.4 shows exemplary collisions with a carton (c) and a hanging cable (d).
- Varying can weight: As an example of disturbances in the process, runs with different weights of the can are included as anomalies. This may be the result of a damaged can or filling errors in a previous processing step. The can used in the normal data weighs 241g while recorded anomalies include 6 different weights between 41g (empty can) and 448g.
- Cable routed at robot: Cables or hoses are often routed along the robot arm for various tools such as screwdrivers or painting equipment. If this cable is not dismantled again for another application, this can affect the current application. Included in this dataset are recordings in which a cable is attached to the robot with cable ties as shown in Figure 7.4 (d).
- Invalid gripping position: An error in the process sequence, such as a worn conveyor belt with extended stopping distance, may result in unusual gripping positions of the can. Camera-based object detection can compensate for these errors as long as the can remains in the camera's field of view. This unusual gripping position nevertheless indicates an unusual process sequence. A time delay is integrated in the software of the robot application for recording unusual gripping positions.
- Unstable platform: In collaborative applications, robot arms are often mounted on mobile platforms. An incorrectly adjusted mobile platform as well as uneven ground can lead to unwanted vibrations and thus affect the robot arm. The dataset contains recordings with three differently unstable mountings of the mobile platform.

7.5.2 *Gripping Errors*

Defects or contamination of the vacuum gripper may cause the can to become detached due to acceleration or centrifugal force. The loss of the can also happen during braking before the target position, which is why there are also samples in which the robot loses the can shortly before placement. To simulate the anomalies *Losing the can* and *Misgrip of can*, the gripper is timed to open during different movements of the robot arm as shown in Figure 7.4 (e) or the can is not gripped at all because the gripper is intentionally switched off.

7.5.3 *Robot axis wear*

The following two anomaly types simulate wear of the robot axes:

- Miscommutation: The wear of mechanical and electrical components decreases the efficiency of the respective robot axis. This decreasing efficiency is simulated via an incorrect commutation of the motor, which we adjust with software parameters. For each case, the recordings contain a single incorrectly commuted axis.
- Increased friction: Dirt or age-related wear can lead to increased friction of individual robot axes. The additional friction is generated by attaching foam at the transition point of two robot axes. The foam is fastened with a tension belt as shown in Figure 7.4 (f). In each sample of this anomaly type, there is one axis with increased friction in one of two strength levels.

7.6 ANALYSIS

In this section, we explain and visualize the characteristics of the data set at the signal level in more detail. To this end, we will first analyze the normal data and then describe how the anomalies deviate from them in the following subsections.

7.6.1 *Normal Data*

Figure 7.5 shows selected signals of the second robot axis from an exemplary recording from the normal data of the voraus-AD dataset. Between the second 0 and 1, the robot moves to the fixed scan position and remains there to detect the can. The minimum and maximum values are close to each other for the first 4.5 seconds since all movements up to this point are identical and the recording always starts at the same time.

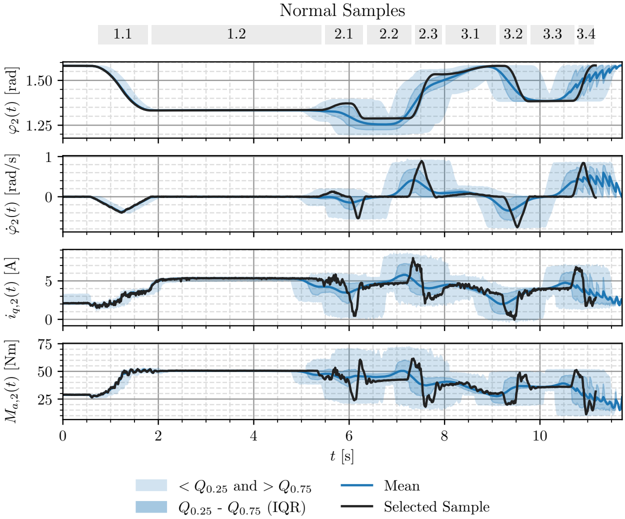
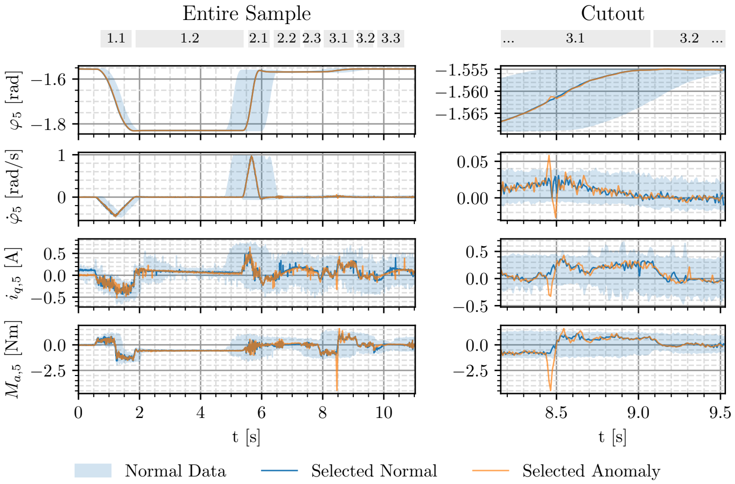


Figure 7.5: Selected signals from the second axis of the normal data within our voraus-AD dataset. The light blue areas represent the different quartiles, while the blue line shows the mean signal and the black line a random sample. At the top, the individual actions from Table 7.1 are shown.

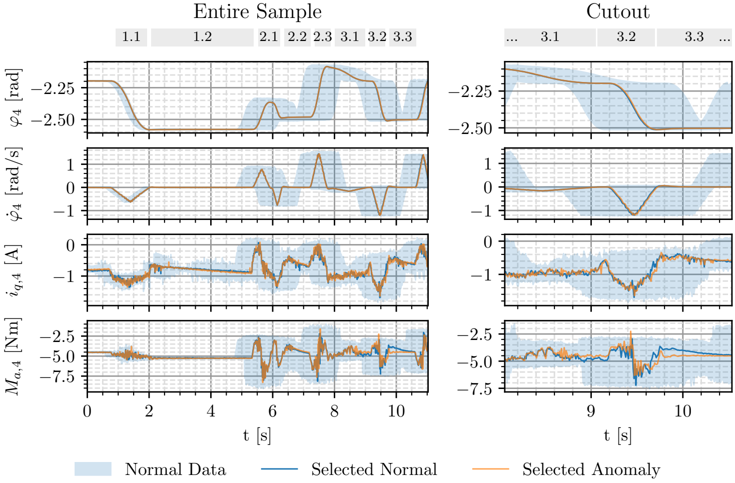
Starting from the visual object detection of the can, the variance is increased due to the temporal and spatial offset caused by the movement to different positions of the can. In the following actions 2-4 of Table 7.1 are performed as marked at the top of Figure 7.5.

7.6.2 *Anomalies*

Figure 7.6 (a) shows selected signals from an anomaly where the robot arm collides with a foam cuboid. The collision occurs at about $t = 8.45$ s as visible on all signals by a deflection in the signal. After the robot arm has deviated from its joint angle target due to the collision the closed-loop control counteracts to correct the path which influences the joint angle φ_5 and its velocity $\dot{\varphi}_5$. The deviation of the torque $M_{s1,5}$ is explained by its physical relationship with external forces acting on the robot.
The signals of the fourth axis from the anomaly *losing the can* are shown in Figure 7.6 (b). An effect on the joint angle φ_4, the joint angular velocity $\dot{\varphi}_4$ and the motor current $i_{q,4}$ is not visually apparent. However, the torque $M_{s1,4}$ of the fourth axis shows a short oscillation at $t = 8.6$ s, when the robot arm loses the can. After the loss of the can, for $8.8 \text{ s} < t < 9.6$ s,



(a) Collision with foam



(b) Losing the can

Figure 7.6: Selected signals of the anomalies *collision with foam* and *losing the can*. The left plot shows the entire sample, while the right side zooms into the time period of the anomalous event. For the normal sample, we visualize the most similar example in the dataset.

Dataset	**AURSAD** [116]	**voraus-AD (ours)**
Application	**screwdriver**	**pick-and-place**
Variance (normal data)	**discrete (screw holes)**	**continuous (rand. position)**
Robot	UR3e	Yu-Cobot
#Axes	6	6
Anomaly types	**4**	**12**
#Normal samples	1420	1367
#Anom. samples	625	755
∅sample duration	**15.3s**	**11s**
Signal frequency	**100 Hz**	**500 Hz**
#Signals	125	130
Raw dataset size	**5.81 GB**	**11.17 GB**

Table 7.4: Comparison between AURSAD and voraus-AD. The main differences are highlighted in bold.

a lower absolute torque is measured which can be explained by the decrease in weight as a result of the loss of the can. In the time around $t = 10$ s, an increased absolute torque is measured compared to the error-free recording, since there is no contact between the conveyor belt and the (lost) can which would dampen the weight force.

7.6.3 *Comparison to AURSAD*

Table 7.4 compares the key characteristics from our dataset with AURSAD [116], which is currently the only comparable dataset in this domain. The different task of the robot (screwdriver vs. pick-and-place in our case) results in different challenges for the detection: The variance of the normal data in AURSAD is nearly discrete, as the screws are driven into pre-defined holes, whereas the samples of our dataset are continuously distributed, since the can is at a random, not pre-defined location on the conveyor belt before gripping it. While AURSAD contains only 4 anomaly types, which are exclusively related to the screw and plate, voraus-AD contains 12 diverse anomaly types, which cover most possible scenarios in reality. Most of our anomaly types (7), such as collisions or axis-wear-related anomalies, do not only occur exclusively in pick-and-place tasks, but are general potential faults in robotics applications. Thus, evaluations on our dataset provide insights for other robotics applications as well. The machine data of voraus-AD was provided with a significantly higher sampling rate (500 Hz vs. 100 Hz), which allows the detection of high-frequency deviations. In contrast to AURSAD, our dataset includes measurements of torque sensors instead of the force on the end effector.

7.6.4 *Evaluation Protocol*

As shown in the previous chapters and introduced in Section 3.4, anomaly detection methods are evaluated using the area under the *receiving operator characteristic* (AUROC). It measures the integral of the true positive rate (TPR) over every false positive rate (FPR) by varying a threshold $\theta \in (-\infty, \infty)$ of a binary decision.

The AUROC should be measured for each anomaly type individually to identify the strengths and weaknesses of a method. For an overall comparison, it should be averaged over all 12 types. We do not recommend calculating the area under one single ROC over all samples and anomaly types, as the number of samples per anomaly type strongly biases the metric in this case.

7.7 CONCLUSION

We introduce a publicly available AD dataset for robot applications which offers a novel level of variety in anomaly types generalizing to many real scenarios without external sensors. In addition to the 130 signals of machine data, we provide highly detailed metadata for the pick-and-place cycles. This benchmark makes the evaluation of AD methods for robot applications more transparent and comparable such that upcoming methods may demonstrate their effectiveness on this dataset. In addition to the standard evaluation, we encourage the research community to work on efficiently dealing with the number of signals or samples during training.

In the following chapter, we introduce a methodology that takes concepts from chapters 4 (DifferNet) and 5 (CS-Flow) and transfer them to multivariate time series, as in voraus-AD. We also evaluate how existing methods for time series, which have been applied mainly to rather few signals so far, perform on our large dataset.

8

MTS-FLOW

In the following, we present a new baseline for anomaly detection on multivariate time series and provide extensive experiments on our newly proposed dataset voraus-AD from Chapter 7. Thereby we aim to detect whether an entire time series contains any anomaly. Similar to DifferNet and CS-Flow (Chapters 4 and 5), our proposed *Multivariate Time Series Flow* (MTS-Flow) is based on density estimation of normal data via Normalizing Flows whose architecture we tailored to multivariate time series. As previously introduced, the estimated likelihoods of the NF are used as an indicator for anomalies: We assume that anomalies have a low likelihood whereas the likelihood of normal data should be high. Instead of using a feature extractor as for the previously proposed vision approaches, density estimation is performed directly on the machine data. To apply Real-NVP-based NFs to time series, we adapt the internal networks by exploiting the temporal structure via convolutions while preserving the concept of signals which we shuffle, permute, transform and recombine inside the coupling blocks. Furthermore, we also enable a temporal analysis of the signals by transferring the principle of localization from Section 4.2.4 which utilizes the input gradient of the anomaly score. Our implementation can be accessed via GitHub[1]https://tnt.uni-hannover.de/vorausAD. This chapter is largely based on our publication "The voraus-AD Dataset for Anomaly Detection in Robot Applications" [15].

8.1 METHOD

Similar to the previously presented methods in Chapters 4 to 6, our MTS-Flow $f : \mathcal{X} \rightarrow \mathcal{Z}$ transforms the unknown data distribution p_X of normal samples $x \in X_{\text{tr}}$ to a target space $\mathcal{Z}$ with known base distribution p_Z. Since f is bijective, this enables us to measure the likelihoods of data points after mapping them to the target space using $z = f(x)$. Using a standard multivariate Gaussian distribution $\mathcal{N}(0, I_d)$ as target

---

[1] https://tnt.uni-hannover.de/vorausAD

distribution with d as the number of dimensions of z, the density of the target space is defined by

$$p_Z(z) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\|z\|_2^2}. \tag{8.1}$$

Following the change of variables formula, the Jacobian of f, the likelihood of data points is given by

$$p_X(x) = p_Z(z) \left| \det \frac{\partial z}{\partial x} \right|. \tag{8.2}$$

The final decision on whether the sample is classified as an anomaly is given by thresholding the likelihood

$$\mathcal{A}(x) = \begin{cases} 1 & \text{for } p_X(x) < \theta \\ 0 & else \end{cases} \tag{8.3}$$

with θ as an adjustable parameter. In practice, the parameter can be set according to a previously defined acceptable false positive rate using the statistics of normal data. In the following, we describe the pipeline of our proposed MTS-Flow in Section 8.1.1 and its training procedure in Section 8.1.2.

8.1.1 *Architecture*

We adapt the design of the Normalizing Flow to exploit the structure of multivariate time series to appropriately model the data distribution by defining the permutation strategy as well as the design of the internal networks. Thereby, we build on the Real-NVP architecture which we describe in detail in Subsection 4.1.3.

Our MTS-Flow processes entire time series with dimensions $T \times S$ with T as the number of time steps and S as the number of signals. We exploit and maintain the temporal structure by using convolutions for the internal functions instead of fully connected blocks as in [95]. Thereby, we follow [13], [14], in which state-of-the-art performance for anomaly detection on image data with convolutions in the internal networks of NFs was shown. The partitioning into signals is maintained as well. However, the signals no longer have their original meaning during the transformation steps and should be interpreted as latent signals at the output, which are independent of each other.

A coupling block of our MTS-Flow is designed as follows: First, the signals of the input are reordered using a permutation matrix $P \in \{0,1\}^{S \times S}$

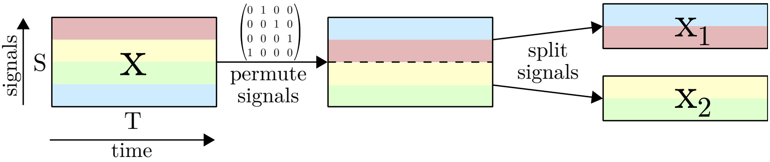

signals
S
X
T
time
$\begin{pmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{pmatrix}$
permute signals
split signals
X_1
X_2

Figure 8.1: Permutation and split procedure inside a coupling block of MTS-Flow: First, the signals are permuted in a fixed, randomly defined, manner. After that, the sample is divided into two components, each comprising half of the signals. Best viewed in color.

which is randomly determined initially for each block. We split the sample along the signal axis evenly into the parts x_1 and x_2 with dimensions $T \times \frac{S}{2}$. This enables the usage of correlations between different signals for density estimation. Figure 8.1 visualizes the permutation and split procedure.

The parts are then fed into the internal networks to transform the counterpart subsequently. The internal networks consist of a sequence of 3 one-dimensional convolutions, whose kernel is moved along the time as a sliding window, and ReLUs as nonlinearities as shown in Figure 8.2. We reduce the number of signals in the hidden layers with a factor r which we refer to as *signal scaling*. After the last convolutional layer, the output is split evenly along the signal axis to obtain the scaling and translation coefficients s and t, matching the dimensions $T \times \frac{S}{2}$ of x_1 and x_2. Since the exponentiation of the scaling coefficients can cause an unstable optimization due to exploding gradients, we apply soft-clamping as in Chapters 4 to 6 for better convergence:

$$\sigma_\alpha(s) = \frac{2\alpha}{\pi} \arctan \frac{s}{\alpha} \tag{8.4}$$

with α as a hyperparameter that controls the clamping magnitude by restricting the values to the interval $(-\alpha, \alpha)$.

8.1.2 *Training*

During training, we optimize the parameters for the internal networks such that the training data in the latent space is normally distributed. As already described in Subsection 4.2.2, the overall model is trained via maximum-likelihood training: The probability of the training data according to the model should be maximized. Instead of maximizing the likelihood, we minimize the negative log-likelihood $-\log p_X(x)$ for convenience as it is equivalent and numerically more practicable. As

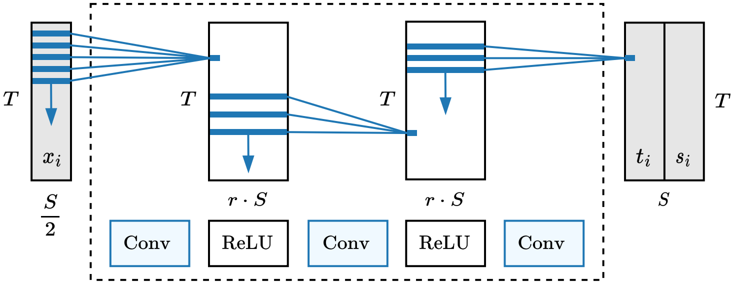


Figure 8.2: Architecture of the internal networks in MTS-Flow.

previously shown in Equations 4.9 to 4.11, this comes down to minimize the following loss function:

$$\begin{aligned} \mathcal{L}(x) &= -\log p_X(x) = -\log p_Z(z) - \log \left|\det \frac{\partial z}{\partial x}\right| \\ \mathcal{L}(x) &= \frac{\|z\|_2^2}{2} - \log \left|\det \frac{\partial z}{\partial x}\right|. \end{aligned} \tag{8.5}$$

8.1.3 *Temporal Analysis*

In addition to the sequence-level detection which provides if any unusual event has occurred, in practice it is often relevant to analyze when this event happened. This helps to find the cause of the error and to possibly eliminate it. We provide a measure that indicates the impact of each time step on the anomaly score. Similar to DifferNet in Chapter 4, we use the gradient of the anomaly score with respect to the inputs by backpropagating the loss from Eq. 8.5. The input gradient $\nabla_x \in \mathcal{R}^{T \times S}$ is aggregated to a gradient for every time step t by taking the ℓ_1-norm over all signals:

$$\nabla_x^t = \sum_{s=1}^{S} |\nabla_x(t,s)|. \tag{8.6}$$

8.2 EXPERIMENTS

In this Section, we first present the experimental setup in Subsection 8.2.1) and then compare our baseline MTS-Flow with previous work in Subsection 8.2.1. Finally, in Subsection 8.2.3 the characteristics of the voraus-AD dataset are varied to explore its impact on the detection performance.

General Parameters	Value
Signal frequency	100 Hz
Batch size	32
Optimizer	Adam [102]
Learning rate	$8 \cdot 10^{-4}$
Learning rate decay	0.1
Decay epochs	11, 61
Total epochs	70
NF Parameters	
Coupling blocks n_{blocks}	4
Signal scaling r	2
Kernel sizes k_1, k_2, k_3	13, 1, 1
Dilation d_1, d_2, d_3	2, 1, 1

Table 8.1: Hyperparameter setting for MTS-Flow

8.2.1 *Implementation Details*

In the following, we describe the implementation details of MTS-Flow and the other baselines we compare. As further baselines we use a PCA-based approach [77], 1-nearest-neighbor distance (1-NN [57]), a convolutional autoencoder (CAE), a LSTM-based VAE [75], a Hidden Markov Model [84] and a graph-augmented Normalizing Flow (GANF [117]). We have used a sampling frequency of 100 Hz for all methods and provide a uniform dimensionality by padding samples to the maximum length for all experiments and all methods, unless otherwise stated.

MTS-FLOW Table 8.1 summarizes the hyperparameters of our method for the following experiments We apply a learning rate decay during training to achieve a fast but stable convergence. The initial learning rate of $8 \cdot 10^{-4}$ is multiplied by a factor of 0.1 in epochs 11 and 61. The training ends after a fixed number of 70 epochs. For the NF, individual kernel sizes and dilation parameters for the 3 convolutional layers inside the internal networks are used. While the first convolution takes 13 sample points with a dilation of 2, the other 2 layers process the signal locally having a kernel size of 1. The number of signals is scaled in the hidden layers with a factor of $r = 2$. In total, the NF is a chain of 4 blocks, each using the same hyperparameters.

1-NN Since the comparison of a sample to its nearest neighbor among normal samples seems to be helpful for humans to identify anomalies (see Fig. 7.6), we utilize the nearest neighbor distance as anomaly score as proposed by [57]. More specifically, we use the ℓ_1-norm to the nearest neighbor in the training set as a better performance was observed compared to the ℓ_2-norm.

PCA For the PCA-based approach from [77], we performed a parameter tuning on the number of components in which 90 principal components explaining 81.1% of the variance have shown to provide the best performance.

CAE As another baseline, we implemented a convolutional autoencoder (CAE) with the ℓ_2 reconstruction error as anomaly score, similar to the SWVAE [71] and work from other domains [40], [42], [63], [118]. Note that [71] was originally applied for 12 signals (instead of 130 here) having a space complexity of $\mathcal{O}(S^2)$ with S as the number of signals which is why we could not apply it to voraus-AD for memory reasons. The CAE includes 1D convolutional layers for the encoder that run over time. For the decoder, we use transposed convolutions. A hyperparameter search gives us the following configuration: The encoder and decoder comprise 3 layers each with a kernel size of 7, strides of 2 and 220 hidden channels. The latent space has 200 dimensions. We optimize this network for 60 epochs with Adam [102] and an initial learning rate of $5.2 \cdot 10^{-4}$ which is multiplied by 0.1 after 55 epochs.

GANF We used the official repository of Dai and Chen to perform experiments with GANF [117]. Hyperparameters were taken from their experiments with SWaT [119] since the task and dataset are similar to our scenario.

HMM For the HMM method by Azzalini et al. [84] we decided for the online approach as it outperformed the offline approach with less time and space complexity. As described by the authors, we determined the number of states by the BIC score, running k from 1 to 12, restricting covariance types for the multivariate Gaussian distributed emission probabilities to spherical and diagonal for runtime reasons. This parameter search yielded the optimal BIC score at k=7 and a covariance matrix with only diagonal entries being nonzero.

LSTM-VAE We used our own reimplementation for the LSTM-VAE from Park et al. [75]. Since we observed over-optimization for constant signal sections by having tiny standard deviations in the output, which resulted in poor reconstruction of the remaining signals, we constrained a lower bound of 0.05 for the standard deviation outputs by adding it to the softplus activation. It has shown that the LSTM-Memory has problems processing the 130 signals for high sampling rates, which is why we evaluated with 10 Hz here. A hyperparameter optimization gave us a configuration with 90 epochs, 128 hidden neurons for the LSTMs, a latent space dimension of 32 and a learning rate of 10^{-3}.

Method	1-NN [57]	PCA [77]	CAE adapts [71]	LSTM-VAE [75]	GANF [117]	HMM [84]	MTS-Flow **(ours)**
Add. Friction	74.8	76.4	89.4 ± 0.2	88.7 ± 1.5	88.5 ± 4.5	88.0 ± 0.7	**96.6** ± 0.6
Miscommutation	80.8	87.0	99.1 ± 0.0	98.1 ± 0.5	98.8 ± 1.1	93.7 ± 1.3	**99.8** ± 0.3
Misgrip can	**100.0**	**100.0**	**100.0** ± 0.0	**100.0** ± 0.0	47.6 ± 13.1	71.0 ± 3.3	95.3 ± 3.3
Losing can	68.7	70.1	72.6 ± 0.3	70.4 ± 2.9	72.1 ± 5.8	88.8 ± 1.0	**96.2** ± 0.4
Add. axis weight	75.0	79.2	93.5 ± 0.1	82.7 ± 1.1	93.2 ± 2.4	89.6 ± 1.2	**94.1** ± 0.7
Coll. foam	69.6	73.9	81.5 ± 0.2	81.5 ± 2.1	81.2 ± 6.2	**89.8** ± 1.5	87.5 ± 1.2
Coll. cables	74.5	75.7	79.6 ± 0.3	77.3 ± 3.6	82.7 ± 6.4	**91.8** ± 1.4	84.7 ± 1.2
Coll. cardboard	82.8	83.6	78.6 ± 0.3	82.8 ± 4.8	77.5 ± 5.8	86.3 ± 1.2	**88.3** ± 1.2
Var. can weight	63.7	64.2	72.3 ± 0.3	71.4 ± 2.1	68.9 ± 8.7	**90.9** ± 2.0	85.1 ± 1.1
Cable at robot	63.0	71.6	83.1 ± 0.3	96.0 ± 1.4	76.6 ± 8.1	84.4 ± 1.1	**100.0** ± 0.0
Invalid grip. pos.	93.4	92.1	88.8 ± 0.3	97.7 ± 1.3	86.6 ± 10.8	91.8 ± 1.4	**100.0** ± 0.0
Unstable platform	83.6	85.9	83.9 ± 0.2	93.6 ± 1.9	84.6 ± 3.8	82.5 ± 1.2	**96.1** ± 0.7
Mean	77.5	80.0	85.2 ± 9.2	86.7 ± 10.1	79.9 ± 12.7	87.4 ± 5.8	**93.6** ± 5.7

Table 8.2: Anomaly detection results on voraus-AD for MTS-Flow and other baselines measured in AUROC percentage. MTS-Flow shows the best performance for most categories and on average.

8.2.2 *Results*

In this section, we evaluate current state-of-the-art methods along with MTS-Flow on voraus-AD. As introduced in Subsection 7.6.4, the evaluation protocol involves measuring the AUROC for each anomaly type and the mean AUROC across all types. The baselines include all the described methods from Section 8.2.1. Table 8.2 summarizes the results including the AUROC for every anomaly category and the mean AUROC over all of them. We report the mean and standard deviation over 9 runs for the non-deterministic training of MTS-Flow, LSTM-VAE, GANF, HMM and CAE with different random initializations. Figure 8.3 shows the underlying ROC curves for the median performance over all runs.

MTS-Flow outperforms all other baselines for 8 out of 12 categories and on average by a large margin of 6.2%. It is notable that the relative differences in performance between the methods vary strongly depending on the anomaly category. For example, in contrast to MTS-Flow or HMM, the comparatively simple approaches 1-NN and PCA can perfectly detect a misgrip of the can, but underperform on average. Hidden Markov Models, on the other hand, detect the misgrip with only 71% AUROC,

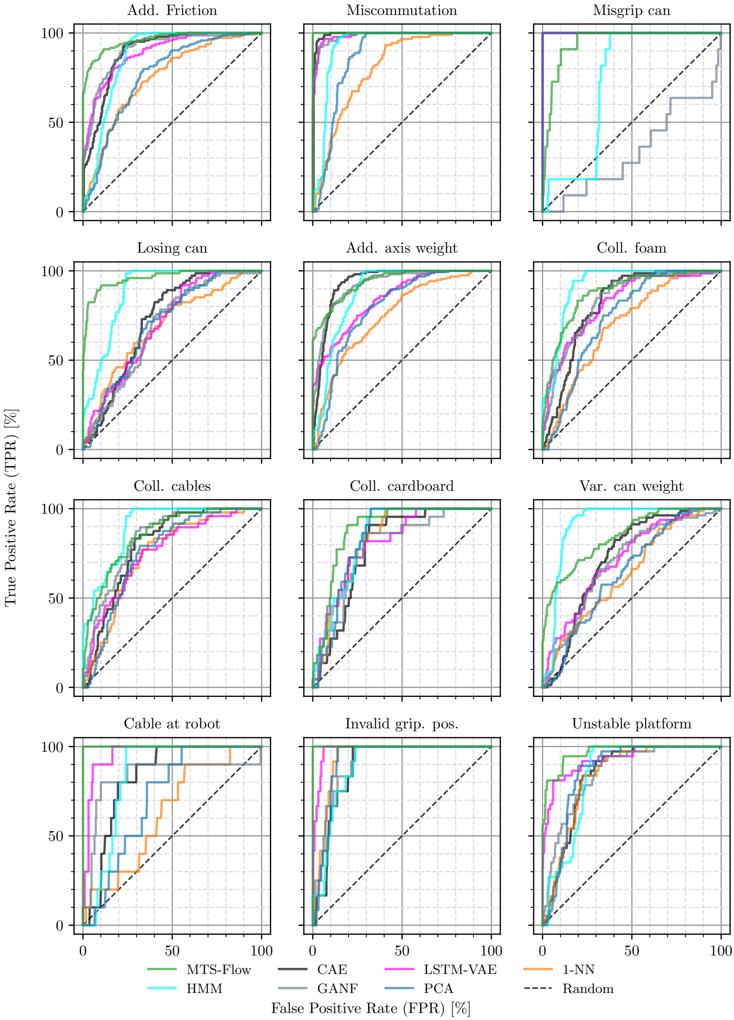


Figure 8.3: Each diagram shows the ROC-Curve for one anomaly type color-coding the methods.

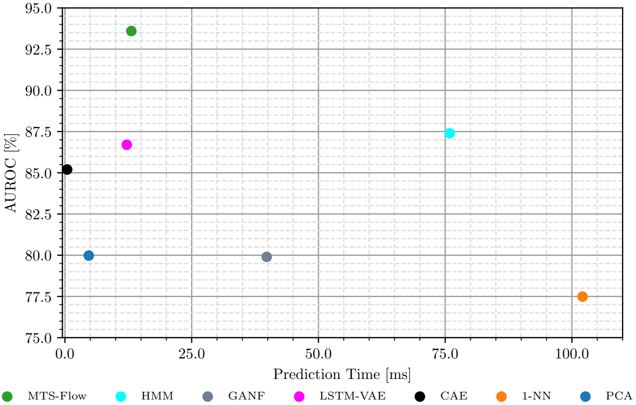


Figure 8.4: Inference time vs. AD performance for all baselines. Times for MTS-Flow, GANF, LSTM-VAE and CAE are given using *pytorch* on a *NVIDIA GeForce RTX 4090* as GPU. Experiments with HMM, PCA and 1-NN were made with the python packages *hmmlearn*, *pytorch-CPU* and *sklearn*, respectively, and measured with a *Intel i9-13900K* CPU.

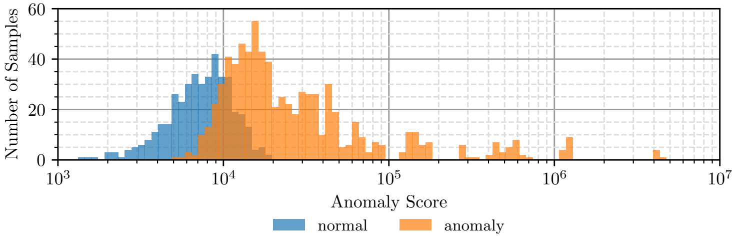


Figure 8.5: Histogram of anomaly scores for normal data and anomalous data for MTS-Flow. In practice, a threshold inside the overlapping interval around 10^4 would be chosen.

but are most sensitive to collisions with about 86 to 92%. Figure 8.4 relates the performance to the inference time of the given methods for samples from voraus-AD at 100 Hz. Although PCA (4.7 ms) and CAE (0.43 ms) and LSTM-VAE (12.2 ms) perform faster than MTS-Flow (13.1 ms), all of these methods are fast enough for most real-time requirements in practice. The nearest neighbor method scales with the training dataset size and might not meet these requirements beyond a certain size (here 102.1 ms). Similarly, the computation time of the HMM approach (here 75.9 ms) is very sensitive to the number of signals as it involves the calculation of determinants with $\mathcal{O}(S^3)$. In contrast, the computation time of our method is not related to the dataset size and scales with $\mathcal{O}(S^2)$ assuming a constant signal scaling in the internal networks.

The distribution of anomaly scores for normal data and anomalies are visualized in Figure 8.5 on a log scale for anomaly scores. While the normal samples show almost a normal distribution of rather small scores, we observe a shifted long-tailed distribution towards large scores for anomalies. Despite there is room for some improvement for many samples in the mid-range where the histograms overlap, however, there is a notable ratio of data points that are clearly separated from the opposite class.

8.2.3 *Ablation Studies*

To get a deeper understanding of MTS-Flow, we experiment with variations regarding all dimensions including signals, time and sample size. For all experiments, 9 runs were used with the same random initializations as before.

Figure 8.6 (a) shows the mean AUROC when using different subsets of signals. We use the categorization from Table 7.2 and refer to the joint group of targets and estimations as *computed* signals. Mechanical signals provide clearly more importance for AD compared to electrical signals improving the performance by 25%. Both measured and computed signals lead to a moderate performance around 86% although this varies more for measured signals. Overall, we observe that each group provides useful information, as the AUROC is above the *random* baseline of 50% in each case, and that using all signals gives the best performance.

We also investigated the influence of the sampling frequency within the range of 10 to 500 Hz as shown in Figure 8.6 (b). Note that we adapted the dilation parameter for the first convolution of every block to 10 for the 500 Hz experiment as otherwise a small temporal context would be considered. The detection improves with raising frequency from 10 Hz to 50 Hz as anomalies may be characterized by high-frequent components

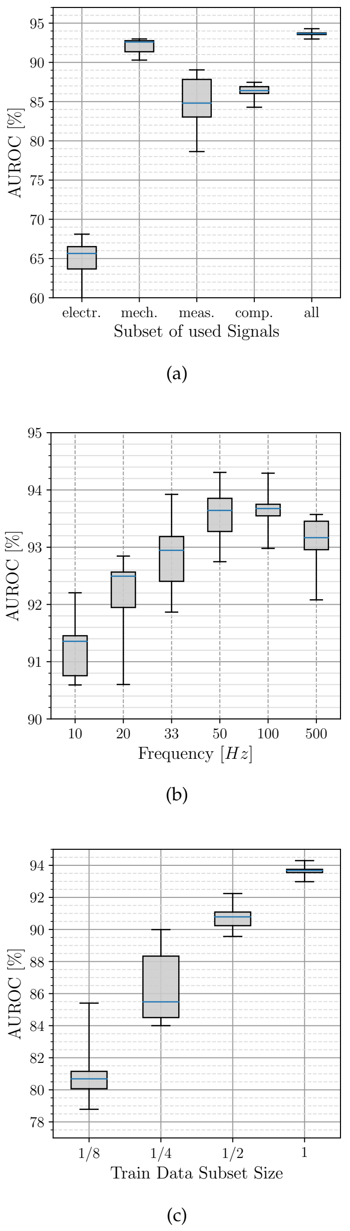


Figure 8.6: Ablation Studies regarding signal types (a), sampling rate (b) and training set size (c) when applying MTS-Flow on voraus-AD.

in this range. This effect is mostly saturated from 50 Hz and slightly reverses for sampling frequencies up to 500 Hz. Since upcoming methods could still benefit from higher frequencies, we provide signals with the original sampling at 500 Hz.

Figure 8.6 (c) outlines the relation between training set size and AD performance. We used random subsets for individual runs and kept the number of training iterations constant over different subset sizes. It turns out that high detection quality is merely achieved with many examples, indicating that detecting anomalies in voraus-AD is not straightforward. Future work may optimize the detection with less data.

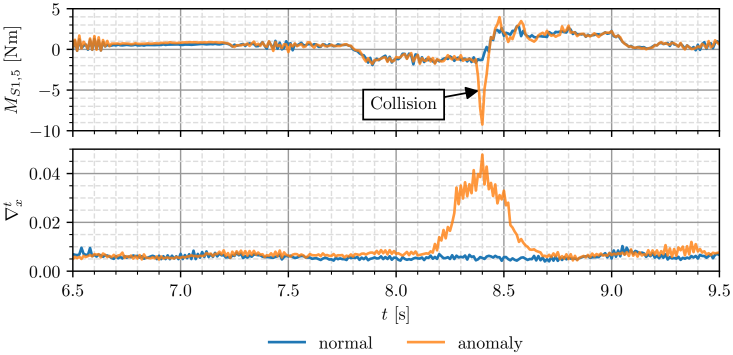
8.2.4 *Temporal Analysis*

We use the input gradient as an indicator for the temporal occurrence of anomalies as described in Section 8.1.3. We evaluate the temporal analysis qualitatively by showing the gradient characteristics for selected examples. Note that a quantitative evaluation is hardly feasible for many cases since the exact timeframe is indeterminable for gradually increasing or decreasing impacts.

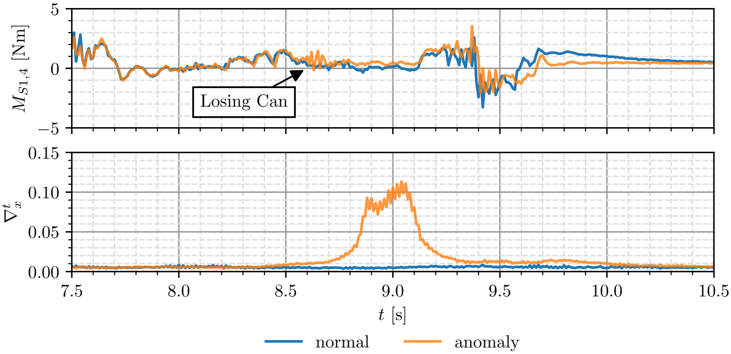
Figure 8.7 visualizes the torque signal of anomalous sequences and the corresponding input gradient ∇_x^t from Eq. 8.6. A collision with a foam as shown in Fig. 8.7 (a), which occurs in the original sample at 8.4 s (upper plot) is clearly visible in the gradient (lower plot) at this time. Note that the receptive field causes some dilatation of this peak around the event. Interestingly, this peak has a delay for the event *losing the can*, shown in Fig. 8.7 (b), since the missing can weight affects the forces in the following movement. For the anomaly-free sample, the gradient is relatively constant at a low level between 0.05 and 0.1 with some noise. In practice, a simple peak detection after smoothing the signal would provide the unusual parts of the signal.

8.3 CONCLUSION

We present MTS-Flow which transfers the success of density estimation with NFs for AD to the data domain of high-dimensional time series via a tailored architecture design. MTS-Flow outperforms previous work regarding sample-wise detection by a large margin of 6.2% and provides a temporal analysis to inspect unusual events further. Future work could work on efficiently dealing with the number of signals or samples during training. The method may also be further enhanced by combining it with an asymmetric student-teacher approach, for example, as in Chapter 6. Moreover, the density estimation could be applied to some other rep-



(a) Collision with foam



(b) Losing the can

Figure 8.7: Temporal analysis: The occurrence of unusual events is temporally determinable by measuring the input gradient. The upper plots show original anomalous torque signals (orange) and its nearest neighbor among normal samples (blue). The lower plots are the individual input gradients which are highly increased when an unusual event occurs.

resentation of the input data, e. g. by transforming it to the frequency domain or by leveraging some feature extractor as for the previously proposed approaches on image data.

9

CONCLUSION

In this thesis, we address the problem of automatic anomaly detection using machine learning, focusing on the industrial context. An anomaly is defined as an observation deviating from the majority such as a defective product. Detecting such anomalies is crucial for ensuring product quality and preventing equipment failures, process deviations or safety hazards in industrial settings. The problem is formulated as a semi-supervised anomaly detection task, where only normal data is available during training with the goal of identifying abnormal observations during inference. Throughout this thesis, we have presented selected methods to deal with this problem on images as well as on time series of robot machine data.

IMAGE-BASED METHODS

As the main contributions, we proposed three image-based anomaly detection methods, namely DifferNet, Cross-Scale-Flow (CS-Flow), and Asymmetric Student-Teacher Network (AST). All three methods leverage a pretrained neural network to extract image features on which a density estimation is learned using Normalizing Flows. Additionally, they provide anomaly localization maps, allowing us to pinpoint the regions of anomalies within the images.

We first introduced DifferNet (Chapter 4), demonstrating that the likelihoods provided by a Normalizing Flow can be effectively used for image-based anomaly detection by utilizing the bijective property of this neural network. This involves a learnable maximum likelihood estimation using a Real-NVP-based flow to obtain the density of the composite of compact feature vectors that were extracted from different scales of the image. The flow transforms the data into a latent space with known density where likelihoods are directly measurable. To identify the anomalous regions in the image, the gradients of a backpropagation of the likelihood up to the input image are visualized.

With CS-Flow (Chapter 5) we have proposed an extension to DifferNet that provides a significant gain in detection performance by learning the density of entire feature maps of different scales by our specially designed flow. Full-resolution feature maps allow anomalies of fine structures to be better represented and the local context to be considered. A custom architecture was introduced that enables the processing of the composite of multi-scale feature maps, where within the internal networks these scales can interact with each other to exploit correlations between them. Since the outputs of CS-Flow are also maps indicating the likelihood of image regions, these outputs provide a direct indication of localization.

AST (Chapter 6) combines the concepts of NF-based density estimation with student-teacher networks. In the latter, a student network is optimized to imitate a teacher network on normal data. Since this imitation usually fails on anomalies, the distance between the network outputs can be used as an anomaly indicator. However, as we discovered that this imitation unintentionally works for some anomalies in existing methods due to an undesired generalization caused by the similar design of student and teacher, we propose a particular form of asymmetry of the networks: While the teacher is realized by a Normalizing Flow, trained for density estimation similar to CS-Flow, the student is a conventional network, similar to ResNet [99]. Since both networks have a substantially different internal structure and the output of the bijective NF-based teacher is more responsive to anomalies, the student is further hindered from imitating the teacher on anomalous data, resulting in higher student-teacher distances and thus a more reliable detection. Besides the application on RGB images, AST also achieves state-of-the-art results on 3D data or their combination.

AD FOR ROBOT MACHINE DATA

To facilitate the evaluation and advancement of anomaly detection methods in robotics, we introduced the voraus-AD dataset (Chapter 7), containing time series of machine data from a robot performing a non-deterministic pick-and-place task. This dataset will represent the first publicly available anomaly detection dataset in robotics with a diverse range of anomalies and signals, providing a valuable resource for future research in this field. We provide a total of 1367 anomaly-free and 755 anomalous time series, divided into 12 anomaly types which include axis wear simulations, gripping errors and several process errors as collisions or faulty setups. The 130 signals comprise mechanical and electrical measures from 6 axes at a sampling rate of 500 Hz.

Furthermore, we adjusted the concepts from the previously introduced image-based AD methods DifferNet and CS-Flow to multivariate time series and conducted extensive experiments on our newly proposed dataset voraus-AD. As before, our method MTS-Flow (Chapter 8) relies on a learned density estimation of normal data using Normalizing Flows, allowing for identifying anomalies by their small likelihood. We leveraged the data structure of the multivariate time series to adapt the flow for this domain. The internal networks of the coupling blocks comprise temporal convolutions that operate on subsets of signals to transform the data distribution to a well-known target distribution. Moreover, we enabled a temporal analysis of the signals by transferring the localization approach from DifferNet, utilizing the input gradient of the anomaly score.

FUTURE WORK

This work has focused on anomaly detection using pre-extracted features from RGB images or direct processing of 3D scans. Future work may investigate how the feature extraction could fine-tuned to other possible domains as X-ray or hyperspectral imaging. These give a different dynamic range with possibly different structures to interpret which may be not optimally processed by RGB-pretrained networks. For example, various self-supervised techniques [120]–[122] could be evaluated as it is unlikely to have access to networks pretrained on labeled large-scale datasets as ImageNet.

In addition, a major challenge in industry is to make the typically resource-intensive deep-learning algorithms executable on small and ideally low-cost embedded devices while fulfilling real-time requirements. Although the algorithms developed here would meet such requirements for many cases, a more efficient implementation or adaptation would make them more widely applicable. In addition to manual adaptation to specific hardware, it would also be beneficial to compress or accelerate the network itself. Possible options are knowledge distillation [67], network pruning [123] and quantization [124].

In this work, it has been assumed that there is a training set that contains only defect-free examples, which is often not a major constraint since anomalies are rare in most scenarios. However, it costs labeling effort to ensure an anomaly-free training set. Active learning could help to determine which examples should be labeled next so that the model learns more efficiently. Since it is expected to find anomalies in this labeling process, the methods could be extended to utilize anomalies directly in training. Except for a few papers [125], [126], active learning for AD is mostly underexplored.

Even though localization is already a first step towards the explainability of the model by identifying the image regions underlying the decision, this technique does not provide a fully comprehensive statement about the error type. The localization gives no information about the attribute that is anomalous in an image region. For example, the defect could represent a deformation, a discoloration, the absence of a component, or the presence of a previously unknown structure. Future work may explore how the decision could be justified in such a way. One key could be interpretable models whose features can be verbalized by associating image content with natural language [127] where a common embedding space such as in CLIP [128] is utilized.

BIBLIOGRAPHY

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. "A survey of network anomaly detection techniques." In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31.

[2] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. "A survey of anomaly detection techniques in financial domain." In: *Future Generation Computer Systems* 55 (2016), pp. 278–288.

[3] Arijit Ukil, Soma Bandyoapdhyay, Chetanya Puri, et al. "IoT healthcare analytics: The importance of anomaly detection." In: *2016 IEEE 30th international conference on advanced information networking and applications (AINA)*. IEEE. 2016, pp. 994–997.

[4] Ayan Chatterjee and Bestoun S Ahmed. "IoT anomaly detection methods and applications: A survey." In: *Internet of Things* 19 (2022), p. 100568.

[5] Hamzeh Alimohammadi and Shengnan Nancy Chen. "Performance evaluation of outlier detection techniques in production timeseries: A systematic review and meta-analysis." In: *Expert Systems with Applications* 191 (2022), p. 116371.

[6] Tamás Czimmermann, Gastone Ciuti, Mario Milazzo, et al. "Visual-based defect detection and classification approaches for industrial applications—A survey." In: *Sensors* 20.5 (2020), p. 1459.

[7] Paul Bergmann, Michael Fauser, David Sattlegger, et al. "MVTec AD–A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9592–9600.

[8] M Abd Al Rahman and Alireza Mousavi. "A review and analysis of automatic optical inspection and quality monitoring methods in electronics industry." In: *Ieee Access* 8 (2020), pp. 183192–183271.

[9] Paul Bergmann, Xin Jin, David Sattlegger, et al. "The MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization." In: *17th International Conference on Computer Vision Theory and Applications* (2022).

[10] Paul Bergmann. "Unsupervised Anomaly Detection and Localization for Visual Quality Inspection." PhD dissertation. Technische Universität München, 2022.

[11] Yibin Huang, Congying Qiu, and Kui Yuan. "Surface defect saliency of magnetic tile." In: *The Visual Computer* 36.1 (2020), pp. 85–96.

[12] 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*. 2021, pp. 1907–1916.

[13] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, et al. "Fully Convolutional Cross-Scale-Flows for Image-based Defect Detection." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 1088–1097.

[14] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, et al. "Asymmetric Student-Teacher Networks for Industrial Anomaly Detection." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 2592–2602.

[15] Jan Thies Brockmann*, Marco Rudolph*, Bodo Rosenhahn, et al. "The voraus-AD Dataset for Anomaly Detection in Robot Applications." In: *Transactions on Robotics* (Nov. 2023). ISSN: 1552-3098. DOI: `10.1109/TRO.2023.3332224`.

[16] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. "Structuring Autoencoders." In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019.

[17] Bastian Wandt, Marco Rudolph, Petrissa Zell, et al. "CanonPose: Self-Supervised Monocular 3D Human Pose Estimation in the Wild." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 13294–13304.

[18] Tom Wehrbein, Marco Rudolph, Bodo Rosenhahn, et al. "Probabilistic Monocular 3D Human Pose Estimation with Normalizing Flows." In: *Proceedings of the IEEE International Conference on Computer Vision* (2021).

[19] Thomas Norrenbrock, Marco Rudolph, and Bodo Rosenhahn. "Take 5: Interpretable Image Classification with a Handful of Features." In: *Progress and Challenges in Building Trustworthy Embodied AI*.

[20] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[22] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[23] Crispin W Gardiner et al. *Handbook of stochastic methods*. Vol. 3. springer Berlin, 1985.

[24] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[25] Guansong Pang, Chunhua Shen, Longbing Cao, et al. "Deep learning for anomaly detection: A review." In: *ACM computing surveys (CSUR)* 54.2 (2021), pp. 1–38.

[26] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, et al. "Machine learning for anomaly detection: A systematic review." In: *Ieee Access* 9 (2021), pp. 78658–78700.

[27] Jingkang Yang, Kaiyang Zhou, Yixuan Li, et al. "Generalized out-of-distribution detection: A survey." In: *arXiv preprint arXiv:2110.11334* (2021).

[28] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, et al. "A unifying review of deep and shallow anomaly detection." In: *Proceedings of the IEEE* 109.5 (2021), pp. 756–795.

[29] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, et al. "A unifying review of deep and shallow anomaly detection." In: *Proceedings of the IEEE* 109.5 (2021), pp. 756–795.

[30] Guansong Pang, Chunhua Shen, Longbing Cao, et al. "Deep learning for anomaly detection: A review." In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–38.

[31] Srikanth Thudumu, Philip Branch, Jiong Jin, et al. "A comprehensive survey of anomaly detection techniques for high dimensional big data." In: *Journal of Big Data* 7.1 (2020), pp. 1–30.

[32] Bernhard Schölkopf, Robert C Williamson, Alex Smola, et al. "Support vector method for novelty detection." In: *Advances in neural information processing systems* 12 (1999).

[33] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." In: *2008 eighth ieee international conference on data mining*. IEEE. 2008, pp. 413–422.

[34] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, et al. "LOF: identifying density-based local outliers." In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[35] Tiago Nazare, Rodrigo de Mello, and Moacir Ponti. "Are pretrained CNNs good feature extractors for anomaly detection in surveillance videos?" In: *arXiv preprint arXiv:1811.08495* (2018).

[36] Karsten Roth, Latha Pemula, Joaquin Zepeda, et al. "Towards total recall in industrial anomaly detection." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14318–14328.

[37] 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*. 2017, pp. 665–674.

[38] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, et al. "Deep autoencoding models for unsupervised anomaly segmentation in brain MR images." In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4*. Springer. 2019, pp. 161–169.

[39] Zhou Wang, Alan C Bovik, Hamid R Sheikh, et al. "Image quality assessment: from error visibility to structural similarity." In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[40] Dong Gong, Lingqiao Liu, Vuong Le, et al. "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1705–1714.

[41] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. "GANomaly: Semi-supervised Anomaly Detection via Adversarial Training." In: *Computer Vision – ACCV 2018*. Cham: Springer International Publishing, 2019, pp. 622–637.

[42] Ye Fei, Chaoqin Huang, Cao Jinkun, et al. "Attribute restoration framework for anomaly detection." In: *IEEE Transactions on Multimedia* (2020).

[43] Paul Bergmann, Michael Fauser, David Sattlegger, et al. "Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4183–4192.

[44] Paul Bergmann, Kilian Batzner, Michael Fauser, et al. "Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization." In: *Int. J. Comput. Vis.* 130.4 (2022), pp. 947–969. DOI: `10.1007/s11263-022-01578-9`. URL: `https://doi.org/10.1007/s11263-022-01578-9`.

[45] Qinfeng Xiao, Jing Wang, Youfang Lin, et al. "Unsupervised anomaly detection with distillated teacher-student network ensemble." In: *Entropy* 23.2 (2021), p. 201.

[46] Guodong Wang, Shumin Han, Errui Ding, et al. "Student-Teacher Feature Pyramid Matching for Anomaly Detection." In: *arXiv preprint arXiv:2103.04257* (2021).

[47] Oliver Rippel, Patrick Mertens, and Dorit Merhof. "Modeling the distribution of normal data in pre-trained deep features for anomaly detection." In: *arXiv preprint arXiv:2005.14140* (2020).

[48] Thomas Defard, Aleksandr Setkov, Angelique Loesch, et al. "PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization." In: *pattern Recognition, ICPR International Workshops and Challenges*. 2021.

[49] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, et al. "Cutpaste: Self-supervised learning for anomaly detection and localization." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9664–9674.

[50] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. "Draem-a discriminatively trained reconstruction embedding for surface anomaly detection." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8330–8339.

[51] Hannah M Schlüter, Jeremy Tan, Benjamin Hou, et al. "Self-supervised out-of-distribution detection and localization with natural synthetic anomalies (nsa)." In: *arXiv preprint arXiv:2109.15222* (2021).

[52] Jouwon Song, Kyeongbo Kong, Ye-In Park, et al. "Anoseg: Anomaly segmentation network using self-supervised learning." In: *arXiv preprint arXiv:2110.03396* (2021).

[53] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers." In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[54] Jerone Andrews, Thomas Tanay, Edward Morton, et al. "Transfer representation-learning for anomaly detection." In: *NeurIPS*. 2019.

[55] J. Ross Quinlan. "Induction of decision trees." In: *Machine learning* 1 (1986), pp. 81–106.

[56] Thomas Cover and Peter Hart. "Nearest neighbor pattern classification." In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[57] Mennatallah Amer and Markus Goldstein. "Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer." In: *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)*. 2012, pp. 1–12.

[58] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space." In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572.

[59] Ozan Sener and Silvio Savarese. "Active Learning for Convolutional Neural Networks: A Core-Set Approach." In: *International Conference on Learning Representations*. 2018. URL: `https://openreview.net/forum?id=H1aIuk-RW`.

[60] Samarth Sinha, Han Zhang, Anirudh Goyal, et al. "Small-GAN: Speeding up GAN Training using Core-Sets." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 9005–9015.

[61] Y. LeCun. "Generalization and Network Design Strategies." In: *Connectionism in Perspective*. Ed. by R. Pfeifer, Z. Schreter, F. Fogelman, et al. an extended version was published as a technical report of the University of Toronto. Zurich, Switzerland: Elsevier, 1989.

[62] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[63] Paul Bergmann, Sindy Löwe, Michael Fauser, et al. "Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders." In: *VISIGRAPP*. 2019.

[64] Shuangfei Zhai, Yu Cheng, Weining Lu, et al. "Deep structured energy based models for anomaly detection." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. 2016, pp. 1100–1109.

[65] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, et al. "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks." In: *Medical image analysis* 54 (2019), pp. 30–44.

[66] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows." In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1530–1538.

[67] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. "Distilling the knowledge in a neural network." In: *arXiv preprint arXiv:1503.02531* 2.7 (2015).

[68] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, et al. "Improved knowledge distillation via teacher assistant." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5191–5198.

[69] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive representation distillation." In: *arXiv preprint arXiv:1910.10699* (2019).

[70] Prasanta Chandra Mahalanobis. "On the generalised distance in statistics." In: *Proceedings of the national Institute of Science of India*. Vol. 12. 1936, pp. 49–55.

[71] Tingting Chen, Xueping Liu, Bizhong Xia, et al. "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder." In: *IEEE Access* 8 (2020), pp. 47072–47081.

[72] Maximilian Sölch, Justin Bayer, Marvin Ludersdorfer, et al. "Variational inference for on-line anomaly detection in high-dimensional time series." In: *arXiv preprint arXiv:1602.07109* (2016).

[73] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *CoRR* abs/1312.6114 (2013).

[74] Larry R Medsker and LC Jain. "Recurrent neural networks." In: *Design and Applications* 5 (2001), pp. 64–67.

[75] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder." In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1544–1551.

[76] Davide Azzalini, Luca Bonali, and Francesco Amigoni. "A minimally supervised approach based on variational autoencoders for anomaly detection in autonomous robots." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2985–2992.

[77] Rachel Hornung, Holger Urbanek, Julian Klodmann, et al. “Model-free robot anomaly detection.” In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 3676–3683.

[78] Eliahu Khalastchi, Meir Kalech, Gal A Kaminka, et al. “Online data-driven anomaly detection in autonomous robots.” In: *Knowledge and Information Systems* 43.3 (2015), pp. 657–688.

[79] Diego Romeres, Devesh K Jha, William Yerazunis, et al. “Anomaly detection for insertion tasks in robotic assembly using Gaussian process models.” In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 1017–1022.

[80] Tie Zhang, Peizhong Ge, Yanbiao Zou, et al. “Robot collision detection without external sensors based on time-series analysis.” In: *Journal of Dynamic Systems, Measurement, and Control* 143.4 (2021).

[81] Daehyung Park, Zackory Erickson, Tapomayukh Bhattacharjee, et al. “Multimodal execution monitoring for anomaly detection during robot manipulation.” In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 407–414.

[82] Leonard E Baum and Ted Petrie. “Statistical inference for probabilistic functions of finite state Markov chains.” In: *The annals of mathematical statistics* 37.6 (1966), pp. 1554–1563.

[83] Daehyung Park, Hokeun Kim, and Charles C Kemp. “Multimodal anomaly detection for assistive robots.” In: *Autonomous Robots* 43 (2019), pp. 611–629.

[84] Davide Azzalini, Alberto Castellini, Matteo Luperto, et al. “Hmms for anomaly detection in autonomous robots.” In: *AAMAS CONFERENCE PROCEEDINGS*. ACM. 2020, pp. 105–113.

[85] Ernst Hellinger. “Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen.” In: *Journal für die reine und angewandte Mathematik* 1909.136 (1909), pp. 210–271.

[86] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks.” In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[87] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. “Why normalizing flows fail to detect out-of-distribution data.” In: *NeurIPS*. 2020.

[88] Charles K Chui, Shao-Bo Lin, and Ding-Xuan Zhou. "Deep neural networks for rotation-invariance approximation and learning." In: *Analysis and Applications* 17.05 (2019), pp. 737–772.

[89] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." In: *ICLR 2017* (2016).

[90] Mathieu Germain, Karol Gregor, Iain Murray, et al. "Made: Masked autoencoder for distribution estimation." In: *International Conference on Machine Learning*. 2015, pp. 881–889.

[91] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, et al. "Improved variational inference with inverse autoregressive flow." In: *Advances in neural information processing systems*. 2016, pp. 4743–4751.

[92] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, et al. "Analyzing inverse problems with invertible neural networks." In: *ICLR*. 2019.

[93] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, et al. "Guided image generation with conditional invertible neural networks." In: *arXiv preprint arXiv:1907.02392* (2019).

[94] Artem Ryzhikov, Maxim Borisyak, Andrey Ustyuzhanin, et al. "Normalizing flows for deep anomaly detection." In: *arXiv preprint arXiv:1912.09323* (2019).

[95] Maximilian Schmidt and Marko Simic. "Normalizing flows for novelty detection in industrial time series data." In: *arXiv preprint arXiv:1906.06904* (2019).

[96] Madson LD Dias, César Lincoln C Mattos, Ticiana LC da Silva, et al. "Anomaly Detection in Trajectory Data with Normalizing Flows." In: *arXiv preprint arXiv:2004.05958* (2020).

[97] Izhak Golan and Ran El-Yaniv. "Deep anomaly detection using geometric transformations." In: *Advances in Neural Information Processing Systems*. 2018, pp. 9758–9769.

[98] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[100] K Simonyan and A Zisserman. "Very deep convolutional networks for large-scale image recognition." In: (2015).

[101] Jia Deng, Wei Dong, Richard Socher, et al. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[102] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *International Conference on Learning Representations (ICLR)*. 2015.

[103] Haoqing Cheng, Heng Liu, Fei Gao, et al. "ADGAN: A Scalable GAN-based Architecture for Image Anomaly Detection." In: *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. Vol. 1. IEEE. 2020, pp. 987–993.

[104] Ke Sun, Bin Xiao, Dong Liu, et al. "Deep high-resolution representation learning for human pose estimation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5693–5703.

[105] Mariana-Iuliana Georgescu, Antonio Barbalau, Radu Tudor Ionescu, et al. "Anomaly detection in video via self-supervised and multi-task learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12742–12752.

[106] 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*. 2022, pp. 98–107.

[107] Kai Zhang, Wangmeng Zuo, and Lei Zhang. "FFDNet: Toward a fast and flexible solution for CNN-based image denoising." In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4608–4622.

[108] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017).

[109] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[110] Eliahu Horwitz and Yedid Hoshen. "An Empirical Investigation of 3D Anomaly Detection and Segmentation." In: *arXiv preprint arXiv:2203.05550* (2022).

[111] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. “Fast point feature histograms (FPFH) for 3D registration.” In: *2009 IEEE international conference on robotics and automation*. IEEE. 2009, pp. 3212–3217.

[112] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[113] Mohd Aiman Kamarul Bahrin, Mohd Fauzi Othman, Nor Hayati Nor Azli, et al. “Industry 4.0: A review on industrial automation and robotic.” In: *Jurnal teknologi* 78.6-13 (2016).

[114] Milos Vasic and Aude Billard. “Safety issues in human-robot interactions.” In: *2013 ieee international conference on robotics and automation*. IEEE. 2013, pp. 197–204.

[115] Danica Kragic, Joakim Gustafson, Hakan Karaoguz, et al. “Interactive, Collaborative Robots: Challenges and Opportunities.” In: *IJCAI*. 2018, pp. 18–25.

[116] Błażej Leporowski, Daniella Tola, Casper Hansen, et al. “Detecting Faults during Automatic Screwdriving: A Dataset and Use Case of Anomaly Detection for Automatic Screwdriving.” In: *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems*. Springer, 2021, pp. 224–232.

[117] Enyan Dai and Jie Chen. “Graph-Augmented Normalizing Flows for Anomaly Detection of Multiple Time Series.” In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: `https://openreview.net/forum?id=45L%5C_dgP48Vd`.

[118] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, et al. “Autoencoder-based network anomaly detection.” In: *2018 Wireless telecommunications symposium (WTS)*. IEEE. 2018, pp. 1–5.

[119] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, et al. “A dataset to support research in the design of secure water treatment systems.” In: *Critical Information Infrastructures Security: 11th International Conference, CRITIS 2016, Paris, France, October 10–12, 2016, Revised Selected Papers 11*. Springer. 2017, pp. 88–99.

[120] Kaiming He, Haoqi Fan, Yuxin Wu, et al. “Momentum contrast for unsupervised visual representation learning.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738.

[121] Ting Chen, Simon Kornblith, Mohammad Norouzi, et al. "A simple framework for contrastive learning of visual representations." In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[122] Mathilde Caron, Ishan Misra, Julien Mairal, et al. "Unsupervised learning of visual features by contrasting cluster assignments." In: *Advances in neural information processing systems* 33 (2020), pp. 9912–9924.

[123] Pavlo Molchanov, Stephen Tyree, Tero Karras, et al. "Pruning Convolutional Neural Networks for Resource Efficient Inference." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: `https://openreview.net/forum?id=SJGCiw5gl`.

[124] Tailin Liang, John Glossner, Lei Wang, et al. "Pruning and quantization for deep neural network acceleration: A survey." In: *Neurocomputing* 461 (2021), pp. 370–403.

[125] Tiago Pimentel, Marianne Monteiro, Juliano Viana, et al. "A generalized active learning approach for unsupervised anomaly detection." In: *stat* 1050 (2018), p. 23.

[126] Tiago Pimentel, Marianne Monteiro, Adriano Veloso, et al. "Deep active learning for anomaly detection." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[127] Mert Yüksekgönül, Maggie Wang, and James Zou. "Post-hoc Concept Bottleneck Models." In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: `https://openreview.net/pdf?id=nA5AZ8CEyow`.

[128] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. "Learning transferable visual models from natural language supervision." In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

Marco Rudolph

Personal Data

Year of Birth: 1996
Place of Birth: Hanover, Germany
email: rudolph@tnt.uni-hannover.de

Work Experience

2018 - *present*	research associate at the **Leibniz University Hannover**, **Institut für Informationsverarbeitung**, Main Focus: Image-Based Anomaly Detection, Thesis: "Industrial Anomaly Detection with Normalizing Flows" Advisor: Prof. Dr.-Ing. Bodo Rosenhahn
2017	Teaching Assistant at the **Leibniz University Hannover**, **Institut für Informationsverarbeitung**, Courses: Technical Project II: Tracking with Matlab
2014-2015	Teaching Assistant at the **Leibniz University Hannover**, **Institut für Systems Engineering**, Courses: Modeling the Dynamic Behavior of Systems

Education

Oct 2018	M. Sc. in Computer Science at the **Leibniz University Hannover**, **Institut für Informationsverarbeitung**, Thesis: "Coding of Human Surface Models Using Neural Networks" Advisor: Prof. Dr.-Ing. Bodo Rosenhahn
Sep 2016	B. Sc. in Computer Science at the **Leibniz University Hannover**, **Institut für Systems Engineering**, Thesis: "Indoor Positioning using Machine Learning" Advisor: Dr.-Ing. habil. Matthias Becker
Jun 2013	Abitur at the **Matthias-Claudius-Gymnasium Gehrden**