TUM School of Computation,
Information and Technology
Technische Universität München

Unsupervised Anomaly Detection and Localization for Visual Quality Inspection

Paul Bergmann

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 24.06.2022 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 19.09.2022 angenommen.

Abstract

Anomaly detection refers to the identification of data samples that deviate from a concept of normality. The reliable detection of anomalies is of great importance in many application areas, such as medical imaging, autonomous driving, or the automated detection of defects in industrial manufacturing processes. In this thesis, we focus on the latter. The early identification of product errors can reduce costs, improve quality, and prevent safety risks.

We develop computer vision methods for the robust detection and precise localization of anomalous structures. The presented algorithms are based on unsupervised deep learning techniques that are trained exclusively on anomaly-free samples. Due to the lack of suitable benchmark datasets, we additionally introduce several new datasets and performance metrics that facilitate a thorough evaluation of anomaly detection models.

We begin with an investigation of autoencoders that compute anomaly scores by per-pixel comparisons between an input and a reconstructed image. We show that there are cases in which they perform poorly even if the reconstruction is only slightly inaccurate. We then develop a new autoencoder variant that instead employs a perceptual error function based on the structural similarity index. In our experiments, it significantly improves over autoencoders that rely on pixelwise residuals. Furthermore, we identify a need for larger anomaly detection datasets.

We then present MVTec AD, a comprehensive real-world dataset for unsupervised anomaly detection in industrial inspection scenarios. We conduct an extensive benchmark of prior art in anomaly detection on this dataset and find that there is considerable room for improvement. Next, we develop a new anomaly detection framework based on Student–Teacher learning that leverages descriptors obtained from pretrained neural networks for anomaly detection. Our experiments show that it substantially improves over existing methods.

Subsequently, we focus on the detection of anomalies that violate logical constraints of the anomaly-free training data, such as missing object parts or permissible objects occurring in invalid locations. We introduce the MVTec LOCO AD dataset, which reveals that existing methods do not perform well in the detection of such logical anomalies. Therefore, we extend our Student–Teacher framework by a global network branch that captures the underlying logical constraints of the anomaly-free training data. Our new method sets a new state of the art in the detection of logical anomalies.

Finally, we tackle the problem of geometric anomaly detection in three-dimensional data. We introduce MVTec 3D-AD, the first comprehensive dataset for this task. We then adapt our Student–Teacher framework to three dimensions and present 3D-ST, an unsupervised method for 3D anomaly detection in point cloud data. It performs significantly better than existing 3D anomaly detection approaches.

Zusammenfassung

Anomalieerkennung bezieht sich auf die Identifizierung von Daten, die von einem Normalzustand abweichen. Die zuverlässige Erkennung von Anomalien ist in vielen Anwendungsbereichen von großer Bedeutung, wie zum Beispiel in der medizinischen Bildgebung, beim autonomen Fahren, oder bei der automatischen Erkennung von Defekten in industriellen Fertigungsprozessen. In dieser Arbeit konzentrieren wir uns auf Letzteres. Die frühzeitige Erkennung von Produktfehlern kann Kosten senken, die Produktqualität verbessern und Sicherheitsrisiken vorbeugen.

Wir entwickeln Bildverarbeitungsmethoden zur robusten Erkennung und präzisen Lokalisierung von anomalen Strukturen. Die vorgestellten Algorithmen basieren auf unüberwachten Deep-Learning-Verfahren, die ausschließlich auf anomaliefreien Daten trainiert werden. Aufgrund des Mangels an geeigneten Benchmark-Datensätzen stellen wir zusätzlich mehrere neue Datensätze und Evaluationsmetriken vor, die eine umfassende Auswertung von Modellen zur Anomalieerkennung ermöglichen.

Wir beginnen mit einer Untersuchung von Autoencodern, die Anomaliewerte durch pixelweise Vergleiche zwischen einem Eingabebild und einem rekonstruierten Bild berechnen. Wir zeigen, dass bereits geringfügige Ungenauigkeiten in der Rekonstruktion zu schlechten Erkennungsraten führen können. Daraufhin entwickeln wir eine neue Art von Autoencodern, die Anomaliewerte basierend auf einem visuellen Ähnlichkeitsmaß, dem Structural Similarity Index, berechnet. Unsere Experimente zeigen eine deutliche Verbesserung gegenüber Autoencodern, die auf pixelweisen Fehlerfunktionen basieren. Außerdem stellen wir einen Bedarf an größeren Forschungsdatensätzen für die Erkennung von Anomalien fest.

Deshalb stellen wir MVTec AD vor, einen umfangreichen Datensatz für die Erkennung von Anomalien in industriellen Inspektionsszenarien. Wir führen einen Vergleich bestehender Methoden zur Anomalieerkennung auf diesem Datensatz durch und stellen erhebliches Verbesserungspotential fest. Anschließend entwickeln wir ein neues Verfahren basierend auf einem Student–Teacher Ansatz. Dieses verwendet Deskriptoren, die aus vortrainierten neuronalen Netzwerken extrahiert werden. Unsere Experimente zeigen signifikante Verbesserungen in der Erkennungsleistung im Vergleich zu bestehenden Methoden.

Anschließend konzentrieren wir uns auf die Erkennung von Anomalien, die bestimmte logische Bedingungen der anomaliefreien Trainingsdaten verletzen, zum Beispiel fehlende Objektteile oder das Auftreten anomaliefreier Objekte an ungültigen Stellen. Wir stellen den MVTec LOCO AD Datensatz vor, der deutlich macht, dass bestehende Methoden oft Probleme bei der Erkennung derartiger Anomalien haben. Daher erweitern wir unsere Student–Teacher Methode um einen globalen Netzwerkzweig, welcher die logischen Zusammenhänge der anomaliefreien Trainingsdaten erfasst. Unsere neue Methode zeigt deutliche Verbesserungen hinsichtlich der Erkennung von logischen Anomalien gegenüber bestehenden Verfahren.

Abschließend befassen wir uns mit der Erkennung geometrischer Anomalien in dreidimensionalen Daten. Wir stellen MVTec 3D-AD vor, den ersten umfassenden Datensatz für diese Anwendung. Anschließend adaptieren wir unsere Student–Teacher Methode auf dreidimensionale Daten und präsentieren 3D–ST, eine unüberwachte Methode zur Erkennung von geometrischen Anomalien in 3D Punktwolken. Unsere Methode erreicht Erkennungsraten, die deutlich über denen bestehender Methoden liegen.

Acknowledgements

During the last few years, I had the great privilege to work with many outstanding people to whom I am deeply indebted. First and foremost, I would like to thank my academic advisor Prof. Carsten Steger, for giving me the opportunity to write this thesis and his continuous support and guidance. I would further like to thank Prof. Rosenhahn and Prof. Günnemann for reviewing this dissertation and completing my examination committee.

Furthermore, I would like to thank all former and current members of the MVTec research department, namely Kilian Batzner, Dr. Tobias Böttger, Dr. Bertram Drost, Prof. Carsten Steger, Michael Fauser, Dr. Patrick Follmann, Dr. Markus Glitzner, Lars Heckler, Philipp Härtinger, Rebecca König, Jan-Hendrik Neudeck, Dr. David Sattlegger, and Prof. Markus Ulrich. Thank you for inspiring discussions and for creating a stimulating work environment. I couldn't have wished for better colleagues.

I would like to give special thanks to the entire *Anomaly Detection Team* at MVTec. Thank you Kilian – I still don't know how you managed to turn countless hours in the lab acquiring dataset images into a fun time. Thank you Micha, for always keeping a clear head, for your attention to detail, and the best code reviews I have ever received. David, for our paper writing sessions and for being an amazing mentor. And Carsten, for sharing your immense machine vision knowledge with me. Thank you all for accompanying me on my anomaly detection journey from start to finish.

I would also like to thank my students. Sindy, Maximilian, Peter, Martin, and Xin, it was a pleasure working with all of you and I have learned at least as much from you as I hope you have learned from me.

Finally, I would like to thank my parents, on whose shoulders I stand. My sisters, Lisa and Meike, for always being there for me. My friends, for being awesome. And Shuying, for your unconditional love and support and for always being by my side. This would not have been possible without all of you.

Contents

1 Introduction

Anomalies are patterns in data that differ from a concept of normality. Finding anomalies is important, as their existence may indicate a faulty system or the occurence of an unexpected external event that requires attention. In research, the anomaly detection problem is studied from various viewpoints and application areas, such as network intrusion detection [Khraisat et al., 2019], credit card fraud detection [Shirodkar et al., 2020], finding abnormalities in healthcare diagnosis records [Fernando et al., 2021], or predictive maintanence of mechanical systems [Theissler et al., 2021]. In science in general, the detection of anomalous events that contradict existing models may lead to new insights and advances in human knowledge itself [Mehrotra et al., 2017].

Another area of research where anomaly detection is highly relevant is computer vision. Here, anomalies occur as deviations from normal data acquired from visual sensors, such as cameras. There are many real-world computer vision problems that benefit from reliable anomaly detection systems. In medical imaging, for example, anomalies may manifest themselves as diseases in brain scans that should be brought to the attention of a medical expert. In autonomous driving, anomalies can take the form of obstacles on the road that need to be detected by an on-board camera system to ensure road traffic safety. A third example are automated visual inspection scenarios, where anomalies can occur in the form of defects in manufactured products in a production line. Figure 1.1 shows example images from computer vision datasets in each of these three areas, namely the MVTec Anomaly Detection dataset [Bergmann et al., 2019a, 2021] for industrial inspection, the Fishyscapes dataset [Blum et al., 2019] for autonomous driving, and the BRATS dataset [Menze et al., 2015] for medical imaging. The MVTec Anomaly Detection dataset is described in detail in Chapter 4. In this thesis, we study the anomaly detection problem from a computer vision perspective.

1.1 Deep Learning for Anomaly Detection

One approach to identify anomalies in an image is to manually specify criteria for normality. In a production line, for example, one could implement an algorithm that measures the diameter of a manufactured object in an image and specify a tolerance interval in which the diameter of the object may vary. When a production error occurs and an object with a diameter that lies outside the specified range is produced, it is marked as an anomaly. While this approach may work for certain problems, it has several drawbacks. First, it requires significant domain knowledge of the investigated problem. Designing features that can reliably discriminate between anomaly-free and anomalous data points is a complex task. Second, this approach does not generalize well. When a different object needs to be inspected, this might require a system designer to come up with an

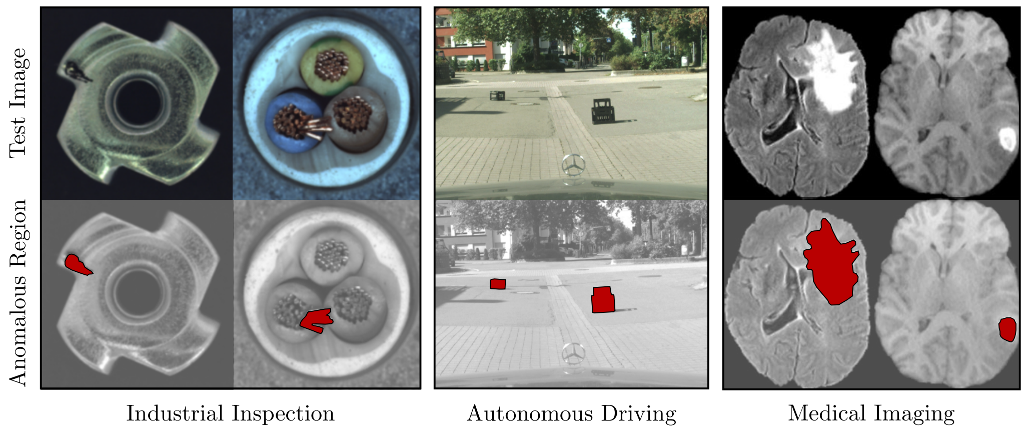


Figure 1.1: Examples for application areas of anomaly detection in computer vision. The first row shows example images of anomaly detection datasets from different domains. The second row highlights anomalous regions in red. Images are taken from the MVTec Anomaly Detection dataset [Bergmann et al., 2021] for industrial inspection, the Fishyscapes dataset [Blum et al., 2019] for autonomous driving, and the BRATS dataset [Menze et al., 2015] for medical imaging.

entirely new set of features. Third, even if one can specify and measure criteria for normality, a simple comparison to a specified range of allowed feature values might be insufficient and more complex approaches are needed. Finally, such manual approaches often require an a priori definiton of all possible anomalies. Unexpected anomalies that were not explicitly programmed may not be found.

To circumvent these problems, anomaly detection can be tackled in a data-driven way using machine learning. Instead of having to manually design criteria that distinguish between anomalous and anomaly-free samples, these models can ideally learn them from a representative set of images. Examples for frequently used models include Support Vector Machines (SVM) [Schölkopf et al., 2001, Li et al., 2003], K-Nearest Neighbor Classifiers (KNN) [Knorr et al., 2000, Ying et al., 2021], or Decision Trees [Reif et al., 2008, Buschjager et al., 2022]. While they often work well on low-dimensional data, their performance degrades when processing high-dimensional inputs such as natural images. Again, one solution to this problem is to train these approaches on handcrafted lower-dimensional representations extracted from the high-dimensional inputs, which comes with the same drawbacks that were discussed above.

More recently, the field of deep learning has brought forward powerful techniques for solving computer vision problems. In image classification [He et al., 2016, Tan and Le, 2019], semantic segmentation [Ronneberger et al., 2015, Jégou et al., 2017], or object detection [Lin et al., 2017, Ren et al., 2017], for instance, deep learning models have demonstrated impressive performance on high-dimensional data. They employ multi-layer neural networks whose parameters are adjusted to minimize a target loss function by performing backpropagation and gradient descent optimization. Instead of relying on handcrafted descriptors, these models learn task-specific features directly from the

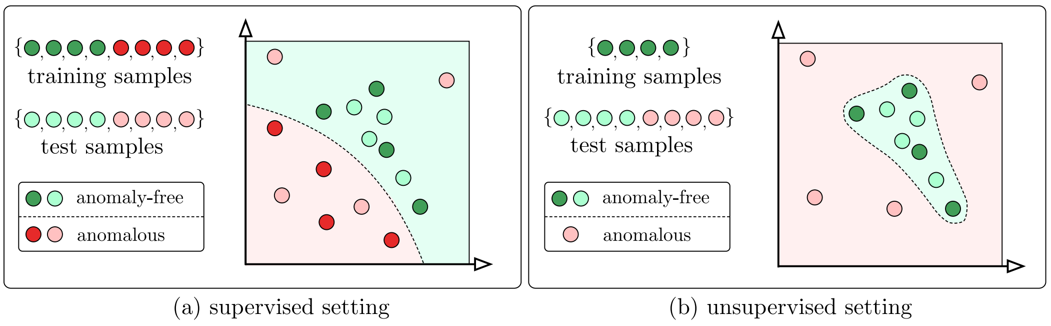


Figure 1.2: Illustration of the anomaly detection problem from both a supervised and an unsupervised perspective. In the supervised setting, anomaly-free as well as anomalous data is required for model training. In the unsupervised case, models are trained only on anomaly-free data.

high-dimensional input data. This also makes deep learning an attractive approach for anomaly detection. Instead of having to explicitly design descriptors and criteria that indicate normality, deep learning provides means to learn both directly from data.

1.2 Supervised and Unsupervised Methods

In machine learning, a fundamental distinction is made between *supervised* and *unsupervised* methods. The former make use of annotations that provide additional information about the dataset samples, such as class labels. Given a set of anomaly-free as well as anomalous data points, a supervised approach could fit a discriminative model that can separate the two classes. Supervised methods rely on the availability of anomalous training samples as well as accurate annotations. In particular, the training samples ideally capture the full range of conceivable anomalies. Unfortunately, achieving this tends to be difficult since the nature of anomalies that may occur in practice is often unknown. Furthermore, some anomalies may occur so rarely that it is practically impossible to collect any training samples for them. Another reason is that obtaining precise annotations can be a tedious and costly task.

A schematic illustration of the supervised setting of the anomaly detection problem is given in Figure 1.2(a). A set of training data points that contains both anomalous and anomaly-free data is used to create a supervised classifier that produces a decision boundary that separates the two classes. During inference, the model attempts to classify unseen test samples. While in this example some of them are correctly classified, its decision boundary is not discriminative enough to separate all anomalies from the anomaly-free test data. This is because some of the anomalous test samples significantly differ from the examples present in the training set.

Alternatively, anomaly detection can be addressed in an unsupervised setting. Unsupervised models are trained exclusively on the provided data without the need for additional annotations. In particular, we want to build anomaly detection models that do not rely on the availability of anomalous training samples. They are trained only on a dataset of anomaly-free data points and can detect anomalous inputs during inference. Unsupervised models for anomaly detection ideally create a tight decision boundary around the anomaly-free training samples, implicitly labeling everything else as anomalous. This is illustrated in Figure 1.2(b). An unsupervised model was fitted to the anomaly-free samples of the same dataset used in the supervised case. In contast to the supervised setting, the unsupervised model now handles all anomalous test inputs correctly. This is because it creates a decision boundary that surrounds the set of anomaly-free samples in the feature space. While the creation of such a decision boundary seems straightforward in this two-dimensional example, it becomes increasingly challenging when the dimensionality of the input data grows and the pairwise distances between all samples converge to the same value [Kriegel et al., 2009, Xia et al., 2015]. This effect is particularly problematic for computer vision datasets, where input samples often contain thousands of input dimensions.

1.3 Different Characteristics of Anomalies

Depending on the application under consideration, the types of anomalies that are of interest can differ significantly. For instance, one might want to create a model that treats images of a certain semantic class as the concept of normality. An example of this is illustrated in Figure 1.3(a). Here, image samples that show a picture of a dog are treated as anomaly-free. When images of a different object class occur, for example, an image of a cat or a tennis ball, these should be identified as anomalous inputs. Such problems tend to exhibit a large intra-class variance. The semantic concept of a dog, for example, can be present in images that are visually very different from each other. Hence, anomalies also need to significantly deviate from the training data manifold such that they can be realiably recognized. Furthermore, it is common that anomalies make up almost the entire image. Therefore, one is often less interested in the localization of the anomaly within the image, but rather in the separation between anomalous and anomaly-free samples.

In other applications, anomalies may occur only as subtle deviations from the anomaly-free training data. In these cases, the intra-class variance within the training set is often limited such that a distinction between anomalous and anomaly-free samples is still possible. An example from the industrial inspection domain is shown in Figure 1.3(b), where images of a hazelnut are inspected. Compared to images of dogs observed in the wild, the hazelnuts do not differ too much from each other. However, in this case, anomalies do not occur as entirely different object classes, but in the form of subtle defects, such as holes or scratches on the surface of the nuts. The defects occur in small, confined regions within the input image, while the rest of the image is considered as anomaly-free. In these applications, it is often desirable to precisely localize the anomalous region within the input image, in addition to making an image-level decision as to whether an anomaly is present or not. The localization provides a visual explanation for why a dataset sample is considered anomalous, which can increase the confidence in

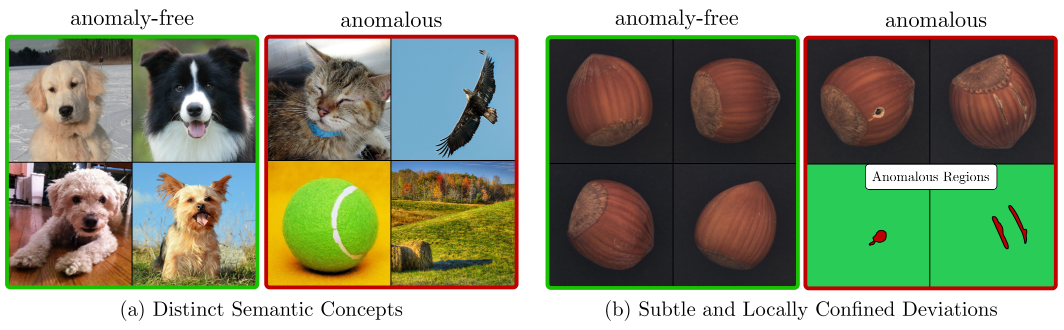

anomaly-free
anomalous
anomaly-free
anomalous
Anomalous Regions
(a) Distinct Semantic Concepts
(b) Subtle and Locally Confined Deviations

Figure 1.3: The characteristics of anomalies that are of interest may vary depending on the application under consideration. For example, anomalies can occur as entirely new semantic concepts (a). They may also appear as locally confined regions that differ only slightly from the anomaly-free training data (b). The latter allows for the pixel-precise localization of the anomalous regions. Images are taken from the ImageNet dataset [Krizhevsky et al., 2012] (left) and the MVTec AD dataset [Bergmann et al., 2021] (right).

the algorithm being used and may be helpful to determine the original cause of the deviation. Furthermore, the localization result may be used to compute features for additional processing steps, e.g., by measuring the geometric properties of a defect.

1.4 Research Question

In this thesis, we study the anomaly detection problem in the unsupervised setting. This is motivated by the fact that in many real-world applications anomalous samples are not available for training. In the past, efforts were made to develop new methods and datasets for this task in one-class or multi-class classification scenarios [An and Cho, 2015, Andrews et al., 2016, Chalapathy et al., 2018, Masana et al., 2018, Roitberg et al., 2019, Ruff et al., 2018, Burlina et al., 2019]. Here, it is assumed that anomalies manifest themselves in the form of images of an entirely different class and a binary image-level decision whether an image is anomalous or not is sufficient. Curiously, we find that little work has been directed towards the development of methods that can localize anomalous regions that only differ in a subtle way from the anomaly-free training data. However, such methods are of great importance in many practical applications, for instance, in industrial inspection. In this thesis, we make progress in this area. In particular, the underlying research question of this thesis is:

Can we create new computer vision models for unsupervised anomaly detection that accurately localize anomalous regions and improve significantly upon existing approaches?

To study this question, we concentrate on industrial inspection scenarios as they constitute an ideal use-case for the field of unsupervised anomaly detection. This is because:

- It is difficult to cover the entirety of possible anomalies with labeled examples, since defects in manufactured products can manifest themselves in very different ways. Furthermore, anomalous training data is often unavailable due to the low error rate of modern production lines.
- The acquisition and annotation of anomalous samples to train supervised anomaly detection models is expensive and unsupervised models can provide a less cost intensive alternative.
- Anomalies often occur in small, confined regions and a precise localization of the defects within the image is desirable.

To cope with the high dimensionality of data obtained from modern computer vision sensors and to circumvent the need for manual feature engineering, we leverage models based on recent advances in deep learning.

1.5 Outline, Contributions, and Publications

The material presented in this thesis is the result of a series of research projects conducted over the past years. Most of these resulted in publications in peer-reviewed scientific conferences or journals and several chapters contain material from these published articles. The following provides a brief summary of the content of each chapter. The respective publications that the individual chapters are based on are also listed.

Chapter 2: Foundations. This chapter briefly summarizes the theoretical prerequisites on which the subsequent parts of this thesis are based. In particular, it formally defines the unsupervised anomaly detection problem and introduces popular base architectures for building anomaly detection systems, such as convolutional autoencoders or generative adversarial networks. We further give a brief introduction to frequently used terminology in deep learning.

Chapter 3: Structural Similarity Autoencoder. Convolutional autoencoders are a popular base architecture for anomaly detection models. Existing approaches typically compare an input image to a reconstructed image by pixelwise error measures. As a result, they are sensitive to slight inaccuracies in the reconstruction. In this chapter, we improve the performance of these methods by using a perceptual loss function based on the structural similarity index (SSIM). In contrast to pixelwise measures, our SSIM-Autoencoder examines inter-dependencies between local image regions.

> Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, Carsten Steger: **Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders;** in: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), Volume 5: VISAPP*, 372-380, 2019, DOI: 10.5220/0007364503720380.

Chapter 4: The MVTec Anomaly Detection Dataset. Research in the field of anomaly detection and localization requires suitable data to create and evaluate models. Unfortunately, no comprehensive dataset for this task existed prior to this research project. To fill this gap, we created the MVTec Anomaly Detection dataset (MVTec AD). It is inspired by industrial manufacturing scenarios and contains over 5000 high-resolution images of 15 object categories. For each category, it provides anomaly-free images for model training as well as test images that contain various types of anomalies, such as scratches or dents on manufactured products. Additionally, we conducted a benchmark of existing methods for unsupervised anomaly detection. Our benchmark indicates a large room for improvement.

> Paul Bergmann, Michael Fauser, David Sattlegger, Carsten Steger: **MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection;** in: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9584-9592, 2019, DOI: 10.1109/CVPR.2019.00982.

> Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, Carsten Steger: **The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection;** in: *International Journal of Computer Vision* 129(4):1038-1059, 2021, DOI: 10.1007/s11263-020-01400-4.

Chapter 5: Student–Teacher Anomaly Detection. Modeling the distribution of descriptors of pretrained networks has proven to be a powerful technique for anomaly detection. However, many existing methods require patch-based evaluations that result in very slow inference times. In addition, they use shallow machine learning models that may prevent them to make use of all available training data, which affects their performance. In this chapter, we introduce a new method for anomaly detection that uses features of pretrained networks. Specifically, we develop a Student–Teacher framework that makes use of all available training features and efficiently computes anomaly scores with a single forward pass. At the time of publication, our method set a new state of the art on the MVTec Anomaly Detection dataset.

> Paul Bergmann, Michael Fauser, David Sattlegger, Carsten Steger: **Uninformed Students: Student–Teacher Anomaly Detection with Discriminative Latent Embeddings;** in: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4183-4192, 2020, DOI: 10.1109/CVPR42600.2020.00424.

Chapter 6: Logical Constraints in Anomaly Detection. Anomalies can take very different forms and a dataset should ideally contain representative examples for all of them. We found that existing datasets, including MVTec AD, focus predominantly on local structural anomalies, such as scratches or dents in manufactured products. They lack anomalies that violate logical constraints of the training data, e.g., permissible objects being present in invalid locations. In this chapter, we present MVTec LOCO, a new dataset that captures both types of anomalies. A benchmark of existing methods reveals considerable room for improvement in the detection of logical anomalies.

Chapter 7: Global Context Anomaly Detection. Building on our findings in Chapter 6, we introduce a new method for unsupervised anomaly detection that significantly improves the joint detection of structural and logical anomalies. In particular, we extend our Student–Teacher method from Chapter 5 by a global network branch that captures long-range dependencies of the anomaly-free training data. The content of Chapter 6 and Chapter 7 is based on the publication:

> Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, Carsten Steger: **Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization;** in: *International Journal of Computer Vision* 130(4):947-969, 2022, DOI: 10.1007/s11263-022-01578-9.

Chapter 8: Unsupervised Detection of Geometric Anomalies in 3D Data. The focus of Chapters 2 through 7 is on the detection of anomalies in two-dimensional color or grayscale images. In the remaining parts of this thesis, we shift our focus to the detection of geometric anomalies in data obtained from industrial 3D sensors. This chapter provides a brief introduction to the technical prerequesites for detecting anomalies in three-dimensional data. In particular, we discuss common data representations, describe an evaluation protocol for 3D anomaly detection, and review related literature.

Chapter 9: The MVTec 3D-AD Dataset for 3D Anomaly Detection. Significant advances were made in many areas of 3D computer vision, such as point cloud registration, 3D classification, 3D semantic segmentation, and rigid pose estimation. However, very little attention is paid to the problem of anomaly detection in the 3D domain. We attribute this to the lack of existing datasets for this task. Therefore, we introduce the MVTec 3D Anomaly Detection dataset, the first comprehensive dataset for unsupervised 3D anomaly detection. A benchmark of existing methods for this task shows that there is still considerable room for improvement.

> Paul Bergmann, Xin Jin, David Sattlegger, Carsten Steger: **The MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization;** in: *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, 202-213, 2022, DOI: 10.5220/0010865000003124.

Chapter 10: Deep Geometric Descriptors for 3D Anomaly Detection. In order to improve upon existing approaches for 3D anomaly detection, we extend our Student–Teacher method from Chapter 5 to the 3D domain. In particular, we describe how to design expressive teacher networks for 3D point clouds that extract deep local geometric descriptors. We introduce a new self-supervised pretraining strategy that does not require any annotations. Our method sets a new state of the art on the MVTec 3D-AD dataset.

Chapter 11: Conclusion. We conclude the thesis by discussing open research questions and possible extensions to this work.

2 Foundations

In this chapter, we formally introduce the problem of unsupervised anomaly detection and review some base architectures that are frequently used to build computer vision models for this task. Additionally, a brief overview of fundamental concepts in deep learning is given. For more thorough introductions to machine learning, deep learning, and computer vision, the reader may refer to the standard literature by Bishop [2006], Goodfellow et al. [2016], Steger et al. [2018], and Szeliski [2011].

An overview of the mathematical notation used throughout this thesis is listed in Table 2.1. For some of the formulas, more details are provided in the following parts of this chapter.

2.1 Unsupervised Anomaly Detection in Image Data

We begin by defining the unsupervised anomaly detection problem when applied to image data. An image $I : D \to \mathbb{R}^C$ is a function that assigns a real-valued vector to each pixel in the image domain $D = \{0, \ldots, H-1\} \times \{0, \ldots, W-1\}$. The height, width, and the number of channels of the image are denoted by $H \in \mathbb{N}^+$, $W \in \mathbb{N}^+$, and $C \in \mathbb{N}^+$, respectively. Commonly used image formats are single-channel grayscale images, 3-channel RGB color images, or multi-channel images obtained from multispectral or hyperspectral cameras.

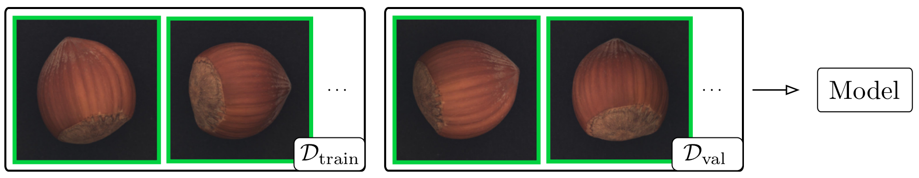
The task is to create a model on a training set of exclusively anomaly-free images $\mathcal{D}_{\text{train}} \subseteq \mathcal{I}$. Here, $\mathcal{I}$ denotes the space of images that is the set of all functions $I : D \to \mathbb{R}^C$. During inference, the model should be able to detect anomalies in images of a test set $\mathcal{D}_{\text{test}} \subseteq \mathcal{I}$. In addition to the training and test datasets, a validation set $\mathcal{D}_{\text{val}} \subseteq \mathcal{I}$ is often used to determine hyperparameters during model training. Like the training set, it only contains anomaly-free samples. It is assumed that all images in the three datasets exhibit the same domain and number of channels. If the images of an application exhibit different spatial dimensions, they can all be scaled to the same width and height before passing them to an anomaly detection model.

Anomalies may occur as any deviation from the training data distribution. However, since we focus on industrial inspection scenarios, we are typically concerned with the unsupervised detection of various types of defects in manufactured products. When we are interested in the accurate localization of anomalous regions in test images, we refer to this task as *anomaly localization*. In contrast, we refer to the problem of making a binary decision between anomaly-free and anomalous images as *anomaly classification*. In this thesis, we are interested in methods that can perform both tasks simultaneously.

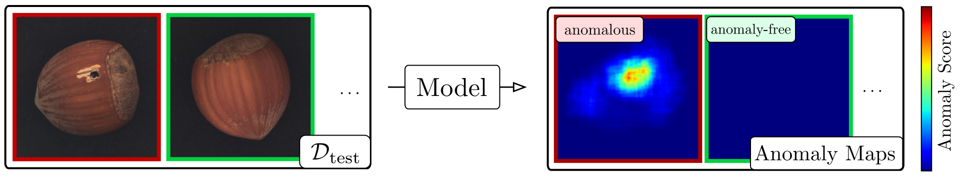
To localize anomalies, a model must assign a real-valued anomaly score to each pixel location $\boldsymbol{p} \in D$ in the form of an anomaly map $A : D \to \mathbb{R}$. Larger values of $A(\boldsymbol{p})$ indicate

	Sets and Spaces
$\mathbb{N}$	Set of non-negative integers.
$\mathbb{N}^+$	Set of positive integers.
$\mathbb{R}$	Set of real numbers.
$S_1 \subseteq S_2$	S_1 is a subset of S_2.
$S_1 \times S_2$	Cartesian product of two sets.
$\lvert S \rvert$	Cardinality of a set.
$\{a, b, c\}$	Set of three elements.
$\{1, 2, \ldots, n\}$	Set of integers from 1 through n.
$[a, b)$	The real interval including a and excluding b.
	Numbers and Arrays
$x \in \mathbb{R}$	A scalar.
$\boldsymbol{x} \in \mathbb{R}^d$	A column vector of dimension d.
$\boldsymbol{A} \in \mathbb{R}^{h \times w}$	A matrix with h rows and w columns.
$\boldsymbol{A}_{i,j}$	The element of the matrix $\boldsymbol{A}$ at index i, j.
$\boldsymbol{x}^T$, $\boldsymbol{A}^T$	The transpose of $\boldsymbol{x}$ and $\boldsymbol{A}$, respectively.
	Functions
$f : X \to Y$	Function f with domain X and range Y.
$f_{\boldsymbol{\theta}}$	Function f depending on parameters $\boldsymbol{\theta}$.
	The parameters $\boldsymbol{\theta}$ are omitted when it serves readability.
$f \circ g$	Composition of functions f after g.
$\mathcal{L}(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$	Loss function with respect to parameters $\boldsymbol{\theta}$.
$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$	Gradient of a loss function with respect to its parameters $\boldsymbol{\theta}$.
	Distributions and Probability Theory
$\boldsymbol{x}$	A vector-valued random variable.
$\Pr(\boldsymbol{x})$	Probability distribution of $\boldsymbol{x}$.
$\Pr(\boldsymbol{x} \vert \boldsymbol{y})$	Conditional probability of $\boldsymbol{x}$ given $\boldsymbol{y}$.
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
$\mathcal{KL}(\Pr(\boldsymbol{x}), \Pr(\boldsymbol{y}))$	KL-divergence between two probability distributions.

Table 2.1: Overview of the mathematical notation used throughout this thesis.



(a) Training Phase



(b) Inference Phase

Figure 2.1: Overview of the anomaly detection problem applied to image data. (a) A model is trained on a set of exclusively anomaly-free training and validation images. (b) During inference, the model must assign an anomaly score to each pixel in the images of the test set. The resulting anomaly maps may be used to derive image-level anomaly scores for anomaly classification.

that a pixel is believed more likely to be part of an anomaly. To make a binary decision whether a pixel contains an anomaly or not, the anomaly scores must be compared to a threshold $t \in \mathbb{R}$. If a score exceeds the threshold, i.e., $A(\boldsymbol{p}) > t$, the pixel $\boldsymbol{p}$ is classified as anomalous. For image-level classification, the pixelwise values of A can be aggregated to derive a single anomaly score, for example, by calculating the mean $\frac{1}{|D|}\sum_{\boldsymbol{p}\in D} A(\boldsymbol{p})$ or the maximum $\max_{\boldsymbol{p}\in D} A(\boldsymbol{p})$. Again, comparing the image-level score to a threshold can be used for making a binary decision if a dataset sample should be considered as anomalous or anomaly-free.

A graphical illustration of the anomaly detection problem on image data is given in Figure 2.1. The training and validation sets contain images of defect-free hazelnuts that are used to create a machine learning model. During inference, the model encounters images of anomaly-free nuts as well as images that exhbit various defects, such as the hole in the surface of a nut. For each image, it produces pixel-accurate anomaly scores as well as a classification decision.

2.2 Evaluation of Anomaly Detection Algorithms

Next, we describe how to evaluate the performance of anomaly detection algorithms. We discuss metrics that assess the anomaly localization performance as well as metrics for anomaly classification. While annotations are not required to train unsupervised anomaly detection models, ground truth information must be available to assess their performance on the test set. For each test image in $\mathcal{D}_{\text{test}}$, we require a pixel-precise ground truth map $G : D \to \{0, 1\}$. For each pixel, it indicates whether an anomaly is present, $G(\boldsymbol{p}) = 1$, or not, $G(\boldsymbol{p}) = 0$.

2.2.1 Anomaly Localization Metrics

Pixel-Level Metrics

Evaluating the performance of anomaly localization algorithms on a per-pixel level treats the classification outcome of each pixel as equally important. A pixel can be classified as either a true positive (TP), false positive (FP), true negative (TN), or false negative (FN). A decision is made by comparing the anomaly score of each pixel to the threshold t, and then comparing the classification outcome to the respective ground truth label. For instance, if a pixel is believed to contain an anomaly, i.e., $A(\boldsymbol{p}) > t$, and it is also labeled as anomalous, i.e., $G(\boldsymbol{p}) = 1$, it is classified as a true positive. For each of the four cases the total number of pixels on the test dataset $\mathcal{D}_{\text{test}}$ is computed as:

$$\text{TP} = \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} \big|\{\boldsymbol{p} \mid G_i(\boldsymbol{p}) = 1\} \cap \{\boldsymbol{p} \mid A_i(\boldsymbol{p}) > t\}\big|, \tag{2.1}$$

$$\text{FP} = \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} \big|\{\boldsymbol{p} \mid G_i(\boldsymbol{p}) = 0\} \cap \{\boldsymbol{p} \mid A_i(\boldsymbol{p}) > t\}\big|, \tag{2.2}$$

$$\text{TN} = \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} \big|\{\boldsymbol{p} \mid G_i(\boldsymbol{p}) = 0\} \cap \{\boldsymbol{p} \mid A_i(\boldsymbol{p}) \leq t\}\big|, \tag{2.3}$$

$$\text{FN} = \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} \big|\{\boldsymbol{p} \mid G_i(\boldsymbol{p}) = 1\} \cap \{\boldsymbol{p} \mid A_i(\boldsymbol{p}) \leq t\}\big|, \tag{2.4}$$

where $\boldsymbol{p} \in D$ and A_i and G_i correspond to the anomaly map and ground truth of the test image with index $i \in \{1, \ldots, |D_{\text{test}}|\}$, respectively. Figure 2.2 shows a schematic illustration of these four pixel-level classification results. Anomaly scores are first thresholded to obtain binary predictions. A comparison to the corresponding ground truth then yields the pixelwise classification results.

Based on these absolute measures, which depend on the total number of pixels in the dataset, relative scores such as the per-pixel false positive rate (FPR), true positive rate (TPR), and precision (PRC) can be derived:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{2.5}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.6}$$

$$\text{PRC} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.7}$$

Apart from these three widely used metrics, another common measure to benchmark segmentation algorithms is the intersection over union (IoU), computed on two sets of pixels. In the context of anomaly localization, one considers the set of all anomalous predictions P_{ano} and the set of all ground truth pixels that are labeled as anomalous G_{ano}, i.e.,

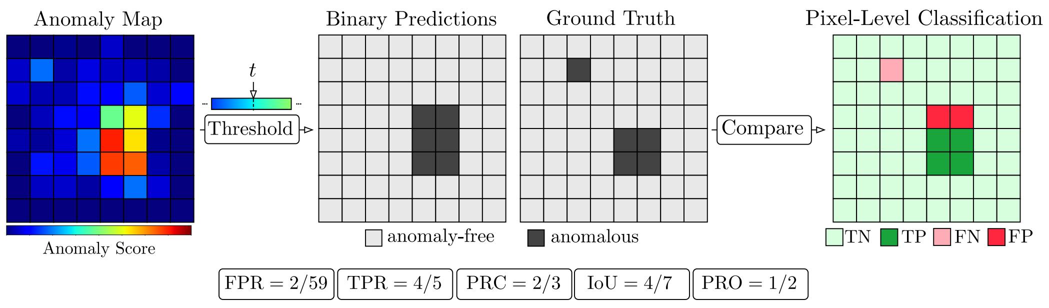


Figure 2.2: Obtaining a classification result for each image pixel. Anomaly scores are first converted into binary predictions by applying a threshold operation. A pixelwise comparison to the ground truth then yields the classification results. These can be used to compute performance metrics, such as the TPR, FPR, PRC, IoU, or PRO.

$$P_{\text{ano}} = \bigcup_{i=1}^{|\mathcal{D}_{\text{test}}|} \{(i, \boldsymbol{p}) \mid \boldsymbol{p} \in D \text{ and } A_i(\boldsymbol{p}) > t\}, \tag{2.8}$$

$$G_{\text{ano}} = \bigcup_{i=1}^{|\mathcal{D}_{\text{test}}|} \{(i, \boldsymbol{p}) \mid \boldsymbol{p} \in D \text{ and } G_i(\boldsymbol{p}) = 1\}. \tag{2.9}$$

Analogous to the relative measures above, the IoU for the class ‘anomalous’ can also be expressed in terms of absolute pixel classification measures:

$$\text{IoU} = \frac{|P_{\text{ano}} \cap G_{\text{ano}}|}{|P_{\text{ano}} \cup G_{\text{ano}}|} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}. \tag{2.10}$$

All of these measures have the advantage that they are straightforward and efficient to compute. However, treating each pixel as entirely independent introduces a bias towards large anomalous regions. Detecting a single anomaly with a large area can make up for the failure to detect numerous smaller ones. Therefore, one often considers metrics that are computed with respect to connected anomalous regions within the ground truth.

Region-Level Metrics

Instead of treating every pixel independently, region-level metrics average the performance over each connected component of the ground truth. This is especially useful if the detection of relatively smaller anomalies is considered equally important to the detection of relatively larger ones. This may be the case in practical applications, depending on the dataset under consideration. Here, we propose the per-region overlap (PRO) metric as an example for a region-level metric.

First, for each test image the ground truth is decomposed into its connected components. A connected component $K \subseteq D$ is a subset of the image domain in which all pixels are labeled anomalous and that are in the immediate vicinity of each other.

In particular, we follow the definition of connected components given by Steger et al. [2018, Chapter 3.4.2.] and use a 4-connectivity. Let $C_{i,j} = \{(i, \boldsymbol{p}) \mid \boldsymbol{p} \in K_j\}$ denote the set of pixels classified as anomalous for a connected component $K_j \subseteq D$ in the ground truth image with index i and P_{ano} denote the set of pixels predicted as anomalous for a threshold t. The per-region overlap can then be computed as

$$\text{PRO} = \frac{1}{k} \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} \sum_{j=1}^{k_i} \frac{|P_{\text{ano}} \cap C_{i,j}|}{|C_{i,j}|}, \tag{2.11}$$

where k_i denotes the number of ground truth components for a single test image and $k = \sum_i k_i$ is the total number of components in $\mathcal{D}_{\text{test}}$. The PRO metric is closely related to the TPR. The important difference is that the PRO metric averages the TPR over each ground truth region instead of averaging over all image pixels. Note that it is not straightforward to adapt other per-pixel measures such as the PRC or the IoU to the per-region case. This is caused by the fact that they make use of the FPR, and false positives cannot be readily attributed to any specific ground truth region.

To illustrate the difference between the PRO metric and the pixelwise measures, Figure 2.2 shows specific values for the above performance metrics computed on a simple toy example. The ground truth contains two anomalous connected components. The thresholded prediction, however, only detects one of them. The PRO metric reflects this and reports a localization accuracy of 1/2. Since the TPR, PRC, and IoU do not operate on a region-level and hence are biased towards the detection of larger connected components, they all report values larger than 1/2.

Threshold-Independent Metrics

All of the above metrics depend on the previous selection of a suitable threshold t, which is a challenging problem in practice. If a suboptimal threshold is selected, the metrics may give a skewed picture of the performance of a method. Furthermore, the desired threshold can depend on the requirements of the specific application under consideration. For instance, if a certain false positive rate is acceptable in practice, this allows the selection of smaller thresholds which can increase the number of true positives.

To decouple the performance evaluation from the choice of any particular threshold, it is common to compute the above metrics at multiple distinct thresholds. Furthermore, it is often desirable to compare two metrics simultaneously since, for example, a high TPR is only useful if the corresponding FPR is low. A way to achieve this is to plot two metrics against each other and compute the area under the resulting curve. A well-known example is the receiver operator characteristic (ROC), which plots the FPR versus the TPR. Another frequently used measure is the precision–recall curve (PR), which plots the true positive rate (recall) versus the precision. In this thesis, we additionally investigate the PRO curve, which plots the FPR versus the PRO, as well as the IoU curve, which shows the FPR versus the IoU.

2.2.2 Anomaly Classification Metrics

Performance metrics for anomaly classification measure how well a method separates anomalous from anomaly-free test samples based on their image-level scores. As for the pixel-level metrics, a dataset sample can be either classified as a true positive, false positive, true negative, or false negative. Given an anomaly score $a_i \in \mathbb{R}$ for each image in the test set, adapting the definition of these four classification results from pixel- to sample-level is straightforward. The anomaly score of each test sample is compared to a classification threshold $t_c \in \mathbb{R}$. If $a_i > t_c$ and any pixel in the ground truth of the respective sample is labeled as anomalous, it is classified as a true positive. The total number of true positive samples can be computed as:

$$\mathrm{TP} = \Bigg| \bigcup_{i=1}^{|\mathcal{D}_{\text{test}}|} \{ i \mid \exists \boldsymbol{p} \in D : G_i(\boldsymbol{p}) = 1 \text{ and } a_i > t_c \} \Bigg|. \tag{2.12}$$

The total number of false positives, true negatives, and false negatives can be computed analogously. From these absolute values, relative measures such as the TPR and FPR can be derived. As for anomaly localization, it is common to evaluate them for multiple thresholds and compute the area under the ROC or PR curve as performance measures.

2.3 Fundamental Concepts in Deep Learning

Neural networks have emerged as a powerful tool for a variety of computer vision tasks, including anomaly detection. There exists a lot of literature on this topic and a comprehensive overview of deep learning theory is well beyond the scope of this thesis. For an introduction, the reader may refer to Goodfellow et al. [2016]. Here, we restrict ourselves to a brief summary of core concepts and an explanation of terminology that is frequently used in the following chapters.

Neural Networks

On an abstract level, a neural network can be written as a function $f_{\boldsymbol{\theta}} : X \to Y$ that processes some input sample $\boldsymbol{x} \in X$ and produces a corresponding output, given a vector of model parameters $\boldsymbol{\theta} \in \mathbb{R}^m$. For instance, the domain X may be selected as a the space of d-dimensional vectors $\mathbb{R}^d$. Other input domains are also possible, e.g., the space of images $\mathcal{I}$. The entries of $\boldsymbol{\theta}$ are also known as the *weights* of the model. The range Y depends on the task that the neural network is supposed to solve. When applied to classification, for example, it may output a distributon over class probabilities. For readability, we often omit the parameter vector in the notation and write $f(\boldsymbol{x})$. The function f is typically realized as a composition of n sequential operations:

$$f = f_n \circ f_{n-1} \circ \dots f_2 \circ f_1. \tag{2.13}$$

Each of the functions f_i is called a *layer* of the neural network. The number of layers is referred to as its *depth.* Neural networks with a large number of layers are termed *deep neural networks* (DNNs).

Network Optimization

Given a set of inputs $\mathcal{D}_{\text{train}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\} \subseteq X$ and corresponding target values $\boldsymbol{y}_i \in Y$, the goal during network optimization is to determine parameters $\boldsymbol{\theta}$ such that for each input, the output of the network closely matches the assigned target value. Optimizing the parameters of a network is also referred to as the *training* of a neural network. Hence, $\mathcal{D}_{\text{train}}$ is also called the *training dataset.* After training, $f_{\boldsymbol{\theta}}$ should ideally generalize to new, unseen samples of a *test dataset* $\mathcal{D}_{\text{test}} \subseteq X$.

A network is trained by adjusting its parameters to minimize a predefined *loss function* $\mathcal{L}(\boldsymbol{\theta}) : \mathbb{R}^m \to \mathbb{R}$. It computes a real-valued score that reflects how accurate the current parameters transform input samples to their respective target values. A common way to define the loss is to specify a function $r : Y \times Y \to \mathbb{R}$ that computes a residual between the desired output of each dataset sample and the actual output produced by the network. The loss over the entire training dataset can then be computed as:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i=1}^{|\mathcal{D}_{\text{train}}|} r(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i). \tag{2.14}$$

For convenience, we may also write this expression as $\mathcal{L}(f_{\boldsymbol{\theta}})$ or $\mathcal{L}(f)$. In practice, the loss function is highly task-specific. In regression problems, for example, a popular choice for the residual r is the L_p-distance between the prediction and the target value, where $p \geq 1$. For a vector $\boldsymbol{x}$ with elements x_i, the L_p-norm is defined as:

$$\|\boldsymbol{x}\|_p = \Big(\sum_i |x_i|^p\Big)^{\frac{1}{p}}.$$

During training, the model parameters can be iteratively adjusted by performing *gradient descent*, i.e., by computing the gradient of the loss function on a set of input samples with respect to the model parameters. The parameters are updated along the negative direction of the gradient:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \lambda \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{2.15}$$

Here, $\boldsymbol{\theta}_k$ denote the model parameters at step $k \in \mathbb{N}$ of the optimization process. At the start of the optimization, the weights $\boldsymbol{\theta}_0$ need to be initialized appropriately. A straightforward strategy is to sample random weights from a predefined probability distribution. A second option is to use the weights of a network that was previously trained on a different dataset as a starting point for the optimization. This technique is also known as *transfer learning.* It may reduce training time and the number of dataset samples required for successful optimization [Zhuang et al., 2021]. The parameter $\lambda \in \mathbb{R}^+$ is known as the *learning rate* and operates as a scaling factor. An appropriate choice of λ is highly important. If chosen too large, this often leads to unstable trainings or divergence. If chosen too small, it may take a long time until convergence, i.e., until a chosen termination criterion is reached.

Since the function $f_{\boldsymbol{\theta}}$ is a composition of a series of other functions, the gradient of the loss can be computed with the backpropagation algorithm [Rumelhart et al., 1986] that computes derivatives by multiple applications of the chain rule. This computation is often referred to as performing a *backward pass* within the neural network since the gradient computation starts at the output layer of the network and ends at its input layer. This is in contrast to the *forward pass*, which describes the computation of an output value for a corresponding input sample.

Computing the gradient over the entire training dataset becomes computationally expensive when $\mathcal{D}_{\text{train}}$ contains a large number of samples, which is common in deep learning applications. Hence, the gradient is usually only approximated by computing the loss function on a subset of the training samples $\{\boldsymbol{x}_{k_1}, \boldsymbol{x}_{k_2} \ldots, \boldsymbol{x}_{k_B}\} \subseteq \mathcal{D}_{\text{train}}$ that is randomly drawn without replacement:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \lambda \nabla_{\boldsymbol{\theta}} \frac{1}{B} \sum_{i=1}^{B} r(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{k_i}), \boldsymbol{y}_{k_i}). \tag{2.16}$$

This approach is also known as *stochastic gradient descent* (SGD). The selected subset of training samples used for gradient computation is called a *training batch* or *minibatch* with *batch size* $B \in \mathbb{N}^+$. Interestingly, using stochastic updates was empirically shown to generalize better to samples of the test set compared to computing the gradient jointly on all dataset samples in some deep learning applications [Xie et al., 2021]. Training batches are drawn until all samples of $\mathcal{D}_{\text{train}}$ were iterated. Iterating the training dataset once is also known as running a *training epoch.* Typically, a network is trained for multiple epochs until the training loss converges.

Many extensions and variations of the stochastic gradient descent method have been proposed for optimizing deep learning models. A typical goal of these is to improve the convergence rate or the training stability. For instance, SGD can be extended by a momentum term [Polyak, 1964], which keeps track of a moving average of previous model updates. In each iteration, the final model update is then a linear combination of the moving average and the new update direction, which results in more robust gradients. Other popular optimization techniques do not use a fixed learning rate, but dynamically adapt λ in each training iteration. Examples for such optimization methods are AdaGrad [Duchi et al., 2011] or the Adam optimizer [Kingma and Ba, 2015].

Model Selection

Once the training is completed, one may ask which of the parameter vectors $\boldsymbol{\theta}_k$ should be used when deploying the network in a practical application. An obvious choice is to use the parameters of the very last iteration or those that yielded the lowest training error during the optimization procedure. However, low training errors do not necessarily correspond to a good performance on the test dataset. Since DNNs often contain millions of model parameters and are expressive enough to closely approximate very complex functions, they may *overfit* the training data by memorization of the correct output for each training sample. This results in low training errors, but poor generalization of the model to unseen data. Therefore, it is typically of interest to select model parameters that are likely to generalize well to $\mathcal{D}_{\text{test}}$. To determine such a model, a common strategy is to regularly evaluate the loss function on a set of validation samples $\mathcal{D}_{\text{val}}$ during training. The samples in $\mathcal{D}_{\text{val}}$ are independent of those in the training set and can be used to judge how well the current model transfers to unseen samples. Once the training has finished, the model parameters that yield the lowest validation error can be selected as the final model.

Network Regularization

To discourage a model from overfitting, a *regularization term* can be added to the loss function. These terms do not contribute directly to the particular task the network is supposed to solve. Instead, they specify solutions in the parameter space that are more desireable than others. One common regularization technique is L_2-regularization. It penalizes solutions in which the parameter vector $\boldsymbol{\theta}$ exhibits a large L_2-norm. This is achieved by adding the term

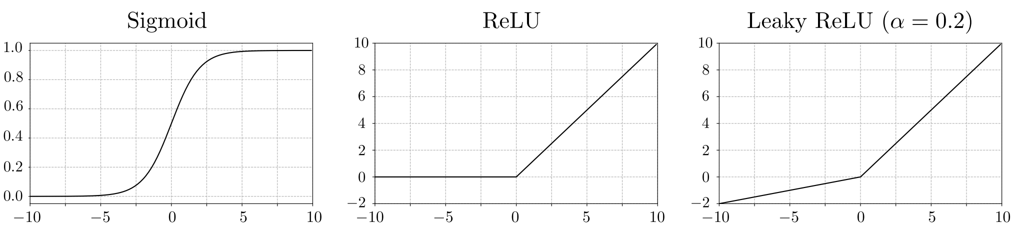
$$\Omega(\boldsymbol{\theta}) = \alpha \boldsymbol{\theta}^T \boldsymbol{\theta} \tag{2.17}$$

to the original loss function of the network. The variable $\alpha \in [0, \infty)$ is a scaling factor that controls the strength of the regularization. A similar effect is achieved by a technique called *weight decay* [Krogh and Hertz, 1991]. It reduces the parameter norm explicitly by multiplying the weights with a constant factor before applying the gradient update. A different method that may prevent a model from overfitting is by adding artificial noise, for example, through *dropout layers* [Srivastava et al., 2014]. During training, these layers randomly set entries of their input array to zero according to a predefined dropout probability.

Convolution Layers

When applying DNNs to images, it is popular to process the input data with a series of convolution layers. Convolving an image with a kernel yields a two-dimensional array of scalar values, which we call a *feature map.* In practice, a convolution layer computes multiple feature maps by applying different filter kernels. The weights of the kernels constitute learnable parameters of the layer. A scalar bias term is added to the result of each convolution, which is also included in the layer's parameter vector.

Neural networks that use convolution layers as key building blocks are termed *convolutional neural networks* (CNNs). Note that in practical implementations, convolution filters are often replaced by cross-correlations, which are convolutions with a flipped kernel. The success of using convolutions in deep learning has several reasons. For example, they can be effectively parallelized and hence efficiently computed on modern graphical processing units (GPUs). Furthermore, the same filter kernel is typically applied to



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\text{LReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases}$$

Figure 2.3: Examples for different activation functions.

the entire input array by evaluating locally strided windows. This is known as *weight sharing*. It limits the number of parameters that must be learned for each convolution, effectively reduces the training time of the deep learning model, improves the training stability, and adds equivariance to translations with respect to the input image.

Since a convolution cannot be computed at pixel locations where the kernel leaves the input array, the input may be artificially extended by applying a *padding* operation. A popular padding technique is zero padding, which extends the input by a border of zero-valued entries.

There exist many extensions and variations to standard convolution layers. For instance, the distance between spatial locations where the convolution kernel is applied can be increased. This is known as a *strided convolution*. It can be used to effectively downsample an input with respect to its spatial dimensions. In *dilated convolutions* [Yu and Koltun, 2016], the filter kernel is spatially extended without increasing the number of kernel parameters to capture a larger area of the input array in each output feature.

Activation Layers

Convolutions are linear operations. This implies that processing an image with a series of convolution layers does not enable them to approximate complex, nonlinear functions. Therefore, convolution layers are often followed by activation layers that introduce nonlinear operations into the network. Typically, these are one dimensional functions that are applied to each element of the input array independently. Many different nonlinearities have been proposed to be used in CNNs [Nair and Hinton, 2010, Hendrycks and Gimpel, 2016]. Figure 2.3 gives an overview of activations that are frequently used in this thesis. The first is a sigmoid activation function that produces values in the interval $(0, 1)$, which is useful to model outputs that resemble probabilities. A second popular activation function is the Rectified Linear Unit, or ReLU. It corresponds to the identity function for positive input values. Otherwise, the output is set to 0. A modification to this is the Leaky ReLU activation, which instead scales negative inputs with a constant scalar factor $\alpha \in [0, \infty)$.

Vanishing Gradients

To model complex functions, neural networks often require a very large number of layers. Unfortunately, as the number of layers in a network grows, its optimization becomes increasingly challenging. In particular, it was shown that the gradient norm in deep networks may attain very small values, which leads to long training times. This problem of *vanishing gradients* may even prevent a network from learning anything meaningful at all [Hochreiter et al., 2001]. It can be addressed by using *residual layers* [He et al., 2016, Veit et al., 2016], which add shortcut connections to a network. They transform their input by any differentiable function and add the original input back to the transformed output. A second technique to stabilize the training and counter vanishing gradients is to use *normalization layers* such as batch normalization [Ioffe and Szegedy, 2015] or instance normalization [Ulyanov et al., 2017]. They enforce their output features to follow a certain predefined distribution, e.g., a Gaussian normal distribution.

Upsampling Layers

Sometimes it is desirable to design networks that generate an image or a high-dimensional feature map from a low-dimensional representation. In such cases, *transposed convolution* or *zooming* layers may be used to upsample an array with respect to its spatial dimensions. Similar to the convolution operation, transposed convolutions contain a set of learnable parameters, i.e., an upsampling kernel. In contrast, zooming layers perform a parameter-free operation based on interpolation algorithms such as nearest neighbor or bilinear interpolation.

Data Augmentation

To prevent DNNs from overfitting and to improve their generalization capabilities, it is common to train them on a large amount of data. For example, the ImageNet dataset [Krizhevsky et al., 2012] provides more than one million samples to create classification models. If such a large number of dataset samples is not available, additional training samples can be created by applying transformations to the original input images. This process of creating a larger dataset from a smaller one is known as *data augmentation*. An image sample can be augmented by applying color transformations, such as a random rescaling of its RGB channels. Another possibility is to apply geometric transformations, such as a random rotation or translation of the pixel coordinates.

2.4 Base Architectures for Unsupervised Anomaly Detection

The landscape of methods for unsupervised anomaly detection is diverse and many approaches have been suggested to tackle the problem. Pimentel et al. [2014] and Ehret et al. [2019] give a comprehensive review of existing work. The following provides a brief overview of network architectures that are frequently used to build computer vision models for unsupervised anomaly detection. We further discuss some existing anomaly detection methods that build on these base architectures.

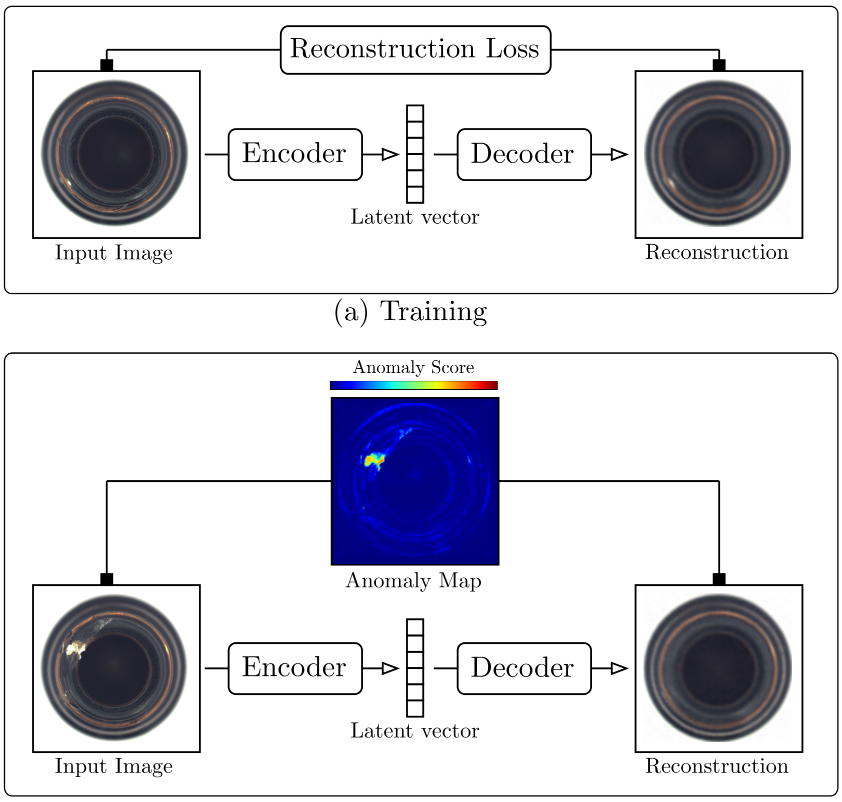


Figure 2.4: Illustration of how an autoencoder can be used for unsupervised anomaly detection. (a) The autoencoder is trained on anomaly-free data only, computing a loss between the training samples and the respective reconstructions. (b) During inference, anomaly scores are derived by a pixelwise comparison between the input and the reconstruction.

2.4.1 Convolutional Autoencoders

Autoencoders (AE) [Masci et al., 2011] are a class of neural networks that attempt to learn a compact representation of their inputs by reconstructing dataset samples through a bottleneck. They consist of two subnetworks, namely an encoder and a decoder. The encoder receives an image and projects it to a low-dimensional latent variable. The decoder then transforms this variable back into an image by applying upsampling operations. For optimization, the network is trained with reconstruction losses. A popular choice is to compute the pixelwise L_2-distance between the inputs and the reconstructed images.

Autoencoders can be trained without any annotations. This makes them popular base architectures for unsupervised anomaly detection methods [Sakurada and Yairi, 2014]. An example is shown in Figure 2.4. In the top subfigure, an autoencoder is trained on a dataset of exclusively anomaly-free images of a bottle mouth. The autoencoder manages to reconstruct the input image quite accurately, despite producing a slightly blurry reconstruction due to the limited capacity of the bottleneck. The bottom subfigure illustrates the output of the trained autoencoder when processing an anomalous test image. In this case, the glass of the bottle mouth is damaged. Since the autoencoder has not observed anomalous samples during training, it fails to reproduce the defect. Instead, it produces an image of an anomaly-free bottle. A direct comparison of the input and the reconstructed image reveals the area where the anomaly occurs.

In practice, the particular choice of the latent dimension can have a significant impact on the anomaly detection performance. If chosen too small, the latent space may not be expressive enough to model all variations of the training dataset, which leads to inaccurate reconstructions. If chosen too large, the model may learn to copy the input data into the latent variable, which would allow it to reconstruct defects on anomalous test images as well.

Various extensions to standard autoencoders exist. Variational autoencoders (VAEs) [Kingma and Welling, 2014] encourage their latent samples to follow a certain prior distribution. This is achieved by making the encoder output the parameters of the posterior distribution of the latent samples given an input image. A similarity measure between the encoder distribution and the prior is then added to the reconstruction loss. During inference, this extension allows to generate new images by randomly drawing samples from the prior and passing them through the decoder network.

VAEs have been frequently employed for unsupervised anomaly detection. An and Cho [2015] define a reconstruction probability for every image pixel by drawing multiple samples from the estimated encoding distribution and measuring the variability of the decoded outputs. Soukup and Pinetz [2018] disregard the trained decoder entirely and instead compute a KL-divergence as an anomaly measure between the prior and the distribution produced by the encoder. This is based on the assumption that anomalous inputs will cause distributions that are very different from those of the prior. Since this approach yields a single anomaly score for an input sample, it cannot be directly applied to anomaly localization. Instead, if an anomaly score needs to be computed for every input pixel, computationally expensive patch-based evaluations must be performed. Similarly, Vasilev et al. [2020] define multiple anomaly measures, either by purely considering latent space behavior or by combining these measures with per-pixel comparisons between the input and the reconstruction. They also compute a single scalar value as an anomaly score.

Autoencoders tend to produce blurry and inaccurate reconstructions [Bergmann et al., 2019b]. This may lead to an increase in false positives when applied to anomaly detection. They might also learn to copy parts of the input data, which would allow them to reconstruct anomalous features during inference. To discourage this behavior, Gong et al. [2019] propose to extend autoencoders with an integrated memory module. Their MemAE selects numerous latent features during training that need to be reused for reconstruction during inference. Building on this idea, Park et al. [2020] introduce MNAD, a method for unsupervised anomaly localization that uses a memory augmented autoencoder instead of a standard one.

2.4.2 Generative Adversarial Networks

Generative adversarial networks (GANs) [Goodfellow et al., 2014] consist of a generator and a discriminator network. As suggested by their name, they are a class of deep generative models that are trained with an adversarial loss function. The task of the generator Gen : $\mathbb{R}^d \rightarrow \mathcal{I}$ is to produce realistic images that ressemble those of the training set. As input, it processes a low-dimensional vector $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ that is drawn from a standard normal distribution. The discriminator Dis : $\mathcal{I} \rightarrow [0, 1]$ attempts to distinguish between the images produced by the generator and the real dataset samples. Its output is interpreted as the probability that the processed input image is real. When trained successfully, the generator produces images that are indistinguishable from real dataset samples. New samples can be created by drawing latent samples from the normal distribution and passing them through the generator. The loss functions of the two networks can be written as:

$$\mathcal{L}(\text{Dis}) = \frac{1}{B}\sum_{i=1}^{B} log(\text{Dis}(I_i)) + log(1 - \text{Dis}(\text{Gen}(\boldsymbol{z}_i))), \tag{2.18}$$

$$\mathcal{L}(\text{Gen}) = \frac{1}{B}\sum_{i=1}^{B} -log(1 - \text{Dis}(\text{Gen}(\boldsymbol{z}_i))). \tag{2.19}$$

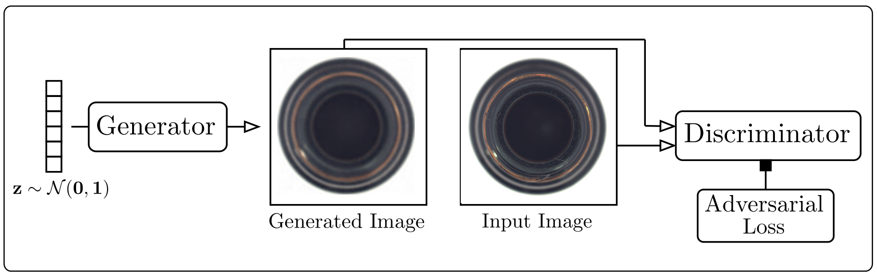
Note that the loss of the generator is just the negative loss of the discriminator where the first term was dropped since the parameters of the generator do not depend on it. The discriminator is encouraged to output values close to 1 for real and values close to 0 for generated images. Conversely, the generator is encouraged to produce images for which the discriminator assigns a high probability for them to be real.

A popular method that uses GANs for anomaly detection is AnoGAN, originally introduced by Schlegl et al. [2017]. In a first step, a generator and discriminator are trained on anomaly-free data. Afterwards, the generator is able to produce new images that resemble the samples of the training set. To detect anomalies in a test image J, the latent space of the generator is searched for a vector $\boldsymbol{z}^*$ that reproduces the image as closely as possible:

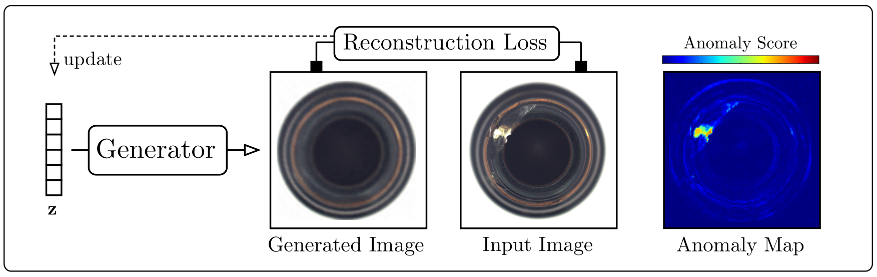
$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} ||J - \text{Gen}(\boldsymbol{z})||_2^2. \tag{2.20}$$

In practice, the sample $\boldsymbol{z}^*$ is obtained by generating a random latent vector that is iteratively updated using gradient descent. The reconstruction error between the input and the generated sample serves as a loss function. Once converged, anomaly scores can be derived by a pixelwise comparison of the input to the reconstruction. The entire process is summarized in Figure 2.5, where the training and inference process of AnoGAN is illustrated on images of a bottle mouth.

Compared to autoencoders, GANs tend to produce sharper reconstructions, which may benefit the anomaly detection performance. However, their training can also be more difficult due to the use of an adversarial loss function [Arjovsky and Bottou, 2017].



(a) Training



(b) Inference

Figure 2.5: Illustration of how a GAN can be used for unsupervised anomaly detection. (a) The GAN is trained on anomaly-free data only. A generator tries to produce samples that the discriminator cannot distinguish from real dataset samples. (b) During inference, the latent space of the generator is searched for a sample that produces an image that closely matches the input image. Anomaly scores are computed by a pixelwise comparison between the input and the generated image.

They are also prone to run into mode collapse, i.e., there is no guarantee that al modes of the distribution of non-defective images are captured by the model. Furthermore, GANs lack an encoder network that determines the latent sample for a corresponding input image. This makes the inference process of AnoGAN computationally expensive, since many iterations of gradient descent need to be performed for each test sample.

To mitigate these issues, Schlegl et al. [2019] introduce f-AnoGAN. It uses a more recent GAN architecture called Wasserstein GAN with Gradient Penalty (WGAN-GP) [Gulrajani et al., 2017], which improves the training stability. They further add an encoder network that is trained to determine $\boldsymbol{z}^*$ for a given input image with a single forward pass. This encoder is trained separately after the GAN training has finished. Similarly, Zenati et al. [2018] propose to use bidirectional GANs [Donahue et al., 2017] to add the missing encoder network for faster inference.

2.4.3 Feature Distribution Models

The methods based on autoencoders or GANs described above are trained using only the anomaly-free training data. A different line of work additionally transfers the knowledge of networks trained on large scale and publicly available datasets for different computer

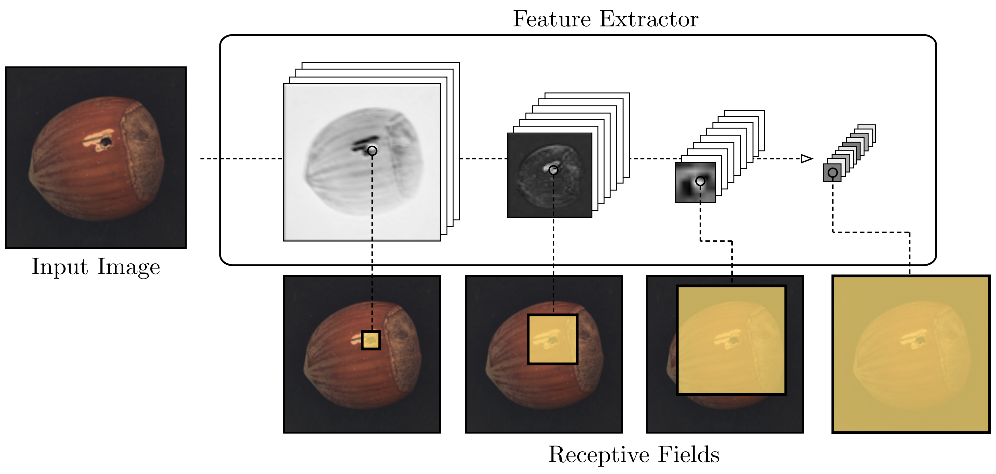
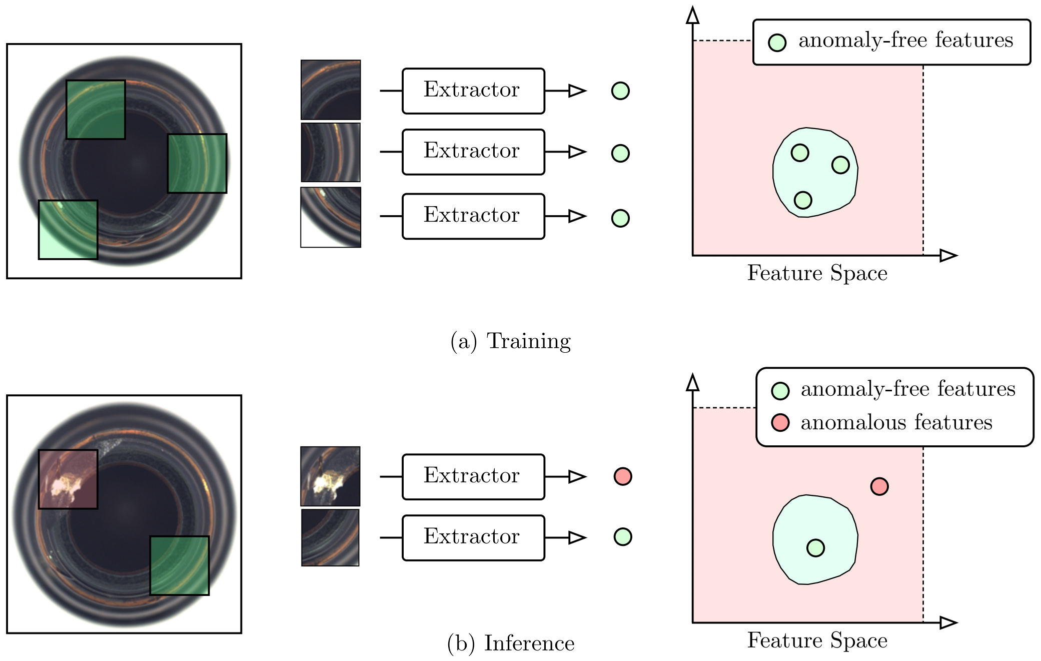


Figure 2.6: Features extracted from a ResNet-18 that was pretrained on the ImageNet dataset. The receptive field of the features is additionally visualized in yellow. The size of the receptive field grows with deeper network layers.

vision tasks. Examples for such datasets are ImageNet [Krizhevsky et al., 2012] for image classification or MS COCO [Lin et al., 2014] for object detection. Once trained, these models can serve as generic feature extractors that produce powerful descriptors that may be used in an anomaly detection system.

Figure 2.6 shows several feature maps extracted from an image of a defective hazelnut when passing it through a ResNet-18 classifier [He et al., 2016] pretrained on ImageNet. ResNet-18 is a popular network architecture used to address image classification problems. Although in principle the network produces a set of feature maps after each layer, here we only show the output of four different layers for simplicity. Since ResNet-18 downsamples the input data to a low-dimensional vector, the spatial dimension of the intermediate feature maps decreases with deeper layers. Simultaneously, the number of extracted feature channels increases. Each entry in a feature map describes the content of a local region within the input image. This region is referred to as the *receptive field*. Its size can be determined by computing the set of all pixels that can potentially influence the value of a feature. Typically, the receptive field grows continuously with deeper layers. This effect is also illustrated in Figure 2.6. Descriptors of early layers are only affected by a small region of the input. The receptive field grows significantly for deeper layers. At some point, it covers the entire input image.

To give an example of how features of pretrained networks may be used for unsupervised anomaly detection, we briefly discuss a method originally presented by Napoletano et al. [2018], named the CNN Feature Dictionary. As feature extractor, it uses a ResNet-18 pretrained for image classification. It maps an input image to a 512-dimensional feature vector. The training of the anomaly detection model is performed by cropping a large number of image patches from the anomaly-free training data at random locations. The patches are zoomed to the input size of the feature extractor and their respective



(a) Training

(b) Inference

Figure 2.7: Illustration of how feature extractors of pretrained networks can be used for unsupervised anomaly detection. (a) Randomly sampled patches of an anomaly-free image are passed through the feature extractor to compute deep descriptors. The feature distribution is modeled with a machine learning model. (b) During inference, image patches that cover anomalous regions produce descriptors that significantly deviate from the anomaly-free ones.

feature vectors are computed. The dimension of the vectors is reduced by applying Principal Component Analysis (PCA) [Hadsell et al., 2006]. In a last step, a K-Means classifier [Lloyd, 1982] is fitted to the extracted data to model the distribution of the anomaly-free descriptors. An overview of this training process is shown in Figure 2.7(a).

During inference, the general idea is that anomaly-free patches will produce descriptors that follow the distribution of the ones from the training data. Patches that contain anomalies, on the other hand, are likely to result in features that have not been observed so far. Hence, they are expected to be far from the cluster centers learned by the K-Means algorithm. An anomaly score for a patch can be derived by computing the minimum distance between its respective descriptor and all cluster centers. To obtain a spatially resolved anomaly map, many patches of a test image need to be evaluated, which is computationally expensive. Figure 2.7(b) gives an overview of the inference phase. Note that the K-Means classifier used in the CNN Feature Dictionary can be easily replaced by other distribution models, such as KNN classifiers [Fix and Hodges, 1989], One-Class Support Vector Machines (OC-SVM) [Schölkopf et al., 2001], or Gaussian Mixture Models (GMM) [Yu et al., 2012].

In Chapter 4, we demonstrate that anomaly detection methods that leverage features of pretrained networks tend to perform better than autoencoder- or GAN-based methods that are trained from scratch. In Chapter 5, we build on this finding and present a Student–Teacher framework for anomaly detection that leverages a pretrained teacher network that outputs descriptors in the form of entire feature maps. Each descriptor captures the content of a local region within the input image. For anomaly detection, an ensemble of student networks is trained on anomaly-free images to reproduce the descriptors of the pretrained teacher. During inference, anomalies are detected where the students fail to correctly predict the output of the teacher. Compared to the CNN Feature Dictionary that requires patch-based evaluations and a random subsampling of training descriptors, our method computes pixel-precise anomaly scores with a single forward pass and makes use of all available training features. Closely following this idea, Salehi et al. [2021] build on this work and train student networks to match feature maps of a single teacher at multiple resolutions.

Numerous other methods have been introduced that transfer descriptors extracted from pretrained networks to the anomaly detection task. Sabokrou et al. [2018] model the distribution of features from the first layers of a pretrained AlexNet with a unimodal Gaussian distribution. The fully convolutional architecture of the employed network allows for efficient feature extraction during training and inference. However, the use of pooling layers rapidly downsamples the input image and leads to a loss in resolution of the output anomaly map. Furthermore, it is challenging for unimodal Gaussian distributions to capture highly complex feature distributions.

Cohen and Hoshen [2020] introduce the SPADE method. They process a given test image by a pretrained network to extract a single feature vector. Then, a subset of the anomaly-free training images is selected. An image is included in the subset if it produces a descriptor that is similar to the one of the test image. Afterwards, spatially resolved feature maps are extracted from all images in the selected subset as well as for the test image under consideration. Anomaly scores are derived by calculating the minimum distance between the anomaly-free features and the test features in each pixel of the feature map.

Other approaches attempt to model the feature distribution with deep normalizing flows [Rudolph et al., 2021, 2022]. In particular, Marchal et al. [2020] show that using these high-capacity deep learning approaches can improve the performance over shallow distribution models.

2.4.4 Methods Not Based on Neural Networks

There are methods that do not make use of deep learning techniques at all. Instead, they operate directly on the raw pixel values of the input images or compute handcrafted features. An example of such a method is the Variation Model [Steger et al., 2018, Chapter 3.4.1.4]. It assumes a geometric alignment of an inspected object throughout the dataset images. It then computes the pixelwise means and standard deviations over all pixels of the training dataset. For images with a single channel this can be written as:

$$\mu(\boldsymbol{p}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i=1}^{|\mathcal{D}_{\text{train}}|} I_i(\boldsymbol{p}), \tag{2.21}$$

$$\sigma(\boldsymbol{p}) = \sqrt{\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i=1}^{|\mathcal{D}_{\text{train}}|} (I_i(\boldsymbol{p}) - \mu(\boldsymbol{p}))^2}. \tag{2.22}$$

Here, $\boldsymbol{p}$ denotes the image coordinate under consideration. If the variations of the training set are extremely small, the standard deviation is often clipped to a predefined value $a \in (0, \infty)$, i.e., by computing $\max(\sigma(\boldsymbol{p}), a)$.

During inference, the pixel values of a test image J are compared against the statistics of the training set using a Mahalanobis distance. An anomaly score for each pixel can be calculated as follows:

$$A(\boldsymbol{p}) = \frac{J(\boldsymbol{p}) - \mu(\boldsymbol{p})}{\sigma(\boldsymbol{p})}. \tag{2.23}$$

If an image contains multiple channels, a Variation Model can be constructed for each of the channels separately. The anomaly scores of the different channels can be combined using an appropriate aggregation function, e.g., by taking the mean or the maximum value over all channels.

A second example for a method that does not rely on deep learning is presented by Böttger and Ulrich [2016]. They extract hand-crafted feature descriptors from defect-free texture images in a compressed-sensing framework. Similar to the CNN Feature Dictionary, the feature vectors describe the content of local patch regions within the training images. The distribution of the training descriptors is then modeled by a Gaussian Mixture Model. During inference, anomalies are detected by evaluating the log-likelihood of the GMM on descriptors extracted from the test images. Low likelihoods indicate anomalous features. To obtain a spatially resolved anomaly map, a patch-sized window is slid across the input image and a likelihood is computed for each pixel. Since this approach was specifically introduced for the inspection of textured surfaces, we refer to this approach as the Texture Inspection Model.

3 Structural Similarity Autoencoder

Convolutional autoencoders have emerged as popular methods for unsupervised anomaly detection on image data. As described in the previous chapter, it is common to obtain anomaly scores by per-pixel comparisons between their input and reconstructed images. In this chapter, we show that this tends to produce many false positives in locations where the reconstruction is only slightly inaccurate, e.g., due to small localization imprecisions of edges. Furthermore, this way of computing anomaly scores performs poorly in the detection of salient differences between the input and reconstructed images when the respective pixels' intensity values are roughly consistent. This prevents such per-pixel measures from localizing anomalies that differ predominantly in structure rather than pixel intensity.

To alleviate these problems, we propose to compute anomaly scores using the structural similarity metric (SSIM) [Wang et al., 2004]. It is a distance measure designed to model a perceptual similarity that is less sensitive to edge alignment and gives importance to salient differences between the input and reconstruction. Our experiments on two real-world datasets show that our SSIM-Autoencoder significantly improves the anomaly localization accuracy over recent approaches for unsupervised anomaly detection that use per-pixel reconstruction error metrics. The content of this chapter is based on the publication *Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders* [Bergmann et al., 2019b].

3.1 Introduction

Autoencoders attempt to reconstruct their inputs in the presence of certain constraints, such as a bottleneck, and thereby manage to capture the essence of high-dimensional data in a lower-dimensional space. It is assumed that the model is unable to reproduce anomalies in the test data that deviate from the training data manifold. As a result, large reconstruction errors indicate anomalies. A brief overview of anomaly detection methods based on autoencoders is given in Section 2.4.1. All of them compute per-pixel errors between the input and the reconstructed images based on an L_p-distance. Baur et al. [2019] add an additional adversarial loss to enhance the visual quality of the reconstructed images. However, during inference, they still compute anomaly scores by a direct comparison of pixel intensities.

In this chapter, we discuss problems that arise when computing anomaly scores by such direct comparisons of individual pixel values. In particular, this approach performs poorly in the detection of defects that arise as structural alterations of the defect-free material where the individual pixel intensity values do not differ significantly. Furthermore, it tends to produce many false positive predictions, even if the reconstructed

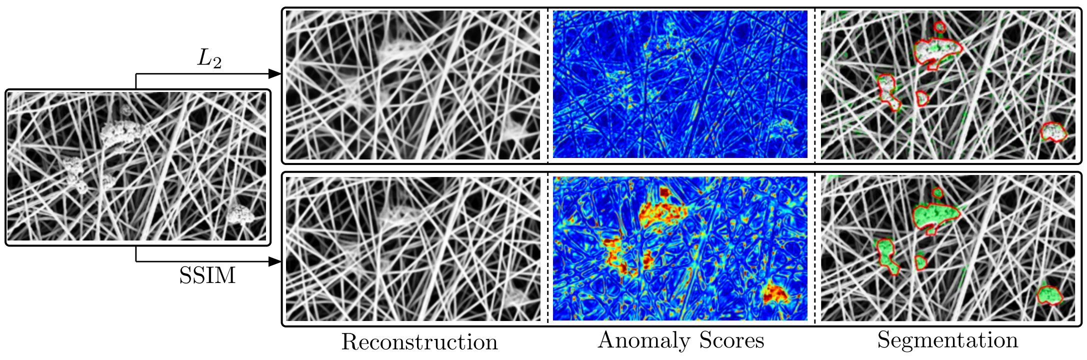


Figure 3.1: An anomalous image of nanofibrous materials is reconstructed by an autoencoder optimizing either the commonly used pixelwise L_2-distance or a perceptual similarity metric based on structural similarity (SSIM). Even though an L_2-Autoencoder fails to properly reconstruct the defects, a per-pixel comparison of the original input and reconstruction does not yield significant residuals that would allow for a successful defect segmentation. The anomaly map using SSIM puts more importance on the visually salient changes made by the autoencoder, enabling for an accurate segmentation of the defects.

images visually differ only slightly from the respective input images. This is because large reconstruction errors may arise already if the reconstruction is shifted by only a few pixels with respect to the original input.

The structural similarity index, on the other hand, computes a similarity measure by comparing local patch regions. Hence, it is less sensitive to small reconstruction inaccuracies. Furthermore, it computes three different statistical measures that model perceptual differences in luminance, contrast, and structure. This allows for the detection of salient differences between the input and the reconstruction in situations where per-pixel error functions yield low residuals. Snell et al. [2017] show that SSIM and the closely related multi-scale version MS-SSIM [Wang et al., 2003] can be used as differentiable loss functions to generate more realistic images in deep learning architectures for tasks such as super resolution. This motivates us to examine its usefulness for the detection of anomalies in an autoencoding framework.

In our experiments on two real-world industrial inspection datasets, switching from per-pixel to perceptual losses yields significant gains in performance. We reach a performance that is on par with other recent approaches for unsupervised anomaly detection that rely on additional model priors such as pretrained networks. Figure 3.1 demonstrates the advantage of perceptual loss functions over a per-pixel L_2-loss on the NanoTWICE dataset of nanofibrous materials [Carrera et al., 2017]. While both autoencoders alter the reconstruction in defective regions, only the anomaly map of the SSIM-autoencoder allows a segmentation of these areas.

3.2 Methodology

A brief introduction to autoencoders is given in Section 2.4.1. Here, we further detail their training and evaluation protocols when applied to anomaly detection problems. We then describe more sophisticated autoencoder variants, i.e., variational and feature matching autoencoders. Finally, we describe the SSIM metric and discuss its advantages over pixelwise error functions.

3.2.1 Autoencoders for Unsupervised Anomaly Detection

Autoencoders attempt to reconstruct an input image $I : D \to \mathbb{R}^C$ through a bottleneck, effectively projecting the input image into a lower-dimensional space, called latent space. An autoencoder consists of an encoder function, i.e., $\text{Enc} : \mathcal{I} \to \mathbb{R}^d$ and a decoder function, i.e., $\text{Dec} : \mathbb{R}^d \to \mathcal{I}$, where d denotes the dimensionality of the latent space. Choosing $d \ll C \times H \times W$ prevents the architecture from simply copying its input and forces the encoder to extract meaningful features from the input patches that facilitate an accurate reconstruction by the decoder. The overall process can be summarized as

$$\hat{I} = \text{Dec}(\text{Enc}(I)) = \text{Dec}(\boldsymbol{z}) \ , \tag{3.1}$$

where $\boldsymbol{z}$ is the latent vector and $\hat{I}$ the reconstruction of the input. In our experiments, the functions Enc and Dec are parameterized by CNNs. Strided convolutions are used to downsample the input feature maps in the encoder and to upsample them in the decoder. Autoencoders can be employed for unsupervised anomaly detection by training them exclusively on anomaly-free image data. During testing, the autoencoder will fail to reconstruct anomalous regions that have not been observed during training, which can thus be segmented by comparing the original input to the reconstruction and computing an anomaly map $A : D \to \mathbb{R}$.

L_2-Autoencoder

To force the autoencoder to reconstruct its input, a loss function must be defined that guides it towards this behavior. For simplicity and computational speed, one often chooses a per-pixel error measure as the loss function, such as the squared L_2-norm:

$$\mathcal{L}_{L_2}(\text{Enc}, \text{Dec}) = \frac{1}{B} \sum_{i=1}^{B} \|I_i - \hat{I}_i\|_2^2, \tag{3.2}$$

where $B \in \mathbb{N}^+$ denotes the number of samples in a training batch. To obtain an anomaly map A_{L_2} during evaluation for a test image J, the per-pixel L_2-distance of J and $\hat{J}$ is computed:

$$A_{L_2}(\boldsymbol{p}) = \|J(\boldsymbol{p}) - \hat{J}(\boldsymbol{p})\|_2^2. \tag{3.3}$$

Variational Autoencoder

Various extensions to the deterministic autoencoder framework exist. VAEs [Kingma and Welling, 2014] impose constraints on the latent variables to follow a certain prior distribution $\Pr(\boldsymbol{z})$. For simplicity, the distribution is typically chosen to be a unit-variance Gaussian. This turns the entire framework into a probabilistic model that enables efficient posterior inference and allows to generate new data from the training manifold by sampling from the latent distribution. The approximation to the posterior distribution $\Pr(\boldsymbol{z}|I)$ that is obtained by encoding an input image can be used to define further anomaly measures. One option is to compute a distance between the two distributions, such as the KL-divergence $\mathcal{KL}(\Pr(\boldsymbol{z}|I)||\Pr(\boldsymbol{z}))$, and indicate anomalies for large deviations from the prior $\Pr(\boldsymbol{z})$ [Soukup and Pinetz, 2018]. However, to use this approach for the pixel-accurate localization of anomalies, a separate forward pass for each pixel of the input image would have to be performed. A second approach for utilizing the posterior that yields a spatial anomaly map is to decode multiple latent samples $\boldsymbol{z}_1, \boldsymbol{z}_2, \dots$ drawn from $\Pr(\boldsymbol{z}|I)$ and to evaluate the per-pixel reconstruction probability $\Pr(I|\boldsymbol{z}_1, \boldsymbol{z}_2, \dots)$ as an anomaly score [An and Cho, 2015].

Feature Matching Autoencoder

Another extension to standard autoencoders was proposed by Dosovitskiy and Brox [2016]. It increases the quality of the produced reconstructions by extracting features from both the input image I and its reconstruction $\hat{I}$ and enforcing them to be equal. Consider $F : \mathcal{I} \to \mathbb{R}^f$ to be a feature extractor that obtains an f-dimensional feature vector from an input image. Then, a regularizer can be added to the loss function of the autoencoder, yielding the loss of the feature matching autoencoder (FM-AE):

$$\mathcal{L}_{\mathrm{FM}}(\mathrm{Enc}, \mathrm{Dec}) = \mathcal{L}_{L_2}(\mathrm{Enc}, \mathrm{Dec}) + \frac{\lambda}{B} \sum_{i=1}^{B} \|F(I_i) - F(\hat{I}_i)\|_2^2 \ . \tag{3.4}$$

Here, $\lambda \in (0, \infty)$ denotes the weighting factor between the two loss terms. F can be parameterized using the first layers of a CNN pretrained on an image classification task. During evaluation, an anomaly map is obtained by comparing the per-pixel L_2-distance of I and $\hat{I}$. The hope is that sharper, more realistic reconstructions will lead to better anomaly maps compared to a standard L_2-Autoencoder.

SSIM-Autoencoder

We show that employing more elaborate architectures such as VAEs or FM-AEs does not yield satisfactory improvements of the anomaly maps over deterministic L_2-Autoencoders in the unsupervised anomaly detection task. They are all based on per-pixel evaluation metrics that assume an unrealistic independence between neighboring pixels. Therefore, they fail to detect structural differences between the inputs and their reconstructions. By adapting the loss and evaluation functions to capture local inter-dependencies between image regions, we are able to significantly improve upon all the aforementioned

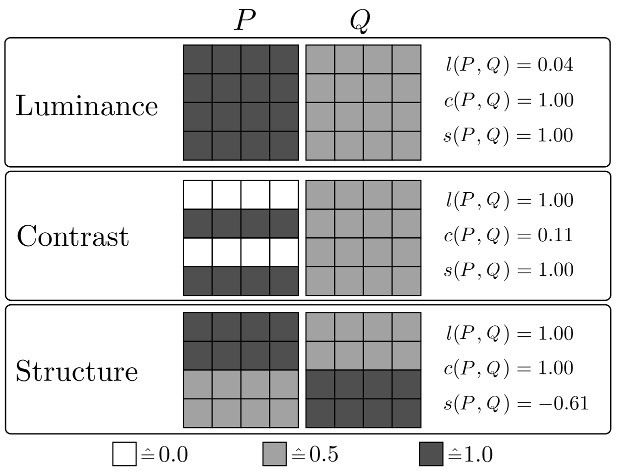


Figure 3.2: Different responsibilities of the three similarity functions employed by SSIM. Example patches P and Q differ in either luminance, contrast, or structure. SSIM is able to distinguish between these three cases, assigning close to minimum similarity values to one of the comparison functions $l(P, Q)$, $c(P, Q)$, or $s(P, Q)$, respectively. An L_2-comparison of these patches yields a constant per-pixel residual value of 0.25 for each of the three cases.

architectures. In the following section, we specifically motivate the use of the structural similarity metric as both the loss function and the evaluation metric for autoencoders to obtain an anomaly map.

3.2.2 Structural Similarity

The SSIM index [Wang et al., 2004] defines a distance measure between two image patches $P : \tilde{D} \to \mathbb{R}$ and $Q : \tilde{D} \to \mathbb{R}$. Each patch contains a single image channel and their domain is defined as $\tilde{D} = \{0, \ldots, K-1\} \times \{0, \ldots, K-1\}$ where $K \in \mathbb{N}^+$ denotes the patch size. SSIM takes into account the patches' similarity in luminance $l(P, Q)$, contrast $c(P, Q)$, and structure $s(P, Q)$:

$$\mathrm{SSIM}(P, Q) = l(P, Q)^\alpha c(P, Q)^\beta s(P, Q)^\gamma \ , \tag{3.5}$$

where $\alpha, \beta, \gamma \in \mathbb{R}$ are user-defined constants to weight the three terms. The luminance measure $l(P, Q)$ is estimated by comparing the patches' mean intensities μ_P and μ_Q. The contrast measure $c(P, Q)$ is a function of the patch variances σ_P^2 and σ_Q^2. The structure measure $s(P, Q)$ takes into account the covariance σ_{PQ} of the two patches. The three

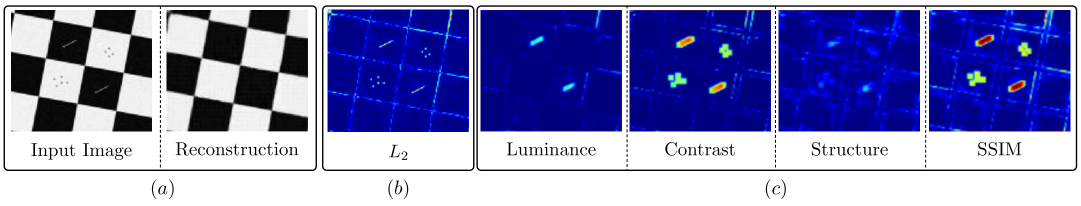


Figure 3.3: A toy example illustrating the advantages of SSIM over the L_2-distance for the segmentation of defects. (a) 128×128 checkerboard pattern with gray strokes and dots that simulate defects. Output reconstruction $\hat{I}$ of the input image I by an L_2-Autoencoder trained on defect-free checkerboard patterns. The defects have been removed by the autoencoder. (b) L_2-anomaly map. (c) Residuals for luminance l, contrast c, structure s, and their pointwise product that yields the final SSIM anomaly map. In contrast to the L_2-anomaly map, SSIM gives more importance to the visually more salient disturbances than to the slight inaccuracies around reconstructed edges.

measures are defined as:

$$l(P,Q) = \frac{2\mu_P\mu_Q + c_1}{\mu_P^2 + \mu_Q^2 + c_1}, \tag{3.6}$$

$$c(P,Q) = \frac{2\sigma_P\sigma_Q + c_2}{\sigma_P^2 + \sigma_Q^2 + c_2}, \tag{3.7}$$

$$s(P,Q) = \frac{2\sigma_{PQ} + c_2}{2\sigma_P\sigma_Q + c_2}. \tag{3.8}$$

The constants $c_1 \in (0,\infty)$ and $c_2 \in (0,\infty)$ ensure numerical stability. In our experiments, we set them to $c_1 = 0.01$ and $c_2 = 0.03$. By substituting (3.6)–(3.8) into (3.5) and setting $\alpha = \beta = \gamma = 1$, the SSIM is given by

$$\mathrm{SSIM}(P,Q) = \frac{(2\mu_P\mu_Q + c_1)(2\sigma_{PQ} + c_2)}{(\mu_P^2 + \mu_Q^2 + c_1)(\sigma_P^2 + \sigma_Q^2 + c_2)}. \tag{3.9}$$

It holds that $\mathrm{SSIM}(P,Q) \in [-1,1]$. In particular, $\mathrm{SSIM}(P,Q) = 1$ if and only if P and Q are identical [Wang et al., 2004]. Figure 3.2 shows the different perceptions of the three similarity functions that form the SSIM index. Each of the patch pairs P and Q has a constant L_2-residual of 0.25 per-pixel and hence assigns relatively low anomaly scores to each of the three cases. SSIM on the other hand is sensitive to variations in the patches' mean, variance, and covariance in its respective anomaly map and assigns low similarity to each of the patch pairs in one of the comparison functions.

To compute the structural similarity between an entire image I and its reconstruction $\hat{I}$, one slides a $K \times K$ window across the image and computes a SSIM value at each pixel location. Since Equation (3.9) is differentiable, it can be employed as a loss function in deep learning architectures that are optimized using gradient descent.

Figure 3.3 illustrates the advantages of SSIM over per-pixel error functions such as L_2 for localizing anomalies. After training an L_2-Autoencoder on defect-free checkerboard

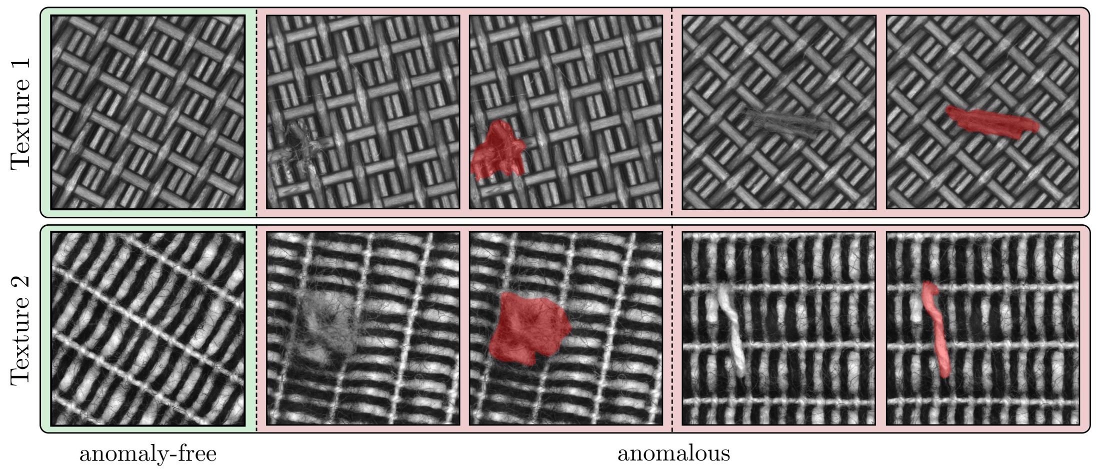


Figure 3.4: Example images from the contributed texture dataset of two woven fabrics. For each texture, the left sample shows an anomaly-free image without any defect that can be used for training. The right images show examples of anomalous textures. Their respective ground truth annotations are visualized in red.

patterns of various scales and orientations, we apply it to an image that contains gray strokes and dots that simulate defects. Figure 3.3(a) shows the input image and its corresponding reconstruction produced by the autoencoder, which removes the defects from the input image. The two remaining subfigures display the anomaly maps when evaluating the reconstruction error with a per-pixel L_2-comparison or SSIM. For the latter, the luminance, contrast, and structure maps are also shown. For the L_2-distance, both the defects and the inaccuracies in the reconstruction of the edges are weighted equally in the anomaly map, which makes them indistinguishable. Since SSIM computes three different statistical features for image comparison and operates on local patch regions, it is less sensitive to small localization inaccuracies in the reconstruction. In addition, it detects defects that manifest themselves in a change of structure rather than large differences in pixel intensity. For the defects added in this particular toy example, the contrast function yields the largest residuals.

3.3 Experiments

3.3.1 Datasets

At the time of conducting this research project, there was a significant shortage of datasets for unsupervised anomaly localization in industrial scenarios. One of the few datasets available was the NanoTWICE dataset for the inspection of nanofibrous materials presented by Carrera et al. [2017]. It contains five defect-free grayscale images of size 1024×700 for training and validation and 40 defective images for evaluation. A sample image of this dataset is shown in Figure 3.1.

Layer	Output Size	Parameters		
		Kernel	Stride	Padding
Input	128x128x1			
Conv1	64x64x32	4x4	2	1
Conv2	32x32x32	4x4	2	1
Conv3	32x32x32	3x3	1	1
Conv4	16x16x64	4x4	2	1
Conv5	16x16x64	3x3	1	1
Conv6	8x8x128	4x4	2	1
Conv7	8x8x64	3x3	1	1
Conv8	8x8x32	3x3	1	1
Conv9	1x1xd	8x8	1	0

Table 3.1: General outline of our autoencoder architecture. The depicted values correspond to the structure of the encoder. The decoder is built as a reversed version of this. Leaky rectified linear units with slope 0.2 are applied as activation functions after each layer except for the output layers of both the encoder and the decoder, in which linear activation functions are used.

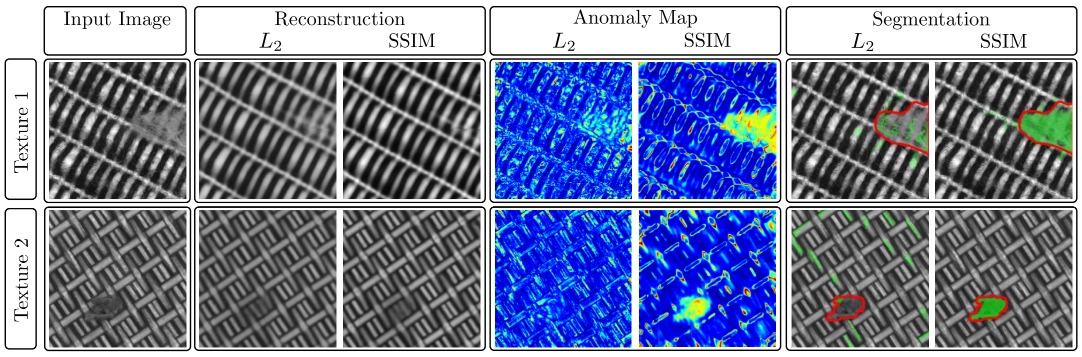


Figure 3.5: Qualitative comparison between reconstructions, anomaly maps, and segmentation results of an L_2-Autoencoder and an SSIM-Autoencoder on two datasets of woven fabric textures. The ground truth regions containing defects are outlined in red while green areas mark the segmentation result of the respective method.

For a more comprehensive assessment of the performance of our SSIM-Autoencoder, we contribute a novel dataset of two woven fabric textures, which is available to the public.[1] We provide 100 defect-free images per texture for training and validation and 50 images that contain various defects such as cuts, roughened areas, and contaminations on the fabric. Pixel-accurate ground truth annotations for all defects are also provided. All images are of size 512×512 pixels and were acquired as single-channel grayscale images. Examples of defective and defect-free textures are shown in Figure 3.4.

[1]`http://www.mvtec.com/company/research/publications`.

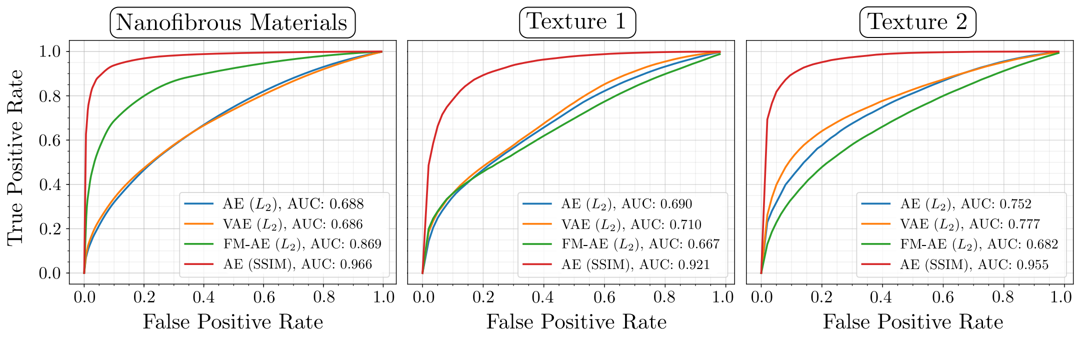


Figure 3.6: Resulting ROC curves of the proposed SSIM-Autoencoder (red line) on the evaluated datasets of nanofibrous materials and the two texture datasets in comparison with other autoencoding architectures that use per-pixel loss functions (green, orange, and blue lines). Corresponding AUC values are given in the legend.

3.3.2 Training and Evaluation Protocol

For all datasets, we train the autoencoders with their respective losses and evaluation metrics, as described in Section 3.2.1. Each architecture is trained on 10 000 defect-free patches of size 128×128, randomly cropped from the given training images. To capture a more global context of the textures, we down-scaled the images to size 256×256 before cropping. Each network is trained for 200 epochs using the ADAM [Kingma and Ba, 2015] optimizer with an initial learning rate of 2×10^{-4}, a weight decay set to 10^{-5}, and a batch size of $B = 64$. The exact parametrization of the autoencoder network shared by all tested architectures is given in Table 3.1. The latent space dimension for our experiments is set to $d = 100$ on the texture images and to $d = 500$ for the nanofibres due to their higher structural complexity. For the VAE, we decode six latent samples from the approximate posterior distribution $\Pr(\boldsymbol{z}|I)$ to evaluate the reconstruction probability for each pixel. The feature matching autoencoder is regularized with the first three convolutional layers of an AlexNet [Russakovsky et al., 2015] pretrained on ImageNet [Krizhevsky et al., 2012] and a weight factor of $\lambda = 1$. For SSIM, the window size is set to $K = 11$ unless mentioned otherwise.

The evaluation is performed by striding over the test images and reconstructing image patches of size 128×128 using the trained autoencoder and computing its respective anomaly map. In principle, it would be possible to set the horizontal and vertical stride to 128. However, at different spatial locations, the autoencoder produces slightly different reconstructions of the same data, which leads to some striding artifacts. Therefore, we decreased the stride to 30 pixels and averaged the reconstructed pixel values. The resulting anomaly maps are thresholded to obtain candidate regions where a defect might be present. An opening with a circular structuring element of diameter 4 is applied as a morphological post-processing to delete outlier regions that are only a few pixels wide [Steger et al., 2018]. We compute the area under the receiver operating characteristic (AU-ROC) as the evaluation metric. The true positive rate is defined as the ratio of pixels correctly classified as defect across the entire dataset. The false positive rate is the ratio of pixels misclassified as defect.

3.3.3 Results

Figure 3.5 shows a qualitative comparison between the performance of the L_2- and the SSIM-Autoencoder on images of the two texture datasets. Although both architectures remove the defect in the reconstruction, only the SSIM anomaly map reveals the defects and provides an accurate segmentation result. The same can be observed for the NanoTWICE dataset, as shown in Figure 3.1.

We confirm this qualitative behavior by numerical results. Figure 3.6 compares the ROC curves and their respective AUC values of our approach using SSIM to the per-pixel architectures. The anomaly localization performance of the latter is often only marginally better than classifying each pixel randomly. For the VAE, we found that the reconstructions obtained by different latent samples from the posterior does not vary greatly. Thus, it could not improve on the deterministic framework. Employing feature matching only improved the segmentation result for the dataset of nanofibrous materials, while not yielding a benefit for the two texture datasets. Using SSIM as the loss and evaluation metric outperforms all other tested architectures significantly. By merely changing the loss function, the achieved AUC improves from 0.688 to 0.966 on the dataset of nanofibrous materials, which is comparable to the numbers reported by Napoletano et al. [2018], where values of up to 0.974 are reported. In contrast to this method, autoencoders do not rely on any model priors such as handcrafted features or pretrained networks. For the two texture datasets, similar gains in performance are observed.

Since the dataset of nanofibrous materials contains defects of various sizes and smaller sized defects contribute less to the overall true positive rate when weighting all pixel equally, we further evaluated the overlap of each detected anomaly region with the ground truth for this dataset and report the p-quantiles for $p \in \{25\%, 50\%, 75\%\}$ in Figure 3.7. For false positive rates as low as 5%, more than 50% of the defects have an overlap with the ground truth that is larger than 91%. This outperforms the results achieved by [Napoletano et al., 2018], who report a minimal overlap of 85% in this setting.

We further tested the sensitivity of the SSIM-Autoencoder to different hyperparameter settings. We varied the latent space dimension d, SSIM window size K, and the size of the patches that the autoencoder was trained on. Table 3.2 shows that SSIM is insensitive to different hyperparameter settings once the latent space dimension is chosen to be sufficiently large. Using the optimal setup of $d = 500$, $k = 11$, and patch size 128×128, a forward pass through our architecture takes 2.23 ms on a Tesla V100 GPU. Patch-by-patch evaluation of an entire image of the NanoTWICE dataset takes 3.61 s on average, which is significantly faster than the runtimes reported by [Napoletano et al., 2018]. Their approach requires between 15 s and 55 s to process a single input image.

Figure 3.8 depicts qualitative advantages that employing a perceptual error metric has over per-pixel distances such as L_2. It displays two defective images from one of

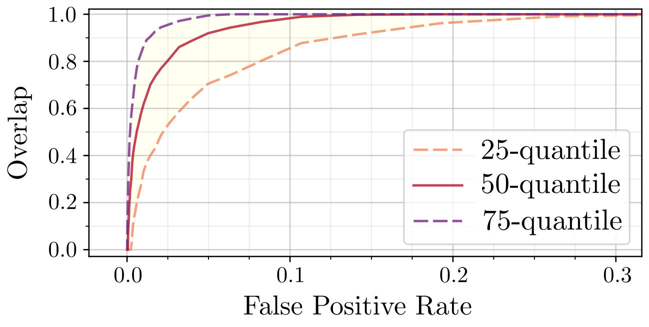


Figure 3.7: Per-region overlap for individual defects between our segmentation and the ground truth for different false positive rates using an SSIM-Autoencoder on the dataset of nanofibrous materials.

Latent dimension	AUC	SSIM window size	AUC	Patch size	AUC
50	0.848	3	0.889		
100	0.935	7	0.965	32	0.949
200	0.961	**11**	**0.966**	64	0.959
500	**0.966**	15	0.960	**128**	**0.966**
1000	0.962	19	0.952		

Table 3.2: Area under the ROC curve (AUC) on NanoTWICE for varying hyperparameters in the SSIM-Autoencoder architecture. Different settings do not significantly alter defect segmentation performance.

the texture datasets, where the top image contains a high-contrast defect of metal pins which contaminate the fabric. The bottom image shows a low-contrast structural defect where the fabric was cut open. While the L_2-norm has problems to detect the low-contrast defect, it easily segments the metal pins due to their large absolute distance in gray values with respect to the background. However, misalignments in edge regions still lead to large residuals in non-defective regions as well, which would make these thin defects hard to segment in practice. SSIM robustly segments both defect types due to its simultaneous focus on luminance, contrast, and structural information and insensitivity to edge alignment due to its patch-by-patch comparisons.

3.4 Conclusion

In this chapter, we demonstrated the advantage of perceptual loss functions over commonly used per-pixel residuals in autoencoding architectures when used for unsupervised anomaly segmentation tasks. Per-pixel losses fail to capture inter-dependencies between local image regions and therefore are of limited use when defects manifest themselves in structural alterations of the defect-free material where pixel intensity values do not differ significantly. We further show that employing probabilistic per-pixel error metrics

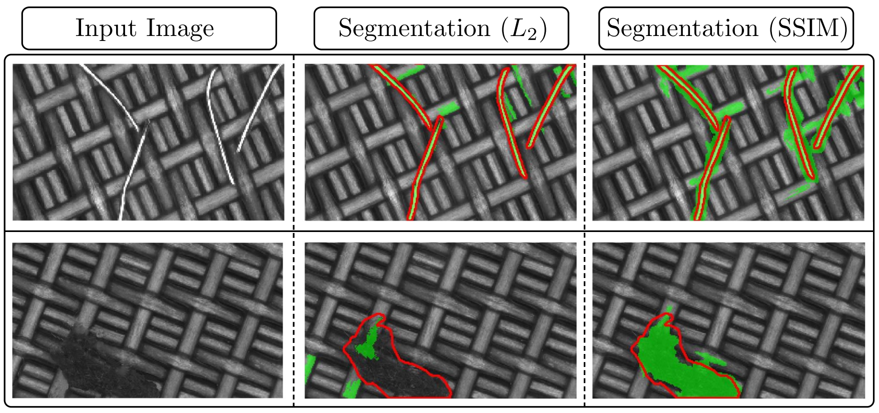


Figure 3.8: In the first row, the metal pins have a large difference in gray values in comparison to the defect-free background material. Therefore, they can be detected by both the L_2 and the SSIM error metric. The defect shown in the second row, however, differs from the texture more in terms of structure than in absolute gray values. As a consequence, a per-pixel distance metric fails to segment the defect while SSIM yields a good segmentation result.

obtained by VAEs or sharpening reconstructions by feature matching regularization techniques do not improve the segmentation result since they do not address the problems that arise from treating pixels as mutually independent.

SSIM, on the other hand, is less sensitive to small inaccuracies of edge locations due to its comparison of local patch regions and takes into account three different statistical measures: luminance, contrast, and structure. We demonstrate that switching from per-pixel loss functions to an error metric based on structural similarity yields significant improvements by evaluating on a challenging real-world dataset of nanofibrous materials and a contributed dataset of two woven fabric materials which we make publicly available. Employing SSIM often achieves an enhancement from almost unusable segmentations to results that are on par with other recent approaches for unsupervised anomaly detection which additionally rely on image priors such as pre-trained networks.

Although our experiments showed that using SSIM can significantly improve the anomaly localization performance of autoencoding architectures in some applications, there is still considerable room for future research. There are still anomalies that our method cannot localize, which requires the continued development of new, improved anomaly detection approaches. We have also found that there is a lack of datasets for anomaly detection. While our new texture dataset is a first step in addressing this gap, it covers only a small portion of the multitude of potential applications that arise in industrial inspection scenarios. Therefore, in the next chapter, we present a much more comprehensive dataset.

4 The MVTec Anomaly Detection Dataset

The development of methods for unsupervised anomaly detection requires data on which to train and evaluate new approaches and ideas. In this chapter, we introduce the MVTec Anomaly Detection (MVTec AD) dataset containing 5354 high-resolution color images of different object and texture categories. It contains normal, i.e., defect-free images intended for training and images with anomalies intended for testing. The anomalies manifest themselves in the form of over 70 different types of defects such as scratches, dents, contaminations, and various structural changes. In addition, we provide pixel-precise ground truth annotations for all anomalies. We conduct a thorough evaluation of recent unsupervised anomaly detection methods based on deep architectures such as convolutional autoencoders, generative adversarial networks, and feature descriptors using pretrained convolutional neural networks, as well as classical computer vision methods. This benchmark indicates that methods that leverage descriptors of pretrained networks perform best, but all evaluated models leave considerable room for improvement. The content of this chapter is based on two publications: *MVTec AD - A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection* [Bergmann et al., 2019a] and *The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection* [Bergmann et al., 2021]. The latter is a significant extension of the first article.

4.1 Introduction

In many areas of computer vision, large-scale datasets have led to incredible advances during the last few years. Consider how closely intertwined the development of new classification methods is with the introduction of datasets such as MNIST [LeCun et al., 1998], CIFAR10 [Krizhevsky and Hinton, 2009], or ImageNet [Krizhevsky et al., 2012]. For the task of unsupervised anomaly detection, we find that there is a lack of comprehensive, large-scale, high-resolution datasets. To fill this gap and to spark further research in the development of new methods in this area, we introduce the MVTec Anomaly Detection (MVTec AD or MAD for short) dataset[1] that facilitates a thorough evaluation of such methods. Some example images are shown in Figure 4.1. We identify industrial inspection tasks as an ideal and challenging real-world use case. Defect-free images of objects or textures are used to train a model that must determine whether an anomaly is present during test time. Unsupervised methods play a significant role here since it is often unknown beforehand which types of defects might occur during manufacturing. In addition, industrial processes are highly optimized to minimize the number

[1]`https://www.mvtec.com/company/research/datasets`.

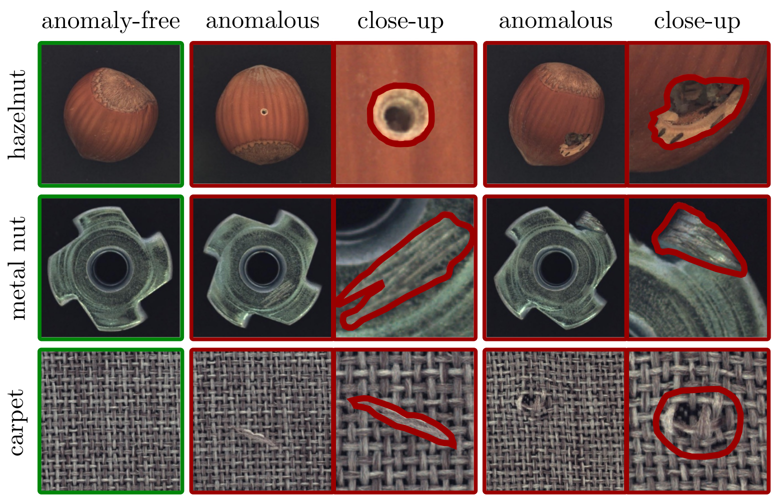


Figure 4.1: Two objects (*hazelnut* and *metal nut*) and one texture (*carpet*) from the MVTec Anomaly Detection dataset. For each of them, one defect-free image and two images that contain anomalies are displayed. Anomalous regions are highlighted in close-up figures together with their pixel-precise ground truth labels. The dataset contains objects and textures from several domains and covers various anomalies that differ in attributes such as size, color, and structure.

of defective samples. Therefore, only a very limited amount of images with defects is available, in contrast to a vast amount of defect-free samples that can be used for training. Ideally, methods should provide a pixel-accurate localization of anomalous regions. All this makes industrial inspection tasks perfect benchmarks for unsupervised anomaly detection methods that work on natural images. Our main contributions presented in this chapter are:

- We introduce a comprehensive dataset for the task of unsupervised anomaly detection in natural image data. It mimics real-world industrial inspection scenarios and consists of 5354 high-resolution images of five unique textures and ten unique objects from different domains. There are 73 different types of anomalies in the form of defects or structural deviations in the objects or textures. For each defect image, we provide pixel-accurate ground truth regions (1888 in total) that allow to evaluate methods for both anomaly classification and localization.

- We conduct a thorough evaluation of recent methods for unsupervised anomaly detection on the dataset. We show that the evaluated methods do not perform equally well across object and defect categories. On average, methods that leverage descriptors of pretrained networks perform best, but all evaluated methods leave considerable room for improvement.

- We provide a thorough discussion on various evaluation metrics and threshold estimation techniques for unsupervised anomaly localization and highlight their advantages and shortcomings. Our evaluations demonstrate the importance of selecting suitable metrics and show that threshold selection is a highly challenging task in practice. In addition, we include a discussion about the runtime and memory consumption of the evaluated methods. These are important criteria for the applicability of the benchmarked methods in real-world scenarios such as automated inspection tasks.

4.2 Existing Datasets for Anomaly Detection

We give a brief overview of datasets that are commonly used for anomaly detection in natural images and demonstrate the need for our new dataset. We distinguish between datasets that are designed for making a binary decision between anomalous and anomaly-free images and datasets that allow for the localization of anomalous regions.

4.2.1 Classification of Anomalous Images

When evaluating methods for anomaly detection in multi-class classification scenarios, a common practice is to adapt existing classification datasets for which class labels are already available. The most prominent examples are MNIST, CIFAR10, and ImageNet. A popular approach is to select an arbitrary subset of classes, re-label them as anomalies, and train an anomaly detection system solely on the remaining inlier classes [An and Cho, 2015, Chalapathy et al., 2018, Ruff et al., 2018, Burlina et al., 2019]. During the testing phase, it is checked whether the trained model is able to correctly predict that a test sample belongs to one of the inlier classes. An alternative approach is to train a classifier on all classes of a single dataset, e.g., MNIST, and use images of an entirely different dataset, e.g., notMNIST[2], as outliers. While these approaches immediately provide a large amount of data for training and testing, the anomalous samples differ significantly from the samples drawn from the training distribution. Therefore, when performing evaluations on such datasets, it is unclear how a proposed method would generalize to data where anomalies manifest themselves in less significant deviations from the training data manifold.

For this purpose, Saleh et al. [2013] propose a dataset that contains six categories of abnormally shaped objects, such as oddly shaped cars, airplanes, and boats, obtained from internet search engines, that should be distinguished from regular samples of the same class in the PASCAL VOC dataset [Everingham et al., 2015]. While this data might be closer to the training data manifold, the decision is again based on entire images rather than finding the parts of the images that make them novel or anomalous.

[2]`http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`

4.2.2 Localization of Anomalous Regions

For the evaluation of methods that segment anomalies in images, only few datasets are available to the public. Many of them are limited to the inspection of textured surfaces or focus on anomaly detection in multi-class semantic segmentation scenarios. The research community lacks comprehensive datasets that allow for the segmentation of anomalous regions in natural images where the anomalies manifest themselves in subtle deviations from the training data.

Carrera et al. [2017] provide NanoTWICE,[3] a dataset of 45 grayscale images that show a nanofibrous material acquired by a scanning electron microscope. Five defect-free images can be used for training. The remaining 40 images contain anomalous regions in the form of specks of dust or flattened areas. Since the dataset only provides a single kind of texture, it is unclear how well algorithms that are evaluated on this dataset generalize to other textures of different domains.

A dataset that is specifically designed for optical inspection of textured surfaces is proposed by Wieler and Hahn [2007]. They provide ten classes of artificially generated grayscale textures with defects weakly annotated in the form of ellipses. Each class comprises 1000 defect-free texture patches for training and 150 defective patches for testing. The annotations, however, are coarse and since the textures were generated by very similar texture models, the variance in appearance between the different textures is insignificant. Furthermore, artificially generated datasets can only be seen as an approximation to the real world.

Huang et al. [2018] introduce a surface inspection dataset of magnetic tiles. It contains 1344 grayscale images of a single texture. Each image is either anomaly-free or contains one of five different surface defects, such as cracks or uneven areas. For each defective image, pixel-precise ground-truth labels are provided. Similarly, Song and Yan [2013] introduce a database of 1800 grayscale images of a single steel surface. It contains six different defect types, such as scratches or surface crazings. Each defect is coarsely annotated with a bounding box.

Blum et al. [2019] introduce Fishyscapes, a dataset intended to benchmark semantic segmentation algorithms with respect to their ability to detect out-of-distribution inputs. They artificially inserted images of novel objects into images of the Cityscapes dataset [Cordts et al., 2016], for which pixel-precise annotations are available. The task is then to train a model for semantic segmentation while at the same time being able to identify certain objects as novelties by leveraging the model's per-pixel uncertainties. In contrast to their dataset, we focus on the one-class setting, where dataset images only show a single object and no training annotations are available. Furthermore, our anomalies manifest themselves in subtle deviations from the input images rather than showing entirely different object classes.

The CAOS (Combined Anomalous Object Segmentation) benchmark introduced by Hendrycks et al. [2019] provides two datasets similar to Fishyscapes. It consists of the StreetHazards and BDD-Anomaly datasets. StreetHazards contains artificially rendered driving scenes with inserted foreign objects. BDD-Anomaly also consists of driving

[3] `http://www.mi.imati.cnr.it/ettore/NanoTWICE/`

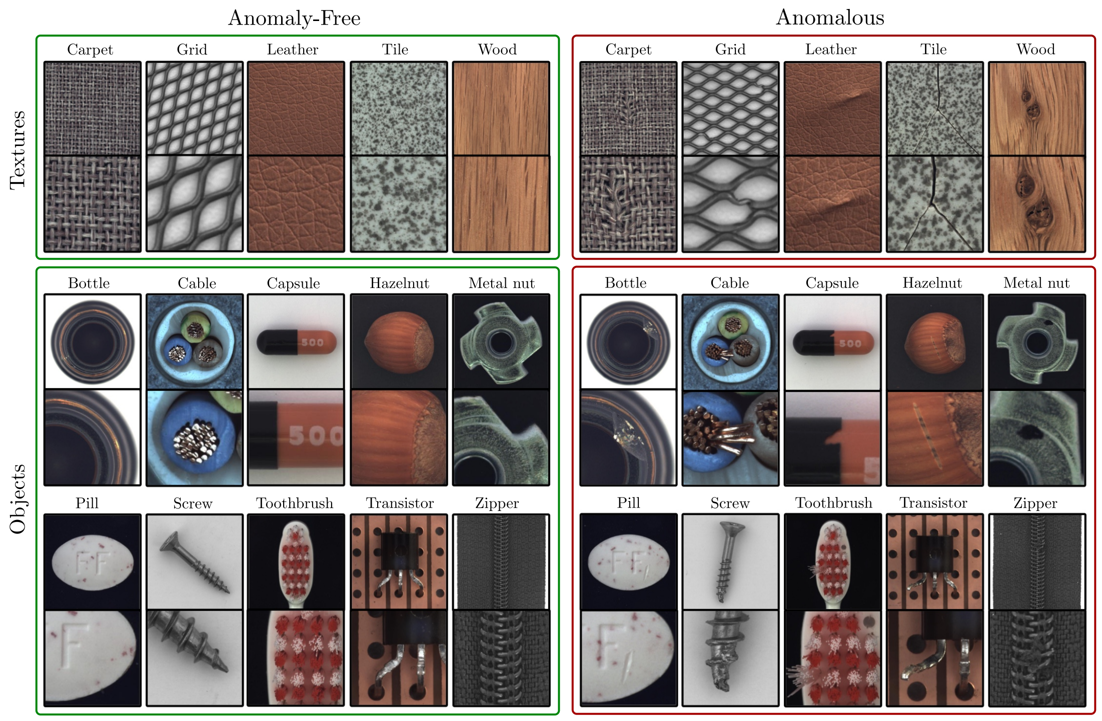


Figure 4.2: Example images for all five textures and ten object categories of the MVTec Anomaly Detection dataset. For each category, an anomaly-free as well as an anomalous example is shown. The top row shows the entire input image. The bottom row gives a close-up view. For anomalous images, the close-up highlights the anomalous regions.

scenes and was derived from the BDD100K dataset [Yu et al., 2020] by selecting two classes as anomalous and removing images containing these classes from the training and validation sets. As in the case of Fishyscapes, the CAOS datasets are geared towards a multi-class setting.

4.3 Description of the Dataset

The MVTec Anomaly Detection dataset comprises 15 categories with 3629 images for training and 1725 images for testing. The training set contains only images without defects. The test set contains both: images containing various types of defects and defect-free images. Table 4.1 gives an overview for each object category. Some example images for every category together with an example defect are shown in Figure 4.2. Five categories cover different types of regular (*carpet*, *grid*) or random (*leather*, *tile*, *wood*) textures, while the remaining ten categories represent various types of objects. Some of these objects are rigid with a fixed appearance (*bottle*, *metal nut*), while others are deformable (*cable*) or include natural variations (*hazelnut*). A subset of objects was acquired in a roughly aligned pose (e.g., *toothbrush*, *capsule*, and *pill*) while others

	Category	# Train	# Test (good)	# Test (defective)	# Defect types	# Defect regions	Image side length	Grayscale
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 4.1: Statistical overview of the MVTec AD dataset. For each category, the number of training and test images is given together with additional information about the defects present in the respective test images.

were placed in front of the camera with a random rotation (e.g., *metal nut*, *screw*, and *hazelnut*). The test images of anomalous samples contain a variety of defects, such as defects on the objects' surface (e.g., *scratches*, *dents*), structural anomalies like distorted object parts, or defects that manifest themselves by the absence of certain object parts. In total, 73 different defect types are present, on average five per category. The defects were manually generated with the aim to produce realistic anomalies as they would occur in real-world industrial inspection scenarios. They greatly vary in size, as shown in a box-and-whisker plot [Tukey, 1977] in Figure 4.3.

All images were acquired using a 2048 $\times$ 2048 pixel high-resolution industrial RGB sensor in combination with two bilateral telecentric lenses [Steger et al., 2018, Chapter 2.2.4.2] with magnification factors of 1:5 and 1:1, respectively. Afterwards, the images were cropped to a suitable output size. All image resolutions are in the range between 700 $\times$ 700 and 1024 $\times$ 1024 pixels. Each dataset image shows a unique physical sample. We did not augment images by taking multiple pictures of the same object in different poses. Since grayscale images are also common in industrial inspection, three object categories (*grid*, *screw*, and *zipper*) are made available solely as single-channel images. The images were acquired under controlled illumination conditions. For some object classes, however, the illumination was altered intentionally to increase variability. We provide pixel-precise ground truth labels for each defective image region. In total, the dataset contains 1888 anomalous regions. All regions were carefully annotated and reviewed multiple times. During the acquisition of the dataset, we generated defects that are confined to local regions, which facilitated a precise labeling of each anomaly. Additionally, pixels on the border of anomalies or lying in ambiguous regions were preferably labelled as anomalous. For locally deformed objects, annotations were created on the deformed area as well as in the region where the deformed object part is expected to be located. Some defects manifest themselves as missing parts. In these cases, we annotated the expected location

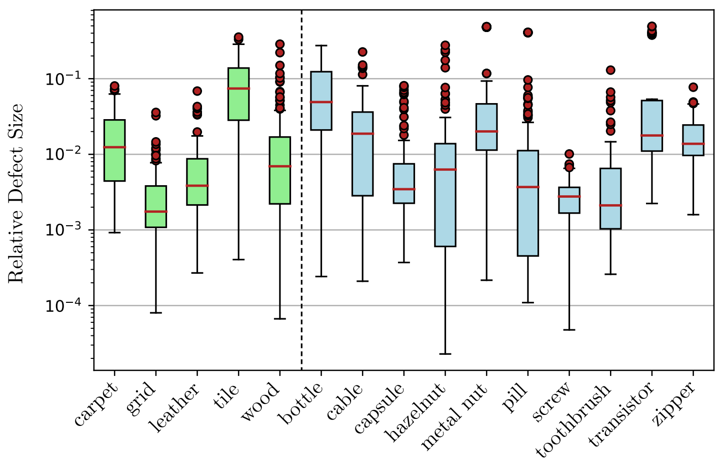


Figure 4.3: Size of anomalies for all textures (green) and objects (blue) in the dataset on a logarithmic scale visualized as a box-and-whisker plot with outliers. Defect areas are reported as the number of pixels within a connected component relative to the total number of pixels within an image. Anomalies vary greatly in size for each dataset category.

of the part as anomalous. Some examples of labels for selected anomalous images are displayed in Figures 4.1 and 4.7.

4.4 Performance Metrics

Assessing the performance of anomaly detection algorithms is challenging. A variety of different metrics are available and each comes with its own advantages and disadvantages. In Section 2.2, we give an overview of the most commonly used performance measures. Threshold independent metrics such as the AU-ROC or the AU-PRO are particularly popular, since determining a suitable threshold is often difficult in practice. Considering these measures for anomaly localization, it is important to note that the test split of our dataset is highly imbalanced in the sense that the number of anomalous pixels is significantly smaller than the number of anomaly-free ones. Only 2.7% of all pixels in the test set are labeled as anomalous. Therefore, thresholds that yield a large FPR result in segmentation results that are no longer meaningful. This is especially the case for industrial applications. There, large false positive rates would lead to a large amount of defect-free parts being wrongly rejected. An example is shown in Figure 4.4, where segmentation results are given for multiple thresholds as a contour plot. The thresholds were selected such that they result in different false positive rates on the input image, ranging from 1%, for which the defect is well detected, to 100%, where the entire image is segmented as anomalous. For FPRs as low as 30% the segmentation result is already degenerated. Therefore, we include metrics in our evaluations that compute the area under the curves only up to a certain false positive rate. To ensure that the maximum attainable values of this performance measure is equal to 1, we normalize the resulting

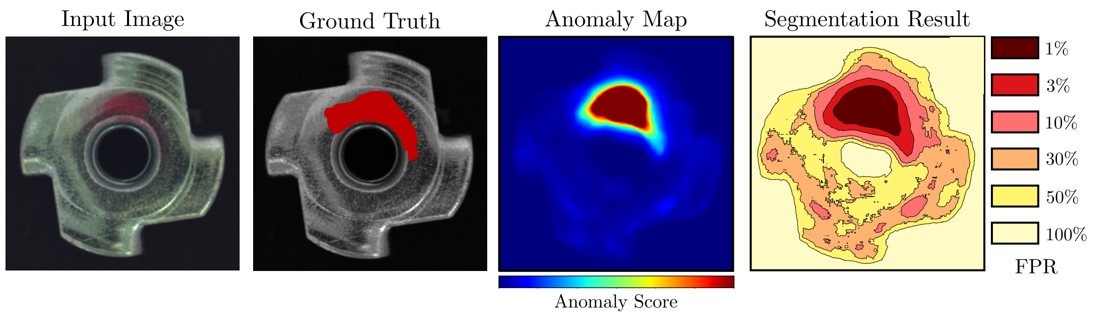


Figure 4.4: Example anomaly map for an anomalous input image of class *metal nut*. Binary segmentation results for multiple thresholds are shown as a contour plot. The thresholds are selected such that a certain false positive rate is achieved on the input image. Due to the large class-imbalance between anomalous and anomaly-free pixels, only results at relatively low FPR yield a satisfactory segmentation of the color defect.

area. Since the PR curve does not use the FPR in its computation, we always evaluate its entire area.

4.5 Threshold Selection

Evaluating anomaly localization algorithms using threshold-independent metrics such as measuring the area under a curve entirely circumvents the need for picking a suitable threshold. However, when employing an algorithm in practice, one must ultimately decide on a threshold value to determine whether a part is classified as defective or not. This is a challenging problem due to the lack of anomalous samples during training time. Even if a small number of anomalous samples was available for threshold estimation, we still consider it preferable to estimate a threshold solely on anomaly-free data. This is because there is no guarantee that the provided samples cover the entire range of possible anomalies and the estimated threshold might perform poorly for other, unknown, types. Instead, we want to find a threshold that separates the distribution of anomaly-free data from the rest of the entire data manifold such that even subtle deviations can be detected.

In this chapter, we consider three threshold estimation techniques for anomaly localization where the thresholds are estimated solely on a set of anomaly-free validation images $\mathcal{D}_{\text{val}}$ prior to testing. In our experiments, we evaluated how well each technique transfers from the validation to the test set and which performance is ultimately achieved when selecting these particular thresholds.

Maximum Threshold: In theory, a method should classify all pixels of the validation images as anomaly-free. To achieve this, one can simply select the threshold to equal the maximum value of all occurring anomaly scores on the validation set. In practice, this is often a highly conservative estimate since already a single outlier pixel with a large anomaly score can lead to thresholds that do not perform well on the test set.

p-Quantile Threshold: To make the estimation more robust against outliers, one can compute a threshold taking the entire distribution of validation anomaly scores into account and allowing for a certain amount of outlier pixels. Here, we investigate the p-quantile, which selects a threshold such that a percentage p of validation pixels is classified as anomaly-free.

k-Sigma Threshold: A third approach is to first compute the mean μ and standard deviation σ over all anomaly scores of the validation set, and then define a threshold to be $t = \mu + k\sigma$. This additionally takes the spread of the distribution of anomaly scores into account. If this distribution can be assumed to perfectly follow a Gaussian distribution, k can also be chosen to achieve a certain false positive rate on the validation set. However, since in practice this might not be the case, the false positive rate on the validation set may differ.

Max-Area Threshold: All estimators discussed so far compute thresholds simply on the one-dimensional distribution of validation anomaly scores and do not take the spatial location of image pixels into account. In particular, they are insensitive to the size of false positive regions, as many small regions are treated equally to a single larger one. In applications where only anomalies of a certain minimum size are expected, one can leverage this information to filter such small false positive regions and determine a threshold by permitting connected components on the validation images that do not exceed a predefined maximum permissible area. This ensures that an anomaly detector that classifies connected components as anomalous based on their area would not detect a single defect on the validation images.

4.6 Benchmark

We conduct a thorough evaluation of recent methods for unsupervised anomaly localization on our dataset. It is intended to serve as a baseline for future methods. In particular, we include a standard L_2-Autoencoder as well as the SSIM-Autoencoder introduced in Chapter 3. We further evaluate f-AnoGAN [Schlegl et al., 2019] and the CNN Feature Dictionary [Napoletano et al., 2018] as representatives of methods based on GANs and pretrained CNNs, respectively. We also consider two traditional methods for our benchmark. The first is the GMM-based Texture Inspection model presented by Böttger and Ulrich [2016]. While their algorithm was originally intended to be applied to images of regular textures, it can also be applied to the objects of our dataset. The second method is the Variation Model [Steger et al., 2018, Chapter 3.4.1.4]. A brief discussion of all evaluated methods can be found in Section 2.4.

We then discuss the strengths and weaknesses of each method on the various objects and textures of the dataset. We show that, while each method can detect anomalies of certain types, none of the evaluated methods excels on the entire dataset. We find that the CNN Feature Dictionary which leverages features of a pretrained network tends to perform better than other approaches. However, all methods leave room for improvement.

We assess the effect of different performance metrics on the evaluation result and compare different threshold estimation techniques. Furthermore, we provide information on inference time and memory consumption for each evaluated method.

4.6.1 Training and Evaluation Protocol

The following paragraphs list the training and evaluation protocols for each method. For each dataset category, we randomly split 10% of the anomaly-free training images into a validation set. The same validation set was used for all evaluated methods.

Fast AnoGAN: For the evaluation of fast AnoGAN (f-AnoGAN), we use the publicly available implementation by the original authors.[4] The GAN's latent space dimension is set to 128 and generated images are of size 64 × 64 pixels, which results in a relatively stable training for all categories of the dataset. GAN training is conducted for 100 epochs using the Adam optimizer with an initial learning rate of 10^{-4} and a batch size of 64. The encoder network for fast inference is trained for 50 000 iterations with an initial learning rate of 5×10^{-5} and batch size of 64. Since the implementation of fast AnoGAN only operates on single-channel images, all input images are converted to grayscale beforehand.

Anomaly maps are obtained by a per-pixel L_2-comparison of the input image with the generated output. For all evaluated dataset categories, training, validation and testing images are zoomed to size 256 × 256 pixels. 50 000 training patches of size 64 × 64 pixels are randomly cropped from the training images. During testing, a patchwise evaluation is performed with a horizontal and vertical stride of 64 pixels.

L_2**- and SSIM-Autoencoder:** For the evaluation of the L_2- and SSIM-Autoencoder, we build on the same network architecture that was described in Chapter 3. It reconstructs patches of size 128 × 128, employing either a per-pixel L_2-loss or a loss based on the structural similarity index (SSIM). Here, we extend the architecture by an additional convolution layer to process images at resolution 256 × 256. We find an SSIM window size of 11 × 11 pixels to work well in our experiments. The latent space dimension is chosen to be 128. Larger latent space dimensions do not yield significant improvements in reconstruction quality while lower dimensions lead to degenerate reconstructions. Training is run for 100 epochs using the Adam optimizer with an initial learning rate of 2×10^{-4} and a batch size of 128.

For each dataset category, 10 000 training samples are augmented from the train split of the original dataset. For textures, randomly sampled patches are cropped evenly across the training images. For objects, we apply a random translation and rotation to

[4]`https://github.com/tSchlegl/f-AnoGAN`

the entire input image and zoom the result to match the autoencoder's input resolution. Additional mirroring is applied where the object permits it.

For the dataset objects, anomaly maps are generated by passing an image through the autoencoder and comparing the reconstruction with its respective input using either per-pixel L_2-comparisons or SSIM. For textures, we reconstruct patches at a stride of 64 pixels and average the resulting anomaly maps. Since SSIM does not operate on color images, for the training and evaluation of the SSIM-Autoencoder all images are converted to grayscale.

Feature Dictionary: We use our own implementation of the CNN Feature Dictionary proposed by [Napoletano et al., 2018], which extracts features from the 512-dimensional average pooling layer of a ResNet-18 pretrained on ImageNet. Principal Component Analysis (PCA) is performed on the extracted features to explain 95% of the variance. K-means is run with 50 cluster centers and the nearest descriptor to each center is stored as a dictionary vector. We extract 100 000 patches of size 128×128 for both the textures and objects. All images are evaluated at their original resolution. A stride of 8 pixels is chosen to create a spatially resolved anomaly map. For grayscale images, the channels are triplicated for feature extraction since the used ResNet-18 operates on three-channel input images.

GMM-Based Texture Inspection Model: For the Texture Inspection Model by Böttger and Ulrich [2016], an optimized implementation is available in the HALCON machine vision library.[5] Images are converted to grayscale, zoomed to an input size of 400×400 pixels, and a four-layer image pyramid is constructed for training and evaluation. On each pyramid level, a separate GMM with dense covariance matrix is trained. The patch size of examined texture regions on each pyramid level is set to 7×7 pixels. We use a maximum of 50 randomly selected images from the original training set for training the Texture Inspection Model. Anomaly maps for each pyramid level are obtained by evaluating the negative log-likelihood for each image pixel using the corresponding trained GMM. We normalize the anomaly scores of each level such that the mean score is equal to 0 and their standard deviation equal to 1 on the validation set. The different levels are then combined to a single anomaly map by averaging the four normalized anomaly scores per pixel position.

Variation Model: To create the Variation Model [Steger et al., 2018, Chapter 3.4.1.4], we use all available training images of each dataset category in their original size and calculate the mean and standard deviation at each pixel location. This works best if the images show aligned objects. Since this is not always the case, we implemented a specific alignment procedure for our experiments on the following six dataset categories. *Bottle* and *metal nut* are aligned using shape-based matching [Steger, 2001, 2002], *grid* and *transistor* using template matching with normalized cross-correlation as the similarity measure [Steger et al., 2018, Chapter 3.11.1.2]. *Capsule* and *screw* are segmented via

[5]`https://www.mvtec.com/products/halcon`

Category	f-AnoGAN	Feature Dictionary	L_2-Autoencoder	SSIM-Autoencoder	Texture Inspection	Variation Model
Carpet	0.253	**0.895**	0.306	0.392	0.855	0.165
Grid	0.626	0.757	0.798	0.847	**0.857**	0.545
Leather	0.584	0.819	0.519	0.389	**0.981**	0.394
Tile	0.252	**0.873**	0.251	0.166	0.472	0.425
Wood	0.517	0.778	0.520	0.530	**0.827**	0.455
Bottle	0.440	**0.906**	0.567	0.703	0.636	0.659
Cable	0.428	**0.815**	0.507	0.368	0.597	0.405
Capsule	0.447	0.791	0.771	0.830	**0.834**	0.802
Hazelnut	0.872	0.913	0.922	0.897	**0.958**	0.849
Metal nut	0.482	**0.701**	0.607	0.501	0.384	0.562
Pill	0.700	**0.872**	0.847	0.803	0.606	0.834
Screw	0.808	0.725	0.864	**0.875**	0.864	0.701
Toothbrush	0.809	0.718	**0.891**	0.841	0.786	0.774
Transistor	0.494	0.590	**0.657**	0.602	0.542	0.554
Zipper	0.202	0.897	0.457	0.515	**0.923**	0.221
Mean	0.528	**0.803**	0.632	0.617	0.741	0.556

Table 4.2: Normalized area under the PRO curve up to an average false positive rate per-pixel of 30% for each dataset category.

thresholding and then aligned by using a rigid transformation which is determined by geometric features of the segmented region.

The anomaly map for a test image is obtained as follows. We define the value of each pixel in the anomaly map by calculating the distance from the gray value of the corresponding test pixel to the trained mean value and divide this distance by a multiple of the trained standard deviation. For multichannel images, this process is done separately for each channel and we obtain an overall anomaly map as the pixelwise maximum of all the channels' individual maps. Note that when a spatial transformation is applied to input images during inference, some input pixels might not overlap with the mean and deviation images. For such pixels, no meaningful anomaly score can be computed. In our evaluation, we set the anomaly score for such pixels to the minimum attainable value of 0. As for the GMM-based Texture Inspection, we use the optimized implementation of the HALCON machine vision library.

4.6.2 Anomaly Localization Results

We begin by comparing the performance of all methods for different threshold independent evaluation metrics, followed by an analysis of each method individually. The computation of curve areas that involve the false positive rate is performed up to an FPR of 0.3 if not mentioned otherwise.

Table 4.2 shows the area under the PRO curve for each method and dataset category. The CNN Feature Dictionary, which leverages pretrained feature extractors, has the best mean performance averaged over all dataset categories. The generative deep learning methods that are trained from scratch perform significantly worse, often only performing on par or inferior to the more traditional approaches, i.e., the Variation Model and the

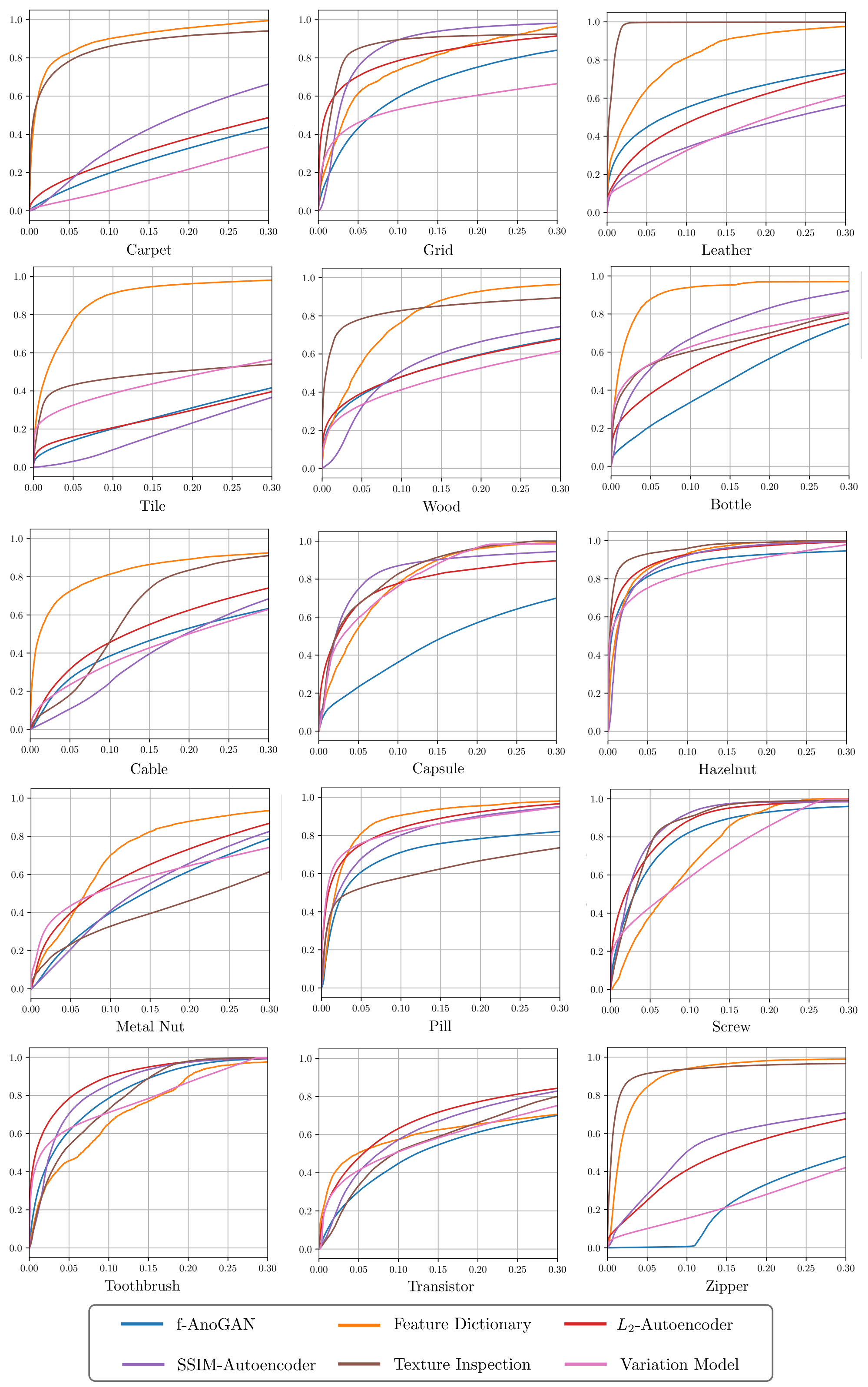


Figure 4.5: PRO curves for each dataset category and all evaluated methods. The per-region overlap (y-axis) is plotted against false positive rates up to 30% (x-axis).

Metric	f-AnoGAN	Feature Dictionary	L_2-Autoencoder	SSIM-Autoencoder	Texture Inspection	Variation Model
AU-PR	0.136 (6)	**0.466 (1)**	0.241 (3)	0.148 (5)	0.299 (2)	0.234 (4)
AU-ROC	0.472 (6)	**0.836 (1)**	0.590 (3)	0.584 (4)	0.656 (2)	0.526 (5)
AU-PRO	0.528 (6)	**0.803 (1)**	0.632 (3)	0.617 (4)	0.741 (2)	0.556 (5)
AU-IoU	0.073 (6)	**0.168 (1)**	0.099 (3)	0.091 (5)	0.100 (2)	0.095 (4)
AU-PRO$_{0.01}$	0.113 (5)	0.201 (3)	0.218 (2)	0.075 (6)	**0.263 (1)**	0.197 (4)
AU-PRO$_{0.05}$	0.249 (6)	0.459 (2)	0.372 (3)	0.279 (5)	**0.488 (1)**	0.328 (4)
AU-PRO$_{1.00}$	0.784 (6)	**0.931 (1)**	0.838 (4)	0.840 (3)	0.890 (2)	0.796 (5)

Table 4.3: Comparison of threshold independent performance metrics. For each metric and evaluated method, the normalized area under the curve is computed and averaged across all dataset categories. The ranking of each method with respect to the evaluated metric is given in brackets. For the ROC, PRO and IoU curves, the area is computed up to an FPR of 30%. The AU-PRO metric is additionally reported for varying integration limits.

GMM-based Texture Inspection. On this dataset, the average performance of the SSIM-Autoencoder is slightly below that of the L_2-Autoencoder. This is likely because the SSIM-Autoencoder does not leverage color information and operates on grayscale images only. For each object and method, the corresponding PRO curves are given in Figure 4.5.

Table 4.3 assesses the influence of different performance metrics on the evaluation result. The mean area under the ROC, PRO, PR, and IoU curves are given for each evaluated method. Areas are averaged over all dataset categories. Additionally, the area under the PRO curve is computed up to three different integration limits: 0.01, 0.05, and 1.0. For each method, its ranking with respect to the current metric and all other methods is given in brackets. When integrating up to a false positive rate of 30% (first four rows), the rankings produced by the investigated metrics are fairly consistent and the CNN Feature Dictionary performs best. f-AnoGAN performs worst for all evaluated metrics. When varying the integration limit of the false positive rate for the AU-PRO metric (last three rows), the ranking of some methods changes significantly. For example, when evaluating the full area under the PRO curve, the CNN Feature Dictionary ranks first, while it ranks only third place if the area is computed only up to an FPR of 1%. This highlights that the choice of the integration limit is important for metrics that involve the false positive rate and one must select it carefully depending on the requirements of the application. For tasks where low false positive rates are crucial, sufficiently small integration limits may be preferred over larger ones.

Table 4.3 further shows that when decreasing the integration limit of the FPR, the area under the PRO curve drops for all methods by more than a factor of 3. This shows that many methods only manage to detect anomalies when at the same time a significant amount of false positive pixels are allowed in the segmentation result. This might limit the applicability of these methods in practice, as illustrated in Figure 4.4. Figure 4.6 shows example curves for all evaluated metrics for the dataset category *zipper*. For this object, defect sizes do not vary as much as for other dataset categories (Figure 4.3), and hence the ROC and PRO curves are similar. The PR curve shows that the precision

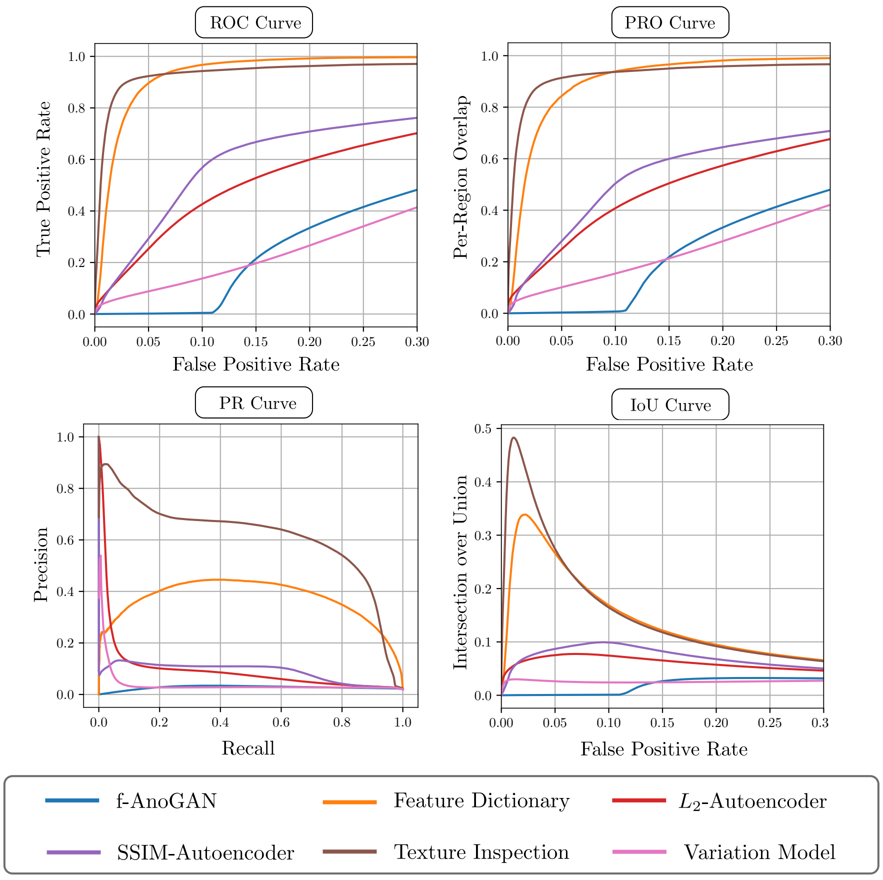


Figure 4.6: Comparison of performance curves for the dataset category *zipper*.

of all methods except the Texture Inspection Model is smaller than 0.5 for most recall values. This indicates that these methods predict more false positive pixels than true positives for any threshold. Compared to the precision, the IoU additionally takes the false negative predictions into account. Therefore, the IoU is bounded by the maximum attained precision value and methods with low overall precision also yield low IoU values for any threshold. For large false positive rates, the IoU converges towards the ratio of the number of ground truth anomalous pixels divided by the total number of pixels in the evaluated dataset.

Figure 4.7 shows an example for each method where anomaly detection worked well, i.e., the thresholded anomaly map substantially overlaps with the ground-truth (left column) and where each method produced an unsatisfactory result (right column). Anomaly scores were thresholded such that an average FPR of 0.01 across the entire test set is achieved. Based on the selected images, we now discuss individual properties of each evaluated method when applied to our dataset.

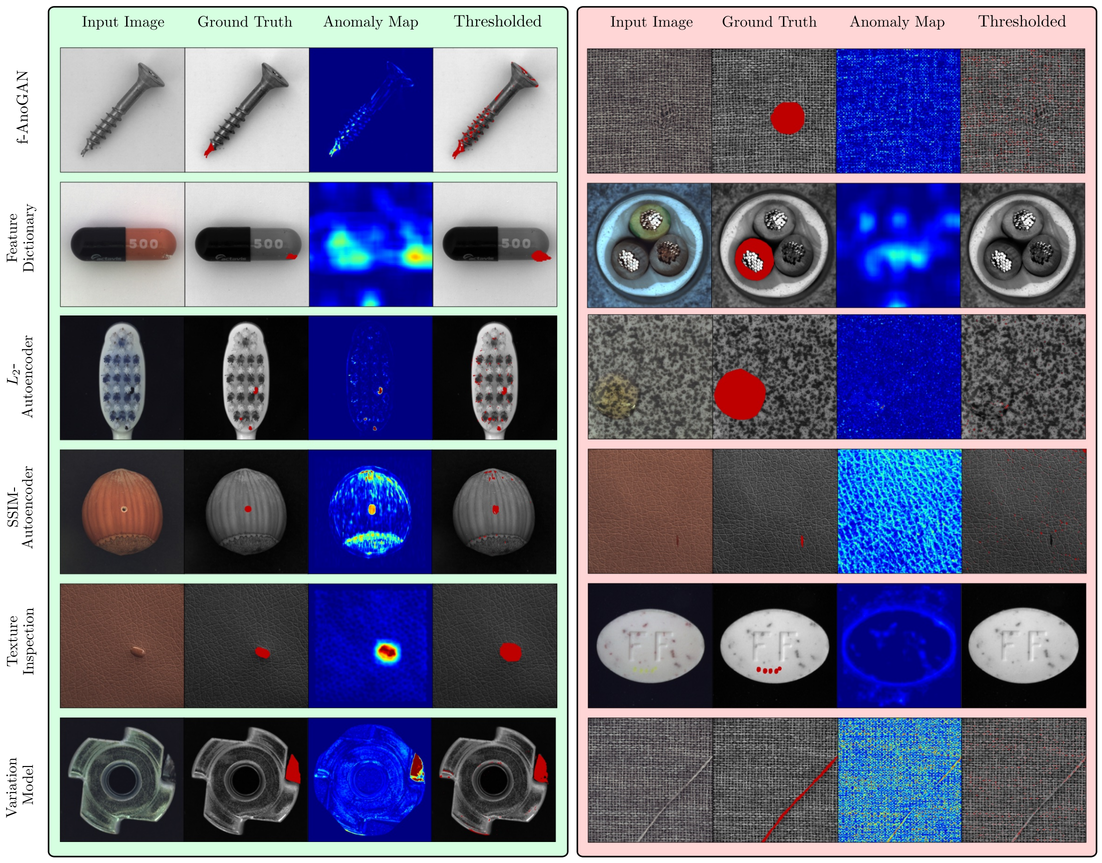


Figure 4.7: Qualitative results for each evaluated method. The left column shows examples where each method worked well. A failure case is shown in the right column. Thresholds were selected such that a false positive rate of 0.01 is achieved on the test set of an evaluated category.

f-AnoGAN: The f-AnoGAN method computes anomaly scores based on per-pixel comparisons between its input and reconstruction. Due to the increased contrast between the screw and the background, it manages to segment the tip of the screw. Because of imperfect reconstructions, however, the method also yields numerous false positives around the objects' edges and around regions where strong reflections are present. It entirely fails to detect the structural anomaly on the carpet since a removal of the defect by reconstruction does not result in an image substantially different from the input.

Feature Dictionary: The CNN Feature Dictionary was originally designed to model the distribution of repetitive texture patches. However, it also yields promising results for anomaly localization on objects when anomalies manifest themselves in features that deviate strongly from the local descriptors of the training data manifold. For example, the small crack on the capsule is well detected. However, since the method randomly subsamples training patches, it yields increased anomaly scores in regions that are underrepresented in the training set, e.g., on the imprint on the left half of the capsule. Additionally, due to the limited capacity of K-Means, the training feature distribution is often insufficiently well approximated. The method does not capture the global context of an object. Hence, it fails to detect the anomaly on the cable cross section, where the inner insulation on the bottom left shows the wrong color, as it is brown instead of blue.

L_2**- and SSIM-Autoencoder:** Both autoencoders rely on accurate reconstruction of their inputs for precise anomaly detection. However, they often fail to reconstruct small details and produce blurry images. Therefore, they tend to yield increased anomaly scores in regions that are challenging to reconstruct accurately, as can be observed on the object boundaries of the hazelnut and the bristles of the toothbrush. Like for f-AnoGAN, the L_2-Autoencoder's per-pixel comparisons result in unsatisfactory anomaly localization performance when the gray-value difference is small between the input and reconstruction, as is the case for the transparent color defect on the tile. Since the SSIM-Autoencoder only operates on grayscale images, it often fails to detect color defects entirely, such as the red color stroke on the leather texture.

GMM-Based Texture Inspection Model: HALCON's Texture Inspection models the distribution of gray-values within local image patches using a GMM. It performs well on uniform texture patterns, for example, those that are present in the dataset category leather. Since it only operates on grayscale images, it often fails to detect color defects such as the one on the pill. Because the boundaries of objects are underrepresented in the training data, it often yields increased anomaly scores in these areas.

Variation Model: For the evaluation of the Variation Model, prior object alignment is performed where possible. It performs well for rigid objects such as the metal nut, which allows for a precise alignment. Due to the applied transformation, not every single input image pixel overlaps with the mean and deviation image. For these background pixels, no meaningful anomaly score can be computed. For dataset categories where an alignment is not possible, e.g., *carpet*, this method fails entirely. Due to the high variance of the gray values in the training images, the model assigns high likelihoods to almost every gray value.

4.6.3 Anomaly Classification Results

In addition to the localization of anomalous regions, it is also of interest how well each method can separate between anomalous and anomaly-free data samples. To compute a single anomaly score from an anomaly map, we select its maximum value. We then compute the area under the ROC curve for each method and dataset category. The results are listed in Table 4.4. They are very similar to the results obtained for anomaly localization. Again, the CNN Feature Dictionary yields the overall best performance with a ROC-AUC of 0.8. The more traditional methods, i.e., the Variation Model and

Category	f-AnoGAN	Feature Dictionary	L_2-Autoencoder	SSIM-Autoencoder	Texture Inspection	Variation Model
Carpet	0.506	0.865	0.554	0.459	**0.917**	0.175
Grid	0.896	0.855	0.940	0.796	**0.980**	**0.980**
Leather	0.814	0.773	0.881	0.565	**0.990**	0.870
Tile	0.845	**0.996**	0.627	0.470	0.925	0.651
Wood	0.891	0.909	0.766	0.666	**0.917**	0.840
Bottle	0.839	**0.980**	0.962	0.483	0.906	0.874
Cable	0.506	**0.814**	0.541	0.626	0.439	0.583
Capsule	0.494	0.709	0.774	0.601	0.552	0.783
Hazelnut	0.946	0.818	0.956	0.695	**0.963**	0.703
Metal nut	0.404	**0.941**	0.610	0.680	0.419	0.753
Pill	0.571	**0.778**	0.700	0.610	0.601	0.686
Screw	**0.785**	0.437	0.736	0.673	0.645	0.727
Toothbrush	0.536	0.733	**0.961**	0.822	0.844	0.888
Transistor	0.712	0.673	0.718	0.624	0.419	**0.829**
Zipper	0.557	0.705	0.739	0.745	**0.936**	0.604
Mean	0.687	**0.800**	0.764	0.634	0.764	0.731

Table 4.4: Area under the ROC curve for classification for each dataset category.

the Texture Inspection Model, perform better than the majority of generative deep learning models.

4.6.4 Threshold Estimation Techniques

In Section 4.5, we discussed various techniques to estimate thresholds purely on a validation set of anomaly-free images. To assess their performance in practice, we computed thresholds on three different categories of the dataset: *bottle*, *pill*, and *wood*. The Maximum threshold simply selects the maximum anomaly score of all validation pixels. For the p-Quantile threshold, we used $p = 0.99$, which means that one percent of all validation pixels will be marked as anomalous by each method. We selected a k-Sigma threshold such that under the assumption of normally distributed anomaly scores, also a quantile of 0.99 is reached. We additionally investigated a Max-Area threshold that allows connected components of anomalous pixels with an area smaller than 0.1% of the area of the entire input image.

Figure 4.8 marks the FPR and PRO values achieved when applying the different thresholds. For each dataset category, the three best performing methods in terms of AU-PRO are displayed. Since the Maximum threshold does not allow a single false positive pixel on the entire validation set, it is the most conservative threshold estimator among the evaluated ones, yielding the lowest false positive rates on the test set. However, in some cases, it entirely fails to produce any true positives as well, due to outliers on the validation set.

All other threshold estimation techniques allow a certain amount of false positives on the validation set. Hence, they also yield increased false positive rates on the test set. Both the p-Quantile and k-Sigma thresholds attempt to fix the false positive rate at one percent. However, due to the inaccurate segmentations of each method, the

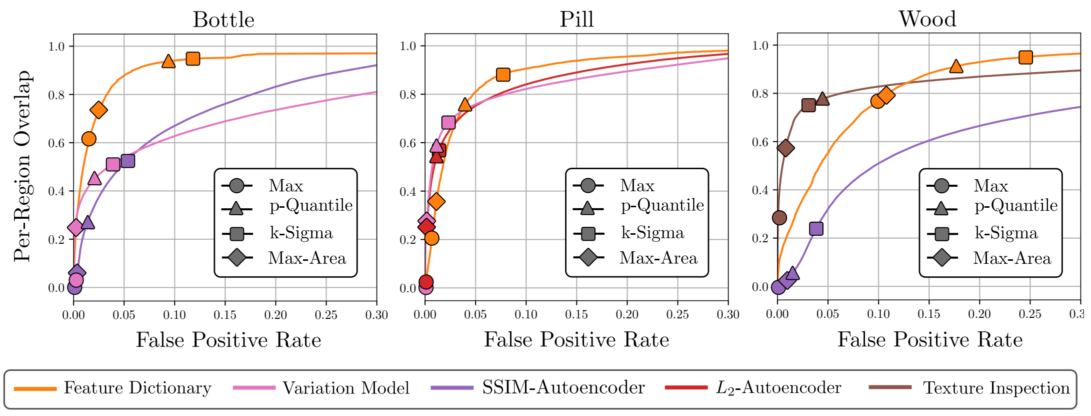


Figure 4.8: Performance of different threshold estimates in terms of FPR and PRO on three different dataset categories. For each category, the three top performing methods are displayed. Thresholds are computed on a validation set of anomaly-free images.

application of each threshold results in a significantly higher FPR. Furthermore, the marker locations of the two thresholds are often very different for the same anomaly detection method, which indicates that the assumption of normally distributed anomaly scores does often not hold in practice. For many of the evaluated methods, the Max-Area threshold is only slightly less conservative than picking the maximum of all anomaly scores. This indicates that already only a slight decrease of the Maximum threshold results in connected components of false positives that one might deem large enough to classify them as anomalies in practice.

Our results show that selecting a suitable threshold for anomaly localization purely on anomaly-free validation images is a highly challenging problem in practice. The same estimator might yield very different results depending on the anomaly detection method and dataset under consideration. For applications that require very low false positive rates, one is at risk of picking too conservative thresholds that fail to detect any anomalies. On the other hand, allowing for too many false positives quickly yields segmentation results that are no longer useful in practice as well.

4.6.5 Time and Memory Consumption

The runtime and required memory of a method during inference are important criteria for its applicability in real-world scenarios. However, since both greatly depend on implementation-specific details, measuring them accurately is challenging. For example, the amount of memory used by deep learning methods can often be greatly reduced when freeing intermediate feature maps during a forward pass. The execution time of an algorithm is directly affected by the specific libraries being used and the amount of exploited potential for parallelization. Hence, we do not provide exact numbers for inference time and memory consumption but rather point out qualitative differences between the evaluated methods.

Method	#Parameters
f-AnoGAN	24.57 M
Feature Dictionary	11.46 M
L_2-Autoencoder	1.20 M
SSIM-Autoencoder	1.20 M

Table 4.5: Approximate number of model parameters of each evaluated deep learning method in millions.

As can be expected, methods performing multiple forward passes through a network have the highest inference times. A particularly extreme example is the CNN Feature Dictionary, which requires several seconds to process a single image. This is due to the patchwise evaluation and the fact that only a single anomaly score is produced for each patch. It is possible to reduce the time by using a larger stride for the patches at the cost of coarser anomaly maps. Methods that require only a single model evaluation per image and run entirely on the GPU, such as the autoencoders evaluated on the objects of the dataset, allow for much faster inference times in the range of a few milliseconds. However, when performing strided evaluations on the textures, multiple forward passes become necessary and their runtime increases to several hundreds of milliseconds. The same is true for f-AnoGAN. For the more traditional methods, i.e., the Variation Model and the Texture Inspection Model, we use optimized implementations of the HALCON machine vision library that entirely run on the CPU and achieve runtimes in the range of tens and hundreds of milliseconds, respectively.

To facilitate a relative comparison of the amount of memory required to perform inference in deep learning models, one commonly reports the total number of model parameters as a lower bound. The number of parameters for each model evaluated in this chapter is given in Table 4.5. Since the Variation Model and the Texture Inspection Model are not based on deep learning and work in an entirely different way, simply counting the number of model parameters and comparing them to the deep learning approaches is not advisable. The Variation Model, for example, stores two model parameters for each image pixel and thus, the total number of parameters is in the same range as one of the evaluated deep learning models. However, the Variation Model does not need to allocate any additional memory and one can still expect the deep learning approaches to consume a lot more memory during inference due to their intermediate computation of high-dimensional feature maps.

4.7 Conclusion

In this chapter, we introduced the MVTec Anomaly Detection dataset, a new dataset for unsupervised anomaly detection that is based on real-world industrial inspection scenarios. The dataset provides the possibility to evaluate unsupervised anomaly detection methods on various texture and object classes with different types of anomalies. Because pixel-precise ground truth labels for anomalous regions in the images are provided, it is possible to evaluate anomaly detection methods for both image-level classification as well as pixel-level segmentation.

We have thoroughly evaluated several methods based on deep learning as well as two classical methods for anomaly detection on our dataset. Our results show that discriminative approaches that leverage descriptors of pretrained networks tend to perform better than methods that learn feature representations from scratch solely on the anomaly-free training data. We have provided information on inference time as well as memory consumption for each evaluated method.

Furthermore, we have discussed properties of common evaluation metrics and threshold estimation techniques for anomaly localization and have highlighted their advantages and shortcomings. We have shown that determining suitable thresholds solely on anomaly-free data is a challenging problem because the performance of each estimator highly varies for different dataset categories and evaluated methods.

5 Student–Teacher Anomaly Detection

In the experiments of the previous chapter, we found that descriptors extracted from pretrained CNNs perform well when applied to unsupervised anomaly detection. However, methods such as the CNN Feature Dictionary rely on patch-based evaluations that result in coarse anomaly maps and slow inference times. In this chapter, we introduce a Student–Teacher framework that detects and precisely localizes anomalous regions in high-resolution images with a single forward pass.

Student networks are trained to regress the output of a descriptive teacher network that was pretrained on a large dataset of patches from natural images. This circumvents the need for prior data annotation. Anomalies are detected where the outputs of the student networks differ from that of the teacher network. This happens when they fail to generalize outside the manifold of anomaly-free training data. The intrinsic uncertainty in the student networks is used as an additional scoring function that indicates anomalies. We compare our method to a number of existing deep learning methods for unsupervised anomaly detection. Our experiments demonstrate significant improvements on a number of real-world datasets, including the MVTec Anomaly Detection dataset.

The content of this chapter is based on the publication *Uninformed Students: Student–Teacher Anomaly Detection with Discriminative Latent Embeddings* [Bergmann et al., 2020].

5.1 Introduction

The performance of many supervised computer vision algorithms is improved by transfer learning, i.e., by using discriminative embeddings from pretrained networks [Kornblith et al., 2019, Sun et al., 2019]. For unsupervised anomaly detection, such approaches are not thoroughly explored. Recent work suggests that these feature spaces generalize well for anomaly detection and even simple baselines outperform generative deep learning approaches [Burlina et al., 2019, Perera and Patel, 2019]. However, the performance of existing methods on large high-resolution image datasets is hampered by the use of shallow machine learning pipelines that require a dimensionality reduction of the used feature space. Moreover, they rely on heavy training data subsampling since their capacity does not suffice to model highly complex data distributions with a large number of training samples.

We propose to circumvent these limitations of shallow models by implicitly modeling the distribution of training features with a Student–Teacher approach. This leverages the high capacity of deep neural networks and frames anomaly detection as a feature regression problem. Given a descriptive feature extractor pretrained on a large dataset of patches from natural images (the teacher), we train an ensemble of student networks

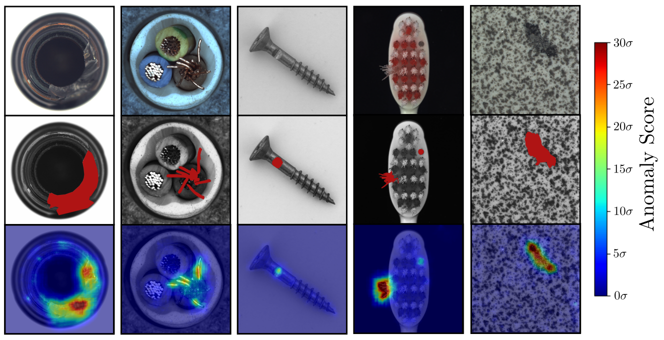


Figure 5.1: Qualitative results of our Student–Teacher method on the MVTec Anomaly Detection dataset. **Top row:** Input images containing defects. **Center row:** Ground truth regions of defects in red. **Bottom row:** Anomaly scores for each image pixel predicted by our algorithm.

on anomaly-free training data to mimic the teacher's output. During inference, the students' predictive uncertainty together with their regression error with respect to the teacher are combined to yield dense anomaly scores for each input pixel. Our intuition is that students will generalize poorly outside the manifold of anomaly-free training data and start to make wrong predictions. Figure 5.1 shows qualitative results of our method when applied to images selected from the MVTec Anomaly Detection dataset introduced in the previous chapter. A schematic overview of the entire anomaly detection process is given in Figure 5.2. Our main contributions are:

- We propose a novel Student–Teacher framework for unsupervised anomaly detection. Local descriptors from a pretrained teacher network serve as surrogate labels for an ensemble of students. Our model can be trained end-to-end on large unlabeled image datasets and make use of all available training data.

- We introduce scoring functions based on the students' predictive variance and regression error to obtain dense anomaly maps for the localization of anomalous regions in natural images. We describe how to extend our approach to detect anomalies at multiple scales by adapting the students' and teacher's receptive fields.

- We compare our method to a number of shallow machine learning classifiers and deep generative models that are fitted directly to the teacher's feature distribution on three different computer vision datasets. We also compare it to recently introduced deep learning methods for unsupervised anomaly detection. At the time of publication, our method achieved state-of-the-art performance.

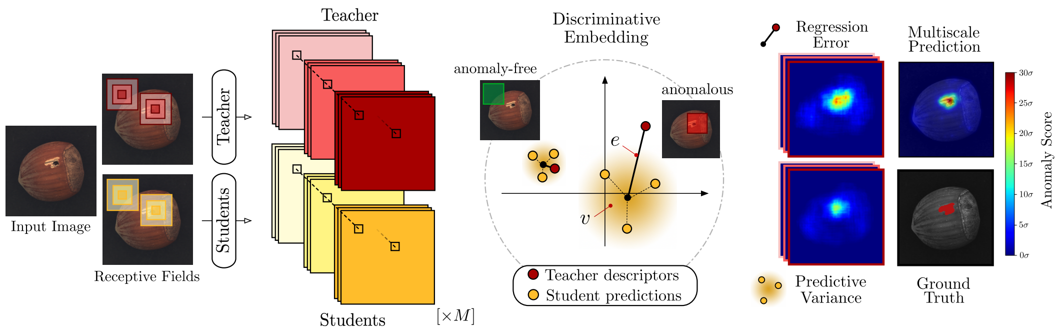


Figure 5.2: Schematic overview of our approach. Input images are fed through a teacher network that densely extracts features for local image regions. An ensemble of M student networks is trained to regress the output of the teacher on anomaly-free data. During inference, the students will yield increased regression errors e and predictive variances v in pixels for which the receptive field covers anomalous regions. Anomaly maps generated with different receptive fields can be combined for anomaly localization at multiple scales.

5.2 Related Work

5.2.1 Anomaly Detection using Pretrained Networks

Promising results have been achieved by transferring discriminative embedding vectors of pretrained networks to the task of anomaly detection by fitting shallow machine learning models to the features of anomaly-free training data. For a brief introduction to such methods, we refer to Section 2.4.3.

Andrews et al. [2016] use activations from different layers of a pretrained VGG network and model the anomaly-free training distribution with a ν-SVM. However, they only apply their method to image classification and do not consider the segmentation of anomalous regions. Similar experiments have been performed by Burlina et al. [2019]. They report superior performance of discriminative embeddings compared to feature spaces obtained from generative models.

Nazare et al. [2018] investigate the performance of different off-the-shelf feature extractors pretrained on an image classification task for the segmentation of anomalies in surveillance videos. Their approach trains a 1-Nearest-Neighbor (1-NN) classifier on embedding vectors extracted from a large number of anomaly-free training patches. Prior to the training of the shallow classifier, the dimensionality of the network's activations is reduced using Principal Component Analysis (PCA). To obtain a spatial anomaly map during inference, the classifier must be evaluated for a large number of overlapping patches, which quickly becomes a performance bottleneck and results in rather coarse anomaly maps. Similarly, Napoletano et al. [2018] extract activations from a pretrained ResNet-18 for a large number of cropped training patches and model their distribution using K-Means clustering after prior dimensionality reduction with PCA. They also perform strided evaluation of test images during inference. Both approaches sample training patches from the input images and therefore do not make use of all possible training features. This is necessary since, in their framework, feature extraction is computationally expensive due to the use of very deep networks that output only a single descriptor per patch. Furthermore, since shallow models are employed for learning the feature distribution of anomaly-free patches, the available training information must be strongly reduced.

To circumvent the need for cropping patches and to speed up feature extraction, Sabokrou et al. [2018] extract descriptors from early feature maps of a pretrained AlexNet in a fully convolutional fashion and fit a unimodal Gaussian distribution to all available training vectors of anomaly-free images. Even though feature extraction is achieved more efficiently in their framework, pooling layers lead to a downsampling of the input image. This strongly decreases the resolution of the final anomaly map, especially when using descriptive features of deeper network layers with larger receptive fields. In addition, unimodal Gaussian distributions will fail to model the training feature distribution as soon as the problem complexity rises.

5.2.2 Open-Set Recognition with Uncertainty Estimates

Our work draws some inspiration from the recent success of open-set recognition in supervised settings such as image classification or semantic segmentation, where uncertainty estimates of deep neural networks have been exploited to detect out-of-distribution inputs using MC Dropout [Kendall and Gal, 2017] or deep ensembles [Lakshminarayanan et al., 2017]. Seeböck et al. [2020] demonstrate that uncertainties from segmentation networks trained with MC Dropout can be used to detect anomalies in retinal OCT images. Beluch et al. [2018] show that the variance of network ensembles trained on an image classification task serves as an effective acquisition function for active learning. Inputs that appear anomalous to the current model are added to the training set to quickly enhance its performance.

Such algorithms, however, demand prior labeling of images by domain experts for a supervised task, which is not always possible or desirable. In our work, we utilize feature vectors of pretrained networks as surrogate labels for the training of an ensemble of student networks. The predictive variance together with the regression error of the ensemble's output mixture distribution is then used as a scoring function to segment anomalous regions in test images.

5.3 Student–Teacher Anomaly Detection

This section describes the core principles of our proposed method. Given a training dataset $\mathcal{D}_{\text{train}}$ of anomaly-free images, our goal is to create an ensemble of *student networks* S_i that can later detect anomalies in test images. This means that they can assign a score to each pixel indicating how much it deviates from the training data manifold. For this, the student models are trained against regression targets obtained from a descriptive *teacher network* T pretrained on a large dataset of natural images. After the training, anomaly scores can be derived for each image pixel from the students' regression error and predictive variance. Given an input image $I : D \to \mathbb{R}^C$ with domain D

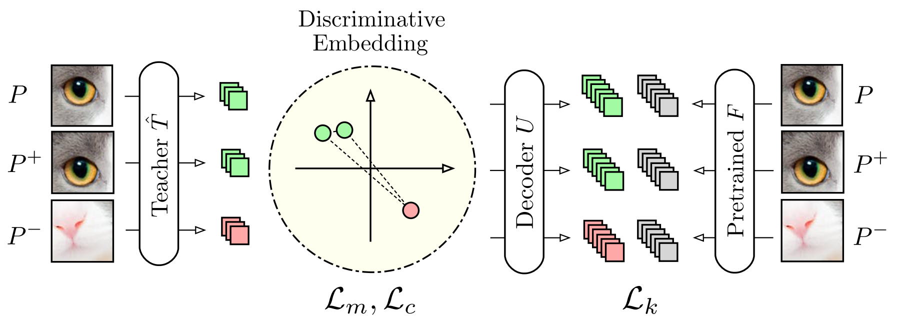


Figure 5.3: Pretraining of the teacher network $\hat{T}$ to output descriptive embedding vectors for patch-sized inputs. The knowledge of a powerful but computationally inefficient network F is distilled into $\hat{T}$ by decoding the latent vectors to match the descriptors of F. We also experiment with embeddings obtained using self-supervised metric learning techniques based on triplet learning. Information within each feature dimension is maximized by decorrelating the feature dimensions within a minibatch.

and number of channels C, each student S_i in the ensemble outputs a feature map that contains a d-dimensional descriptor for each image pixel $\boldsymbol{p} \in D$. By design, we limit the students' receptive field, such that each descriptor describes a square local image region $P_{\boldsymbol{p}}$ within I centered at $\boldsymbol{p}$. The teacher T has the same network architecture as the student networks. However, it remains constant and extracts descriptive embedding vectors for each pixel of the input image I that serve as deterministic regression targets during student training.

5.3.1 Learning Local Patch Descriptors

We begin by describing how to efficiently construct a descriptive teacher network T using metric learning and knowledge distillation techniques. In many existing works for anomaly detection with pretrained networks, feature extractors only output single feature vectors for patch-sized inputs or spatially heavily downsampled feature maps [Napoletano et al., 2018, Sabokrou et al., 2018]. In contrast, our teacher network T efficiently outputs descriptors for every possible square of a fixed side length within the input image. T is obtained by first training a network $\hat{T}$ to embed patch-sized images $P : \tilde{D} \to \mathbb{R}^C$ into a metric space of dimension d using only convolution and max-pooling layers. The domain of each patch is $\tilde{D} = \{0, \ldots, p-1\} \times \{0, \ldots, p-1\}$, where $p \in \mathbb{N}^+$ denotes the patch size. Fast dense local feature extraction for an entire input image can then be achieved by a deterministic network transformation of $\hat{T}$ to T as described in [Bailer et al., 2017]. This yields significant speedups compared to previously introduced methods that perform patch-based strided evaluations. To let $\hat{T}$ output semantically strong descriptors, we investigate both self-supervised metric learning techniques as well as distilling knowledge from a descriptive but computationally inefficient pretrained network. A large number of training patches P can be obtained by random crops from any image database. Here, we use ImageNet [Krizhevsky et al., 2012].

Knowledge Distillation. Patch descriptors obtained from deep layers of CNNs trained on image classification tasks perform well for anomaly detection when modeling their distribution with shallow machine learning models [Napoletano et al., 2018, Nazare et al., 2018]. However, the architectures of such CNNs are usually highly complex and computationally inefficient for the extraction of local patch descriptors. Therefore, we distill the knowledge of a powerful pretrained network F into $\hat{T}$ by matching the output of F with a decoded version of the descriptor obtained from $\hat{T}$:

$$\mathcal{L}_k(\hat{T}) = \frac{1}{\hat{B}} \sum_{i=1}^{\hat{B}} ||U(\hat{T}(P_i)) - F(P_i)||_2^2. \tag{5.1}$$

Here, U denotes a fully connected decoder network that upsamples the d-dimensional output of $\hat{T}$ to the output dimension of the pretrained network's descriptor. The variable $\hat{B}$ refers to the batch size used to pretrain the teacher network.

Metric Learning. If for some reason pretrained networks are unavailable, one can also learn local image descriptors in a fully self-supervised way [Danon et al., 2019]. Here, we investigate the performance of discriminative embeddings obtained using triplet learning. For every randomly cropped patch P, a triplet of patches (P, P^+, P^-) is augmented. Positive patches P^+ are obtained by small random translations around P, changes in image luminance, and the addition of Gaussian noise. The negative patch P^- is created by a random crop from a randomly selected different image. In-triplet hard negative mining with anchor swap [Vassileios Balntas and Mikolajczyk, 2016] is used as a loss function for learning an embedding sensitive to the L_2-metric:

$$\mathcal{L}_m(\hat{T}) = \frac{1}{\hat{B}} \sum_{i=1}^{\hat{B}} \max\{0, \delta + \delta_i^+ - \delta_i^-\}. \tag{5.2}$$

Here, $\delta \in (0, \infty)$ denotes the margin parameter and in-triplet distances δ^+ and δ^- are defined as:

$$\delta_i^+ = ||\hat{T}(P_i) - \hat{T}(P_i^+)||_2^2, \tag{5.3}$$

$$\delta_i^- = \min\{||\hat{T}(P_i) - \hat{T}(P_i^-)||_2^2, ||\hat{T}(P_i^+) - \hat{T}(P_i^-)||_2^2\}. \tag{5.4}$$

Descriptor Compactness. As proposed by Tian et al. [2017], we minimize the correlation between descriptors within one minibatch of inputs P to increase the descriptors' compactness and remove unnecessary redundancy:

$$\mathcal{L}_c(\hat{T}) = \sum_{k \neq l} \boldsymbol{C}_{k,l}^2, \tag{5.5}$$

where $\boldsymbol{C}_{k,l}$ denotes the entries of the correlation matrix computed over all descriptors $\hat{T}(P)$ in the current minibatch. The final training loss for $\hat{T}$ is then given as

$$\mathcal{L}(\hat{T}) = \lambda_k \mathcal{L}_k(\hat{T}) + \lambda_m \mathcal{L}_m(\hat{T}) + \lambda_c \mathcal{L}_c(\hat{T}), \tag{5.6}$$

where λ_k, λ_m, $\lambda_c \in [0, \infty)$ are weighting factors for the individual loss terms. Figure 5.3 summarizes the entire learning process for the teacher's discriminative embedding.

5.3.2 Ensemble of Student Networks for Deep Anomaly Detection

Next, we describe how to train student networks S_i to predict the teacher's output on anomaly-free training data. We then derive anomaly scores from the students' predictive uncertainty and regression error during inference. First, the vector of component wise means $\boldsymbol{\mu} \in \mathbb{R}^d$ and standard deviations $\boldsymbol{\sigma} \in \mathbb{R}^d$ over all training descriptors is computed for data normalization. Descriptors are extracted by applying T to each image in the dataset $\mathcal{D}_{\text{train}}$. We then train an ensemble of $M \geq 1$ randomly initialized student networks S_i, $i \in \{1, \ldots, M\}$ that possess the identical network architecture as the teacher T. For an input image I, each student outputs its predictive distribution over the space of possible regression targets for each local image region $P_{\boldsymbol{p}}$ centered around the pixel $\boldsymbol{p}$. Note that the students' architecture allows us to obtain dense predictions for each image pixel with only a single forward pass, without having to actually crop the patches $P_{\boldsymbol{p}}$. The students' output vectors are modeled as a Gaussian distribution $\Pr(\mathbf{y}|P_{\boldsymbol{p}}) = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_{\boldsymbol{p}}^{S_i}, s)$ with constant covariance $s \in \mathbb{R}$, where $\boldsymbol{\mu}_{\boldsymbol{p}}^{S_i}$ denotes the prediction made by S_i for the pixel $\boldsymbol{p}$. Let $\tilde{\mathbf{y}}_{\boldsymbol{p}}$ denote the teacher's respective descriptor that is to be predicted by the students. The log-likelihood training criterion for each student network and a single dataset sample then simplifies to the squared L_2-distance in feature space:

$$E(I) = \frac{1}{|D|} \sum_{\boldsymbol{p} \in D} ||\boldsymbol{\mu}_{\boldsymbol{p}}^{S_i} - (\tilde{\mathbf{y}}_{\boldsymbol{p}} - \boldsymbol{\mu})\text{diag}(\boldsymbol{\sigma})^{-1}||_2^2, \tag{5.7}$$

where $\text{diag}(\boldsymbol{\sigma})^{-1}$ denotes the inverse of the diagonal matrix filled with the values in $\boldsymbol{\sigma}$. The training loss over the dataset samples in a minibatch of B elements can then be written as $\mathcal{L}(S_i) = \frac{1}{B}\sum_{j=1}^{B} E(I_j)$.

Scoring Functions for Anomaly Detection. Having trained each student to convergence, a mixture of Gaussians can be obtained at each image pixel by equally weighting the ensemble's predictive distributions. From it, measures of anomaly can be obtained in two ways: First, we propose to compute the regression error of the mixture's mean $\boldsymbol{\mu}_{\boldsymbol{p}}$ with respect to the teacher's surrogate label:

$$e_{\boldsymbol{p}} = ||\boldsymbol{\mu}_{\boldsymbol{p}} - (\tilde{\mathbf{y}}_{\boldsymbol{p}} - \boldsymbol{\mu})\text{diag}(\boldsymbol{\sigma})^{-1}||_2^2 \tag{5.8}$$

$$= \left|\left|\frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\mu}_{\boldsymbol{p}}^{S_i} - (\tilde{\mathbf{y}}_{\boldsymbol{p}} - \boldsymbol{\mu})\text{diag}(\boldsymbol{\sigma})^{-1}\right|\right|_2^2. \tag{5.9}$$

The intuition behind this score is that the student networks will fail to regress the teacher's output within anomalous regions during inference since the corresponding descriptors have not been observed during training. Note that $e_{\boldsymbol{p}}$ can be computed even if $M = 1$, where only a single student is trained.

As a second measure of anomaly, we compute for each pixel the predictive uncertainty of the Gaussian mixture as defined by Kendall and Gal [2017], assuming that the student networks generalize similarly for anomaly-free regions and differently in regions that

	Layer	Output Size	Parameters	
			Kernel	Stride
$p = 17$	Input	$17 \times 17 \times 3$		
	Conv1	$12 \times 12 \times 128$	5×5	1
	Conv2	$8 \times 8 \times 256$	5×5	1
	Conv3	$4 \times 4 \times 256$	5×5	1
	Conv4	$1 \times 1 \times 128$	4×4	1
	Decode	$1 \times 1 \times 512$	1×1	1
$p = 33$	Input	$33 \times 33 \times 3$		
	Conv1	$29 \times 29 \times 128$	3×3	1
	MaxPool	$14 \times 14 \times 128$	2×2	2
	Conv2	$10 \times 10 \times 256$	5×5	1
	MaxPool	$5 \times 5 \times 256$	2×2	2
	Conv3	$4 \times 4 \times 256$	2×2	1
	Conv4	$1 \times 1 \times 128$	4×4	1
	Decode	$1 \times 1 \times 512$	1×1	1
$p = 65$	Input	$65 \times 65 \times 3$		
	Conv1	$61 \times 61 \times 128$	5×5	1
	MaxPool	$30 \times 30 \times 128$	2×2	2
	Conv2	$26 \times 26 \times 128$	5×5	1
	MaxPool	$13 \times 13 \times 128$	2×2	1
	Conv3	$9 \times 9 \times 128$	5×5	1
	MaxPool	$4 \times 4 \times 128$	2×2	2
	Conv4	$1 \times 1 \times 256$	4×4	1
	Conv5	$1 \times 1 \times 128$	1×1	1
	Decode	$1 \times 1 \times 512$	1×1	1

Table 5.1: General outline of our network architecture for training teachers $\hat{T}$ with different receptive field sizes $p \in \{17, 33, 65\}$. Leaky rectified linear units with slope 0.05 are applied as activation functions after each convolution layer.

contain novel information unseen during training:

$$v_{\boldsymbol{p}} = \frac{1}{M} \sum_{i=1}^{M} ||\boldsymbol{\mu}_{\boldsymbol{p}}^{S_i}||_2^2 - ||\boldsymbol{\mu}_{\boldsymbol{p}}||_2^2. \tag{5.10}$$

To combine the two scores, we compute the means e_μ, v_μ and standard deviations e_σ, v_σ of all $e_{\boldsymbol{p}}$ and $v_{\boldsymbol{p}}$, respectively, over a validation set of anomaly-free images. Summation of the normalized scores then yields the final anomaly score:

$$\tilde{e}_{\boldsymbol{p}} + \tilde{v}_{\boldsymbol{p}} = \frac{e_{\boldsymbol{p}} - e_\mu}{e_\sigma} + \frac{v_{\boldsymbol{p}} - v_\mu}{v_\sigma}. \tag{5.11}$$

Figure 5.4 illustrates the basic principles of our anomaly detection method on the MNIST dataset, where images with label 0 were treated as the normal class and all other classes were treated as anomalous. Since the images of this dataset are very small, we extracted a single feature vector for each image using $\hat{T}$ and trained an ensemble

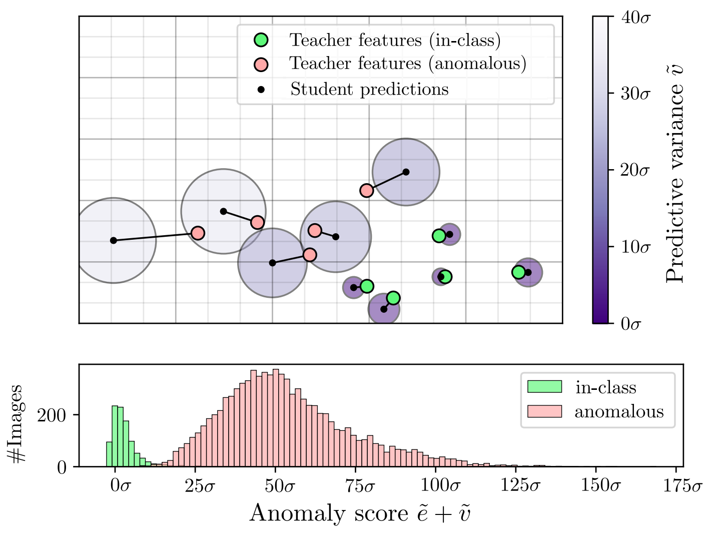


Figure 5.4: Embedding vectors visualized for ten samples of the MNIST dataset. Larger circles around the students' mean predictions indicate increased predictive variance. Being only trained on a single class of training images, the students manage to accurately regress the features solely for this class (green). They yield large regression errors and predictive uncertainties for images of other classes (red). Anomaly scores for the entire dataset are displayed in the bottom histogram.

of $M = 5$ patch-sized students to regress the teacher's output. This results in a single anomaly score for each input image. Feature descriptors were embedded into 2D using Multidimensional Scaling [Borg and Groenen, 2003] to preserve their relative distances.

5.3.3 Multi-Scale Anomaly Detection

If an anomaly only covers a small part of the teacher's receptive field of size p, the extracted feature vector predominantly describes anomaly-free traits of the local image region. Consequently, the descriptor can be predicted well by the students and anomaly detection performance will decrease. One could tackle this problem by downsampling the input image. This would, however, lead to an undesirable loss in resolution of the output anomaly map.

Our framework allows for explicit control over the size of the students' and teacher's receptive field p. Therefore, we can detect anomalies at various scales by training multiple student–teacher ensemble pairs with varying values of p. At each scale, an anomaly map with the same size as the input image is computed. Given L ensemble pairs with different receptive fields, the normalized anomaly scores $\tilde{e}_{\boldsymbol{p}}^{(l)}$ and $\tilde{v}_{\boldsymbol{p}}^{(l)}$ of each scale l can

Method	L_k	L_m	L_c	MNIST	CIFAR-10
OCGAN				0.9750	0.6566
1-NN				0.9753	0.8189
KMeans				0.9457	0.7592
OC-SVM				0.9463	0.7388
L_2-AE				0.9832	0.7898
VAE				0.9535	0.7502
Ours	✓		✓	**0.9935**	**0.8196**
Ours	✓	✓	✓	0.9926	0.8035
Ours		✓	✓	**0.9935**	0.7940
Ours	✓			0.9917	0.8021

Table 5.2: Results on MNIST and CIFAR-10. For each method, the average area under the ROC curve is given, computed across each dataset category. For our algorithm, we evaluate teacher networks trained with different loss functions. ✓ corresponds to setting the respective loss weight to 1, otherwise it is set to 0.

be combined by simple averaging:

$$\frac{1}{L}\sum_{l=1}^{L}\left(\tilde{e}_{\boldsymbol{p}}^{(l)}+\tilde{v}_{\boldsymbol{p}}^{(l)}\right). \tag{5.12}$$

5.4 Experiments

To demonstrate the effectiveness of our approach, an extensive evaluation on a number of datasets is performed. We measure the performance of our Student–Teacher framework against existing pipelines that use shallow machine learning algorithms to model the feature distribution of pretrained networks. To do so, we compare to a K-Means classifier, a One-Class SVM (OC-SVM), and a 1-NN classifier. They are fitted to the distribution of the teacher's descriptors after prior dimensionality reduction using PCA [Hadsell et al., 2006]. We also experiment with deterministic and variational autoencoders as deep distribution models over the teacher's discriminative embedding. The L_2 reconstruction error and reconstruction probability [An and Cho, 2015] are used as the anomaly score, respectively. We further compare our method to recently introduced generative and discriminative deep learning anomaly detection models and report improved performance. We want to stress that the teacher has not observed images of the evaluated datasets during pretraining to avoid an unfair bias.

As a first experiment, we perform an ablation study to find suitable hyperparameters. Our algorithm is applied to a one-class classification setting on the MNIST [LeCun et al., 1998] and CIFAR-10 [Krizhevsky and Hinton, 2009] datasets. We then evaluate on the much more challenging MVTec Anomaly Detection (MVTec AD) dataset introduced in the previous chapter. To highlight the benefit of our multi-scale approach, an additional ablation study is performed on MVTec AD, which investigates the impact of different receptive fields on the anomaly detection performance.

For our experiments, we use identical network architectures for the student and teacher networks, with receptive field sizes $p \in \{17, 33, 65\}$. All architectures are simple CNNs with only convolutional and max-pooling layers, using leaky rectified linear units with slope 5×10^{-3} as activation function. Table 5.1 shows the specific architectures used in our experiments.

For the pretraining of the teacher networks $\hat{T}$, triplets augmented from the ImageNet dataset are used. Images are zoomed to equal width and height sampled from $\{4p, 4p + 1, \dots, 16p\}$ and a patch of side length p is cropped at a random location. A positive patch P^+ for each triplet is then constructed by randomly translating the crop location within the interval $\{-\frac{p-1}{4}, \dots, \frac{p-1}{4}\}$. Gaussian noise with standard deviation 0.1 is added to P^+. All images within a triplet are randomly converted to grayscale with a probability of 0.1. For knowledge distillation, we extract 512-dimensional feature vectors from the fully connected layer of a ResNet-18 that was pretrained for classification on the ImageNet dataset. For the triplet loss, the margin parameter is chosen as $\delta = 1$. For network optimization, we use the Adam optimizer [Kingma and Ba, 2015] with an initial learning rate of 2×10^{-4}, a weight decay of 10^{-5}, and a batch size of 64. Each teacher network outputs descriptors of dimension $d = 128$ and is trained for 5×10^4 iterations.

5.4.1 MNIST and CIFAR-10

Before considering the problem of anomaly localization, we evaluate our method on the MNIST and CIFAR-10 datasets, adapted for one-class classification. Five students are trained on only a single class of the dataset, while during inference images of the other classes must be detected as anomalous. Each image is zoomed to the students' and teacher's input size p and a single feature vector is extracted by passing it through the patch-sized networks $\hat{T}$ and $\hat{S}_i$. We examine different teacher networks by varying the weights $\lambda_k, \lambda_m, \lambda_c$ in the teacher's loss function $\mathcal{L}(\hat{T})$. The patch size for the experiments in this subsection is set to $p = 33$. As a measure of anomaly detection performance, the area under the ROC curve is evaluated.

Shallow and deep distributions models are trained on the teacher's descriptors of all available in-distribution samples. For the deterministic L_2-Autoencoder (L_2-AE) and the variational autoencoder (VAE), we use a fully connected encoder architecture of shape 128–64–32–10 with leaky rectified linear units of slope 5×10^{-3}. The decoder is constructed in a manner symmetric to the encoder. Both autoencoders are trained for 100 epochs at an initial learning rate of 10^{-2} using the Adam optimizer and a batch size of 64. A weight decay rate of 10^{-5} is applied for regularization. To evaluate the reconstruction probability of the VAE, five independent forward passes are performed for each feature vector. For the One-Class SVM (OC-SVM), a radial basis function kernel is used. K-Means is trained with 10 cluster centers and the distance to the single closest cluster center is evaluated as the anomaly score for each input sample. For 1-NN, the feature vectors of all available training samples are stored and tested during inference. For all shallow machine learning models, the dimensionality of the teacher descriptors is reduced using PCA, retaining 95% of the variance. We additionally report numbers for

OCGAN [Perera et al., 2019], a recently proposed generative model directly trained on the input images.

Table 5.2 shows our results. Our approach outperforms the other methods for a variety of hyperparameter settings. Distilling the knowledge of the pretrained ResNet-18 into the teacher's descriptor yields slightly better performance than training the teacher in a fully self-supervised way using triplet learning. Reducing descriptor redundancy by minimizing the correlation matrix yields improved results. On average, shallow models and autoencoders fitted to our teacher's feature distribution outperform OCGAN but do not reach the performance of our approach. Since for 1-NN, every single training vector can be stored, it performs exceptionally well on these small datasets. On average, however, our method still performs better than all evaluated approaches.

5.4.2 MVTec Anomaly Detection Dataset

For all our experiments on MVTec AD, input images are zoomed to $W = H = 256$ pixels. We train on anomaly-free images for 100 epochs with batch size 1. This is equivalent to training on a large number of patches per batch due to the limited size of the networks' receptive field. We use Adam with initial learning rate 10^{-4} and weight decay 10^{-5}. Teacher networks were trained with $\lambda_k = \lambda_c = 1$ and $\lambda_m = 0$, as this configuration performed best on MNIST and CIFAR-10. Ensembles contain $M = 3$ students trained on three different patches sizes $p \in \{17, 33, 65\}$.

We first compare the performance of our Student–Teacher model against the methods evaluated in the previous chapter. The anomaly localization performance is measured by the area under the PRO curve up to an integration limit of 0.3 is listed in Table 5.3. On average, our method performs significantly better than all other evaluated approaches. It also outperforms all other methods on the majority of dataset categories. Similar results are obtained when evaluating the anomaly classification performance, i.e., the ability of each method of making a binary decision between anomalous and anomaly-free samples. Table 5.4 lists the respective areas under the ROC curve. Again, our Student–Teacher method yields an average performance that is substantially above those of the other approaches. Regarding the 15 indivual dataset categories, it obtains the highest AU-ROC value on 11 of them. Figure 5.5 displays additional qualitative results of our method.

Other Feature Distribution Models. We further compare the performance of our method to other distribution models fitted to the feature space of our teacher network. We choose the same hyperparameters as for the experiments on the MNIST and CIFAR experiments. To train shallow classifiers on the teacher's output descriptors, a subset of 5000 feature vectors is randomly sampled from the teacher's feature maps. Their dimension is then reduced by PCA, retaining 95% of the variance. For comparability, we train each algorithm with a single receptive field of size $p = 65$.

Table 5.5 shows our results. Our method consistently outperforms all other evaluated algorithms for almost every dataset category. The shallow machine learning algorithms fitted directly to the teacher's descriptors after applying PCA do not manage to perform

Category	Ours Multi-Scale	f-AnoGAN	Feature Dictionary	L_2-Autoencoder	SSIM-Autoencoder	Texture Inspection	Variation Model
Carpet	0.879	0.253	**0.895**	0.306	0.392	0.855	0.165
Grid	**0.952**	0.626	0.757	0.798	0.847	0.857	0.545
Leather	0.945	0.584	0.819	0.519	0.389	**0.981**	0.394
Tile	**0.946**	0.252	0.873	0.251	0.166	0.472	0.425
Wood	**0.911**	0.517	0.778	0.520	0.530	0.827	0.455
Bottle	**0.931**	0.440	0.906	0.567	0.703	0.636	0.659
Cable	**0.818**	0.428	0.815	0.507	0.368	0.597	0.405
Capsule	**0.968**	0.447	0.791	0.771	0.830	0.834	0.802
Hazelnut	**0.965**	0.872	0.913	0.922	0.897	0.958	0.849
Metal nut	**0.942**	0.482	0.701	0.607	0.501	0.384	0.562
Pill	**0.961**	0.700	0.872	0.847	0.803	0.606	0.834
Screw	**0.942**	0.808	0.725	0.864	0.875	0.864	0.701
Toothbrush	**0.933**	0.809	0.718	0.891	0.841	0.786	0.774
Transistor	**0.666**	0.494	0.590	0.657	0.602	0.542	0.554
Zipper	**0.951**	0.202	0.897	0.457	0.515	0.923	0.221
Mean	**0.914**	0.528	0.803	0.632	0.617	0.741	0.556

Table 5.3: Normalized area under the PRO curve up to an average false positive rate per-pixel of 30% for each dataset category.

Category	Ours Multi-Scale	f-AnoGAN	Feature Dictionary	L_2-Autoencoder	SSIM-Autoencoder	Texture Inspection	Variation Model
Carpet	**0.953**	0.506	0.865	0.554	0.459	0.917	0.175
Grid	**0.981**	0.896	0.855	0.940	0.796	0.980	0.980
Leather	0.947	0.814	0.773	0.881	0.565	**0.990**	0.870
Tile	**0.999**	0.845	0.996	0.627	0.470	0.925	0.651
Wood	**0.991**	0.891	0.909	0.766	0.666	0.917	0.840
Bottle	**0.990**	0.839	0.980	0.962	0.483	0.906	0.874
Cable	0.787	0.506	**0.814**	0.541	0.626	0.439	0.583
Capsule	**0.925**	0.494	0.709	0.774	0.601	0.552	0.783
Hazelnut	**0.991**	0.946	0.818	0.956	0.695	0.963	0.703
Metal nut	0.891	0.404	**0.941**	0.610	0.680	0.419	0.753
Pill	**0.922**	0.571	0.778	0.700	0.610	0.601	0.686
Screw	**0.860**	0.785	0.437	0.736	0.673	0.645	0.727
Toothbrush	**1.000**	0.536	0.733	0.961	0.822	0.844	0.888
Transistor	0.794	0.712	0.673	0.718	0.624	0.419	**0.829**
Zipper	**0.944**	0.557	0.705	0.739	0.745	0.936	0.604
Mean	**0.932**	0.687	0.800	0.764	0.634	0.764	0.731

Table 5.4: Area under the ROC curve for anomaly classification for each dataset category.

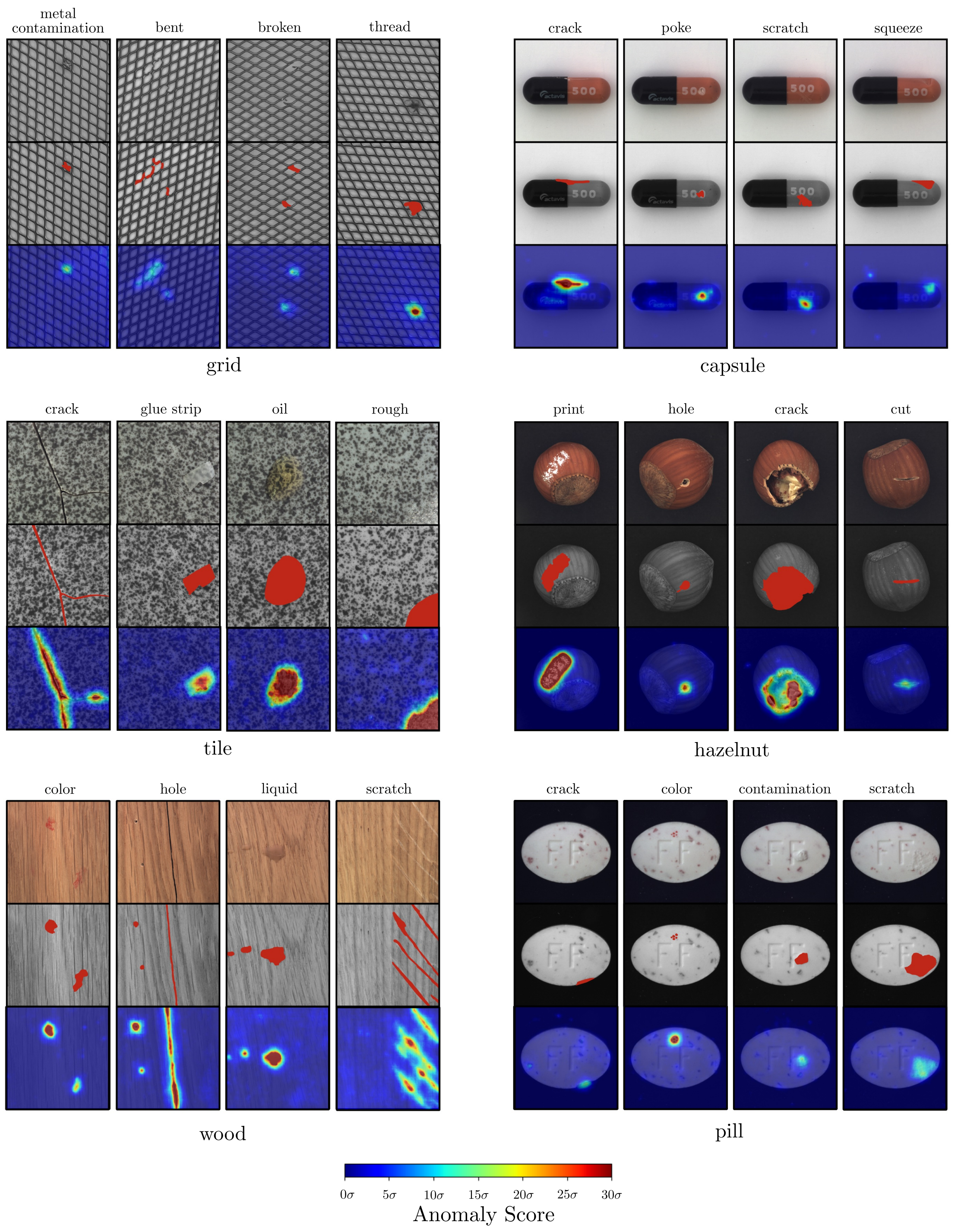


Figure 5.5: Qualitative results of our Student–Teacher method on selected textures (left) and objects (right) of the MVTec Anomaly Detection dataset. Our algorithm performs robustly across various defect categories, such as color defects, contaminations, and structural anomalies. **Top row:** Input images containing defects. **Center row:** Ground truth regions of defects in red. **Bottom row:** Anomaly scores for each image pixel predicted by our algorithm.

Category	Ours $p = 65$	1-NN	OC-SVM	K-Means	L_2-Autoencoder	Variational Autoencoder
Carpet	**0.695**	0.512	0.355	0.253	0.456	0.501
Grid	**0.819**	0.228	0.125	0.107	0.582	0.224
Leather	**0.819**	0.446	0.306	0.308	**0.819**	0.635
Tile	**0.912**	0.882	0.722	0.779	0.897	0.870
Wood	0.725	0.502	0.336	0.441	**0.727**	0.628
Bottle	**0.918**	0.898	0.850	0.495	0.910	0.897
Cable	**0.865**	0.806	0.431	0.513	0.825	0.654
Capsule	**0.916**	0.631	0.554	0.387	0.862	0.526
Hazelnut	**0.937**	0.861	0.616	0.698	0.917	0.878
Metal nut	**0.895**	0.705	0.319	0.351	0.830	0.576
Pill	**0.935**	0.725	0.544	0.514	0.893	0.769
Screw	**0.928**	0.604	0.644	0.550	0.754	0.559
Toothbrush	**0.863**	0.675	0.538	0.337	0.822	0.693
Transistor	0.701	0.680	0.496	0.399	**0.728**	0.626
Zipper	**0.933**	0.512	0.355	0.253	0.839	0.549
Mean	**0.857**	0.640	0.479	0.423	0.790	0.639

Table 5.5: Comparison of the anomaly localization performance of different feature distribution models. The area under the PRO curve up to an average false positive rate per-pixel of 30% for each dataset category is used as a performance measure.

satisfactorily for most of the dataset categories. This shows that their capacity does not suffice to accurately model the large number of available training samples. As was the case in our previous experiment on MNIST and CIFAR-10, 1-NN yields the best results amongst the shallow models. Utilizing a large number of training features together with deterministic autoencoders increases the performance, but still does not match the performance of our approach. Interestingly, the shallow methods fitted to the discriminative embedding of the teacher yield similar performance to current generative methods for anomaly localization such as f-AnoGAN and the SSIM-Autoencoder. This indicates that there is indeed a gap between methods that learn representations for anomaly detection from scratch and methods that leverage discriminative embeddings as prior knowledge.

Anomaly Detection on Multiple Scales. Table 5.6 shows the performance of our algorithm for different receptive field sizes $p \in \{17, 33, 65\}$ and when combining multiple scales. For some objects, such as *bottle* and *cable*, larger receptive fields yield better results. For others, such as *wood* and *toothbrush*, the inverse behavior can be observed. Combining multiple scales enhances the performance for many of the dataset categories and yields the best overall performance. A qualitative example highlighting the benefit of our multi-scale anomaly localization is visualized in Figure 5.6.

Failure Cases. Figure 5.7 shows examples where our method fails to localize anomalies. In particular, our method may fail to detect very small and subtle defects, such as the scratch on the surface of the *pill*. This is because in such cases, the teacher network does not produce descriptors that differ substantially from the ones of the anomaly-

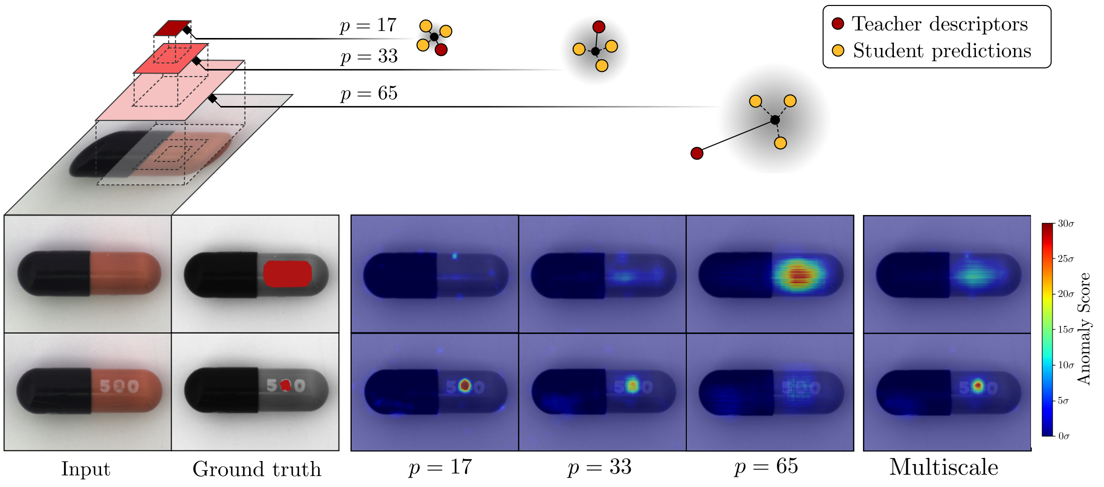


Figure 5.6: Anomaly detection at multiple scales: The architecture with receptive field of size $p = 17$ manages to accurately segment the small scratch on the capsule (top row). However, defects at a larger scale such as the missing imprint (bottom row) become problematic. For increasingly larger receptive fields, the segmentation performance for the larger anomaly increases while it decreases for the smaller one. Our multi-scale architecture mitigates this problem by combining multiple receptive fields.

Category	$p = 17$	$p = 33$	$p = 65$	Multi-Scale
Carpet	0.795	**0.893**	0.695	0.879
Grid	0.920	0.949	0.819	**0.952**
Leather	0.935	**0.956**	0.819	0.945
Tile	0.936	**0.950**	0.912	0.946
Wood	**0.943**	0.929	0.725	0.911
Bottle	0.814	0.890	0.918	**0.931**
Cable	0.671	0.764	**0.865**	0.818
Capsule	0.935	0.963	0.916	**0.968**
Hazelnut	0.971	**0.965**	0.937	0.965
Metal nut	0.891	0.928	0.895	**0.942**
Pill	0.931	0.959	0.935	**0.961**
Screw	0.915	0.937	0.928	**0.942**
Toothbrush	**0.946**	0.944	0.863	0.933
Transistor	0.540	0.611	**0.701**	0.666
Zipper	0.848	0.942	0.933	**0.951**
Mean	0.866	0.900	0.857	**0.914**

Table 5.6: Performance of our algorithm on the MVTec AD dataset for different receptive field sizes p. Combining anomaly scores across multiple receptive fields shows increased performance for many of the dataset's categories. We report the normalized area under the PRO curve up to an average false-positive rate of 30%.

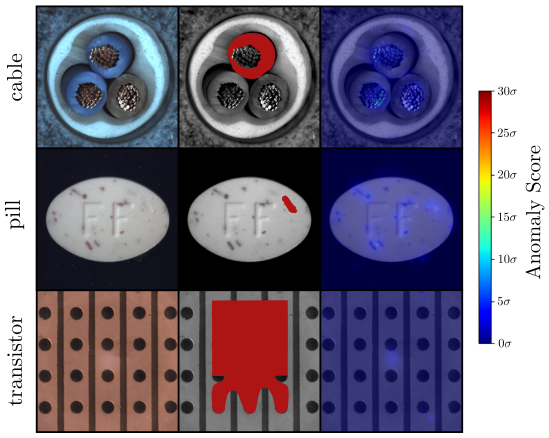


Figure 5.7: Examples of failure cases of our Student–Teacher method. Our method tends to fail on samples with very subtle anomalies, such as the scratch on the *pill*. Furthermore, violations of logical constraints are problematic, e.g., the missing *transistor* or the misplaced wire in the *cable*.

free training data. Furthermore, our method often does not perform well when defects manifest themselved in the form of violations of logical constraints, such as missing objects or objects being placed in invalid locations. Examples for such anomalies are the missing *transistor* or the *cable*, where a yellow wire was replaced by a blue one. Since our method is by design restricted to the analysis of local receptive fields, it produces low anomaly scores for the background patches of the transistor, as well as patches showing a blue wire, since both have appeared in the anomaly-free training set.

5.5 Conclusion

In this chapter, we have introduced a novel framework for the challenging problem of unsupervised anomaly detection in natural images. Anomaly scores are derived from the predictive variance and regression error of an ensemble of student networks, trained against embedding vectors from a descriptive teacher network. Ensemble training can be performed end-to-end and purely on anomaly-free training data without requiring prior data annotation. Our approach can be easily extended to detect anomalies at multiple scales. We demonstrate improvements over recent methods on a number of real-world computer vision datasets for anomaly classification and localization.

Our Student–Teacher method constitutes a significant step forward in the development of methods for unsupervised anomaly detection in 2D images. However, we also found that it may fail to detect anomalies that violate long-range dependencies of the anomaly-free training data. One reason for this is because our method is designed to inspect a large number of local receptive fields independent from each other. In the following two chapters, we address this particular problem in greater detail and propose a possible solution.

6 Logical Constraints in Unsupervised Anomaly Detection

Anomalies manifest themselves in very different ways and an ideal benchmark dataset should contain representative examples for all of them. We find that existing datasets, including our MVTec AD dataset from Chapter 4, are biased towards local structural anomalies such as scratches, dents, or contaminations. In particular, they lack anomalies in the form of violations of logical constraints, e.g., permissible objects occurring in invalid locations. In this chapter, we contribute a new dataset based on industrial inspection scenarios that evenly covers both types of anomalies. We provide pixel-precise ground truth data for each anomalous region and define a generalized evaluation metric that addresses localization ambiguities that can arise for logical anomalies. An initial benchmark on our new dataset reveals that existing approaches tend to be biased towards the detection of one of the two types of anomalies. Furthermore, all methods show considerable room for improvement in the detection of logical anomalies.

The content of this chapter is based on the publication titled *Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization* [Bergmann et al., 2022a].

6.1 Introduction

Deviations from the anomaly-free training data can appear in a variety of ways. On the one hand, entirely new local structures may occur that are not present during training. On the other hand, an image can also be considered anomalous if certain underlying logical or geometrical constraints of the training data are violated. To illustrate the difference between these two, we created a synthetic toy dataset. All anomaly-free images display exactly one black circle at a random location on a flat white background. We introduced two different types of anomalies. The first one is a simple color variation. The second type of anomaly is characterized by the fact that there are two black circles in a single image instead of one.

Numerous existing unsupervised anomaly detection methods model the distribution of local features extracted from pretrained networks. They excel at the detection of anomalies such as the color defect in our toy dataset. They are, however, inherently limited to the information inside the receptive field of their descriptors. This makes it difficult to detect anomalies that violate long-range dependencies. In Figure 6.1, we demonstrate this by considering three test images of our toy dataset, one of which is anomaly-free, one shows a color defect, and one contains an additional circle. The center row shows anomaly maps calculated by our Student–Teacher method presented

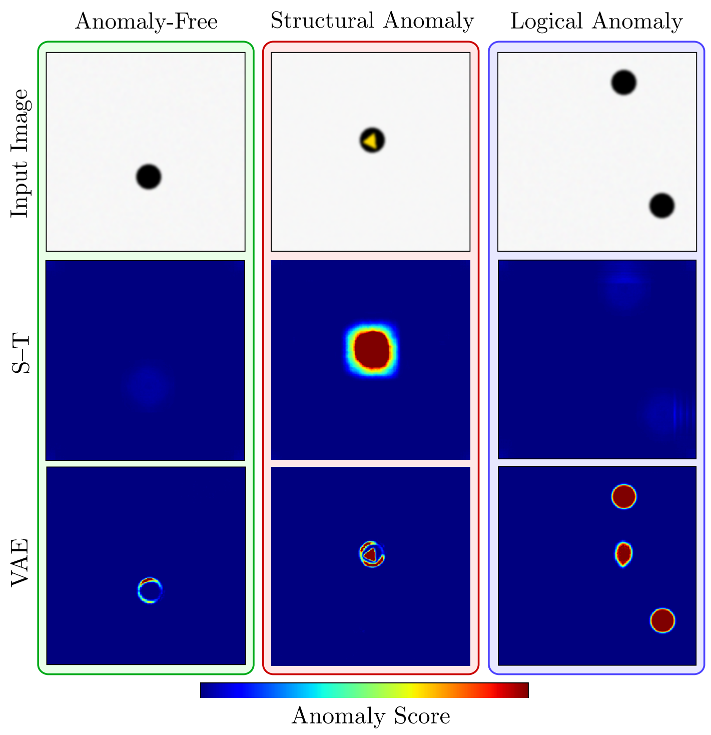


Figure 6.1: Qualitative results of the Student–Teacher (S–T) method and a variational autoencoder (VAE) on a simple toy dataset. Anomaly maps are shown for an anomaly-free image, an image containing a structural anomaly (a color defect), and a logical anomaly (two circles being present instead of one). S–T inspects local image regions and therefore only detects the color defect. The VAE captures the global context of images in its bottleneck. It finds both anomalies, but also produces many false positives due to slight inaccuracies in the reconstructions.

in the previous chapter. It clearly identifies and localizes the color defect. The two circles, however, are not predicted as anomalous because each individual circle does not constitute an anomaly and the receptive field of the method is not large enough to understand the long-range relationships in the image.

Methods that are based on generative models such as Variational Autoencoders (VAEs) [An and Cho, 2015, Vasilev et al., 2020] or Generative Adversarial Networks (GANs) [Goodfellow et al., 2014, Schlegl et al., 2017] have the potential to capture information from the entire image [Liu et al., 2020]. Consequently, they are potentially able to detect anomalies such as the extra black circle in our toy dataset. However, they also tend to produce blurry and inaccurate reconstructions, which leads to an increase in false positives, and are often outperformed by the local methods mentioned above. The bottom row of Figure 6.1 shows anomaly maps calculated by a VAE on our toy dataset. This method accomplishes to identify the two circles as anomalous but produces many false positives in the anomaly-free test image.

Motivated by these observations, we classify an anomaly as either a *structural anomaly* or a *logical anomaly* and demonstrate that existing methods indeed perform very differently on these two classes. We define structural anomalies as new visual structures that occur in locally confined regions and that do not exist in the anomaly-free data. Logical anomalies, on the other hand, violate underlying logical constraints in the data and potentially require a method to capture long-range dependencies. In our toy example, we would classify the color defect as a structural anomaly since the yellow color adds a local structure that has never been observed during training. The additional circle in the top right corner of Figure 6.1 does not introduce any new local structure. The anomaly manifests itself through the violation of the logical constraint that there should always be exactly one circle in the image. Hence, we classify it as a logical anomaly. Note that it is not always straightforward to make a clear distinction between structural and logical anomalies and corner cases may exist.

Existing datasets [Song and Yan, 2013, Carrera et al., 2017, Huang et al., 2018, Bergmann et al., 2021] identify the task of visual inspection of industrially manufactured products as a typical real-world example for unsupervised anomaly detection. All of them focus on the detection of structural anomalies and therefore favor methods that perform well on this type of anomaly. Logical anomalies, however, do occur in manufacturing processes, e.g., as an incorrect wiring of a circuit, a shift in the fill level of a vial, or the absence of an essential component. The development of methods that are capable of detecting logical anomalies is hindered by the availability of suitable data. This creates the need for a dataset that takes both structural and logical anomalies into account with equal importance. We intend to alleviate this need by introducing a new dataset that is also inspired by industrial inspection scenarios but balances the number of logical and structural anomalies.

In summary, our key contributions in this chapter are:

- We introduce a new dataset for the evaluation of unsupervised anomaly detection algorithms that covers both structural and logical anomalies. It contains 3644 images of five distinct object categories inspired by real-world industrial inspection scenarios. Structural anomalies occur as scratches, dents, or contaminations in the manufactured products. Logical anomalies violate underlying constraints, e.g., a permissible object being present in an invalid location or a required object not being present at all.

- To compare the performance of different methods on our dataset, a suitable performance measure is needed. We find that commonly used metrics are not directly applicable to assess the capability of methods to detect logical anomalies. To this end, we introduce a performance metric that takes the different modalities of the defects present in our dataset into account. This performance measure is a generalization of the PRO metric, an established performance measure for unsupervised anomaly detection.

- We conduct an initial benchmark on our new dataset. The results show that existing anomaly detection methods tend to be biased towards the detection of

Category	#Train	# Val	# Test (good)	# Test (structural)	# Test (logical)	# Defect types	Image width	Image height
Breakfast Box	351	62	102	90	83	22	1600	1280
Screw Bag	360	60	122	82	137	20	1600	1100
Pushpins	372	69	138	81	91	8	1700	1000
Splicing Connectors	354	59	119	85	108	21	1700	850
Juice Bottle	335	54	94	94	142	18	800	1600
Total	1772	304	575	432	561	89	-	-

Table 6.1: Statistical overview of the MVTec LOCO AD dataset. For each category, the number of training, validation, and test images is given. Test images are split into anomaly-free images and images that contain structural or logical anomalies. Additionally, the number of different defect types and the image size is reported for each category.

one of the two types of anomalies and that there is particularly large room for improvement in the detection of logical anomalies.

6.2 Datasets for Unsupervised Anomaly Detection

The availability of challenging datasets such as ImageNet [Krizhevsky et al., 2012], MS-COCO [Lin et al., 2014], or Cityscapes [Cordts et al., 2016] has largely contributed to recent successes in various fields of computer vision. In Chapter 4, we showed that for the task of unsupervised anomaly localization comparatively few datasets exist. Here, we show that all of them are primarily designed for the detection of what we refer to as structural anomalies.

Huang et al. [2018] introduce a surface inspection dataset of magnetic tiles. It contains 1344 grayscale images of a single texture. Test images contain various structural anomalies such as cracks or uneven areas. Similarly, Carrera et al. [2017] present NanoTWICE, a dataset of 45 grayscale images of a nanofibrous material acquired by a scanning electron microscope. Anomalies occur in the form of flattened areas or specks of dust. Both datasets only provide textured images, which require a method to focus on local repetitive patterns. Hence, these datasets are inherently unsuited for assessing the ability of a method to capture long-range dependencies and logical constraints.

The Fishyscapes dataset [Blum et al., 2019] is intended to assess the anomaly detection performance of semantic segmentation algorithms for autonomous driving. The task is to train a supervised model on the Cityscapes dataset and, during inference, to localize anomalous objects that were inserted artificially into the test images. The anomalies only consist of objects not present in the training set. This enables their detection based on local, patch-based visual features.

Our MVTec Anomaly Detection dataset presented in Chapter 4 contains 73 types of anomalies, such as contaminations or scratches on manufactured products. The vast majority of anomalies in the dataset matches our definition of structural anomalies. Hence, an evaluation on this dataset alone does not give sufficient insight into how well a method detects logical anomalies.

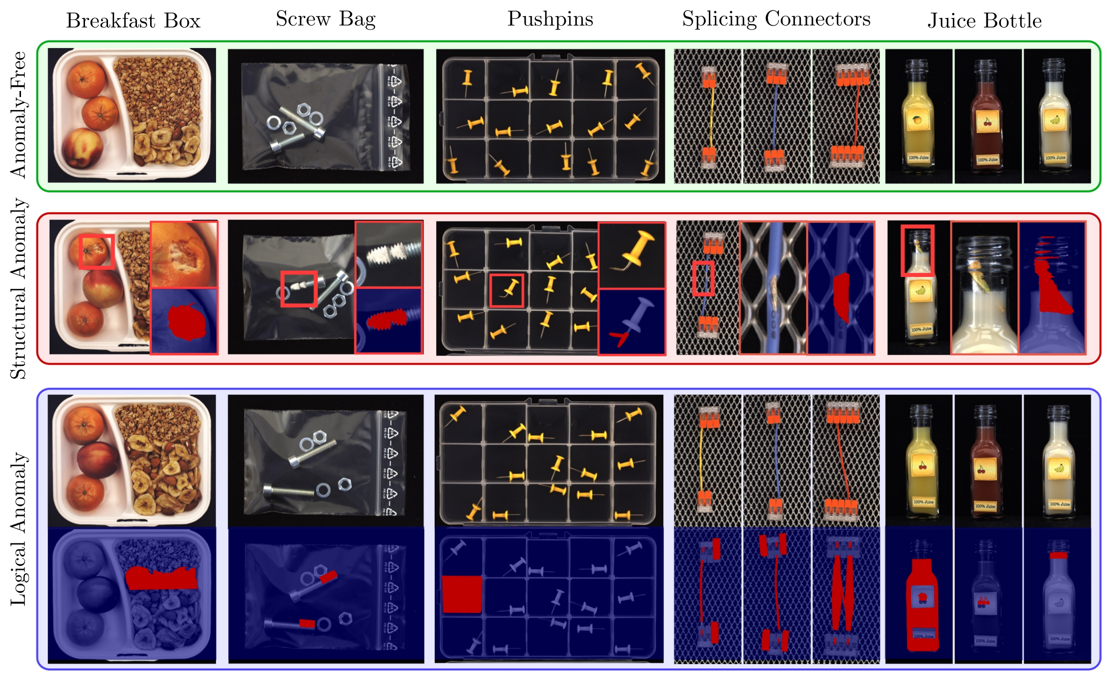


Figure 6.2: Example images of the MVTec LOCO AD dataset for each of the five dataset categories. Each category contains anomaly-free train, validation, and test images. Additional test images contain various structural and logical anomalies. Pixel-precise ground truth annotations are provided for all anomalies.

6.3 The Logical Constraints Anomaly Detection Dataset

To compare the ability of anomaly detection methods to understand logical constraints, we need suitable datasets. As discussed above, few datasets exist for unsupervised anomaly detection in general. Industrial inspection scenarios have been identified as a prime example for unsupervised anomaly detection tasks. This is underlined by the fact that the majority of the existing datasets [Song and Yan, 2013, Carrera et al., 2017, Bergmann et al., 2019b, Huang et al., 2018, Bergmann et al., 2019a] are inspired by such applications.

None of them, however, set an explicit focus on the joint detection of structural and logical anomalies. To this end, we introduce the MVTec **Lo**gical **Co**straints **A**nomaly **D**etection (MVTec LOCO AD) dataset.[1]

6.3.1 Description of the Dataset

MVTec LOCO AD consists of five object categories from industrial inspection scenarios. We provide a total of 1772 images for training, 304 for validation, and 1568 for testing. Figure 6.2 shows example images for each of the dataset categories. The training sets

[1]`https://www.mvtec.com/company/research/datasets/mvtec-loco`

consists of only anomaly-free images. Machine learning methods typically require data for validating their performance during training or for adjusting hyperparameters. To ensure that the choice of the validation data does not add a bias to evaluations and benchmarks, we define a specific validation set. Like the training images, the validation images are free of any anomalies. The test set contains anomaly-free images and images with various types of logical and structural anomalies. All three sets are independent of each other in the sense that they consist of images of distinct physical objects and that there is no overlap between them. An overview of the image statistics of our dataset is shown in Table 6.1, including the number and size of training, validation, and test images as well as the number of different defect types for each category.

Each dataset category possesses certain logical constraints that need to be fulfilled. Anomaly-free images of the category *breakfast box* always contain exactly two tangerines and one nectarine that are always located on the left-hand side of the box. Furthermore, the ratio and relative position of the cereals and the mix of banana chips and almonds on the right-hand side are fixed. A *screw bag* contains exactly two washers, two nuts, one long screw, and one short screw. Each compartment of the box of *pushpins* contains exactly one pushpin. Exactly two *splicing connectors* with the same number of cable clamps are linked by exactly one cable. In addition, the number of clamps has a one-to-one correspondence to the color of the cable and the cable has to terminate in the same relative position on its two ends such that the whole construction exhibits a mirror symmetry. Each *juice bottle* is filled with one of three differently colored liquids and carries exactly two labels. The first label is attached to the center of the bottle and displays an icon that determines the type of liquid. The second is attached to the lower part of the bottle with the text "100% Juice" written on it. The fill level is the same for each bottle. Violations to any of these constraints constitute logical anomalies.

The third row of Figure 6.2 shows examples of logical defects, which manifest themselves in the following ways. The breakfast box contains too many banana chips and almonds. The screw bag contains two long screws and lacks a short one. One compartment of the box of pushpins does not contain any pushpin. For the splicing connectors, we show three different types of defects. On the left, the two splicing connectors do not have the same number of clamps, in the center, the color of the cable does not match the number of clamps and, on the right, the cable terminates in different positions. We also present three different types of defects for the juice bottle. On the left, the icon does not match the type of juice. In the middle, the icon is slightly misplaced. Finally, on the right the fill level of the bottle is too high.

The center row of Figure 6.2 depicts examples of structural anomalies. They manifest themselves as a damaged tangerine, a broken screw, a bent pushpin, a corrupt insulation of a cable, and a contamination inside a juice bottle.

6.3.2 Annotations and Labeling Policies

For all anomalies present in the dataset, we provide pixel-precise ground-truth annotations. Structural anomalies are typically straightforward to annotate. Each pixel of an anomalous image that introduces a local visual structure that is not present in the anomaly-free images is marked as anomalous. In the example of the damaged tangerine in Figure 6.2, all pixels that fall into the damaged region are annotated. Labeling logical defects, however, proves to be a challenging task. As an example, Figure 6.2 depicts a pushpin missing in one of the compartments. Consider two methods, one that marks the whole compartment as anomalous, while the other one only marks a region with the size and shape of a pushpin inside the compartment. In this case, one would probably consider both methods as equally successful.

Our labeling policy and the newly introduced evaluation metric take such ambiguities into account. In our dataset, the union of all areas of the image that could potentially be the cause for the anomaly is labeled as anomalous. However, to achieve a perfect score, a method is not necessarily required to predict the whole ground truth area as anomalous.

To reflect this, we introduce a generalization of the per-region overlap. To calculate the PRO metric, real-valued anomaly scores are thresholded to obtain a binary prediction for each pixel in the test set. Then, the percentage of correctly predicted pixels is computed for each annotated defect region in the ground-truth. The average over all defects yields the final PRO value. Recall that PRO is very similar to computing the average true positive rate (TPR) over all pixels. The advantage of PRO is that it weights defect regions of different size as equally important.

In our dataset, we do not necessarily require a method to segment all pixels of an annotated area. Continuing the example of the missing pushpin, it is sufficient for a method to segment an area the size of one pushpin within the empty compartment. To meet this requirement, we propose a generalized version of the PRO metric that saturates once the overlap with the ground truth exceeds a certain saturation threshold.

6.3.3 The Saturated Per-Region Overlap (sPRO)

Let $C_{i,j} = \{(i, \boldsymbol{p}) \mid \boldsymbol{p} \in K_j\}$ denote the set of pixels classified as anomalous for a connected component $K_j \subseteq D$ in the ground truth image with index i and let $s_{i,j}$ be corresponding saturation thresholds such that $0 < s_{i,j} \leq |C_{i,j}|$. For a set P_{ano} of pixels in the dataset that are predicted as anomalous, we define the saturated per-region overlap (sPRO) as

$$\text{sPRO} = \frac{1}{k} \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} \sum_{j=1}^{k_i} \min\left(\frac{|C_{i,j} \cap P_{\text{ano}}|}{s_{i,j}}, 1\right), \tag{6.1}$$

where k_i denotes the number of ground truth components for a single test image and $k = \sum_i k_i$ is the total number of components in $\mathcal{D}_{\text{test}}$. Note that this is indeed a generalization of the PRO metric because sPRO = PRO if $s_{i,j} = |C_{i,j}|$ for all (i, j). An illustrative example of the sPRO metric with a single ground-truth region is shown in Figure 6.3. Here, one pushpin is missing in one of the box compartments. The annotated area comprises the entire compartment while the saturation threshold s is set to the predetermined size of a single pushpin, which is much smaller. Hence, all

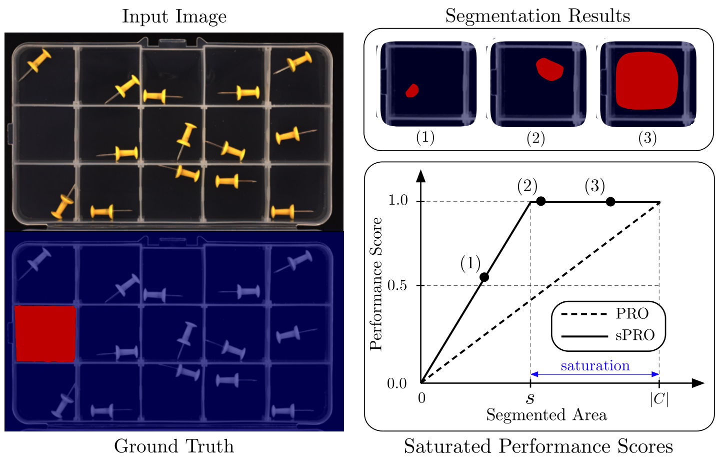


Figure 6.3: Schematic illustration of the introduced sPRO evaluation metric. For an annotated ground truth component, a saturation threshold s is selected. Once the overlap of the predicted region with the ground truth exceeds s, we consider the anomaly segmentation task solved.

predictions P for which the overlap with the ground truth exceeds s fully solve the segmentation task.

Similar to the TPR and PRO, sPRO does not take false positive predictions into account. Hence, we report its value together with the associated false positive rate (FPR). False positive predictions are defined as all pixels that are predicted as anomalous but are not covered by any annotated region. To obtain evaluation results that are independent of the binarization value used to turn real-valued anomaly scores into binary predictions, we make use of the sPRO curve. It is created analogously to the common PRO or ROC curves by computing the sPRO value for various binarization thresholds and plotting them against the corresponding FPR value. As our main performance measure, we compute the area under the sPRO curve up to a limited false positive rate and normalize it to obtain a score between 0 and 1. This is motivated by the fact that anomaly segmentation results at large false positive rates are no longer meaningful. They should, therefore, be excluded from the computation of a performance measure such as the area under the sPRO curve.

6.3.4 Selection of Saturation Thresholds

We selected suitable saturation thresholds for each of the 89 individual defect types that occur in our dataset and provide them alongside with our dataset. The following paragraphs provide further details on our labeling process and the selection of saturation thresholds for various types of anomalies.

Structural anomalies: For structural anomalies, the entire annotated area should be segmented. We therefore set $s = |C|$, which yields the original PRO metric. An example is the broken screw in the second row of Figure 6.2, for which the entire broken area should be segmented as anomalous.

Missing objects: For missing objects, we annotated the entire area in which the object could potentially occur. The corresponding saturation threshold is chosen to be equal to the area of the missing object. We determined the distribution of the area of an object in our dataset by manually annotating numerous instances of the same object. We then selected a value for s from the lower end of this distribution. In the bottom row of Figure 6.2, a pushpin is missing in one of the compartments. Since pushpins can potentially occur at every location in the compartment, its entire area is annotated. The corresponding saturation threshold is set to the size of a single pushpin.

Additional objects: Some test images contain too many instances of an object. In such cases, all instances of the object are annotated. The saturation threshold is set to the area of the extraneous objects. An example is shown in the second row of Figure 6.5, where an additional cable is present between the two splicing connectors. Since it is not clear which of the two cables represents the anomaly, we annotate both of them. The corresponding saturation threshold is set to the area of one cable, i.e., half of the annotated region. On the one hand, this allows a method to obtain a perfect score even if it only marks one of the two cables as an anomaly. On the other hand, a method which marks both of them is not penalized.

Violation of other logical constraints: Besides the presence of additional or the absence of required objects, our MVTec LOCO AD dataset contains various test images that violate a different form of logical constraints. One example is shown in the last row of Figure 6.2, where the juice bottle filled with orange juice carries the label of the cherry juice. Both the orange juice and the label with the cherry are present in the training set. The logical anomaly arises due to the erroneous combination of the two in the same image. One could either mark the area filled with juice or the cherry as anomalous. Hence, our annotation is given by the union of the two regions. Since the segmentation of the cherry within the image is sufficient to solve the anomaly localization task, s is selected as the area of the cherry.

6.4 Benchmark

We conduct an initial benchmark of established methods for anomaly localization on our MVTec LOCO AD dataset. In particular, we evaluate a deterministic autoencoder (AE), a variational autoencoder (VAE), and the memory-guided autoencoder (MNAD). All autoencoders localize anomalies by an L_2-comparison of the input with its reconstruction. We further evaluated f-AnoGAN as a representative for GAN-based methods. For methods that leverage features of pretrained networks, we evaluated SPADE as well as the Student–Teacher anomaly detection model. As an additional baseline, we in-

Category	Vertical flip	Horizontal flip	Random rotation	Color jitter
Breakfast Box			✓	✓
Screw Bag	✓	✓	✓	✓
Pushpins	✓	✓	✓	✓
Splicing Connectors	✓	✓	✓	✓
Juice Bottle			✓	✓

Table 6.2: Overview of the dataset augmentation techniques applied during training to each of the object categories present in our dataset.

cluded the Variation Model. For a brief description of the evaluated methods, we refer to Section 2.4.

6.4.1 Dataset Augmentation

To facilitate the training of data-hungry deep learning models, we designed the acquisition of our dataset in a way that permits an easy augmentation of the images. In our experiments, we used the following image augmentations:

- Vertical flip with probability $\frac{1}{2}$.
- Horizontal flip with probability $\frac{1}{2}$.
- Random rotation by up to 3° around the center of the image.
- Random jitter of brightness, contrast, and saturation of the image.

Not all of these augmentations are suited for every type of object in our dataset. We provide an overview of the augmentations applied to each object in Table 6.2. The augmented datasets were used for all three autoencoders and f-AnoGAN. SPADE, the Student–Teacher model, and the Variation Model did not require augmented data.

6.4.2 Experiment Setup

Here, we give detailed information on the training and evaluation for each method.

Deterministic and Variational Autoencoders: For both autoencoders, we use the base architecture as listed in Table 6.3. For the VAE, the last convolution layer of the encoder is duplicated to estimate the variance. We trained for 500 epochs using Adam with an initial learning rate of 10^{-4}, a weight decay of 10^{-5}, and a batch size of 16. During inference, anomaly scores are derived by a pixelwise comparison of the input images and their reconstructions.

f-AnoGAN: We used the publicly available implementation from the original authors.[2] As required by their method, we zoomed all images to size 64×64 pixels and converted them to grayscale prior to training and evaluation.

[2]`https://github.com/tSchlegl/f-AnoGAN`

Layer	Output Size	Parameters		
		Kernel	Stride	Padding
Input	256x256x3			
Conv1	128x128x32	4x4	2	1
Conv2	64x64x32	4x4	2	1
Conv3	32x32x64	4x4	2	1
Conv4	16x16x64	4x4	2	1
Conv5	8x8x64	4x4	2	1
Conv9	1x1x32	8x8	1	0

Table 6.3: General outline of the autoencoder architecture. The depicted values correspond to the structure of the encoder. The decoder is built as a reversed version of this. Batch normalization and leaky rectified linear units with slope 0.05 are applied after each layer except for the output layers of both the encoder and the decoder.

For the training of the GAN, the dimension of the latent space was set to 128. The optimization was done using Adam with an initial learning rate of 10^{-4}, no weight decay, and a batch size of 64. The GAN was trained for 100 epochs. After each training iteration of the generator, the discriminator was trained for 5 iterations.

The training of the encoder network was performed with an initial learning rate of 5×10^{-5}, no weight decay, and a batch size of 64 and runs for 5×10^{4} iterations. During inference, anomaly scores are derived by a pixelwise comparison between the input and the reconstructed image.

MNAD: We used the publicly available implementation from the original authors[3] with a small modification. Instead of predicting a future video frame, we implemented the reconstruction of the original input images. The memory module was initialized with 10 memory items of dimension 512. The output dimension of the image encoder was set to 32. For optimization, we used Adam with an initial learning rate of 2×10^{-5}, no weight decay, and a batch size of 4. The weights for feature compactness and feature separateness were set to $\lambda_c = \lambda_s = 10^{-1}$. The training was run for 500 epochs in reconstruction mode on images of size 256 $\times$ 256 pixels.

SPADE: We used our own implementation of the SPADE method. As a feature extractor, we used a Wide ResNet50-2 pretrained on ImageNet. The images were zoomed to a size of 224 $\times$ 224. For feature extraction, we used the last convolution layers of the first, second, and third block of the network. For the image-level nearest-neighbor computation, we used $K = 50$ nearest neighbors. On the pixel-level we used $\kappa = 1$ nearest neighbors. The resulting anomaly maps were smoothed using a Gaussian filter with $\sigma = 4$.

Student–Teacher: We used our own implementation of the Student–Teacher method. All images were zoomed to a size of 256 $\times$ 256 pixels prior to training and evaluation. The student networks were trained with 3 different receptive fields of sizes $p \in \{17, 33, 65\}$

[3]`https://github.com/cvlab-yonsei/MNAD`

Method	Breakfast Box	Screw Bag	Pushpins	Splicing Connectors	Juice Bottle	Mean
VM	0.168	0.253	0.254	0.125	0.325	0.225
f-AnoGAN	0.223	0.348	0.336	0.195	0.569	0.334
MNAD	0.080	0.344	0.357	0.442	0.472	0.339
AE	0.189	0.289	0.327	0.479	0.605	0.378
VAE	0.165	0.302	0.311	0.496	0.636	0.382
SPADE	0.372	0.331	0.234	0.516	0.804	0.451
S–T	**0.496**	**0.602**	**0.523**	**0.698**	**0.811**	**0.626**

Table 6.4: Quantitative results on the MVTec LOCO AD dataset. The normalized area under the sPRO curve up to an average false positive rate per pixel of 5% is computed separately for the structural and logical anomalies. The table reports the mean of both values. The best-performing method is highlighted in boldface.

pixels. For each receptive field, we used an ensemble of 3 students, which resulted in a total of 9 trained models per object category. For optimization, we used Adam with an initial learning rate of 10^{-4}, weight decay of 10^{-5}, and a batch size of 1. As anomaly score, we evaluated the predictive variance of the student networks and their regression errors with respect to the pretrained teacher network.

Variation Model: The Variation Model calculates the mean and standard deviation at each pixel location over the entire training set of each object in our dataset. This works best if the images show aligned objects. In the MVTec LOCO AD dataset, the *breakfast boxes* are already aligned. We aligned the *pushpins* and *juice bottles* using shape-based matching [Steger, 2001, 2002]. The *screw bags* and *splicing connectors* were not transformed at all for our experiments.

The pixels of the anomaly map show the absolute difference of the test image to the mean of the training images in multiples of the standard deviation of the training images. This is done separately for each channel and we obtained the overall anomaly map as the average over all channels. If a spatial transformation is applied during inference, some pixels might not overlap with the mean and deviation images. For such pixels, no meaningful anomaly score can be computed and we therefore set it to the minimum attainable value of 0.

6.4.3 Experiment Results

To assess the difference in performance between the detection of structural and logical anomalies, we split the test set into two subsets. Each subset exclusively contains defective test images with structural or logical anomalies, respectively. The anomaly-free test images are included in both sets. For each subset, we computed the area under the sPRO curve up to a false positive rate of 0.05. The joint localization performance for both types of anomalies was measured by the average of the two individual areas.

Table 6.4 shows the results of all evaluated methods for each of the five dataset categories. Similar to the results on the MVTec AD dataset in Chapter 4, the methods based on pretrained features, i.e., SPADE and our Student–Teacher method, perform better than the methods trained from scratch. The latter achieves the best performance

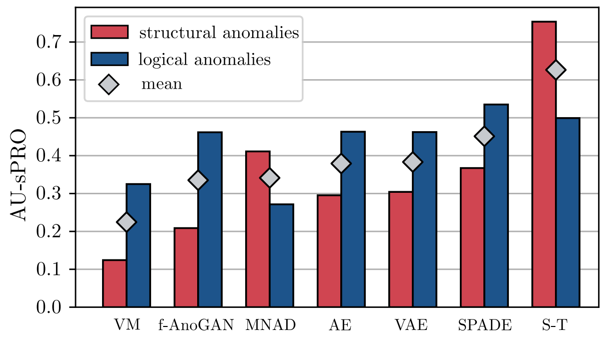


Figure 6.4: Difference in anomaly localization performance for both structural and logical anomalies on the MVTec LOCO AD dataset.

on all of the five dataset categories. However, all methods leave considerable room for future improvement.

Figure 6.4 displays the performance of each method when structural or logical anomalies are treated separately. All evaluated methods show a bias towards the detection of one type of anomaly. Our Student–Teacher method, for example, exhibits a strong bias towards the detection of structural anomalies. Regarding the detection of logical anomalies, it is no longer the best-performing method and its localization accuracy is comparable to generative models such as the AE, VAE, and f-AnoGAN.

Very similar results are obtained when assessing the anomaly classification performance of each method on our new dataset, as shown in Figure 6.6. We derive image-level anomaly scores for each evaluated method by computing the maximum anomaly score over all pixels in a given anomaly map. We then report the area under the ROC curve for each dataset category, again separating logical and structural anomalies. The Student–Teacher method performs best in the detection of structural anomalies. However, its performance on the logical ones is significantly lower. The other methods show a relatively balanced classification performance between logical and structural anomalies.

Figure 6.5 provides additional qualitative results for all evaluated methods. Anomaly images are shown for four test images of our MVTec LOCO AD dataset. Two of them contain structural anomalies, i.e., the flipped splicing connector and the contamination in the juice bottle. The other two contain logical anomalies, i.e., the additional red cable between the two splicing connectors and the banana logo on the bottle filled with orange juice.

While the Student–Teacher approach detected the structural anomalies reliably, it failed to detect the logical anomalies due to its limited receptive field. The SPADE method, on the other hand, failed to detect the flipped splicing connector, while it managed to localize the remaining three anomalies. The deterministic and the variational autoencoder both yielded large residuals in the parts of the images that are challenging to reconstruct, e.g., on the cables between the two splicing connectors. While the memory module in MNAD reduced the number of false positive predictions and improved upon the basic autoencoders in the detection of structural anomalies, it failed to detect the

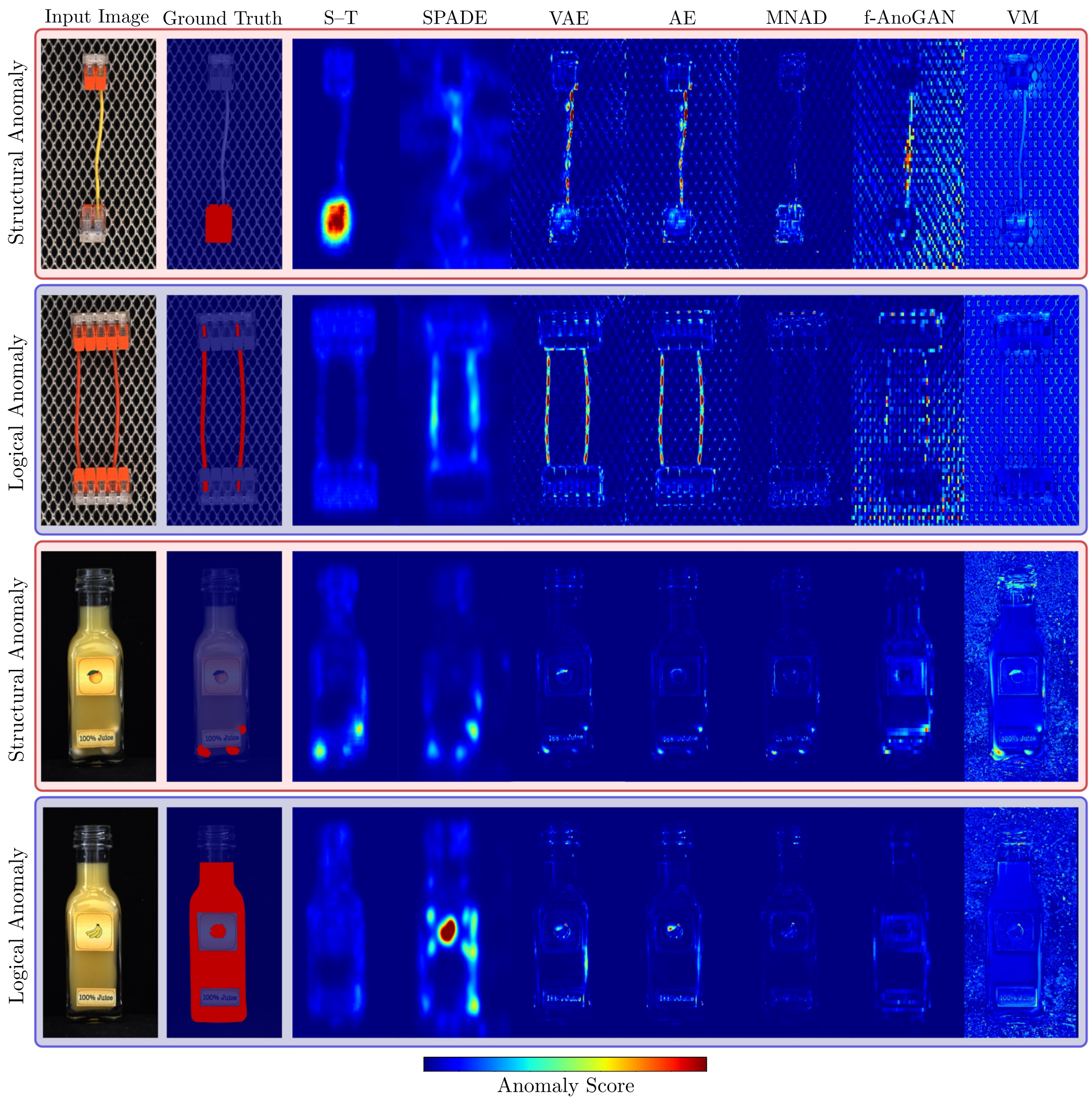


Figure 6.5: Qualitative results for each evaluated method on our MVTec LOCO AD dataset. The first and third row contain examples of structural anomalies, i.e. the flipped connector and the contamination in the juice bottle. The second and third row contain examples of logical anomalies, i.e., a second cable being present between the two connectors and the banana label on the bottle filled with orange juice.

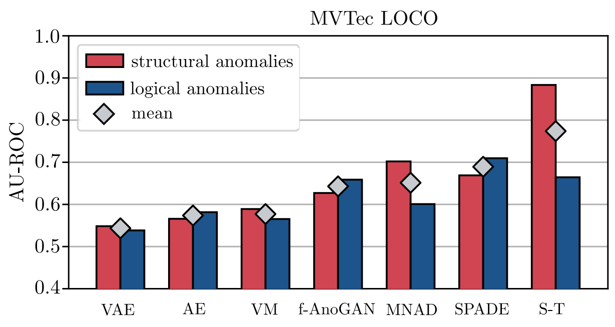


Figure 6.6: Image-level classification results on the presented MVTec LOCO dataset (top) and the MVTec AD dataset (bottom).

logical anomalies. Similar to the deterministic and variation autoencoders, f-AnoGAN yielded numerous false positive predictions in areas that are difficult to accurately reconstruct. The Variation Model requires a pixel-precise alignment of the inspected objects. Since this is not possible for the splicing connectors, it did not perform well for this dataset category. For the juice bottle, it managed to detect parts of the structural anomaly.

6.5 Conclusion

This chapter is based on the observation that anomalies in natural images can manifest themselves in many different ways. We defined two categories of anomalies, which we call structural and logical anomalies. Previous work predominantly concentrated on the development of datasets and methods for the detection of structural ones. We therefore created a new dataset for the unsupervised localization of anomalies that focuses on the detection of both structural and logical anomalies. Pixel-precise ground truth annotations are provided for each anomalous test image. Furthermore, we introduced a new performance metric that takes the different modalities of the two anomaly types into account.

We conducted an initial benchmark of existing methods for unsupervised anomaly detection on our new dataset. Our results showed that existing methods tend to be biased towards the detection of one of the two types of anomalies and that there remains considerable room for improvement, especially for the detection of logical anomalies. In the following chapter, we will introduce a new method that performs significantly better in the joint detection of structural and logical anomalies.

7 Global Context Anomaly Detection

In this chapter, we introduce **G**lobal **C**ontext **A**nomaly **D**etection (**GCAD**), a new method for the unsupervised localization of anomalies that sets a new state of the art in the joint detection of structural and logical anomalies. It consists of a local and a global network branch. The local branch is based on our Student–Teacher method. It inspects confined regions independent of their spatial locations in the input image and is primarily responsible for the detection of entirely new local structures. The global branch learns a globally consistent representation of the training data through a bottleneck that enables the detection of violations of long-range dependencies, a key characteristic of many logical anomalies. Extensive evaluations on MVTec LOCO and MVTec AD show the superiority of our approach in the detection of logical anomalies, as well as in the combined localization of both anomaly types.

The content of this chapter is based on the publication titled *Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization* [Bergmann et al., 2022a].

7.1 Introduction

The results from the previous chapter indicate that while our Student–Teacher method performs well in the detection of structural anomalies, it is by design limited to the inspection of locally confined regions of the anomaly-free training data. This makes it inherently unsuited to detect violations of long-range dependencies that extend beyond its receptive field. Competing methods based on generative models, such as convolutional autoencoders or GANs, have the potential to capture these global constraints in their low-dimensional latent variable. However, their performance is limited by their tendency to produce inaccurate reconstructions. In this chapter, we contribute a new method for the unsupervised pixel-precise localization of anomalies that combines the strengths of compression-based models and methods based on local features extracted from pretrained networks. This enables us to improve the joint detection of structural and logical anomalies compared to prior art.

Our method consists of a local and a global network branch, each of which we show to be primarily responsible for the detection of structural and logical anomalies, respectively. The local branch is based on the Student–Teacher framework from Chapter 5, which was the best performing method in the detection of structural anomalies in our initial benchmark on the MVTec LOCO dataset.

The global branch intends to overcome the difficulty to capture the entire context of an input image. Similar to autoencoders, it consists of a global feature encoder network

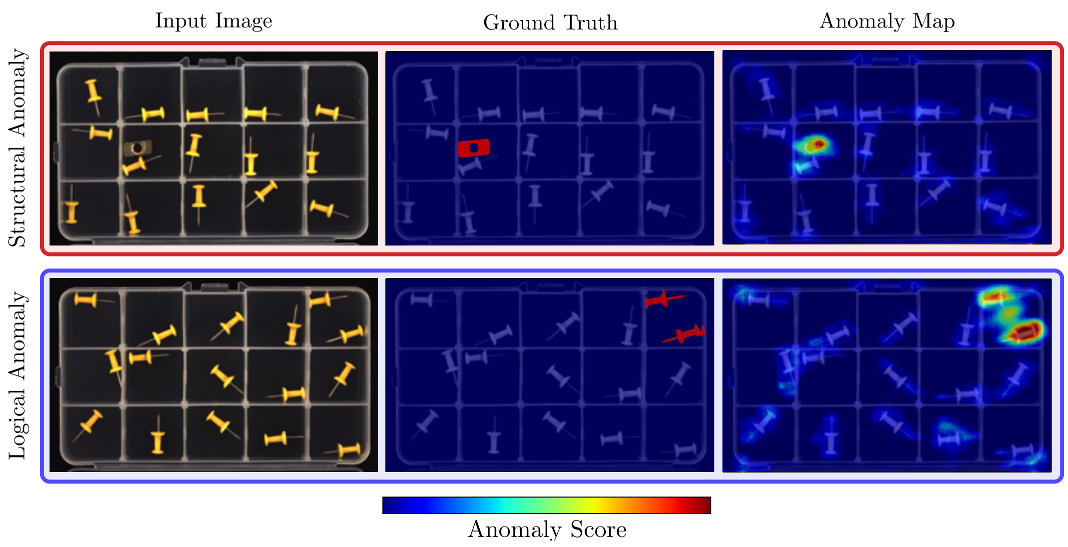


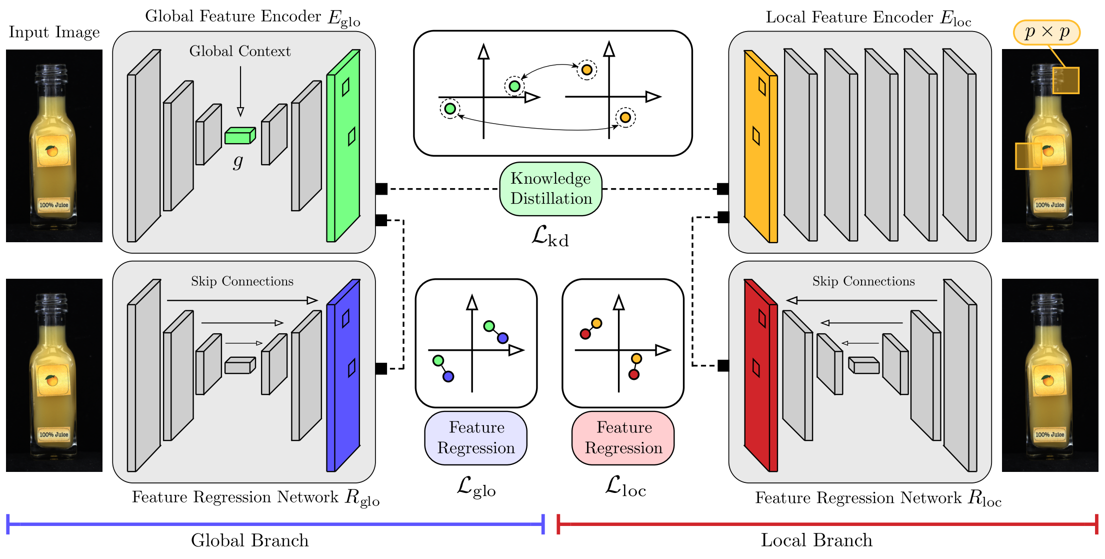
Figure 7.1: Qualitative results of our GCAD method on two samples of the MVTec LOCO dataset. The top and bottom rows show an example of a structural and a logical anomaly, respectively. Our proposed method successfully localizes the anomaly in both images.

that produces a globally consistent representation of the training data through a low-dimensional bottleneck. Simultaneously, a high-capacity regression network attempts to reproduce this feature encoding, while disregarding any underlying global constraints. During inference, deviations between the two networks indicate anomalies. Instead of evaluating per-pixel reconstruction errors within the global feature encoder directly, shifting the anomaly score computation to a learned feature space facilitates the accurate matching of higher-dimensional features through a low-dimensional bottleneck, which greatly reduces the number of false positives.

Extensive evaluations against state-of-the-art methods show the superiority of our approach in the detection of logical anomalies, as well as in the combined localization of both anomaly types. Figure 7.1 shows an illustrative example of our new method applied to samples from the MVTec LOCO dataset.

7.2 Description of Our Method

Given a training dataset $\mathcal{D}_{\text{train}}$ of anomaly-free images, our goal is to localize anomalies in test images $\mathcal{D}_{\text{test}}$ by assigning a real-valued anomaly score to each image pixel. All images $I : D \to \mathbb{R}^C$ possess the same domain D and the same number of channels C. Our method consists of two main branches, one of which is primarily responsible for the localization of structural anomalies and the other one for the localization of logical anomalies. The following paragraphs give details about the two branches and highlight the characteristics that enable them to detect the two different anomaly types. A schematic overview of our approach is given in Figure 7.2.



Input Image
Global Feature Encoder E_{glo}
Global Context
g
Knowledge Distillation
$\mathcal{L}_{\text{kd}}$
Local Feature Encoder E_{loc}
$p \times p$
Skip Connections
Feature Regression
$\mathcal{L}_{\text{glo}}$
Feature Regression
$\mathcal{L}_{\text{loc}}$
Skip Connections
Feature Regression Network R_{glo}
Feature Regression Network R_{loc}
Global Branch
Local Branch

Figure 7.2: Schematic overview of our approach. A global feature encoder E_{glo} is trained against descriptors from a pretrained local feature encoder E_{loc} through a bottleneck to capture the global context of the anomaly-free training data. Each encoder is assigned a high-capacity regression network R_{glo} and R_{loc}, respectively, that matches the output of its respective feature encoder. The joint training of E_{glo} and R_{glo} facilitates the accurate matching of higher-dimensional features through a low-dimensional bottleneck.

Local Model Branch. Our first branch is motivated by the recent success of anomaly detection methods that model the distribution of local features extracted from pretrained CNNs. Such methods achieve state-of-the-art performance on established anomaly localization benchmarks, in which the majority of anomalies match our definition of structural anomalies. In particular, we base this branch of our model on the Student–Teacher method. Since this method computes anomaly scores for locally confined image regions independent of their spatial position in the input image, we refer to this branch as the *local branch* of our model.

It consists of an encoder network E_{loc} which is pretrained on a large number of image patches cropped from the ImageNet dataset. During pretraining, E_{loc} is encouraged to extract expressive descriptors for local image patches via knowledge distillation from a pretrained classification network. In particular, we employ the same feature extractor as used in our Student–Teacher method, which distills the knowledge of a ResNet-18 [He et al., 2016] classifier trained on ImageNet into a dense patch descriptor network via fast dense feature extraction [Bailer et al., 2017]. A detailed description of the network architecture of E_{loc} and the pretraining protocol on ImageNet can be found in Chapter 5. After pretraining, the weights of E_{loc} remain fixed when optimizing our anomaly detection model. Formally, E_{loc} produces a descriptor of dimension d_{loc} at each pixel location, i.e., $E_{\text{loc}}(I) \in \mathbb{R}^{H \times W \times d_{\text{loc}}}$. Each feature describes a local patch of size

$p \times p$ within the original input image. This is achieved by choosing an architecture for E_{loc} with a limited receptive field.

The local branch additionally contains a regression network R_{loc} that is initialized with random weights and is trained to match the output of E_{loc} on the anomaly-free training data. It outputs a feature map of a shape identical to the one produced by E_{loc}, i.e., $R_{\text{loc}}(I) \in \mathbb{R}^{d_{\text{loc}} \times h \times w}$. We use a high-capacity network with skip connections for this task and minimize the squared L_2-norm:

$$\mathcal{L}_{\text{loc}}(R_{\text{loc}}) = \frac{1}{B} \sum_{i=1}^{B} \| E_{\text{loc}}(I_i) - R_{\text{loc}}(I_i) \|_2^2 \, . \tag{7.1}$$

Here, B denotes the size of the current minibatch. If, during inference, an image contains novel local structures that have not been observed during training and that fall within the receptive field of the pretrained feature extractor, E_{loc} will produce novel local descriptors with which R_{loc} is unfamiliar. This leads to large regression errors. Hence, we expect the local branch of our model to perform well in the detection of structural anomalies.

Global Model Branch. E_{loc} inspects only a limited receptive field of size $p \times p$ pixels and, in particular, does not encode the positional composition of the extracted training features. Hence, our local branch is inherently ill-suited for the detection of anomalies that violate long-range dependencies, which is characteristic for many logical anomalies such as missing or additional objects in the input image. To compensate for this, we add a second branch to our model that analyzes the global context of the entire input image. Therefore, we refer to this branch as the *global branch* of our model. Its design is inspired by the observation in the previous chapter that methods that compress the input data to a low-dimensional bottleneck possess the ability to capture logical constraints and fail to reproduce input images that violate them.

Our global branch consists of two networks, E_{glo} and R_{glo}. The first is an encoder network that produces a descriptor of dimension d_{glo} at each pixel location, $E_{\text{glo}}(I) \in \mathbb{R}^{H \times W \times d_{\text{glo}}}$. Similar to an autoencoder, E_{glo} is encouraged to produce feature maps that are globally consistent with the training data. To this end, E_{glo} produces its encoding over a low-dimensional bottleneck of dimension g. Contrary to autoencoders, E_{glo} does not reconstruct the input image. It is trained by distilling the knowledge of the local feature encoder E_{loc} into the global branch. To let the descriptors of E_{glo} match the output dimension of E_{loc}, we introduce an upsampling network U that performs a series of 1×1 convolutions. For training, we minimize

$$\mathcal{L}_{\text{kd}}(E_{\text{glo}}, U) = \frac{1}{B} \sum_{i=1}^{B} \| E_{\text{loc}}(I_i) - U(E_{\text{glo}}(I_i)) \|_2^2 \, . \tag{7.2}$$

In principle, anomaly scores could be computed by comparing the features of E_{loc} directly to those of $U \circ E_{\text{glo}}$. However, our ablation studies show that the high-dimensional and detailed feature maps of E_{loc} can only be approximately reproduced by E_{glo} due to

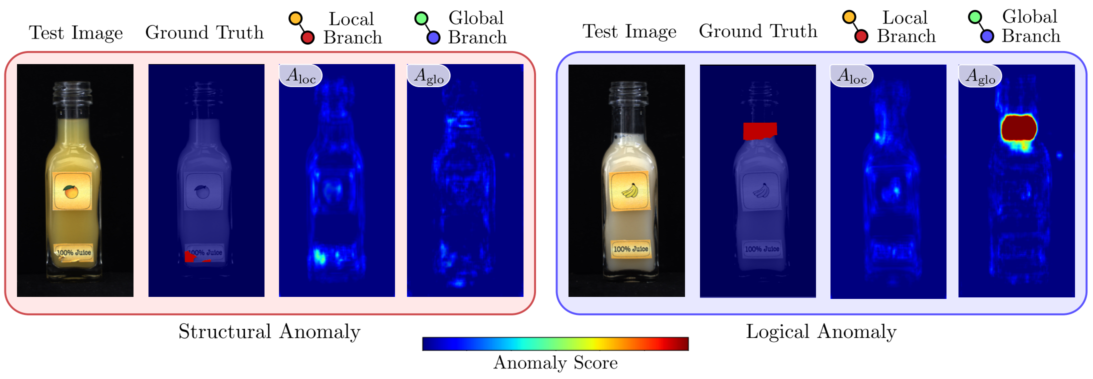


Figure 7.3: Visualization of anomaly maps A_{loc} and A_{glo} for a structural and a logical anomaly. The damaged label in the left column is an example of a structural anomaly. The local branch is able to detect this type of anomaly while the global one does not contribute much information. In the right column, a wrong fill level constitutes a logical anomaly. The local branch is not able to detect this because no new local structure is present in the image. Since the global branch takes the entire image content into account, it is able to successfully segment the anomalous region.

its low-dimensional bottleneck. A direct comparison would lead to many false positives in the anomaly images due to inaccurate feature reconstructions. To circumvent this problem, the second network of the global branch, R_{glo}, is trained to match the output of E_{glo} using the loss term

$$\mathcal{L}_{\text{glo}}(E_{\text{glo}}, R_{\text{glo}}) = \frac{1}{B} \sum_{i=1}^{B} \| E_{\text{glo}}(I_i) - R_{\text{glo}}(I_i) \|_2^2 . \tag{7.3}$$

R_{glo} is intended to accurately transform local image regions into the corresponding feature vectors without taking into account the underlying logical constraints of the training data. To make this possible, R_{glo} does not contain any bottleneck and is designed as a high-capacity network with skip connections.

The difference in architecture between E_{glo} and R_{glo} is crucial to our method. The high capacity of R_{glo} allows it to accurately reproduce the features of E_{glo}, which reduces the number of false positive detections compared to the reconstruction error between E_{glo} and $U \circ E_{\text{glo}}$. The skip connections enable R_{glo} to solve the regression task without capturing the global context of the training data. Thus, the outputs of E_{glo} and R_{glo} differ for anomalous test images that violate global constraints. This allows the localization of logical anomalies that require the analysis of long-range dependencies.

Combination of the Two Branches. We train the whole model end-to-end using the sum of the individual loss terms normalized by the respective depth of the matched features, i.e.,

$$\begin{aligned}\mathcal{L}(R_{\text{loc}}, R_{\text{glo}}, E_{\text{glo}}, U) = {} & \tfrac{1}{d_{\text{loc}}}\mathcal{L}_{\text{kd}}(E_{\text{glo}}, U) \\ & + \tfrac{1}{d_{\text{glo}}}\mathcal{L}_{\text{glo}}(E_{\text{glo}}, R_{\text{glo}}) \\ & + \tfrac{1}{d_{\text{loc}}}\mathcal{L}_{\text{loc}}(R_{\text{loc}}).\end{aligned} \tag{7.4}$$

Due to the joint optimization of $\mathcal{L}_{\text{kd}}$ and $\mathcal{L}_{\text{glo}}$, the global feature encoder is encouraged to learn meaningful descriptors for the training data and simultaneously output a representation that can be easily matched by the feature regression network. Computing residuals in a learned feature space facilitates the accurate matching of higher-dimensional features through a low-dimensional bottleneck.

Scoring Functions for Anomaly Localization. During inference, pixelwise anomaly scores for a test image $J : D \to \mathbb{R}^C$ can be computed by comparing the features of the image encoder networks to the features of the respective regression network, i.e., by computing $A_{\text{loc}} = ||E_{\text{loc}}(J) - R_{\text{loc}}(J)||_2^2$ and $A_{\text{glo}} = ||E_{\text{glo}}(J) - R_{\text{glo}}(J)||_2^2$, respectively. Here, the norm is taken over the respective feature dimension, i.e., d_{loc} and d_{glo}. Large regression errors indicate anomalous pixels. A_{loc} is mainly responsible for detecting structural anomalies, while A_{glo} enables the network to detect logical anomalies, as illustrated in Figure 7.3. Since the weights of both E_{glo} and R_{glo} are randomly initialized, there exists no training incentive for the two networks to behave differently for structural anomalies. Our experiments show that the global branch is indeed mainly responsible for the detection of logical anomalies, while the local branch performs much better at the detection of structural anomalies.

To obtain an anomaly map for the entire model, we calculate A_{loc} and A_{glo} for all images in the validation set after training the model. We then compute the respective means, μ_{loc} and μ_{glo}, and standard deviations, σ_{loc} and σ_{glo}, of all resulting scores. During inference, we normalize the individual anomaly maps and define the combined anomaly map by $A = \frac{A_{\text{loc}} - \mu_{\text{loc}}}{\sigma_{\text{loc}}} + \frac{A_{\text{glo}} - \mu_{\text{glo}}}{\sigma_{\text{glo}}}$. Note that the validation set of our dataset only contains anomaly-free images. Here, we use the corresponding anomaly images only to adjust the scale of anomaly scores of the two network branches.

Anomaly Detection on Multiple Scales. The choice of the receptive field p for E_{loc} can have a significant impact on the anomaly localization performance, especially when anomalies vary greatly in size. To be less dependent on the particular choice of the receptive field, we train multiple models with varying values of p. The anomaly maps of the models are combined by computing their pixelwise average. Let P be the set of all evaluated receptive fields and $A^{(p)}$ be the anomaly map obtained from a model with receptive field $p \in P$. The maps of different receptive fields are combined by computing $\frac{1}{|P|}\sum_{p \in P} A^{(p)}$.

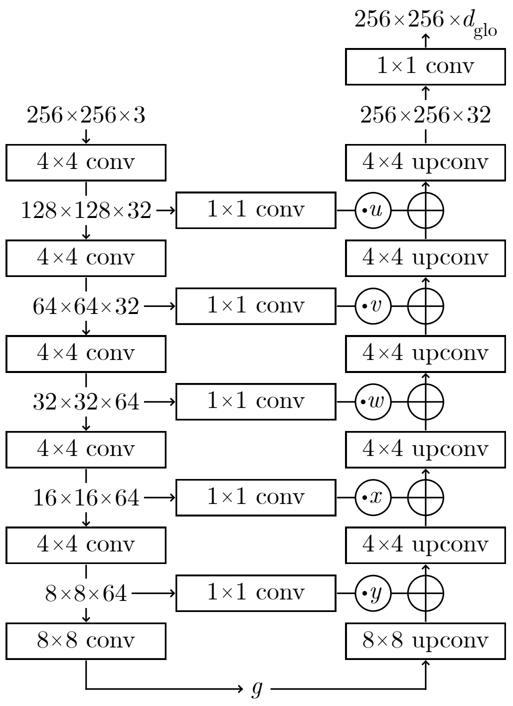


Figure 7.4: Architecture of E_{glo} with a g-dimensional bottleneck. Transposed convolutions are denoted by "upconv." All 4×4 convolutions use a stride of 2 and are followed by a Leaky ReLU activation. The 1×1 convolutions in the skip connections have the same number of feature maps in their output as in their input. Their outputs are scaled by the respective skip weight. Then, they are added element wise to the output of the corresponding transposed convolution.

7.3 Experiments on the MVTec LOCO AD Dataset

Here, we give detailed information on the training and evaluation of our GCAD method on the MVTec LOCO AD dataset. We then compare its performance to the methods benchmarked in the previous chapter.

7.3.1 Training and Evaluation Protocol

All input images were zoomed to $H = W = 256$ pixels. For optimization, we used Adam [Kingma and Ba, 2015] with an initial learning rate of 10^{-4} and a weight decay of 10^{-5}. We trained our method on the augmented training images for 500 epochs, following the same augmentation strategy as described in the previous chapter. E_{glo} outputs feature maps of depth $d_{\text{glo}} = 10$ and the capacity of its global context vector was set to $g = 32$. For E_{loc}, we used the same network architecture and training protocol as in our Student–Teacher method described in Chapter 5. Its feature vectors are of depth $d_{\text{loc}} = 128$. We trained our method using two receptive fields of sizes $p \in \{17, 33\}$ and combined their outputs for multi-scale anomaly detection.

Figure 7.4 shows the architecture of the global feature encoder E_{glo}. We initialized the five skip weights u, v, w, x, and y with a value of 1 prior to training. Then, we linearly decreased the skip weights after each epoch, starting with the upper levels. After 100 epochs, all skip weights were set to a value of 0, meaning that information could only flow through the g-dimensional bottleneck. We empirically observed that progressively fading out the weights of the skip connections facilitated the optimization, yielding lower values of $\mathcal{L}_{\text{kd}}$ on the training and validation set.

The d_{glo}-dimensional features output by E_{glo} are transformed into d_{loc}-dimensional features by an upsampling network U. It consists of three 1×1 convolutions with non-linearities in between. The output of U is matched with the descriptors given by the pretrained network E_{loc}. Finally, the two regression networks R_{loc} and R_{glo} follow a U-Net architecture [Ronneberger et al., 2015]. We use a publicly accessible implementation[1] with five downsampling blocks, five upsampling blocks, and a latent dimension of size $16 \times 16 \times 1024$.

Prior to training, we normalize the features of the pretrained network E_{loc}. For each of the d_{loc} feature dimensions, we compute the mean and the standard deviation of all descriptors on the training dataset. We then update the weights in the final layer of E_{loc} to output normalized features.

For the first 50 epochs, we only trained the global feature encoder E_{glo}, the upsampling network U, and the local regression network R_{loc}. In the remaining 450 epochs, the global regression network R_{glo} was optimized as well.

7.3.2 Experiment Results

Table 7.1 shows the anomaly localization performance of all evaluated methods on our dataset for each dataset category. Our method consistently outperforms all other evaluated methods for all but one of the dataset categories. The top chart in Figure 7.5 displays the localization performance of all methods when structural or logical anomalies are treated separately. Unlike most other methods, GCAD does not show a bias towards the detection of one type of anomaly. Our method significantly outperforms all other approaches in the detection of logical anomalies, while maintaining a high performance at the detection of structural anomalies. In particular, our method performs best when considering the average performance for both anomaly types. Qualitative results of our method for structural and logical anomalies are shown in Figure 7.6 and Figure 7.7.

The bottom bar chart in Figure 7.5 shows the results when considering the task of anomaly classification. We derive image-level anomaly scores by finding the maximum anomaly score over all pixels in a given anomaly map and compute the area under the ROC curve. Similar to our experiments on anomaly localization, our GCAD method performs significantly better than all other evaluated methods in the detection of logical and the joint detection of structural and logical anomalies.

Figure 7.8 shows some failure cases of our method. GCAD may fail when anomalies are very small in size, e.g., for the broken pushpin in the top left compartment. It

[1] `https://github.com/jvanvugt/pytorch-unet`

Method	Breakfast Box	Screw Bag	Pushpins	Splicing Connectors	Juice Bottle	Mean
VM	0.168	0.253	0.254	0.125	0.325	0.225
f-AnoGAN	0.223	0.348	0.336	0.195	0.569	0.334
MNAD	0.080	0.344	0.357	0.442	0.472	0.339
AE	0.189	0.289	0.327	0.479	0.605	0.378
VAE	0.165	0.302	0.311	0.496	0.636	0.382
SPADE	0.372	0.331	0.234	0.516	0.804	0.451
S–T	0.496	**0.602**	0.523	0.698	0.811	0.626
GCAD	**0.502**	0.558	**0.739**	**0.798**	**0.910**	**0.701**

Table 7.1: Quantitative results on the MVTec LOCO AD dataset. The normalized area under the sPRO curve up to an average false positive rate per pixel of 5% is computed separately for the structural and logical anomalies. The table reports the mean of both values. The best-performing method is highlighted in boldface.

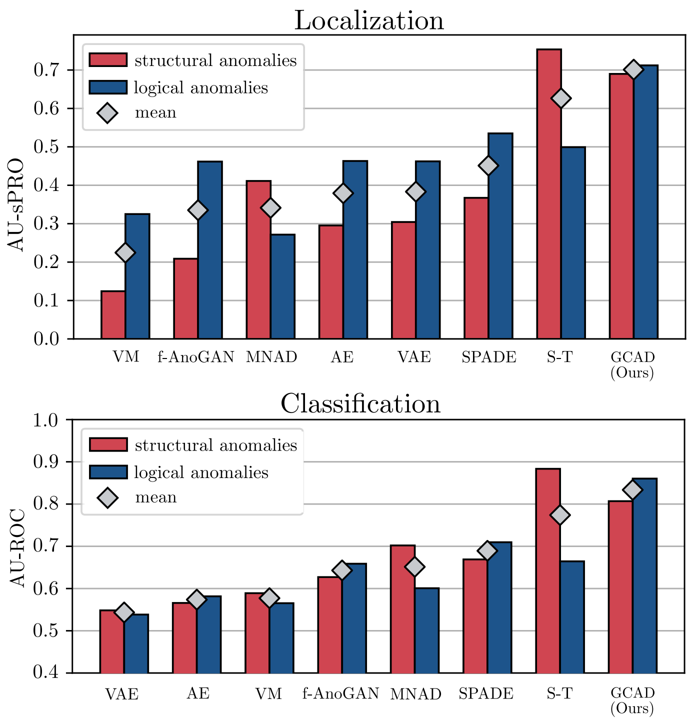


Figure 7.5: Difference in anomaly detection performance for both structural and logical anomalies on the MVTec LOCO AD dataset. The top chart shows the performance for anomaly localization measured by the area under the sPRO curve. The bottom chart shows the performance for anomaly classification in terms of the area under the ROC curve.

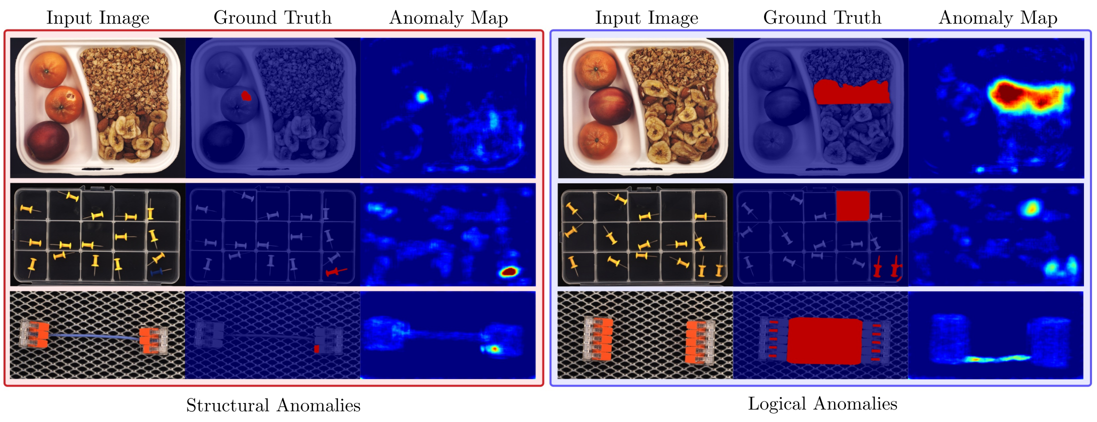


Figure 7.6: Qualitative results of our method on the MVTec LOCO AD dataset for both structural and logical anomalies. The damaged tangerine, the blue pushpin, and the broken connector are structural anomalies. The wrong ratio of cereals and banana chips in the breakfast box, the additional yellow pushpin, and the missing cable between the two connectors constitute logical anomalies.

can also fail to capture very challenging logical constraints, such as enforcing a fixed number of objects that can potentially appear almost anywhere in the input image. The second row of Figure 7.8 depicts such an example in which the *screw bag* contains an additional washer. We show a third failure case of our method in which anomalies manifest themselves in very subtle and intricate differences compared to the anomaly-free images. In the last row of Figure 7.8, no almonds are mixed into the banana chips in the bottom right compartment.

7.3.3 Ablation Studies

To assess the sensitivity of our method with respect to the chosen hyperparameters, we performed various ablation studies. The results are shown in Figure 7.9.

Global Context Dimension. We begin by analyzing the impact of the global context dimension g of the global feature encoder E_{glo}. When the dimension of the latent space was too small, E_{glo} struggled to output meaningful feature maps and the anomaly detection performance declined for both types of anomalies. When the global context dimension was increased, the overall detection performance was not affected substantially. However, we observed a slight decline in the detection of logical anomalies, while the localization of structural anomalies improved. The increased capacity of E_{glo} led to fewer false positives in the global anomaly detection branch while, at the same time, E_{glo} captured the global context of the data less reliably. This is due to the fact that choosing a latent dimension that is too large allows the global feature encoder to copy parts of its input directly into the latent representation. This phenomenon can also be observed in other bottleneck architectures, such as autoencoders. While the mean performance

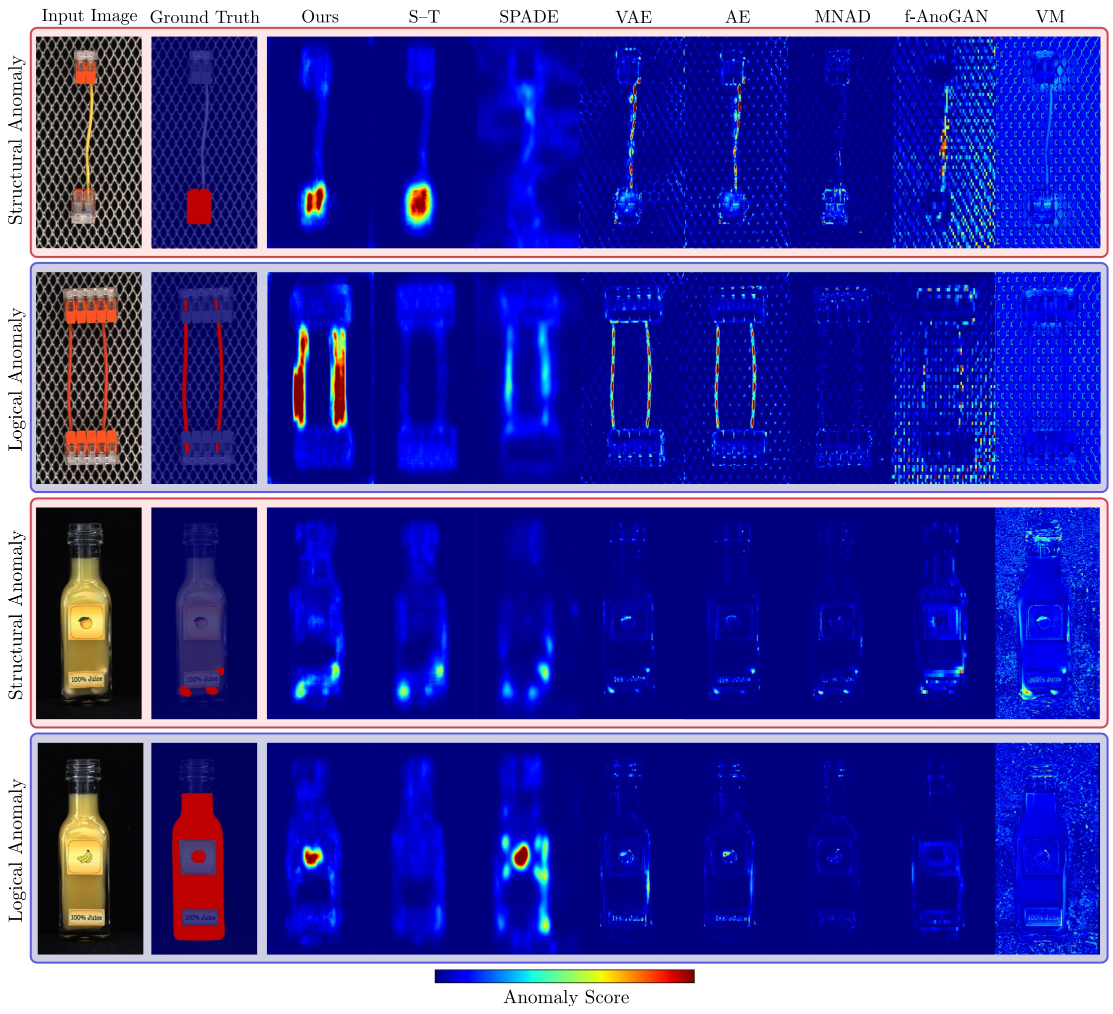


Figure 7.7: Qualitative results for each evaluated method on our MVTec LOCO AD dataset. The first and third row contain examples of structural anomalies, i.e. the flipped connector and the contamination in the juice bottle. The second and third row contain examples of logical anomalies, i.e., a second cable being present between the two connectors and the banana label on the bottle filled with orange juice.

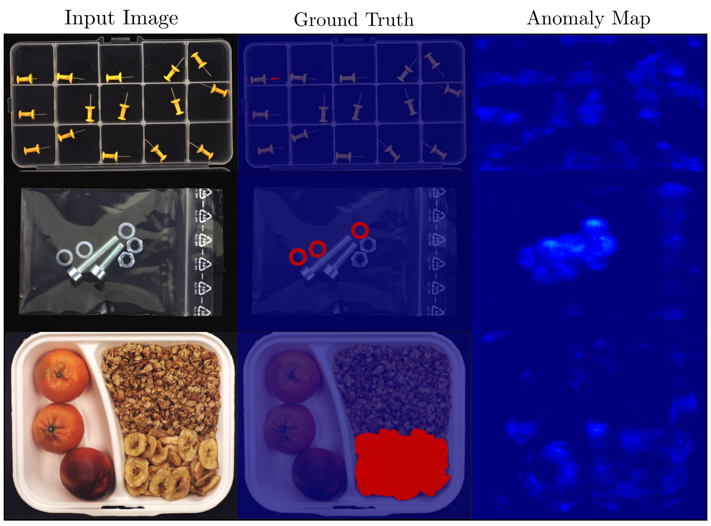


Figure 7.8: Qualitative examples for which our GCAD method fails to localize anomalies.

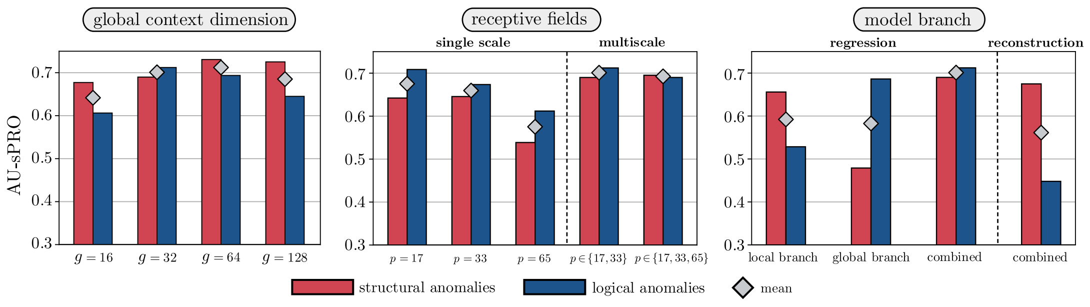


Figure 7.9: Performance of our algorithm when varying different hyperparameters during training or evaluation.

is slightly better for $g = 64$, the best balance between the detection of structural and logical anomalies is achieved for $g = 32$.

Receptive Field. We also assessed the performance of our proposed method with respect to the size of the receptive field of the local feature encoder E_{loc}. Figure 7.9 shows the difference in performance when evaluating our approach for single receptive fields of sizes 17, 33, and 65, as well as when combining the anomaly images of multiple receptive fields together. Our method yielded a similar mean performance for receptive fields of size 17 and 33, while the performance dropped for very large values of p. When combining multiple receptive fields together, the performance for both structural and logical anomaly detection could be enhanced.

Model Branch. Figure 7.9 also evaluates the responsibility of the different branches of our method with respect to the anomaly localization performance. We compared the

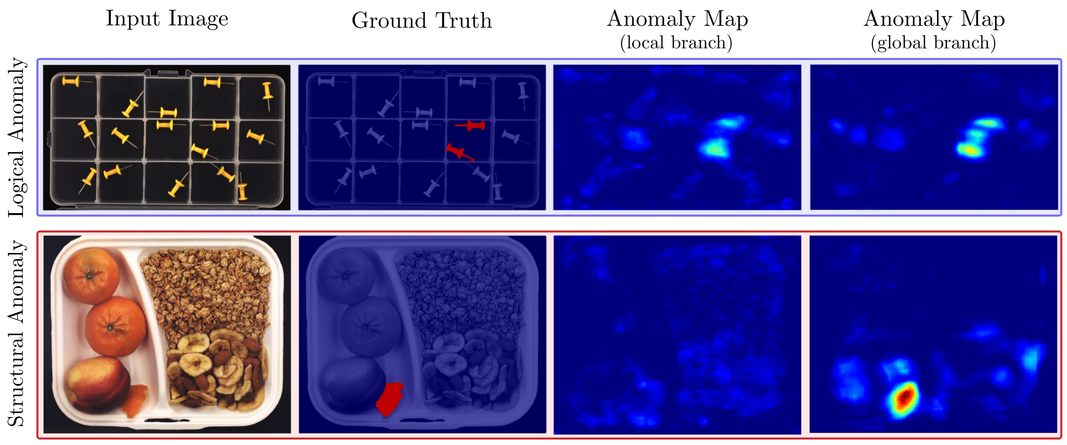


Figure 7.10: Qualitative examples for which the local branch works better in the detection of a logical anomaly than the global branch and vice versa. In the top row, the global branch produces more false positive predictions than the local branch in the detection of the two pushpins. In the bottom row, the local branch fails to localize the contamination in the breakfast box.

performance of the local anomaly maps A_{loc} to that of the global anomaly maps A_{glo} and found that, indeed, A_{loc} performed much better in the detection of structural anomalies. While our local branch is similar to the Student–Teacher approach, we do not train a computationally expensive ensemble to additionally evaluate the intrinsic uncertainty of R_{loc}. This comes at a small cost of structural anomaly detection performance. A_{glo}, on the other hand, yielded a better performance on the logical anomalies. Combining A_{loc} and A_{glo} improved the performance for both structural and logical anomalies.

This indicates that some of the logical anomalies are better detected by the local branch and some structural ones by the global branch. We illustrate this in Figure 7.10. Certain logical anomalies can be detected by the local branch, e.g., two pushpins being present in a single compartment, since both pushpins fall into the receptive field of the local feature extractor E_{loc}. The global branch also detects this anomaly. However, it also tends to produce more false positive predictions than the local branch since it has to reconstruct the entire input image over a low dimensional bottleneck. In this case, the global branch benefits from the performance of the local branch on this logical anomaly. There also exist cases in which the global branch contributes to a better detection of structural anomalies. In the bottom row of Figure 7.10, a piece of a tangerine is present as a contamination in the breakfast box. Since the texture of the contamination matches that of a tangerine, the local branch does not detect this anomaly. The global branch, however, analyzes the entire image context and can encode that there are already two tangerines present in the input image. Therefore, it manages to localize this structural anomaly.

Feature Regression vs. Reconstruction. Next, we compare using R_{glo} for the detection of logical anomalies to simply evaluating the reconstruction error of the global feature

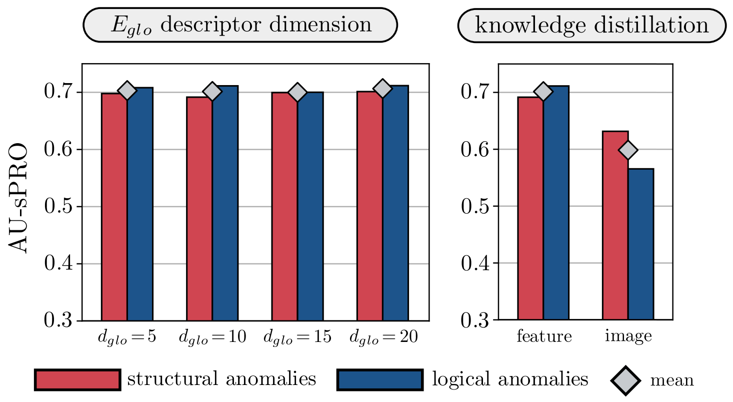


Figure 7.11: The left barplot examines the impact of the output dimension of E_{glo} on the anomaly localization performance. The plot on the right shows the difference in performance for different knowledge distillation targets.

encoder E_{glo} with respect to the pretrained features after upsampling. Figure 7.9 shows that evaluating the reconstruction error performed significantly worse than our feature regression approach. This is because the reconstruction of 128-dimensional pretrained features through a small bottleneck is challenging and leads to many false positives. Our approach circumvents this problem by shifting the feature matching task to a lower-dimensional, learned feature space.

Descriptor Dimension of E_{glo}. We investigate the impact of the output dimension d_{glo} of the global feature encoder E_{glo} on the anomaly detection performance. The plot on the left-hand side of Figure 7.11 indicates that our method performed well for various values of d_{glo} and is not highly sensitive to this parameter.

Knowledge Distillation. Finally, we assess the benefit of distilling knowledge of pretrained descriptors into the global branch of our method. For comparison, we distilled knowledge from the original input images by changing $\mathcal{L}_{\text{kd}}$ to

$$\mathcal{L}_{\text{kd}}(E_{\text{glo}}, U) = \frac{1}{B} \sum_{i=1}^{B} \|I_i - U(E_{\text{glo}}(I_i))\|_2^2 \,. \tag{7.5}$$

The plot on the right-hand side of Figure 7.11 shows that the distillation of features from pretrained networks into E_{glo} greatly enhanced the anomaly localization performance for both structural and logical anomalies.

Variation of the Integration Limit. We further investigate the performance of our method when computing the sPRO metric up to various integration limits. The results are listed in Table 7.2. Our GCAD method yields the overall best performance, independent of the particular chosen integration limit. While the performance of each method

Method	AU-sPRO$_{0.01}$	AU-sPRO$_{0.05}$	AU-sPRO$_{0.10}$	AU-sPRO$_{0.30}$	AU-sPRO$_{1.00}$
VM	0.086	0.225	0.314	0.493	0.740
f-AnoGAN	0.152	0.334	0.442	0.624	0.827
MNAD	0.176	0.339	0.447	0.643	0.853
AE	0.166	0.378	0.499	0.699	0.882
VAE	0.162	0.382	0.506	0.705	0.884
SPADE	0.225	0.451	0.587	0.790	0.927
S–T	0.402	0.626	0.717	0.836	0.937
GCAD (Ours)	**0.462**	**0.701**	**0.787**	**0.891**	**0.962**

Table 7.2: Area under the sPRO curve for different integration limits.

seemingly increases with larger limits, we would like to stress that results at large false positive rates are often not meaningful in practice, as already motivated in Chapter 4. Therefore, we recommend to evaluate the performance on our MVTec LOCO dataset at integration limits no larger than 0.3.

Variation of Saturation Thresholds. In this paragraph, we analyze the sensitivity of the sPRO metric with respect to the manually selected saturation thresholds. We evaluated each method in our benchmark ten times with thresholds sampled uniformly from an interval ranging from 0.5 to 1.5 times the original threshold. In case of defects for which the saturation threshold was chosen to be equal to the annotated area, we did not vary the threshold. The ranking of the evaluated methods was stable across all ten runs, with the exception of two runs in which two methods switched between the sixth and seventh rank.

7.4 Experiments on the MVTec AD Dataset

In addition to the ones on our MVTec LOCO AD dataset, we performed experiments on MVTec AD. We split all test images of the dataset into two subsets. The first contains only images with defects that match our definition of structural anomalies. The second comprises all images that contain at least one logical anomaly. Of the 1258 anomalous test images, we identified 37 to contain defects that match our definition of logical anomalies. We list them in Table 7.3. For each of the logical anomalies, the saturation threshold for the sPRO metric was chosen to be the whole area of the ground truth label. We performed a separate evaluation of each method on structural and logical anomalies, respectively. For all methods, we used the same hyperparameters as on the MVTec LOCO AD dataset. The data augmentation strategies for each evaluated object are listed in Table 7.4.

The top bar chart in Figure 7.12 shows the anomaly localization performance of each evaluated method on MVTec AD. The results are similar to the ones on the MVTec LOCO AD dataset. Our method outperformed all other methods at the combined detection of structural and logical anomalies. The Student–Teacher approach performed slightly better at the detection of structural anomalies. However, its performance dropped significantly for the logical anomalies in the dataset. As for MVTec LOCO, we also compute the AU-ROC values for the image-level classification task on

Category	Defect Name	# Images	Image IDs
cable	cable swap	12	all images
	combined	3	{5, 7, 9}
capsule	faulty imprint	2	{4, 5}
transistor	cut lead	10	all images
	misplaced	10	all images

Table 7.3: Images of the MVTec AD dataset that match our description of logical anomalies.

Category	Vertical flip	Horizontal flip	Random rotation	Color jitter
Bottle	✓	✓	✓	✓
Cable			✓	✓
Capsule			✓	✓
Carpet	✓	✓	✓	✓
Grid	✓	✓	✓	✓
Hazelnut	✓	✓	✓	✓
Leather	✓	✓	✓	✓
Metal Nut			✓	✓
Pill			✓	✓
Screw	✓	✓	✓	✓
Tile	✓	✓	✓	✓
Toothbrush		✓	✓	✓
Transistor		✓	✓	✓
Wood	✓	✓	✓	✓
Zipper	✓	✓	✓	✓

Table 7.4: Overview of the dataset augmentation techniques applied during training to each of the object categories present in the MVTec AD dataset.

the MVTec AD dataset. The results are shown in the bottom bar chart in Figure 7.12. While our proposed method performs slightly worse in the detection of structural anomalies than the Student–Teacher method and SPADE, it excels in the classification of the logical anomalies on this dataset.

Figure 7.13 shows qualitative results for all evaluated methods on the object categories *transistor* and *cable*. The first and third row contain examples of structural anomalies, i.e., a damaged transistor surface and bent wires in a cable cross section. The other two show examples of logical anomalies. In the second row, the transistor is entirely missing. In the fourth row, the top yellow cable has been replaced by a blue one. Our method reliably detects all four defects. The Student–Teacher model performs well on the structural anomalies but entirely fails to localize the logical anomalies due to its limited receptive field. The SPADE method manages to detect the missing transistor and performs well on the structural anomalies, but has difficulties to localize the more subtle logical anomaly in the image of the cable. All methods based on autoencoders tend to yield increased anomaly scores on the structural anomalies. However, they also produce many false positives in areas that are difficult to accurately reconstruct, i.e., the reflections on the wires of the cable. Both the VAE and the deterministic AE show a tendency to detect both of the logical anomalies. This is not the case for MNAD, for which the high-capacity memory module allows to reconstruct the areas that contain

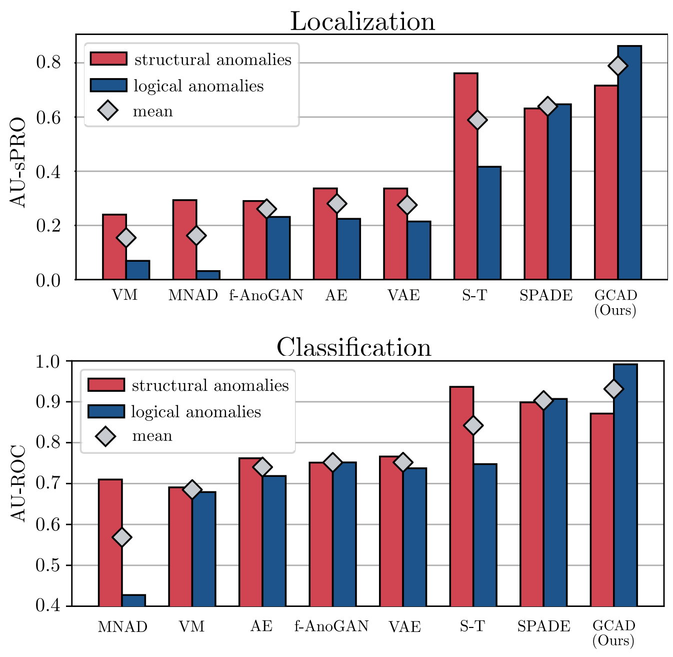


Figure 7.12: Difference in anomaly detection performance for both structural and logical anomalies on the MVTec AD dataset. The top chart shows the area under the sPRO curve for anomaly localization. The bottom chart shows the area under the ROC curve for anomaly classification.

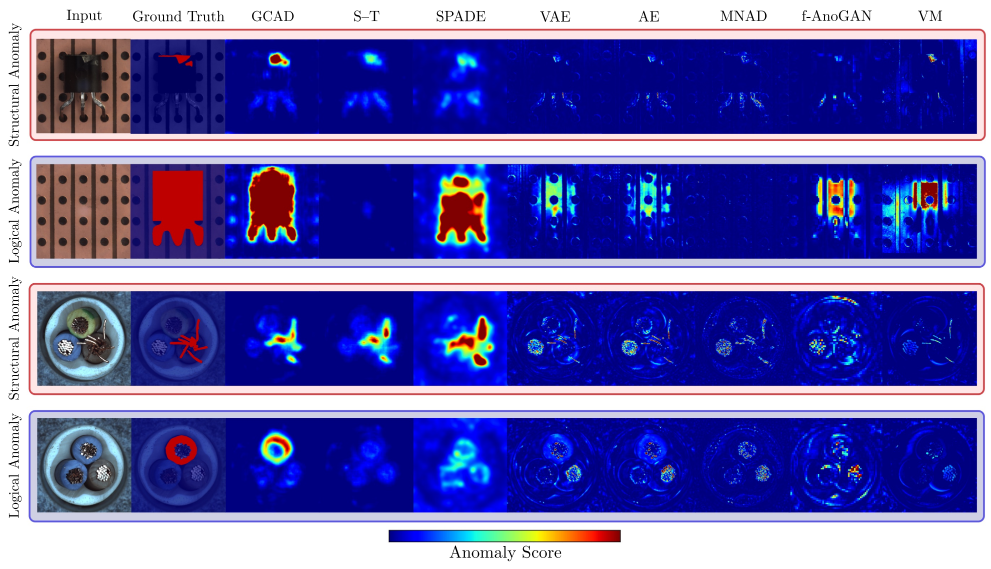


Figure 7.13: Qualitative results for each evaluated method on the MVTec AD dataset. The first and third row contain examples of structural anomalies, i.e., the damaged transistor and the bent wires in the cable cross section. The second and fourth row contain examples of logical anomalies, i.e., the transistor being entirely missing and a blue cable being present instead of a yellow one.

logical anomalies. Similarly to the autoencoders, f-AnoGAN yields many false positives on areas that are challenging to reconstruct. For the missing transistor, however, it manages to capture the logical constraint that a transistor should always be present. The Variation Model manages to detect parts of the damaged transistor as well as its absence. It also yields increased anomaly scores on the bent wires. However, it fails to localize the logical anomaly on the cable.

7.5 Conclusion

In this chapter, we developed Global Context Anomaly Detection, a new method that permits the joint localization of both structural and logical anomalies. It consists of two branches, each of which is primarily intended for the detection of structural and logical anomalies, respectively. The first branch is inspired by our Student–Teacher framework that performs well in the detection of structural anomalies. The second branch learns an embedding of the anomaly-free training data that captures its underlying logical constraints. This is achieved by compressing the input images via a low-dimensional bottleneck.

We performed extensive experiments on the MVTec LOCO and the MVTec AD dataset. Our approach sets a new state of the art in the joint detection of structural and logical anomalies.

8 Unsupervised Detection of Geometric Anomalies in 3D Data

In the previous chapters, we studied anomaly detection from a two-dimensional perspective. In particular, we presented datasets that contain, and methods that process images obtained from color or grayscale cameras. These sensors are frequently used in many real-world applications, since they tend to be affordable and user-friendly. However, certain applications require entirely different sensors in an anomaly detection system.

Hyperspectral cameras, for example, can reveal anomalies in agricultural [Adão et al., 2017] or medical [Lu and Fei, 2014] inspection tasks that manifest themselves through deviations in the spectral signature of a material. Another example are 3D sensors that provide accurate geometric information about a captured scene. They allow for the detection of geometric anomalies that may not be perceivable in the flat projection of a 2D camera. There exists a range of different types of 3D sensors. Time-of-Flight sensors recover 3D information by measuring the time delay between the emission of a light ray and its detection. One particular instance of such sensors are LIDARS, which often occur in autonomous driving applications [Behley et al., 2019]. Another example are X-ray Computed Tomography (CT) sensors that are frequently employed in medical imaging [Jnawali et al., 2018] or airport security applications [Flitton et al., 2015]. A third example are structured light sensors that project a light pattern into the scene that is recorded with a standard 2D camera. Depth information can be recovered through triangulation. Such cameras have found their way, for instance, into industrial manufacturing pipelines [Drost et al., 2017].

While for 2D images, the anomaly detection problem has received increased attention over recent years from the research community, it is still relatively unexplored when applied to 3D data. In the remaining parts of this thesis, we tackle this problem and introduce a new dataset as well as a new method for the unsupervised detection of geometric anomalies. In this chapter, we briefly discuss the technical prerequisites to conduct anomaly detection in three dimensions. We explain the role of different data representations, such as depth images, voxel grids, and point clouds. We then show how our previous evaluation protocol of the anomaly detection problem for 2D images can be adapted to the 3D domain. Finally, we give a brief overview of existing literature.

8.1 Different Representations of 3D Data

Depending on the type of 3D sensor, the data format in which geometric information is generated may differ. In the following, we discuss several frequently returned data formats, namely XYZ and depth images, voxel grids, and point clouds.

Representation as XYZ or Depth Images

An XYZ image represents geometric information as a three-channel image on a two-dimensional grid. Each pixel in the image maps to a single 3D coordinate, where each of the three coordinates is stored in one of the image channels, respectively. The coordinates are typically expressed with respect to the local coordinate system of the 3D sensor. Formally, we write an XYZ image as a function $I : D \to \mathbb{R}^3$ that maps from a two-dimensional pixel domain $D = \{0, 1, \ldots, H-1\} \times \{0, 1, \ldots, W-1\}$ to a 3D coordinate $\boldsymbol{x} \in \mathbb{R}^3$. Here, H and W denote the width and the height of the image, respectively.

In practice, it may happen that a sensor fails to recover the 3D information at some of the image pixels. In such cases, it is common to assign a placeholder value, such as the zero vector $(0, 0, 0)^T$. It may be tempting to simply ignore these invalid values when processing XYZ images with an anomaly detection model. However, the underlying cause of the failed reconstruction may be the occurrence of an anomaly in the captured scene. Hence, it is important to treat all pixels within an XYZ image equally, regardless of whether they contain valid or invalid 3D information.

An example XYZ image is displayed in the top row of Figure 8.1. It describes the 3D surface of a cookie. The sample is taken from the MVTec 3D Anomaly Detection dataset [Bergmann et al., 2022b], which we present in detail in Chapter 9. White areas indicate invalid pixels for which no geometric information is available. The figure additionally shows RGB color values that directly correspond to each 3D point. If color information is provided by a 3D sensor, it can be used as an additional input feature.

If the intrinsic parameters of the sensor are known and a camera projection center exists, the x and y coordinates of a 3D point can be recovered by constructing a ray that runs through the camera center and the corresponding pixel in the XYZ image. Intersecting this ray with a plane that is parallel to the x and y axis and at distance z from the camera center yields the full 3D information. Therefore, it is common to simplify an XYZ image by only storing its z-channel, i.e., by writing it as $I : D \to \mathbb{R}$. Such images are referred to as *depth images*.

Representation as Voxel Grids

XYZ and depth images both have the disadvantage that they cannot represent 3D points that are occluded by points that are relatively closer to the camera center. To circumvent this limitation, geometric information can be represented as a three-dimensional volume that is partitioned into a regularly spaced grid of small volume elements. A single volume element is referred to as a *voxel*, and the entire grid is called a *voxel grid*. Each voxel indicates whether the volume in 3D space that it represents is occupied or not. Hence, a voxel grid can be written as a function $V : D \to \{0, 1\}$, where the domain of the grid is defined as $D = \{0, 1, \ldots, W-1\} \times \{0, 1, \ldots, H-1\} \times \{0, 1, \ldots, L-1\}$. Here, W, H, and L denote the width, height, and number of layers of the grid, respectively.

Again, in some cases it may be desirable to attach additional information such as color values to each voxel element. This can be achieved by directly storing the respective information inside the voxel grid. Each voxel then maps to a multidimensional vector,

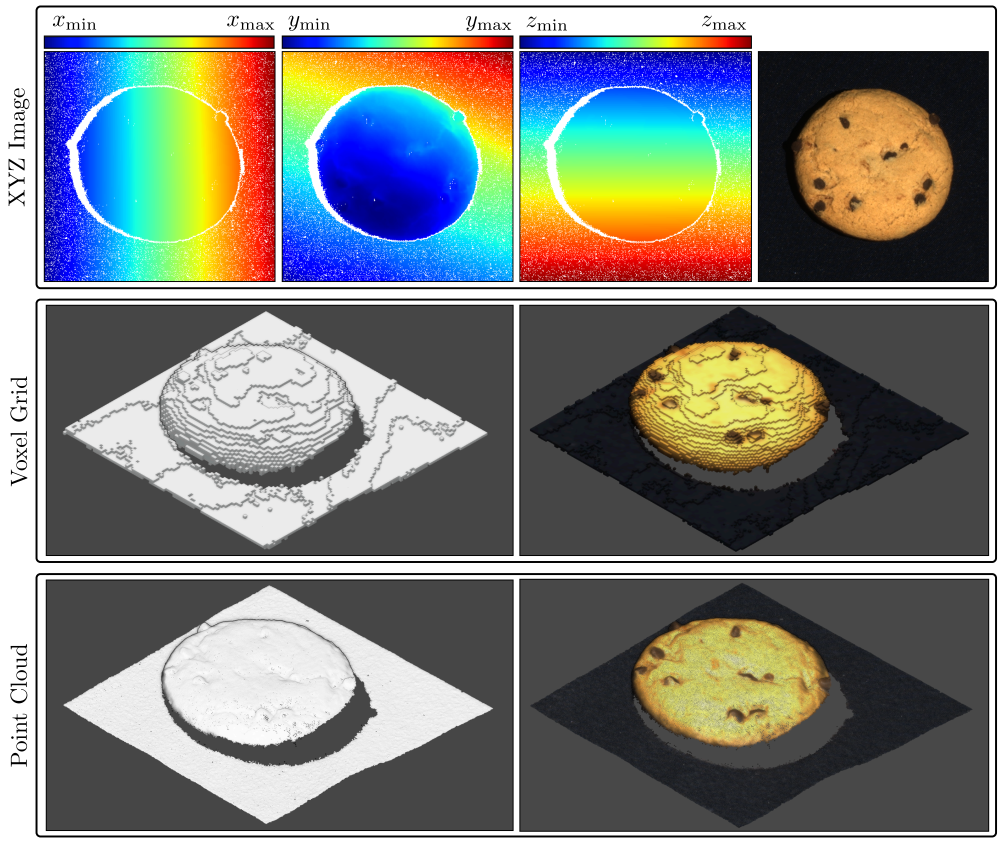


Figure 8.1: Visualization of frequently used data formats to represent geometric data. An anomaly-free sample taken from the category *cookie* of the MVTec 3D-AD dataset is displayed as an XYZ image, a Voxel Grid, and a Point Cloud. The corresponding color information is displayed as an additional attribute attached to each 3D point.

i.e., $V : D \to \mathbb{R}^d$. In this case, a suitable placeholder value must be defined that identifies empty voxel elements.

Continuing the example of the cookie, its voxelized representation is shown in the second row of Figure 8.1. Each dimension of the grid is divided into $W = H = L = 128$ voxel elements. The left image shows the binary voxel grid, while the right additionally displays color information attached to the voxel elements.

Representation as Point Clouds

Note that in the voxelized representation of the cookie, the majority of voxel elements are empty and, therefore, not displayed. Since an empty cell consumes just as much memory as an occupied one, voxel grids tend to be inefficient when used to represent scenes that lead to sparsely populated grids, such as surface data. In such cases, a possibility to represent the data more compactly is through *point clouds*.

A point cloud $P = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$ is an unordered set of 3D points $\boldsymbol{x}_i \in \mathbb{R}^3$. In contrast to depth images or voxel grids, the points in P do not lie on a regular grid. Hence, they can accurately model different areas of 3D space with varying degrees of density. Since empty space is modeled implicitly through the absence of points, point clouds are particularly memory efficient when representing sparse scenes. However, algorithms that process point clouds directly need to be carefully designed such that, ideally, they are invariant to the order in which the points are presented to the model. Furthermore, point clouds do not provide an explicit definition of what constitutes neighboring points that could be used to process them directly with convolution layers.

For point clouds, additional input features such as color vectors can be stored in a separate set $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \dots\}$, where each element $\boldsymbol{c}_i \in \mathbb{R}^d$ corresponds to a multidimensional vector that stores the attribute values attached to each point in P. The third row in Figure 8.1 shows a visualization of a cookie in point cloud format. The left image displays each 3D coordinate in gray, while the right image shows the same point cloud with its additional RGB vectors.

8.2 Performance Evaluation for 3D Anomaly Detection

Given a dataset of anomaly-free training samples $\mathcal{D}_{\text{train}}$ obtained from a 3D sensor, the task is to train a model that, during inference, can detect anomalous samples in a test dataset $\mathcal{D}_{\text{test}}$ and precisely localize anomalous geometric structures within them. In Section 2.1, we defined an evaluation protocol for the anomaly detection problem when applied to two-dimensional color or grayscale images. We now discuss the changes that need to be made to this protocol when considering 3D data instead.

Anomaly Classification

Regarding anomaly classification, the problem remains to assign a single anomaly score to each sample in the test dataset $\mathcal{D}_{\text{test}}$. This is independent of the particular data format in which the 3D data is represented in. Hence, the same performance measures as for the 2D case can be used to assess the anomaly classification performance. In particular, we will report the area under the ROC curve.

Anomaly Localization

To define a protocol that evaluates a models ability to localize anomalous geometric structures, it is necessary to make assumptions about the particular format the 3D data is represented in. If the samples of an anomaly detection dataset are given as XYZ images, it is straightforward to adapt the evaluation protocol used for color or grayscale images as described in Section 2.1. In particular, a method must assign an anomaly score to each pixel of the XYZ images in the test set. Corresponding ground truth maps that indicate whether a pixel, i.e., a 3D point, is considered anomalous or not can then be used to compute established evaluation metrics for anomaly localization, such as the area under the PRO curve. Note that representing 3D data as XYZ images allows to

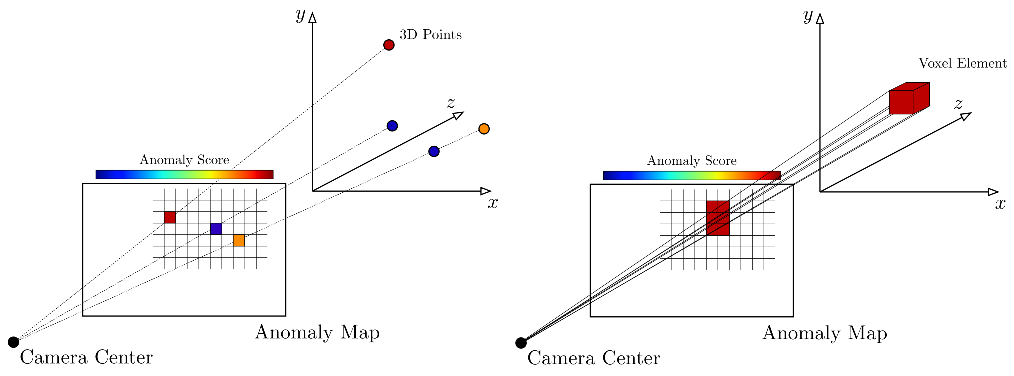


Figure 8.2: Evaluation of methods that process point clouds or voxel grids on anomaly detection datasets that contain XYZ images. Anomaly scores are projected into the image plane by using the sensor's internal camera parameters. If multiple anomaly scores are assigned to the same pixel, the maximum over all scores is selected.

annotate invalid pixels as anomalous. This is an important property, since anomalies may also manifest themselves through the absence of points in a scene.

Evaluating the anomaly localization performance on voxel or point cloud data requires different evaluation protocols. For voxel grids, one option is to provide voxel-precise ground truth volumes that indicate whether a voxel element contains an anomaly or not. Unfortunately, compared to the annotation of anomalies in XYZ image data, annotating anomalies in 3D voxel grids is complicated and time consuming. In particular, when anomalies manifest themselves through the absence of certain geometries, the annotation must accurately model the missing three-dimensional structure. Similarly, a direct evaluation of anomaly localization algorithms on point cloud data requires the creation of annotations that mark certain continuous parts in 3D space as anomalous.

To reduce the complexity to create accurate ground truth annotations, we focus on anomaly detection datasets that contain XYZ images throughout the remaining parts of this thesis. Once an anomaly score is assigned to each pixel within the XYZ images of the test set, we can compute the same performance metrics as as described in Section 2.1 to assess the anomaly localization performance. For methods that process and assign anomaly scores to XYZ images directly, applying this evaluation protocol is straightforward. To evaluate approaches that operate on point clouds or voxel grids instead, the XYZ images of a dataset first need to be converted to the respective data format. The resulting anomaly scores then need to be projected into the plane of the input image through the internal parameters of the camera. This process is illustrated in Figure 8.2. If a method produces anomaly scores as a set of unordered 3D points, the anomaly scores can be directly assigned to the image pixels that these points are projected to. If a pixel is assigned multiple scores, the maximum value is selected. Similarly, if a model assigns anomaly scores to individual voxel elements, the eight corners of each voxel can be projected to their respective image coordinates. Each pixel within the resulting region is assigned the respective anomaly score.

8.3 Deep Learning Models for 3D Anomaly Detection

The design of any anomaly detection system significantly depends on the format of the input data. Here, we briefly discuss how the different forms of 3D data can be processed with deep learning models and give an overview of existing models for the unsupervised detection of anomalies in 3D data.

Like color or grayscale images, XYZ and depth images can be processed with 2D convolution layers. This makes it possible to adapt some of the previously discussed anomaly detection methods to these data formats, such as convolutional autoencoders or f-AnoGAN. Note however, that it is not straightforward to transfer the methods that leverage descriptors from pretrained networks to find anomalies in XYZ images, since they rely on domain-specific pretrainings.

Compared to XYZ and depth images, voxel grids exhibit an additional spatial dimension. To process them with convolution layers, the convolution kernels need to be extended by adding a third dimension [Maturana and Scherer, 2015]. Then, the regular grid structure of the voxel grid makes it possible to design CNNs whose architecture highly resembles those of networks that operate on 2D image grids.

Designing deep learning models that process point clouds comes with two key challenges. First, the input points are not stored on a regular grid and the immediate neighborhood of each point is not explicitly defined. This prevents points clouds to be transformed with 2D or 3D convolution layers directly. To address this problem, it is possible to obtain local neighborhoods by computing sets of neighboring points for each input point through the k-nearest neighbor algorithm [Hu et al., 2020]. The resulting graph structure can then be used to process the point cloud, for example, with graph convolution layers [Zhang et al., 2019]. Second, a model that processes point clouds should ideally be invariant with respect to the order the input points are presented to the network. This is because changing the order of the input points does not affect the geometric structure that they represent. To achieve this, deep learning models that process point clouds commonly use permutation-invariant operations such as computing the maximum or the average value over a set of feature vectors [Charles et al., 2017]. For a comprehensive overview of existing techniques to process point clouds with deep learning models, we refer to Bello et al. [2020].

There exist only very few methods that were explicitly introduced for the task of unsupervised 3D anomaly detection. Previous work stems from the medical domain and operates on voxel data. Simarro Viana et al. [2021] introduce an extension of f-AnoGAN to voxel grids. As for the 2D case, a GAN is trained to generate voxel grids that mimic the training distribution using 3D convolutions. Subsequently, an encoder network is trained that maps input samples to the corresponding latent samples of the generator. During inference, anomaly scores are derived for each voxel element by comparing the input voxels to the reconstructed ones. Similarly, Bengs et al. [2021] present an autoencoder-based method that also operates on voxel data. A variational autoencoder is trained to reconstruct voxel grids through a low-dimensional latent variable. Anomaly scores are again derived by a per-voxel comparison of the input to its reconstruction.

9 The MVTec 3D Anomaly Detection Dataset

In the previous chapter, we found that there exist only very few methods that tackle the 3D anomaly detection problem. We believe that an important reason for this is the lack of publicly available datasets that can be used to develop new approaches and ideas. In this chapter, we introduce the first comprehensive 3D dataset for the task of unsupervised anomaly detection and localization. It is inspired by real-world visual inspection scenarios in which a model has to detect various types of geometric defects on manufactured products, even if it is trained only on anomaly-free data. We employed a high-resolution industrial 3D sensor to acquire depth scans of 10 different object categories. For all object categories, we present a training and validation set, each of which solely consists of scans of anomaly-free samples. The corresponding test sets contain samples showing various defects such as scratches, dents, holes, contaminations, or deformations. Precise ground-truth annotations are provided for every anomalous test sample. An initial benchmark of 3D anomaly detection methods on our dataset indicates a considerable room for improvement.

The content of this chapter is based on the publication titled *The MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization* [Bergmann et al., 2022b], which was recognized with the Best Industrial Paper Award at the *17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*.

9.1 Introduction

The increased availability and precision of modern 3D sensors has led to significant advances in the field of 3D computer vision. The research community has used these devices to create datasets for a wide variety of real-world problems, such as point cloud registration [Zeng et al., 2017], classification [Wu et al., 2015], 3D semantic segmentation [Chang et al., 2015, Dai et al., 2017], 3D object detection [Armeni et al., 2016], and rigid pose estimation [Drost et al., 2017, Hodaň et al., 2020]. The development of new and improved algorithms relies on the availability of such high-quality datasets.

Regarding anomaly detection in two-dimensional image data, numerous synthetic and real-world benchmark datasets exist. They cover various domains, e.g., autonomous driving [Blum et al., 2019], video anomaly detection [Li et al., 2013, Lu et al., 2013, Sultani et al., 2018], or industrial inspection scenarios [Huang et al., 2018, Bergmann et al., 2019a, 2021, Carrera et al., 2017, Song and Yan, 2013]. These datasets have led

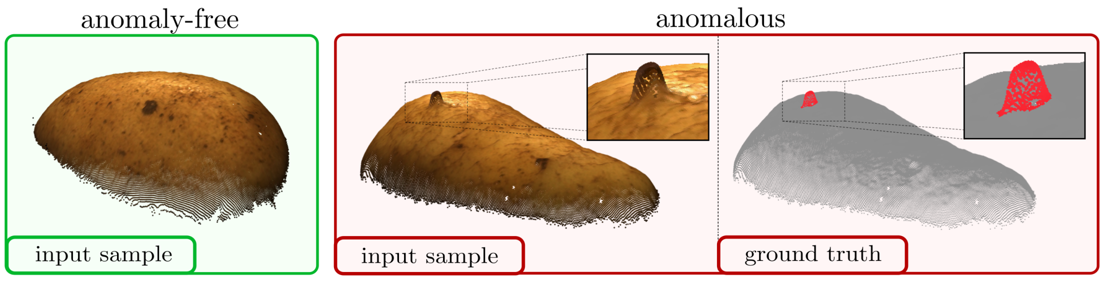


Figure 9.1: Two samples representing the category *potato* of our new dataset. The sample on the left is anomaly-free while the one on the right contains a nail stuck through the surface of the potato. We depict the point clouds overlaid with the associated color values. For the anomalous sample, we additionally display the annotated ground truth.

to the development of numerous methods that are intended to operate on 2D color or grayscale images.

At the time of conducting this research project, there was no comprehensive 3D dataset designed explicitly for the unsupervised detection and localization of anomalies. Existing methods for this problem are evaluated on two medical benchmarks that were originally introduced for the supervised detection of diseases in brain magnetic resonance (MR) scans. Menze et al. [2015], Bakas et al. [2017], and Baid et al. [2021] present the multimodal brain tumor image segmentation benchmark (BRATS). It consists of 65 multi-contrast MR scans of glioma patients. Each sample is provided as a dense voxel grid and tumors were annotated by radiologists in each image slice of the scan. Similarly, Liew et al. [2018] provide the Anatomical Tracings of Lesions After Stroke (ATLAS) dataset. It consists of 304 MR scans with corresponding ground truth annotations of brain lesions. Both these datasets provide 3D information by stacking multiple grayscale images to form a dense voxel grid.

To fill this gap and spark further interest in the development of new methods, we introduce a real-world dataset for the task of unsupervised 3D anomaly detection and localization. Given a set of exclusively anomaly-free 3D scans of an object, the task is to detect and localize various types of anomalies. Figure 9.1 shows two prototypical samples of our new dataset. Like the datasets introduced in the previous chapters, it is inspired by industrial inspection scenarios. Our main contributions are as follows:

- We introduce the first comprehensive dataset for unsupervised anomaly detection and localization in three-dimensional data. It consists of 4147 high-resolution 3D point cloud scans from 10 real-world object categories. While the training and validation sets only contain anomaly-free data, the samples in the test set contain various types of anomalies. Precise ground truth annotations are provided for each anomaly.[1]

[1]The dataset is publicly available at https://www.mvtec.com/company/research/datasets.

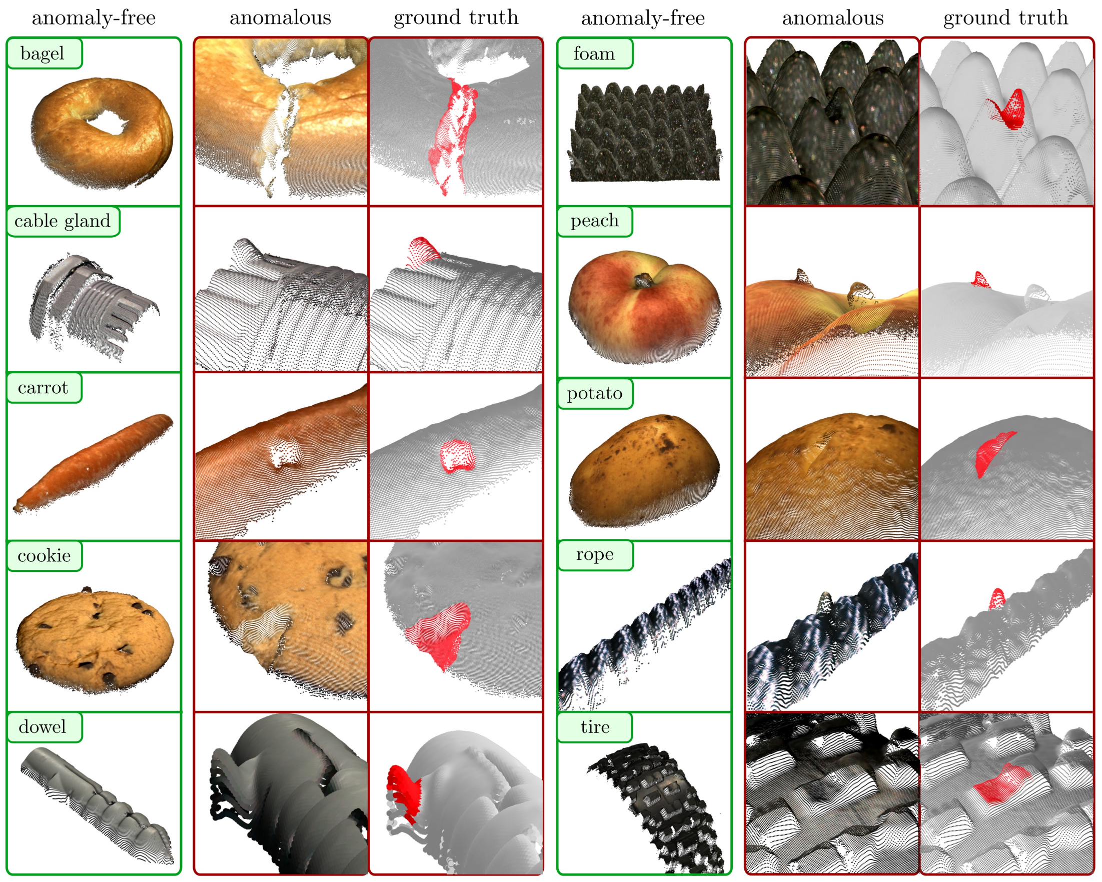


Figure 9.2: Examples for all 10 dataset categories of the MVTec 3D-AD dataset. For each category, the left column shows an anomaly-free point cloud with RGB values projected onto it. The second column shows a close-up view of an anomalous test sample. Anomalous points are highlighted in the third column in red. Note that the background planes were removed for better visibility.

- We evaluate current methods that were specifically designed for unsupervised 3D anomaly localization. Our initial benchmark reveals that existing methods do not perform well on our dataset and that there is considerable room for future improvement.

9.2 Description of the Dataset

The MVTec 3D-AD dataset consists of 4147 scans acquired by a high-resolution industrial 3D sensor. For each of the 10 object categories, a set of anomaly-free scans is provided for model training and validation. The test set contains both, anomaly-free scans as well as object samples that contain various types of structural anomalies, such

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 9.1: Statistical overview of the MVTec 3D-AD dataset. For each category, we list the number of training, validation, and test images. Test images are split into anomaly-free images and images containing anomalies. We report the number of different defect types, the number of annotated regions, and the size of the XYZ images for each category.

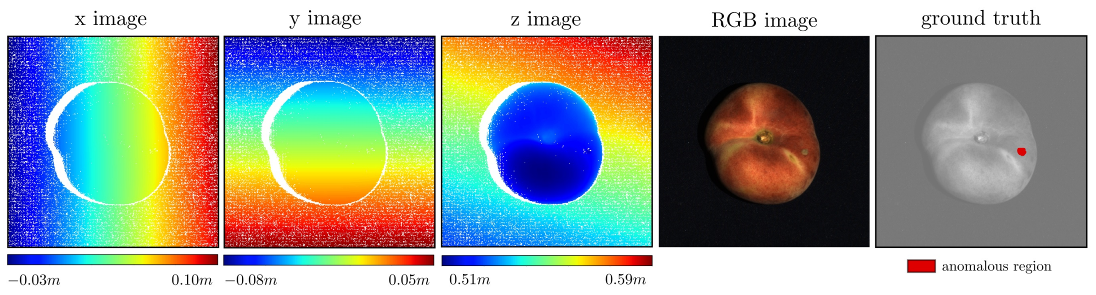


Figure 9.3: Visualization of the provided data for one anomalous test sample of the dataset category *peach*. In addition to three images that encode the 3D coordinates of the object, RGB information as well as a pixel-precise ground-truth image are provided.

as scratches, dents, or contaminations. The defects were devised and fabricated as they would occur in real-world inspection scenarios.

Five of the object categories in our dataset exhibit considerable natural variations from sample to sample. These are *bagel*, *carrot*, *cookie*, *peach*, and *potato*. Three more objects, *foam*, *rope*, and *tire*, have a standardized appearance but can be easily deformed. The two remaining objects, *cable gland* and *dowel*, are rigid. In principle, inspecting the last two could be achieved by comparing an object's geometry to a CAD model. However, unsupervised methods should be able to detect anomalies on all kinds of objects and the creation of a CAD model might not always be desirable or practical in a real application. An example point cloud for each dataset category is shown in Figure 9.2. The figure also displays some anomalies together with the corresponding ground truth annotations. The images of the *bagel* and the *cookie* show cracks in the objects. The surfaces of the *cable gland* and the *dowel* exhibit geometrical deformations. There is a hole in the *carrot* and some contaminations on the *peach* and the *rope*. Parts of the *foam*, the *potato*, and the *tire* are cut off. These are prototypical examples of the 41 types of anomalies present in our dataset. More statistics on the dataset are listed in Table 9.1.

9.2.1 Data Acquisition and Preprocessing

All dataset scans were acquired with a Zivid One$^+$ Medium,[2] an industrial sensor that records high-resolution 3D scans using structured light. The data is provided by the sensor as a three-channel XYZ image with a resolution of 1920×1200 pixels. The channels represent the x, y, and z coordinates with respect to the local camera coordinate frame. The (x, y, z) values of the image provide a one-to-one mapping to the corresponding point cloud. In addition, the sensor acquires complementary RGB values for each (x, y, z) pixel. It was statically mounted to view all objects of each individual category from the same angle. We performed a calibration of the internal camera parameters that allows to project 3D points into the respective pixel coordinates [Steger et al., 2018]. The scene was illuminated by an indirect and diffuse light source.

For each dataset category, we specified a fixed rectangular domain and cropped the original XYZ and RGB images to reduce the amount of background pixels in the samples. The acquisition setup as well as the preprocessing are very similar to real-world applications where an object is usually located in a defined position and the illumination is chosen to best suit the task. In addition, our setup enables and simplifies data augmentation. All objects were recorded on a dark background and the preprocessing leaves a sufficient margin around the objects to allow for the application of various data augmentation techniques, such as crops, translations, or rotations. This enables the use of our dataset for the training of data-hungry deep learning methods, as demonstrated by our experiments in Section 9.3.

Figure 9.3 shows the data provided for the anomalous test sample of the *peach* displayed in Figure 9.2. The image is of size 600×600 pixels, cropped from the original sensor scan. The first three images visualize the x, y, and z coordinates of the dataset sample, respectively. White pixels mark areas where the sensor did not return any 3D information due to, e.g., occlusions, reflections, or sensor inaccuracies. The corresponding RGB and ground truth annotation images are also displayed.

9.2.2 Ground-Truth Annotations

We provide precise ground-truth annotations for each anomalous sample in the test set. Anomalies were annotated in the 3D point clouds. Since there is a one-to-one mapping of the 3D points to their respective pixel locations in the XYZ image, we make the annotations available as two-dimensional regions. This procedure allows us to additionally label invalid sensor pixels and thus annotate anomalies that manifest themselves through the absence of points. For example, an anomaly might lead to a failure of 3D reconstruction and therefore yield invalid pixels in the 3D image. Furthermore, if an anomaly is visible in the RGB image and its corresponding color pixels are not already included in the ground truth label, we append these pixels to the annotation.

An example ground truth mask is shown in Figure 9.3, where a contamination is present on the *peach*. In Figure 9.2, further annotations are visualized when projected to the valid 3D points of a scene. The size of the individual connected components of the

[2]https://www.zivid.com/zivid-one-plus-medium-3d-camera

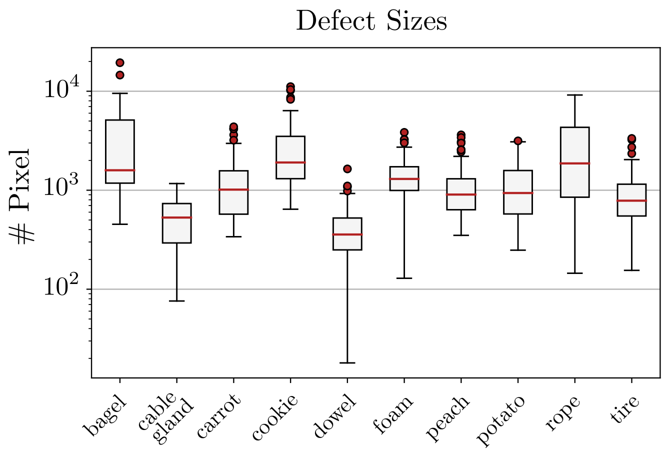


Figure 9.4: Size of anomalies for all objects in the dataset visualized as a box-and-whisker plot. Defect areas are reported as the total number of pixels within an annotated connected component. Anomalies vary greatly in size for each dataset category.

anomalies present in the test set varies greatly, from a few hundred to several thousand pixels. Figure 9.4 visualizes their distribution as a box-and-whisker plot with outliers on a logarithmic scale [Tukey, 1977].

9.2.3 Performance Evaluation

To assess the anomaly localization performance of a method on our dataset, we require it to output a real-valued anomaly score for each (x, y, z) pixel of the test set. In contrast to only assigning anomaly scores to all valid 3D points of the test samples, this allows the detection of anomalies that manifest themselves through invalid sensor pixels or missing 3D structures. These anomaly scores are converted to binary predictions using a threshold. We then compute the per-region overlap (PRO) metric, which is defined as the average relative overlap of the binary prediction with each connected component of the ground truth. This process is repeated for multiple thresholds and a curve is constructed by plotting the resulting PRO values against the corresponding false positive rates. The final performance measure is computed by integrating this curve up to a limited false positive rate and normalizing the resulting area to the interval $[0, 1]$. For a more thorough definition of the PRO metric, we refer to Section 2.2.

We want to stress that, when working with our dataset, we strongly discourage to calculate the area under the PRO curve up to high false positive rates. We recommend to select the integration limit no larger than 0.3. This is due to the fact that the anomalous regions are very small compared to the size of the images. At large false positive rates, the amount of erroneously segmented pixels would be significantly larger than the number of actually anomalous pixels. This would lead to segmentation results that are no longer meaningful in practice.

Our dataset can also be used to assess the performance of algorithms that make a binary decision for each sample if it contains an anomaly or not. In this case, we report the area under the ROC curve as a standard classification metric.

9.3 Benchmark

To examine how existing 3D anomaly localization methods perform on our new dataset, we conducted an initial benchmark. It is intended to serve as a baseline for future methods. Very few methods have been proposed explicitly for this task and all of them operate on voxel data. This is mainly due to the fact that these methods are originally intended to process MR or CT scans that consist of several layers of intensity images. As representatives from this class of methods, we include Voxel f-AnoGAN [Simarro Viana et al., 2021] and our own implementation of a convolutional Voxel AE [Bengs et al., 2021] in our benchmark. Predecessors of these methods were developed for 2D image data. The main difference between the 2D and 3D methods is the use of 2D convolutions on images and 3D convolutions on voxel data, respectively. Therefore, these methods are easily adapted to process depth images and we include them in our benchmark as well. In addition to these deep learning methods, we evaluate the performance of variation models [Steger et al., 2018] on voxel data and depth images. They detect anomalies by calculating the pixel- or voxel wise Mahalanobis distance to the training data distribution. All evaluated methods can either operate solely on the 3D data or can additionally process the color information attached to each 3D point. We therefore also compare the difference in performance when adding color information to the models.

9.3.1 Training and Evaluation Protocol

Data Representation. To represent dataset samples as voxel grids, we first compute a global 3D bounding box over the entire training set for each dataset category. Then, a voxel grid is placed at the center of the bounding box. The length of each side of the grid is is chosen to be equal to the longest side of the bounding box. If only 3D data is processed, occupied and empty voxels are assigned the values 1 and -1, respectively. If RGB information is added, empty voxels are assigned the vector $(-1, -1, -1)$. Occupied voxels are assigned the average RGB value of all points that fall inside the same grid cell.

For methods that process depth images, invalid pixels are assigned a distance of 0. If color information is included, the RGB channels are appended to the single-channel depth image. For both, the voxel grids and depth images, the RGB values are scaled to the interval $[0, 1]$.

Methods on Voxel Grids. For all voxel-based methods, we use grids of size $64 \times 64 \times 64$ voxels. To choose the latent dimension of the compression-based methods, we performed an ablation study, which is described in Section 9.3.2. Anomaly scores are computed by a voxel wise comparison of the input with its reconstruction.

For the implementation of Voxel f-AnoGAN, we use the same network architecture as proposed by Simarro Viana et al. [2021]. The GAN and the encoder network are both trained for 50 epochs on the augmented version of our dataset with an initial learning rate of 0.0002 and a batch size of 2 using the Adam optimizer [Kingma and Ba, 2015]. The weight for the gradient penalty loss of the GAN is set to 10 and one generator training iteration is performed for every 5 iterations of the discriminator training.

The Voxel Autoencoder consists of an encoder and a decoder network. Their architecture is the same as the one of the encoder and the generator in Voxel f-AnoGAN, respectively. We train for 50 epochs on the augmented version of our dataset with a batch size of 2 using the Adam optimizer with an initial learning rate of 0.0001. The voxel grids of the samples of our dataset are sparsely populated and the majority of voxels is empty. We found that this leads to problems when training the Voxel AE if each voxel is weighted equally in the reconstruction loss. In this case, the model tends to simply output an empty voxel grid to minimize the reconstruction error. To address this imbalance, we introduce a loss weight $w \in (0, 1)$ that is computed as the fraction of empty voxels in the training set. During training, the loss at each voxel is then multiplied by w if the voxel is occupied and by $(1 - w)$ otherwise.

For the Voxel Variation Model, we first compute the mean and standard deviation of the training data at each voxel. During inference, anomaly scores are derived by computing the voxel wise Mahalanobis distance of each test sample to the training distribution.

Methods on Depth Images. Our implementations of Depth f-AnoGAN and the Depth AE both process images at a resolution of 256×256 pixels. Input images are zoomed using nearest neighbor interpolation for depth, and bilinear interpolation for color images. Anomaly scores are derived by a per-pixel comparison of the input images and their reconstructions.

The Depth f-AnoGAN consists of three sub-networks, i.e., an encoder, a discriminator, and a generator. The architecture of the encoder is given in Table 9.2. It consists of a stack of 10 convolution blocks that compress an input image of size 256×256 pixels and C channels to a d-dimensional latent vector. Each convolution block except the last one is followed by an instance normalization layer [Ulyanov et al., 2017] and a LeakyReLU with slope 0.05. The architecture of the discriminator is identical to the one of the encoder except that $d = 1$. The generator produces an image of size 256×256 pixels and C channels from a latent variable with d dimensions. Its architecture is symmetric to the one in Table 9.2 in the sense that convolutions are replaced by transposed convolution layers. Both, the GAN and the encoder network, are trained for 50 epochs using a batch size of 4 and an initial learning rate of 0.0002 using the Adam optimizer.

For the encoder and decoder of the Depth AE, we use the same architecture as for the encoder and generator of Depth f-AnoGAN, respectively. It is trained for 50 epochs using the Adam optimizer at a batch size of 32 and an initial learning rate of 0.0001.

The Depth Variation Model processes images with their original resolution and computes the mean and standard deviation over the entire training set at each image pixel.

Layer	Output Size	Parameters		
		Kernel Size	Stride	Padding
Image	$256 \times 256 \times C$			
conv1	$128 \times 128 \times 32$	4	2	1
conv2	$64 \times 64 \times 32$	4	2	1
conv3	$32 \times 32 \times 32$	4	2	1
conv4	$32 \times 32 \times 32$	3	1	1
conv5	$16 \times 16 \times 64$	4	2	1
conv6	$16 \times 16 \times 64$	3	1	1
conv7	$8 \times 8 \times 128$	4	2	1
conv8	$8 \times 8 \times 64$	3	1	1
conv9	$8 \times 8 \times 32$	3	1	1
conv10	$1 \times 1 \times d$	8	1	0

Table 9.2: Encoder architecture of Depth f-AnoGAN and the Depth AE.

Again, anomaly scores are derived by computing the pixel wise Mahalanobis distance from the training distribution.

Dataset Augmentation. Since the evaluated methods, except for the Variation Models, require large amounts of training data, we use data augmentation to increase the size of the training set. For each object category, we first estimate the normal vector of the background plane, which is constant across samples. We then rotate each dataset sample around this normal vector by a certain angle and project the resulting points and corresponding color values into the original 2D image grid using the internal camera parameters. We augment each training sample 20 times by randomly sampling angles from the interval $[-5°, 5°]$.

Computation of Anomaly Maps. All voxel-based methods compute an anomaly score for each voxel element. However, comparing their performance on our dataset requires them to assign an anomaly score to each pixel in the original XYZ images. We therefore project the anomaly scores to pixel coordinates using the internal camera parameters of the 3D sensor. For each voxel element, we project all 8 corner points and compute the convex hull of the resulting projected points. All image pixels within this region are assigned the respective anomaly score of the voxel element. If a pixel is assigned multiple anomaly scores, we select their maximum.

Methods on depth images already assign a score to each pixel. The anomaly maps of Depth f-AnoGAN and the Depth AE are zoomed to the original image size using bilinear interpolation.

9.3.2 Experiment Results

Table 9.3 lists quantitative results for each evaluated method for the localization of anomalies. For each dataset category, we report the normalized area under the PRO curve with an upper integration limit of 0.3. We further report the mean performance over all categories.

			bagel	cable gland	carrot	cookie	dowel	foam	peach	potato	rope	tire	mean
3D Only	Voxel	GAN	0.440	**0.453**	**0.825**	**0.755**	**0.782**	**0.378**	**0.392**	0.639	**0.775**	**0.389**	**0.583**
		AE	0.260	0.341	0.581	0.351	0.502	0.234	0.351	**0.658**	0.015	0.185	0.348
		VM	**0.453**	0.343	0.521	0.697	0.680	0.284	0.349	0.634	0.616	0.346	0.492
	Depth	GAN	0.111	0.072	0.212	0.174	0.160	0.128	0.003	0.042	0.446	0.075	0.143
		AE	0.147	0.069	0.293	0.217	0.207	0.181	0.164	0.066	0.545	0.142	0.203
		VM	0.280	0.374	0.243	0.526	0.485	0.314	0.199	0.388	0.543	0.385	0.374
3D + RGB	Voxel	GAN	**0.664**	0.620	0.766	**0.740**	0.783	0.332	0.582	0.790	0.633	0.483	**0.639**
		AE	0.467	**0.750**	**0.808**	0.550	0.765	**0.473**	**0.721**	**0.918**	0.019	0.170	0.564
		VM	0.510	0.331	0.413	0.715	0.680	0.279	0.300	0.507	0.611	0.366	0.471
	Depth	GAN	0.421	0.422	0.778	0.696	0.494	0.252	0.285	0.362	0.402	**0.631**	0.474
		AE	0.432	0.158	**0.808**	0.491	**0.841**	0.406	0.262	0.216	**0.716**	0.478	0.481
		VM	0.388	0.321	0.194	0.570	0.408	0.282	0.244	0.349	0.268	0.331	0.335

Table 9.3: Anomaly localization results. The area under the PRO curve is reported for an integration limit of 0.3 for each evaluated method and dataset category. The best performing methods are highlighted in boldface.

			bagel	cable gland	carrot	cookie	dowel	foam	peach	potato	rope	tire	mean
3D Only	Voxel	GAN	0.383	0.623	0.474	0.639	0.564	0.409	0.617	0.427	0.663	0.577	0.538
		AE	0.693	0.425	0.515	**0.790**	0.494	0.558	0.537	0.484	0.639	0.583	0.572
		VM	**0.750**	**0.747**	**0.613**	0.738	**0.823**	0.693	**0.679**	**0.652**	0.609	**0.690**	**0.699**
	Depth	GAN	0.530	0.376	0.607	0.603	0.497	0.484	0.595	0.489	0.536	0.521	0.524
		AE	0.468	0.731	0.497	0.673	0.534	0.417	0.485	0.549	0.564	0.546	0.546
		VM	0.510	0.542	0.469	0.576	0.609	**0.699**	0.450	0.419	**0.668**	0.520	0.546
3D + RGB	Voxel	GAN	**0.680**	0.324	0.565	0.399	0.497	0.482	0.566	0.579	0.601	0.482	0.517
		AE	0.510	0.540	0.384	0.693	0.446	0.632	0.550	0.494	**0.721**	0.413	0.538
		VM	0.553	**0.772**	0.484	**0.701**	0.751	0.578	0.480	0.466	0.689	0.611	**0.609**
	Depth	GAN	0.538	0.372	0.580	0.603	0.430	0.534	0.642	**0.601**	0.443	0.577	0.532
		AE	0.648	0.502	**0.650**	0.488	**0.805**	0.522	**0.712**	0.529	0.540	0.552	0.595
		VM	0.513	0.551	0.477	0.581	0.617	**0.716**	0.450	0.421	0.598	**0.623**	0.555

Table 9.4: Anomaly classification results. We report the area under the ROC curve. The best-performing methods are highlighted in boldface.

The first six rows in Table 9.3 show the performance of each method when trained only on the 3D sensor data without providing any color information. In this case, the Voxel f-AnoGAN performs best on average and on the majority of all dataset categories. It is followed by the Voxel VM, which shows the best performance on one of the objects. The Voxel AE performs worse than the other two voxel-based methods. This is because it tends to produce blurry and inaccurate reconstructions.

On average, each voxel-based method performs better than its depth-based counterpart. Among all depth-based methods, the Depth Variation Model performs best. We found that the Depth AE and Depth f-AnoGAN produce many false positives in the anomaly maps around invalid pixels in the input data.

Figure 9.5 shows corresponding qualitative anomaly localization results. For visualization purposes, the predicted anomaly scores were projected onto the input point clouds. For each dataset sample, the corresponding ground truth is visualized in red. While most of the methods are able to localize some of the defects in our dataset, they also yield a large number of false positive predictions, either on the objects' surfaces, around the objects' edges, or in the background. Due to the reconstruction inaccuracies of the Voxel AE, it can only detect the larger and more salient anomalies in our dataset such as the one depicted in Figure 9.5.

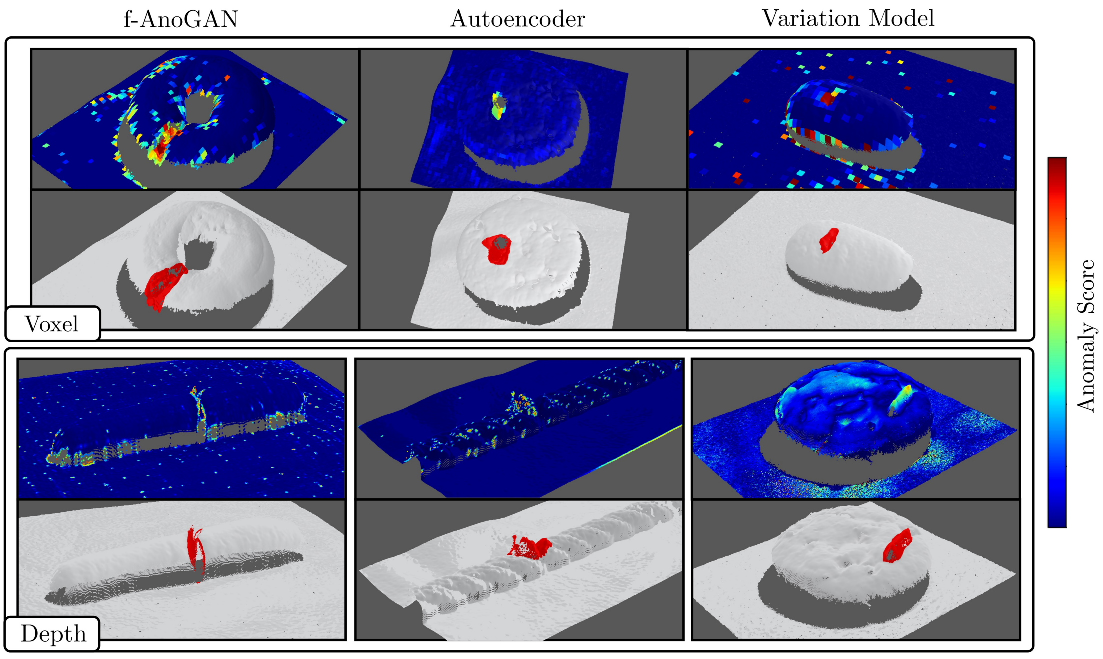


Figure 9.5: Qualitative anomaly localization results in which each individual method performed well. The top image visualizes the anomaly scores as an overlay to the input point cloud. The bottom image shows the corresponding ground-truth annotation in red. The displayed methods were only trained on the 3D data without adding color information.

In addition to evaluating each method on 3D data only, we report the performance of the methods when trained with RGB features at each 3D point. The results are listed in the bottom six rows of Table 9.3. Adding RGB information improves the performance of all methods except for the Variation Models. Since the RGB images do not contain invalid pixels, the Depth AE and Depth f-AnoGAN benefit most from the color information. Nevertheless, the voxel-based methods still outperform their depth-based counterparts. Again, Voxel f-AnoGAN shows the best overall performance. For some object categories, however, the Voxel AE performs better than Voxel f-AnoGAN when including color information.

We further provide results for the classification of dataset samples as either anomalous or anomaly-free. Since this requires a method to output a single anomaly score for each dataset sample, we compute the maximum anomaly score of each anomaly map. As performance measure, we compute the area under the ROC curve. Table 9.4 lists the results. Here, the Voxel Variation Model achieves the best overall performance. On average, many of the evaluated methods perform only slightly better than a random classifier. This is because they tend to spuriously produce high anomaly scores in anomaly-free areas as also observed in Figure 9.5.

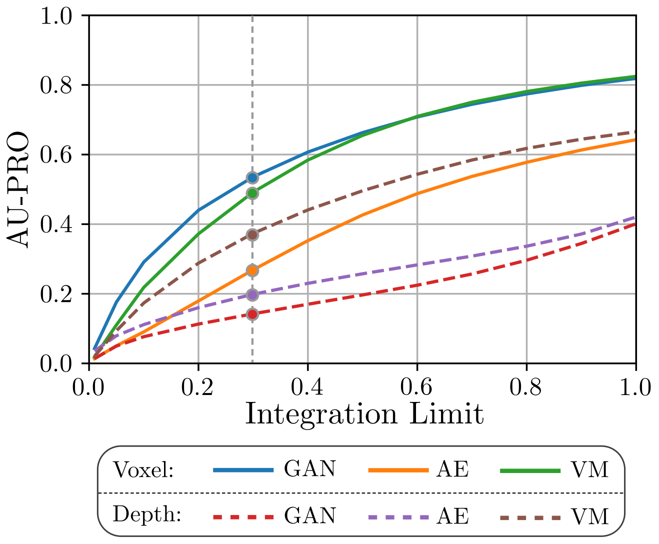


Figure 9.6: Dependence of AU-PRO on the integration limit. The results of our benchmark are reported at a limit of 0.3.

Performance at Different Integration Limits. As discussed in Section 9.2.3, it is important to select a suitable integration limit to compute the area under the PRO curve. To illustrate this, Figure 9.6 shows the dependence of the performance of each evaluated method on the integration limit. All methods show a monotonic increase in their AU-PRO. When integrating to a false positive rate of 1, the Voxel f-AnoGAN and the Voxel VM surpass an AU-PRO of 0.8, which would suggest that these methods are close to solving the task. However, evaluating at large integration limits include binary segmentation masks where the number of false positive predictions is extremely high. Since the area of the defects present in our dataset is very small compared to the area of anomaly-free pixels, such segmentation results are no longer meaningful. Therefore, we selected an integration limit of 0.3 in our evaluation.

Latent Dimensions of Compression-Based Methods. To select suitable latent dimensions for the evaluated compression-based methods, we perform an ablation study. Their mean performance over all object categories is given in Table 9.5. For the experiments conducted above, we use the respective latent dimension that yielded the best mean performance in the ablation study.

Quality of Reconstructions. For the methods based on AEs or GANs, the anomaly detection performance significantly depends on the quality of their reconstructions. To get an impression of the reconstruction quality, Figure 9.7 shows two examples for each evaluated method. To visualize the voxel-based methods, voxel grids are converted to point clouds by applying a threshold to each cell. A cell is classified as occupied if it

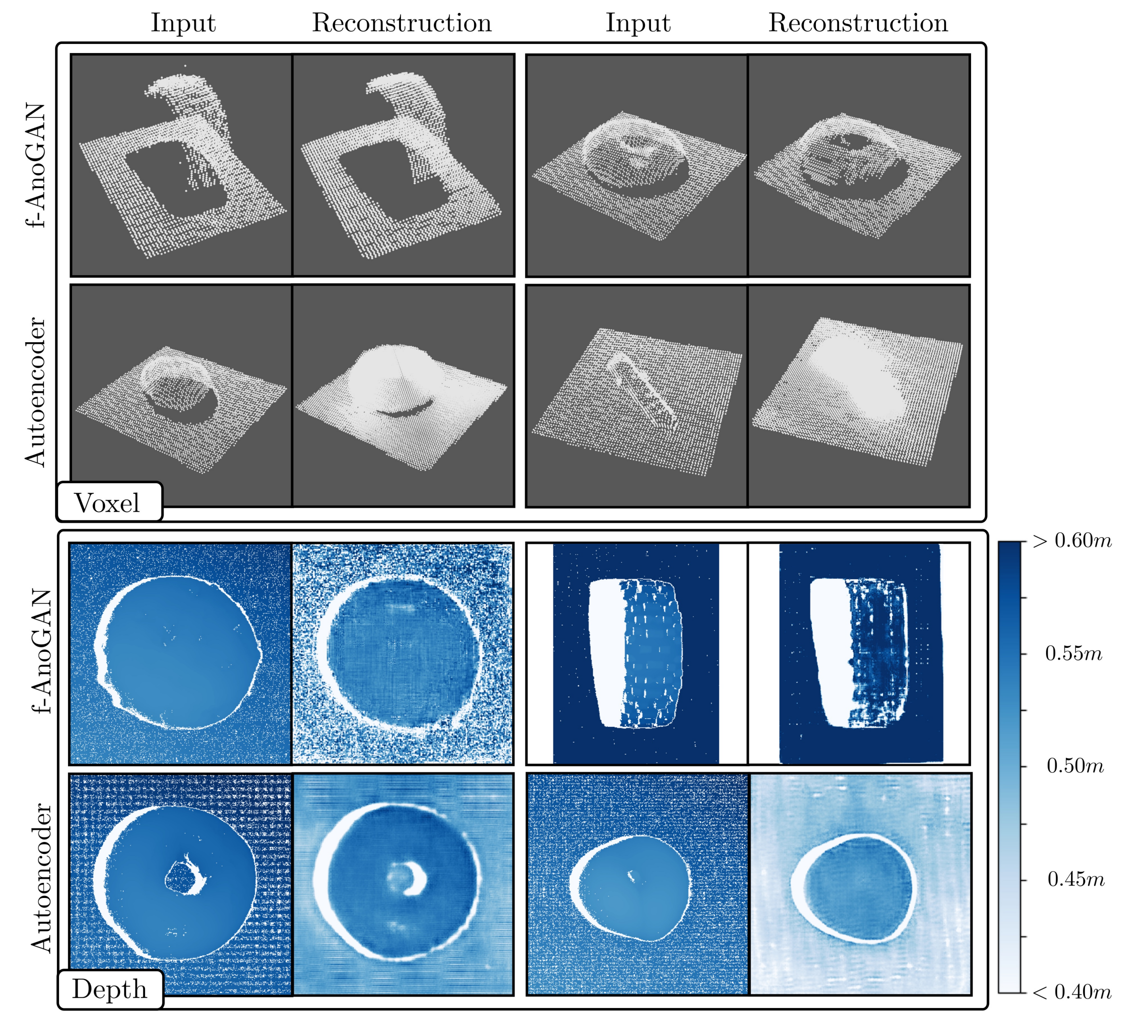


Figure 9.7: Examples of reconstructions for each compression-based method. For voxel-based methods, inputs and reconstructions are visualized as point clouds. For depth-based methods, they are shown as depth images.

		Latent Dimension		
		128	512	2048
Voxel	GAN	0.536	**0.583**	0.555
	AE	**0.348**	0.269	0.305
Depth	GAN	**0.143**	0.137	0.135
	AE	0.199	**0.203**	0.199

Table 9.5: Difference in performance when varying the latent dimension of each compression-based method. We list the area under the PRO curve up to an integration limit of 0.3. The best performing setting is highlighted in boldface.

contains a value of 0.9 or higher. The Voxel AE tends to produce blurry reconstructions around the objects' surfaces. The Voxel f-AnoGAN does not have this problem. However, it sometimes fails to produce parts of the input. For the depth-based methods, inputs and reconstructions are visualized as depth images. Darker shades of blue indicate points that are further from the camera center. White points indicate invalid pixels. Both, the Depth f-AnoGAN and the Depth AE show problems reconstructing noisy areas that exhibit many invalid pixels.

9.4 Conclusion

We presented a comprehensive 3D dataset for the task of unsupervised detection and localization of anomalies. The conceptualization and acquisition of the dataset was inspired by real-world visual inspection tasks. It consists of over 4000 point clouds depicting instances of ten different object categories. The data was acquired using a high-resolution structured light 3D sensor. About 1000 samples of the dataset contain various types of anomalies and we provide precise ground truth annotations for all of them.

We performed an initial benchmark of the few existing methods showing that there is significant room for improvement. In particular, the accuracy of the evaluated methods is insufficient for them to be used in real-world industrial applications.

10 Deep Geometric Descriptors for 3D Anomaly Detection

Our initial benchmark conducted on the MVTec 3D-AD dataset showed that methods for unsupervised anomaly detection in 3D data leave considerable room for improvement. In this chapter, we present a new method for the unsupervised detection of geometric anomalies in high-resolution 3D point clouds that significantly improves the detection rate compared to existing approaches. In particular, we propose an adaptation of our Student–Teacher anomaly detection framework to three dimensions. A student network is trained to match the output of a pretrained teacher network on anomaly-free point clouds. When applied to test data, regression errors between the teacher and the student allow reliable localization of anomalous structures. To construct an expressive teacher network that extracts dense local geometric descriptors, we introduce a novel self-supervised pretraining strategy. The teacher is trained by reconstructing local receptive fields and does not require annotations. Extensive experiments on the MVTec 3D Anomaly Detection dataset highlight the effectiveness of our approach, which outperforms the existing methods by a large margin. Ablation studies show that our approach meets the requirements of practical applications regarding performance, runtime, and memory consumption.

10.1 Introduction

In recent years, significant progress has been made in the field of 3D computer vision in various research areas such as 3D classification, 3D semantic segmentation, and 3D object recognition. Many new methods build on earlier achievements in their counterparts in 2D, which operate with natural image data. However, the transition from 2D to 3D poses additional challenges, e.g., the need to deal with unordered point clouds and sensor noise. This has led to the development of new network architectures and training protocols specific to three dimensions. In this chapter, we follow the practice in other areas of computer vision and take inspiration from recent advances in 2D anomaly detection to devise a powerful 3D method.

More specifically, we build on the success of using descriptors from pretrained neural networks for unsupervised anomaly detection. A common protocol is to extract these descriptors as intermediate features from networks trained on the ImageNet [Krizhevsky et al., 2012] dataset. Models based on pretrained features were shown to perform better than ones trained with random weight initializations [Burlina et al., 2019, Bergmann et al., 2020, Cohen and Hoshen, 2020]. In particular, they typically outperform methods based on convolutional autoencoders or generative adversarial networks.

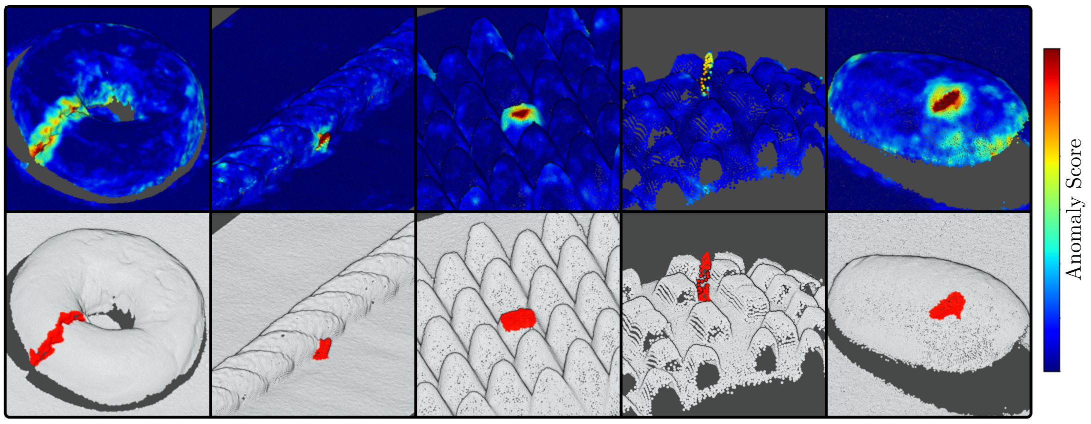


Figure 10.1: Qualitative results of our 3D–ST method on the MVTec 3D Anomaly Detection dataset. Our method reliably localizes geometric anomalies in test point clouds, even though it has been trained only on anomaly-free samples. Top row: Anomaly scores for each 3D point predicted by our algorithm. Bottom row: Ground truth annotations of anomalous points in red.

So far, there is no established pretraining protocol for unsupervised anomaly detection in 3D point clouds. Existing work addresses the extraction of local 3D features that are highly task-specific. For point cloud registration, feature extractors often heavily downsample the input data or operate only on a small number of input points. This makes them ill-suited for anomaly localization in 3D. In this work, we develop a novel approach for pretraining local geometric descriptors that transfer well to this task. We then use this pretraining strategy to introduce a new method that outperforms existing approaches in the localization of geometric anomalies in high-resolution 3D point clouds. In particular, our key contributions are:

- We present 3D Student–Teacher, the first method for unsupervised anomaly detection that operates directly on 3D point clouds. Our method is trained only on anomaly-free data and it localizes geometric anomalies in high-resolution test samples with a single forward pass. We propose an adaptation of the well-established Student–Teacher framework for anomaly detection to three dimensions. A student network is trained to match deep local geometric descriptors of a pretrained teacher network. During inference, anomaly scores are derived from the regression errors between the student's predictions and the teacher's targets. Our method sets a new state of the art on the MVTec 3D-AD dataset presented in Chapter 9. It performs significantly better than existing methods which use voxel grids and depth images.

- We develop a self-supervised training protocol that allows the teacher network to learn generic local geometric descriptors that transfer well to the 3D anomaly detection task. The teacher extracts a geometric descriptor for each input point by aggregating local features within a limited receptive field. A decoder network is trained to reconstruct the local geometry encoded by the descriptors. Our pretraining strategy provides explicit control over the receptive field and dense feature extraction for a large number of input points. This allows us to compute anomaly scores for high-resolution point clouds without the need for intermediate subsampling.

10.2 Learning Deep 3D Descriptors

To adapt our Student–Teacher method to three dimensional data, a pretrained network that extracts local geometric features from 3D data is required. Geometric feature extraction is commonly used in 3D applications such as 3D registration or 3D pose estimation. The community has recently shifted from designing hand-crafted descriptors [Tombari et al., 2010, Salti et al., 2014] to learning-based approaches.

One line of work learns low-dimensional descriptors on local 3D patches cropped from larger input point clouds. In 3DMatch [Zeng et al., 2017] and PPFNet [Deng et al., 2018b], supervised metric learning is used to learn embeddings from annotated 3D correspondences. PPF-FoldNet [Deng et al., 2018a] pursues an unsupervised strategy where an autoencoder is trained on point pair features extracted from the local patches. Similarly, Kehl et al. [2016] introduce an autoencoder that is trained on patches of RGB-D images to obtain local features. These methods have the disadvantage that a separate patch needs to be cropped and processed for each feature. This quickly becomes computationally intractable for a large number of points.

To mitigate this problem, recent 3D feature extractors attempt to densely compute features for high-resolution inputs. Choy et al. [2019] propose FCGF, a fully convolutional approach to local geometric feature extraction for 3D registration. They design a network with sparse convolutions to efficiently processes high-resolution voxel data. Given a large number of precisely annotated local correspondences, their approach is trained using contrastive losses that encourage matching local geometries to be close in feature space. PointContrast [Xie et al., 2020] learns descriptors for 3D registration in a self-supervised fashion and does not rely on human annotations. Correspondences are automatically derived by augmenting a pair of overlapping views from a single 3D scan. While being computationally efficient, these methods require a prior voxelization that can lead to discretization inaccuracies. Furthermore, all of the discussed methods are designed to produce feature spaces that are ideally invariant to 3D rotations of the input data. In unsupervised anomaly detection, however, anomalies can manifest themselves precisely through locally rotated geometric structures. Such differences should therefore be reflected in the extracted feature vectors. This calls for the development of a different pretraining strategy that is sensitive to local rotations.

10.3 Student-Teacher Anomaly Detection in Point Clouds

In this section, we introduce 3D Student–Teacher (3D-ST), a versatile framework for the unsupervised detection and localization of geometric anomalies in high-resolution

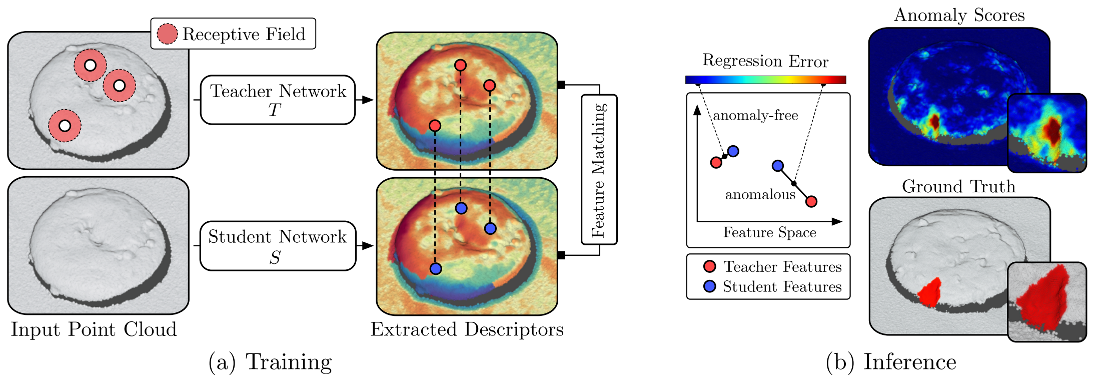


Figure 10.2: (a) Training of our proposed 3D-ST method on anomaly-free point clouds. A student network S is trained to match the local descriptors of a pretrained teacher network T. (b) Computation of anomaly scores during inference. Anomaly scores are derived from the regression error between the student and the teacher network. Increased regression errors correspond to anomalous 3D points.

3D point clouds. We build on the recent success of leveraging local descriptors from pretrained networks for anomaly detection and propose an adaptation of the 2D Student–Teacher method [Bergmann et al., 2020] to 3D data.

Given a training dataset of anomaly-free input point clouds, our goal is to create a model that can localize anomalous regions in test point clouds, i.e., to assign a real-valued anomaly score to each point. To achieve this, we design a dense feature extraction network T, called *teacher network*, that computes local geometric features for arbitrary point clouds. For anomaly detection, a *student network* S is trained on the anomaly-free point clouds against the descriptors obtained from T. During inference, increased regression errors between S and T indicate anomalous points. An overview of our approach is illustrated in Figure 10.2.

To pretrain the teacher, we present a self-supervised protocol. It works on any generic auxiliary 3D point cloud dataset and requires no human annotations.

10.3.1 Self-Supervised Learning of Dense Local Geometric Descriptors

We begin by describing how to construct a descriptive teacher network T. An overview of our pretraining protocol is displayed in Figure 10.3. Given an input point cloud $P \subset \mathbb{R}^3$ containing 3D points, its purpose is to produce a d-dimensional feature vector $\boldsymbol{f_x} \in \mathbb{R}^d$ for every $\boldsymbol{x} \in P$. The vector $\boldsymbol{f_x}$ describes the local geometry around the point $\boldsymbol{x}$, i.e., the geometry within its receptive field.

Local Feature Aggregation. The network architecture of T has two key requirements. First, it should be able to efficiently process high-resolution point clouds by computing a feature vector for each input point without downsampling the input data. Second, it requires explicit control over the receptive field of the feature vectors. In particular, it

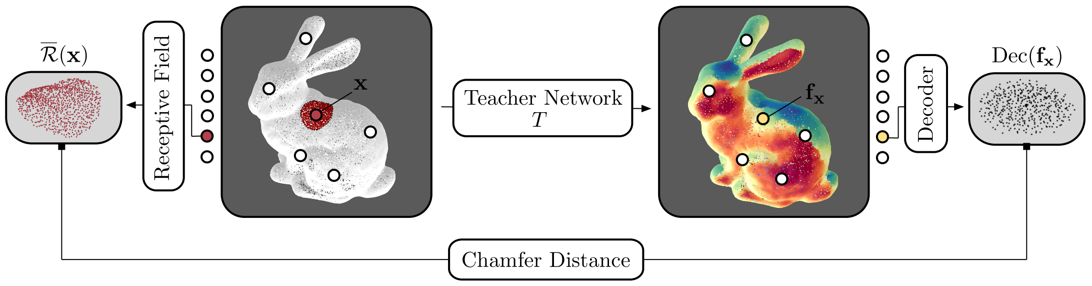


Figure 10.3: Overview of our proposed self-supervised pretraining strategy. A teacher network is trained to output local geometric descriptors for each 3D point in the input sample with a single forward pass. Simultaneously, a decoder network transforms randomly sampled descriptors of the teacher and attempts to fit the local receptive field around its respective input point.

has to be possible to efficiently compute all points within the receptive field of an output descriptor.

To meet these requirements, we construct the k-nearest neighbor graph of the input point cloud and initialize $\boldsymbol{f_x} = \mathbf{0}$. We then pass the input sample through a series of residual blocks, where each block updates the feature vector of each 3D point $\boldsymbol{x}$ from $\boldsymbol{f_x} \in \mathbb{R}^d$ to $\tilde{\boldsymbol{f}}_{\boldsymbol{x}} \in \mathbb{R}^d$. These blocks are inspired by RandLA-Net [Hu et al., 2020, 2021], an efficient and lightweight neural architecture for semantic segmentation of large-scale point clouds. In semantic segmentation tasks, the absolute position of a point is often related to its class, e.g., in autonomous driving datasets. Here, we want our model to produce features that describe the local geometry of an object independent of its absolute location. We therefore make the residual blocks translation-invariant by removing any dependency on absolute coordinates. This significantly increases the performance when used for anomaly detection as underlined by the results of our experiments in Section 10.4.

The architecture of our residual blocks is visualized in Figure 10.4(a). The input features are first passed through a shared MLP, followed by two local feature aggregation (LFA) blocks. The output features are added to the input after processing both by an additional shared MLP. The features are transformed by a series of residual blocks and a final shared MLP with a single hidden layer that maintains the dimension of the descriptors, i.e., $\tilde{\boldsymbol{f}}_{\boldsymbol{x}} \in \mathbb{R}^d$.

The purpose of the LFA block is to aggregate the geometric information from the local vicinity of each input point. To this end, it computes the nearest neighbors $\mathrm{knn}(\boldsymbol{x}) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$ of all $\boldsymbol{x} \in P$ and a set of local geometric features G for each point pair defined by

$$G(\boldsymbol{x}, \boldsymbol{x}_j) = (\boldsymbol{x} - \boldsymbol{x}_j) \odot \|(\boldsymbol{x} - \boldsymbol{x}_j)\|_2^2 \quad \text{where } j \in \{1, \ldots, k\}. \tag{10.1}$$

The operator $\odot$ denotes the concatenation operation and $\|\cdot\|_2^2$ denotes the L_2-norm. Since G only depends on the difference vectors between neighboring points, our network is by design invariant to translations of the input data. Our experiments show that this invariance of our local feature extractor is crucial for anomaly detection performance.

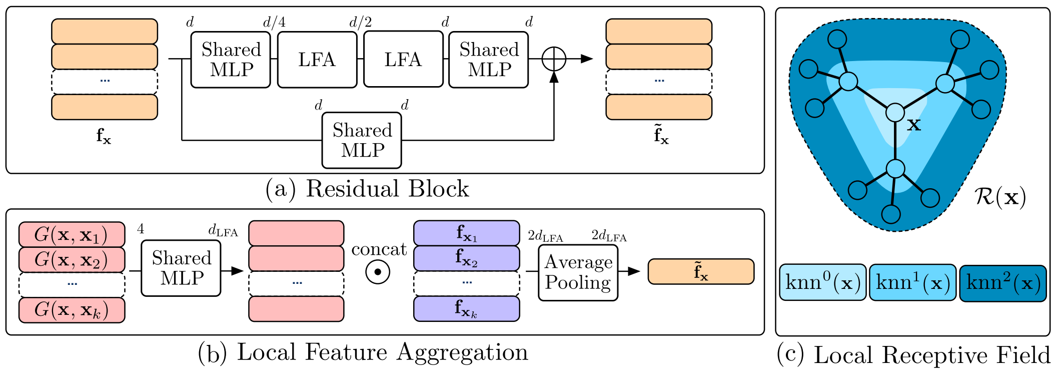


Figure 10.4: Overview of our network architecture. (a) Residual block that performs a series of local feature aggregation steps to update the feature vectors. (b) The local feature aggregation block aggregates geometric information of surrounding points. (c) Visualization of the receptive field of a point $\boldsymbol{x}$.

Therefore, we make this small but important change to the LFA block. A schematic description of such a block is given in Figure 10.4(b).

For each LFA block, the set of geometric features $G(\boldsymbol{x}, \boldsymbol{x}_j)$ is passed through a shared MLP producing feature vectors of dimension d_{LFA}. These are concatenated with the set of input features $\{\boldsymbol{f}_{\boldsymbol{x}_1}, \ldots, \boldsymbol{f}_{\boldsymbol{x}_k}\}$. The output feature vector of the LFA block $\tilde{\boldsymbol{f}}_{\boldsymbol{x}}$ is obtained by an average-pooling operation of the concatenated features, yielding a feature vector of dimension $2d_{\text{LFA}}$.

Reconstructing Local Receptive Fields. To pretrain T in a self-supervised fashion, we employ a decoder network that reproduces the local receptive field of a feature vector. The design of our network architecture allows an efficient computation of all points within the receptive field $\mathcal{R}(\boldsymbol{x})$ of a point $\boldsymbol{x}$, i.e., all points that affect the feature vector $\boldsymbol{f}_{\boldsymbol{x}}$. Each LFA block depends on the features of the surrounding nearest neighbors $\text{knn}(\boldsymbol{x})$. Whenever an LFA block is executed, $\mathcal{R}(\boldsymbol{x})$ grows by one hop in the nearest-neighbor graph. The receptive field can therefore be obtained by iteratively traversing the nearest neighbor graph:

$$\mathcal{R}(\boldsymbol{x}) = \bigcup_{l=0}^{L} \text{knn}^l(\boldsymbol{x}), \quad \text{where} \quad \text{knn}^l(\boldsymbol{x}) = \bigcup_{\boldsymbol{y} \in \text{knn}^{l-1}(\boldsymbol{x})} \text{knn}(\boldsymbol{y}) \tag{10.2}$$

and $\text{knn}^0 = \{\boldsymbol{x}\}$. L denotes the total number of LFA blocks in the network. Figure 10.4(c) visualizes this definition of the receptive field.

The decoder $\text{Dec} : \mathbb{R}^d \to \mathbb{R}^{3\times m}$ upsamples a feature vector to produce m 3D points by applying an MLP. For pretraining, we extract descriptors from an input point cloud by passing it through the local feature extractor. We then randomly sample a set of points Q from the input. For each $\boldsymbol{x} \in Q$, we compute the receptive fields $\mathcal{R}(\boldsymbol{x})$ and pass their respective feature vectors through the decoder. To train the decoder, we minimize the

Chamfer distance [Barrow et al., 1977] between the decoded points and the receptive fields. Since our network architecture is not aware of the absolute coordinates of $\boldsymbol{x}$, we additionally compute the mean $\bar{\boldsymbol{x}}$ of all $\boldsymbol{x} \in \mathcal{R}(\boldsymbol{x})$ and subtract it from each point, yielding the set $\overline{\mathcal{R}}(\boldsymbol{x}) = \mathcal{R}(\boldsymbol{x}) - \bar{\boldsymbol{x}}$. The loss function for our self-supervised training procedure can then be written as:

$$\mathcal{L}_C(T, \mathrm{Dec}) = \frac{1}{|Q|} \sum_{\boldsymbol{x} \in Q} \mathrm{Chamfer}(\mathrm{Dec}(\boldsymbol{f_x}), \overline{\mathcal{R}}(\boldsymbol{x})). \tag{10.3}$$

Data Normalization. For our teacher network to be applied to any point cloud not included in the pretraining dataset, some form of data normalization is required. Since our network operates on the distance vectors of neighboring points, we choose to normalize the input data with respect to these distances. More specifically, we compute the average distance between each point and its nearest neighbors over the entire training set, i.e.,

$$s = \frac{1}{k\ |P|} \sum_{\boldsymbol{x} \in P} \sum_{\boldsymbol{y} \in \mathrm{knn}(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2. \tag{10.4}$$

We then scale the coordinates of each data sample in the pretraining dataset by $1/s$. This allows us to apply the teacher network to arbitrary point cloud datasets, as long as the same data normalization technique is used.

10.3.2 Matching Geometric Features for 3D Anomaly Detection

Finally, we describe how to employ the pretrained teacher network T to train a student network S for anomaly detection. Given a dataset of anomaly-free point clouds, we first calculate the scaling factor s for this dataset as defined in (10.4). The weights of T remain constant during the entire anomaly detection training. S exhibits the identical network architecture as T and is initialized with uniformly distributed random weights. Each training point cloud $P_t \subset \mathbb{R}^3$ is passed through both networks, T and S, to compute dense features $\boldsymbol{f}_{\boldsymbol{x}}^T$ and $\boldsymbol{f}_{\boldsymbol{x}}^S$ for all $\boldsymbol{x} \in P_t$, respectively. The weights of S are optimized to reproduce the geometric descriptors of T by computing the feature wise L_2-distance as an error function:

$$E(P_t) = \frac{1}{|P_t|} \sum_{\boldsymbol{x} \in P_t} \left\| \boldsymbol{f}_{\boldsymbol{x}}^S - (\boldsymbol{f}_{\boldsymbol{x}}^T - \boldsymbol{\mu})\ \mathrm{diag}(\boldsymbol{\sigma})^{-1} \right\|_2^2. \tag{10.5}$$

The training loss over the dataset samples in a minibatch of B elements can then be written as $\mathcal{L}_{ST}(S) = \frac{1}{B} \sum_{j=1}^{B} E(P_{t,j})$. We transform the teacher features to be centered around $\mathbf{0}$ with unit standard deviation. This requires the computation of the component wise means $\boldsymbol{\mu} \in \mathbb{R}^d$ and standard deviations $\boldsymbol{\sigma} \in \mathbb{R}^d$ of all teacher features over the whole training set. We denote the inverse of the diagonal matrix filled with $\boldsymbol{\sigma}$ by $\mathrm{diag}(\boldsymbol{\sigma})^{-1}$.

During inference, anomaly scores $A(\boldsymbol{x})$ are derived for each point $\boldsymbol{x} \in P_i$ in a test point cloud $P_i \subset \mathbb{R}^3$. They are given by the regression errors between the respective features of the student and the teacher network, i.e.,

$$A(\boldsymbol{x}) = \|\boldsymbol{f}_{\boldsymbol{x}}^S - (\boldsymbol{f}_{\boldsymbol{x}}^T - \boldsymbol{\mu})\ \mathrm{diag}(\boldsymbol{\sigma})^{-1}\|_2^2. \tag{10.6}$$

The intuition behind this is that anomalous geometries produce features that the student network has not observed during training, and is hence unable to reproduce. Large regression errors indicate anomalous geometries.

10.4 Experiments

10.4.1 Experiment Setup

To demonstrate the effectiveness of our approach, we perform extensive experiments on the MVTec 3D Anomaly Detection (MVTec 3D-AD) dataset. We benchmark the performance of our 3D-ST method against existing methods for unsupervised 3D anomaly detection. In particular, we follow the initial benchmark on MVTec 3D-AD from the previous chapter and compare 3D-ST against the Voxel f-AnoGAN, the Voxel Autoencoder, and the Voxel Variation Model. The benchmark also includes their respective counterparts that process depth images instead of voxel grids by exchanging 3D with 2D convolutions.

Teacher Pretraining. To pretrain the teacher network (cf. Section 10.3.1), we generate synthetic 3D scenes using objects of the ModelNet10 dataset [Wu et al., 2015]. It consists of over 5000 3D models divided into 10 different object categories.

We generate a scene of our pretraining dataset by randomly selecting 10 samples from ModelNet10 and scaling the longest side of their bounding box to 1. The objects are rotated around each 3D axis with angles sampled uniformly from the interval $[0, 2\pi]$. Each object is placed at a random location sampled uniformly from $[-3, 3]^3$. Point clouds are created by selecting a fixed number of points from the scene using farthest point sampling [Moenning and Dodgson, 2003]. The training and validation datasets consist of 1000 and 50 point clouds, respectively. Our experiments show that using such a synthetic dataset for pretraining yields local descriptors that are well suited for 3D anomaly detection. In our ablation studies, we additionally investigate the use of real-world datasets from different domains for pretraining, namely Semantic KITTI [Geiger et al., 2012, Behley et al., 2019], MVTec ITODD [Drost et al., 2017], and 3DMatch [Zeng et al., 2017].

The teacher network T consists of 4 residual blocks and processes $n = 64000$ input points. We perform experiments using two different feature dimensions $d \in \{64, 128\}$. The shared MLPs in all network blocks are implemented with a single dense layer, followed by a LeakyReLU activation with a negative slope of 0.2. The input and output dimensions of each shared MLP are given in Figure 10.4. For local feature aggregation, a nearest neighbor graph with $k = 32$ neighbors is constructed. The pretraining runs for 250 epochs using the Adam optimizer with an initial learning rate of 10^{-3} and a weight decay of 10^{-6}. At each training step, a single input sample is fed through the teacher network. To generate reconstructions of local receptive fields, 16 randomly selected descriptors from the output of T are passed through the decoder network, which is implemented as an MLP with input dimension d, two hidden layers of dimension 128, and an output layer that reconstructs $m = 1024$ points. Each hidden layer is followed

		bagel	cable gland	carrot	cookie	dowel	foam	peach	potato	rope	tire	mean
Voxel	GAN	0.440	0.453	0.825	0.755	0.782	0.378	0.392	0.639	0.775	0.389	0.583
	AE	0.260	0.341	0.581	0.351	0.502	0.234	0.351	0.658	0.015	0.185	0.348
	VM	0.453	0.343	0.521	0.697	0.680	0.284	0.349	0.634	0.616	0.346	0.492
Depth	GAN	0.111	0.072	0.212	0.174	0.160	0.128	0.003	0.042	0.446	0.075	0.143
	AE	0.147	0.069	0.293	0.217	0.207	0.181	0.164	0.066	0.545	0.142	0.203
	VM	0.280	0.374	0.243	0.526	0.485	0.314	0.199	0.388	0.543	0.385	0.374
PCD	3D-ST$_{64}$	0.939	0.440	0.984	0.904	0.876	**0.633**	0.937	**0.989**	0.967	0.507	0.818
	3D-ST$_{128}$	**0.950**	**0.483**	**0.986**	**0.921**	**0.905**	0.632	**0.945**	0.988	**0.976**	**0.542**	**0.833**

Table 10.1: Anomaly localization results for each evaluated method and dataset category. The area under the PRO curve is reported for an integration limit of 0.3. The best performing method is highlighted in boldface.

by a LeakyReLU activation with negative slope of 0.05. After the training, we select the model with the lowest validation error as the teacher network.

Anomaly Detection. The student network S in our 3D-ST method has the same network architecture as the teacher. It is trained for 100 epochs on the anomaly-free training split of the MVTec 3D-AD dataset. We train with a batch size of $B = 1$. This is equivalent to processing a large number of local patches per iteration due to the limited receptive field of the employed networks. We use Adam with an initial learning rate of 10^{-3} and weight decay 10^{-5}. Each point cloud is reduced to $|P| = 64000$ input points using farthest point sampling. For inference, we select the student network with the lowest validation error.

The evaluation on MVTec 3D-AD requires to predict an anomaly score for each pixel in the original (x, y, z) images. To do this, we apply harmonic interpolation [Evans, 2010] to the pixels that were not assigned anomaly scores by our method. We follow the standard evaluation protocol of MVTec 3D-AD and compute the per-region overlap (PRO) and the corresponding false positive rate for successively increasing anomaly thresholds. We then report the area under the PRO curve (AU-PRO) integrated up to a false positive rate of 30%. We normalize the resulting values to the interval $[0, 1]$.

10.4.2 Experiment Results

Table 10.1 shows quantitative results of each evaluated method on every object category of MVTec 3D-AD for anomaly localization. The top three rows list the performance of the voxel-based methods. The following three rows list the performance of the respective methods on 2D depth images. The bottom two rows show the performance of our 3D-ST method on 3D point cloud data, evaluated for two different descriptor dimensions $d \in \{64, 128\}$. Our method performs significantly better than all other methods on every dataset category. Increasing the descriptor dimension from 64 to 128 yields a slight overall improvement of 1.5 percentage points.

Table 10.2 lists the performance of each evaluated method for anomaly classification. We compute a single anomaly score for each dataset sample by taking the maximum over all anomaly scores. The area under the ROC curve is reported as performance metric. Our 3D-ST method achieves the best performance on the majority of dataset categories.

		bagel	cable gland	carrot	cookie	dowel	foam	peach	potato	rope	tire	mean
Voxel	GAN	0.383	0.623	0.474	0.639	0564	0.409	0.617	0.427	0.663	0.577	0.538
	AE	0.693	0.425	0.515	0.790	0.494	0.558	0.537	0.484	0.639	0.583	0.572
	VM	0.750	**0.747**	0.613	0.738	0.823	0.693	0.679	0.652	0.609	**0.690**	0.699
Depth	GAN	0.530	0.376	0.607	0.603	0.497	0.484	0.595	0.489	0.536	0.521	0.524
	AE	0.468	0.731	0.497	0.673	0.534	0.417	0.485	0.549	0.564	0.546	0.546
	VM	0.510	0.542	0.469	0.576	0.609	**0.699**	0.450	0.419	0.668	0.520	0.546
PCD	3D-ST$_{64}$	0.805	0.409	0.809	**0.905**	0.736	0.588	0.743	0.648	**0.978**	0.496	0.712
	3D-ST$_{128}$	**0.862**	0.484	**0.832**	0.894	**0.848**	0.663	**0.763**	**0.687**	0.958	0.486	**0.748**

Table 10.2: Anomaly classification results for each evaluated method and dataset category. The area under the ROC curve is reported. The best performing method is highlighted in boldface.

Qualitative results of our method are shown in Figure 10.1. 3D-ST manages to localize anomalies over a range of different object categories, such as the crack in the *bagel*, the contamination on the *rope* and the *tire*, or the cut in the *foam* and the *potato*. Additional qualitative results are shown in Figure 10.5.

Many anomaly detection applications require particularly low false positive rates. We therefore report the mean performance of all evaluated methods when varying the integration limit of the PRO curve in Figure 10.6. Our method outperforms all other evaluated methods for any chosen integration limit. The relative difference in performance is particularly large for lower integration limits. This makes our approach well-suited for practical applications.

10.4.3 Ablation Studies

We additionally perform various ablation studies with respect to the key hyperparameters of our proposed method. Figure 10.7 shows the dependency of the mean performance of our method on the number of input points $|P|$, the feature dimension d, or the number of nearest neighbor points k used for local feature aggregation. We additionally visualize the inference time and the memory consumption of each model during training and evaluation.[1] We find that our method is insensitive to the choice of each hyperparameter. In particular, the mean performance of each evaluated model outperforms the best performing competing model from the baseline experiments by a large margin. The mean performance of our model grows monotonically with respect to each considered hyperparameter. It eventually saturates, whereas the inference time and memory consumption continue to increase super-linearly.

Feature Space of the Teacher Network. We depict the effectiveness of our pretraining strategy in Figure 10.8. The left bar plot shows the mean performance with respect to changes in the training strategy of our method. The first bar indicates how the performance changes when we initialize the teacher's weights randomly and perform no pretraining. As expected, the performance drops significantly. The next bar shows the performance when concatenating the absolute point coordinates of each 3D point to the

[1] All models were implemented using the PyTorch library [Paszke et al., 2019]. Inference times and memory consumption were measured on an NVIDIA Tesla V100 GPU.

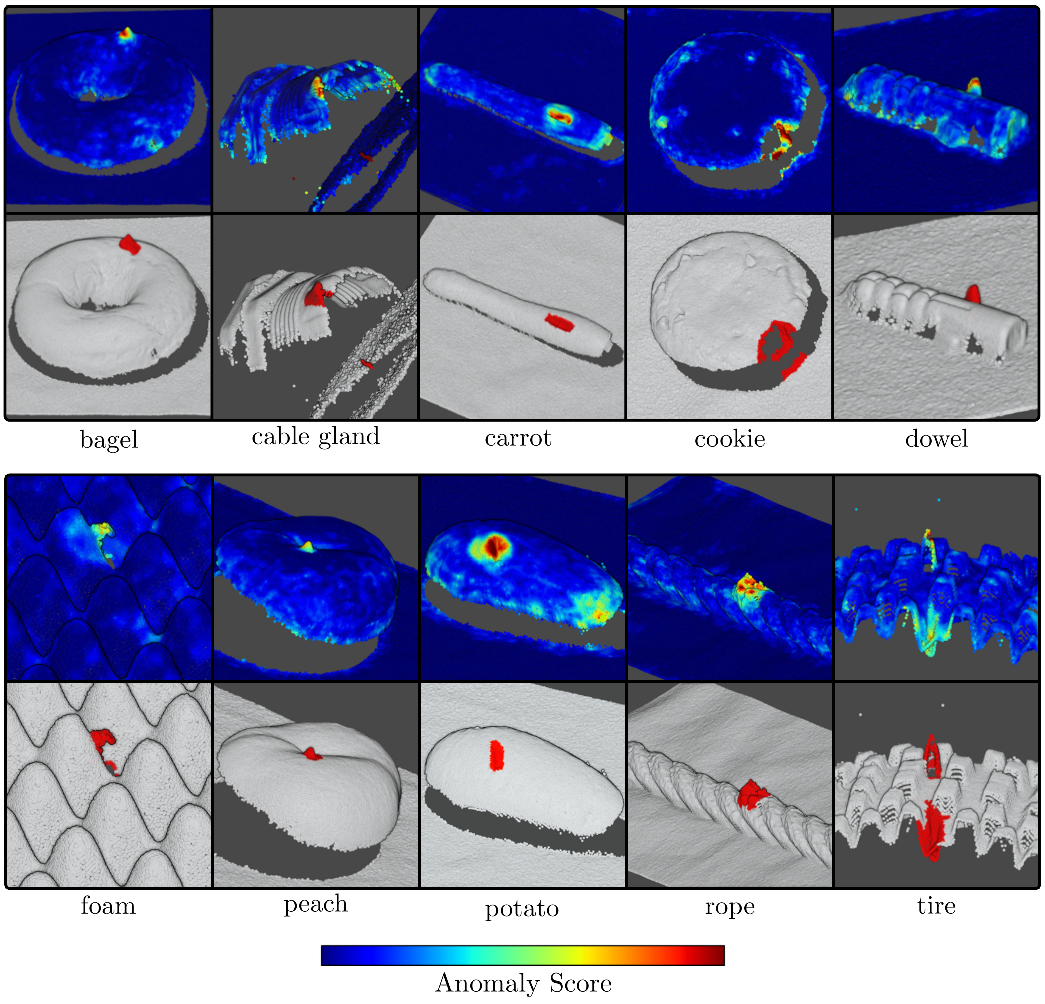


Figure 10.5: Additional qualitative results of our method on the MVTec 3D-AD dataset. Top row: Anomaly scores for each 3D point predicted by our algorithm. Bottom row: Ground truth annotations of anomalous points in red.

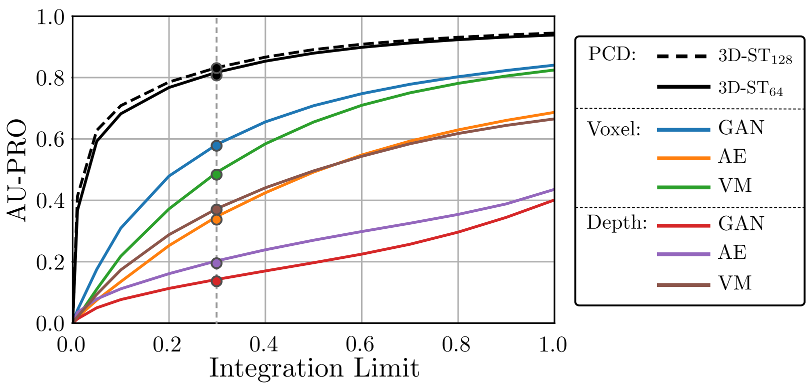


Figure 10.6: Performance of each method with respect to the integration limit. The AU-PRO at an integration limit of 0.3 is marked by a vertical line. In real-world scenarios, the performance at lower integration limits is of particular importance.

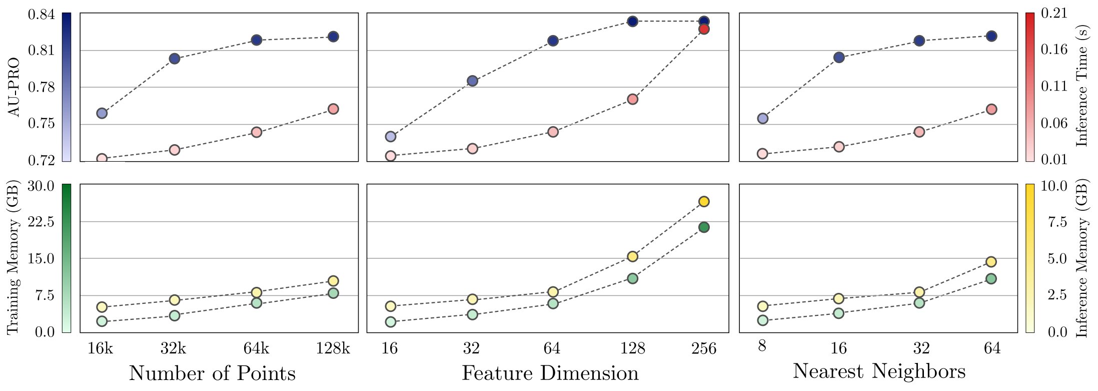


Figure 10.7: Performance of our method when changing various key hyperparameters, i.e., the number of input points, the feature dimension, and the number of nearest neighbors used for local feature aggregation.

local feature aggregation function G, as proposed in [Hu et al., 2020]. This no longer makes our network translation invariant and decreases the performance. This indicates that translation invariance is indeed important for our network architecture and that our modification to the local feature aggregation module has a significant impact. The third bar shows the performance of our method when trying to additionally incorporate rotation invariance. We achieve this by augmentation of the training data with randomly sampled rotations such that locally rotated geometries are also considered as anomaly-free. The performance is still significantly below our method, which indicates that sensitivity to local rotations is beneficial for 3D anomaly detection.

Pretraining Dataset. In most of our experiments, we use synthetically generated scenes created from objects from the ModelNet10 dataset as described above. Our pretraining strategy does not require any human annotations and can operate on arbitrary input

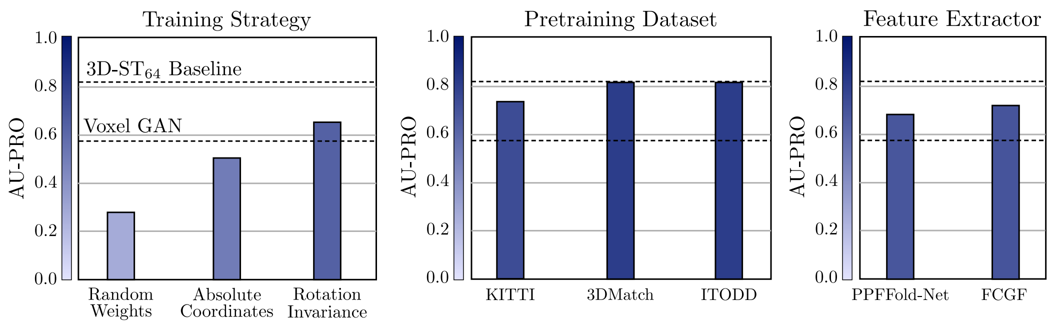


Figure 10.8: Sensitivity of our method to changes in the pretraining strategy of the teacher network. We show the effects of a random teacher initialization, adding absolute point coordinates, and rotation invariant features. We further examine the influence of different pretraining datasets and feature extractors.

point clouds. We are thus interested in whether the performance varies when using different pretraining datasets. As first experiment, we randomly select 1000 training scenes from the Semantic KITTI autonomous driving dataset, which is captured with a LIDAR sensor. Secondly, we pretrain a teacher on all samples of 3DMatch, an indoor dataset originally designed for point cloud registration. Finally, we use all samples of the MVTec ITODD dataset [Drost et al., 2017], an industrial dataset originally designed for 3D pose estimation. The center bar chart in Figure 10.8 shows the mean performance of our method when using these three datasets for pretraining, compared to our baseline model. We find that our method does not strongly depend on the specific dataset chosen for pretraining when using ITODD or 3DMatch. We observe a slight performance gap for the KITTI dataset, which is likely due to the large domain shift.

Feature Extractor. We additionally test the performance of our method when our pretrained teacher network is replaced by descriptors obtained from different feature extractors. In particular, we compare against features obtained from PPFFold-Net [Deng et al., 2018a] and FCGF [Choy et al., 2019]. For both, we use publicly available pretrained models.[2] PPF-FoldNet outputs a single 512-dimensional descriptor for patches cropped from a local neighborhood of 1024 points around each input point. Since it requires patch-based feature extraction, producing descriptors for a large number of input points becomes prohibitively slow. We therefore only extract 1000 descriptors for each point cloud with PPF-FoldNet. FCGF outputs 32-dimensionsal descriptors and was pretrained in a supervised fashion on the 3DMatch dataset by finding correspondences for 3D registration. Since it requires a prior voxelization of the input data, we select a voxel size of 3.5 mm and extract descriptors for 64000 points.

We train our student network to match the features extracted from these pretrained networks instead of our proposed teacher network. The feature dimension of the output layer of our student network is adapted to match the feature dimension of the descriptors.

[2]`https://github.com/XuyangBai/PPF-FoldNet`, `https://github.com/chrischoy/FCGF`

The results are shown in the right bar plot in Figure 10.8. Transferring the features of both networks yields better performance than the Voxel GAN, which is the previously best-performing method that was trained from scratch. This underlines the effectiveness of using pretrained geometric descriptors for 3D anomaly detection. Both extractors do not reach the performance of our proposed pretraining strategy that is specifically designed for the anomaly detection problem.

10.5 Conclusion

We proposed 3D-ST, a new approach to the challenging problem of unsupervised anomaly detection in 3D point clouds. Our method is trained exclusively on anomaly-free samples. During inference, it localizes geometric structures that deviate from the ones present in the training set. In particular, we proposed an adaptation of Student–Teacher anomaly detection from 2D to 3D. Existing 3D methods are trained from random weight initializations. In contrast to this, our method leverages the descriptiveness of deep local geometric features extracted from a pretrained network. To address the lack of pretraining protocols for 3D anomaly detection, we introduced a self-supervised strategy based on the reconstruction of local receptive fields. This enables the training of teacher networks that produce dense local geometric descriptors for arbitrary 3D point clouds.

For anomaly detection, a student network matches the geometric descriptors of the teacher on anomaly-free data. During inference, anomaly scores are derived for each 3D point by computing the regression error between its associated student and teacher descriptors. Extensive experiments on the MVTec 3D Anomaly Detection dataset show that our method outperforms all existing methods by a large margin. We performed various ablation studies which demonstrate that our method is computationally efficient and robust to the choice of hyperparameters and pretraining datasets.

11 Conclusion

In this thesis, we contributed to the field of unsupervised anomaly detection and localization from a computer vision perspective. We briefly summarize our contributions and point the reader to possible directions for future research.

11.1 Summary

Anomaly Detection in Color and Grayscale Images

In Chapter 3, we introduced the SSIM-Autoencoder, a new variant of convolutional autoencoders that computes anomaly scores based on the structural similarity index. Compared to methods that derive anomaly scores from per-pixel residuals, our approach is less sensitive to small inaccuracies in the reconstruction and improves the detection of anomalies that occur as structural differences between the input and reconstructed images when the respective pixels' color values are roughly consistent. During the development of our new autoencoder, we found that there is a lack of datasets for unsupervised anomaly detection. Therefore, we contributed a texture dataset of two woven fabric materials to evaluate our approach.

Since this texture dataset it is still relatively small and only covers a single application scenario, we introduced the MVTec Anomaly Detection dataset (MVTec AD) in Chapter 4. It allows for the evaluation of unsupervised anomaly detection methods on various texture and object classes with different types of anomalies. An initial benchmark of existing methods indicated considerable room for improvement.

The comparison of different methods on MVTec AD revealed that descriptors extracted from pretrained networks work well when transferred to the anomaly detection problem. Unfortunately, these models often rely on patch-based feature extraction and shallow machine learning models, which limits their performance. To overcome these limitations, in Chapter 5, we introduced a framework for unsupervised anomaly detection based on Student–Teacher learning. We designed a pretrained teacher network that efficiently extracts dense local feature descriptors with a single forward pass. For anomaly detection, an ensemble of student networks is trained to predict the output of the teacher network on anomaly-free data. During inference, anomalies are detected through increased regression errors and predictive variances of the students. The approach can be easily extended to detect anomalies at multiple scales. Our experiments show that our method performs significantly better than existing methods on a number of anomaly detection datasets.

In Chapter 6, we find that existing datasets for unsupervised anomaly detection predominantly focus on structural anomalies that manifest themselves through novel visual structures that occur in locally confined regions, such as scratches, dents, or contaminations in manufactured products. This is also the case for our MVTec AD dataset. However, anomalies may also appear as violations of underlying logical constraints of the anomaly-free data. Therefore, we created a new dataset that equally focuses on the detection of both structural and logical anomalies. We further introduced a new performance metric that takes the different modalities of the two anomaly types into account. An initial benchmark showed that existing methods do not perform well in the detection of logical anomalies.

In Chapter 7, we introduced Global Context Anomaly Detection (GCAD), a new method that allows for the joint localization of both structural and logical anomalies. It extends our Student–Teacher framework by a global model branch that enables the detection of logical anomalies. Extensive experiments showed that our approach performed equally well in the detection of structural and logical anomalies and improved the state of the art in the joint detection of both.

Geometric Anomaly Detection in Three-Dimensional Data

In the remaining parts of the thesis, we shifted our focus to the detection of geometric anomalies in three-dimensional data. We found that currently, there exist only very few methods that consider this task. We attribute this to the lack of suitable datasets that can be used to develop new methods. Therefore, in Chapter 9, we presented the MVTec 3D Anomaly Detection dataset, a comprehensive 3D dataset for the task of unsupervised anomaly detection and localization. It consists of ten different object categories from industrial manufacturing scenarios, acquired with a high-resolution 3D sensor. It contains various geometric anomalies that appear in the surface of the inspected objects. An initial benchmark of existing methods revealed considerable room for improvement.

To improve over existing 3D anomaly detection methods, we extended our Student–Teacher method to point cloud data in Chapter 10. In particular, we introduced a self-supervised pretraining protocol to create teacher networks that efficiently extract local geometric descriptors from any point cloud. For anomaly detection, a student network matches the geometric descriptors of the teacher on anomaly-free data. During inference, anomaly scores are derived for each input point by computing the regression error between its associated student and teacher descriptors. Experiments on the MVTec 3D Anomaly Detection dataset show that our 3D Student–Teacher method performs better than all existing methods by a large margin. We performed various ablation studies which demonstrate that our method is computationally efficient and robust to the choice of hyperparameters and pretraining datasets.

11.2 Future Research

In this thesis, we contributed several new datasets and methods for unsupervised anomaly detection. While our datasets provide a solid foundation to create and benchmark new methods, there is still room for the development of new datasets due to the wide variety of conceivable anomaly detection applications and types of anomalies that may occur in practice. Hence, we encourage researchers to strive for the creation of even more diverse anomaly detection datasets that challenge existing methods. In particular, it would be of interest to create datasets that cover data modalities that are entirely different from those investigated in this thesis. For instance, anomaly detection in data obtained from hyperspectral cameras or X-Ray sensors is not yet thoroughly explored.

Regarding anomaly detection methods, our approaches achieve detection rates that make them well-suited to be deployed in many real-world anomaly detection systems. However, they still do not achieve perfect localization and classification accuracies. Especially for the detection of logical or particularly subtle anomalies, there is still room for improvement. The methods developed in this thesis greatly rely on the availability of descriptors extracted from pretrained neural networks. Therefore, a promising avenue for future research is the development of new pretraining protocols that lead to better detection rates or more efficient and lightweight network architectures that reduce the computational overhead. On the other hand, it would also be interesting to investigate if it is possible to improve the anomaly detection performance of methods that are trained from scratch to match the performance of methods that employ pretrained feature extractors. While the former are often straightforward to be transferred to new input domains, the latter rely on domain-specific pretraining protocols that may not be available for a particular application domain.

Bibliography

T. Adão, J. Hruška, L. Pádua, J. Bessa, E. Peres, R. Morais, and J. J. Sousa. Hyperspectral imaging: A review on uav-based sensors, data processing and applications for agriculture and forestry. *Remote Sensing*, 9(11), 2017. doi: 10.3390/rs9111110.

J. An and S. Cho. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. *SNU Data Mining Center, Technical Report*, 2015.

J. T. Andrews, T. Tanay, E. J. Morton, and L. D. Griffin. Transfer Representation-Learning for Anomaly Detection. In *Anomaly Detection Workshop at the International Conference on Machine Learning (ICML)*, 2016.

M. Arjovsky and L. Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *International Conference on Learning Representations (ICLR)*, 2017.

I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543, 2016. doi: 10.1109/CVPR.2016.170.

U. Baid et al. The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification. *arXiv preprint arXiv:2107.02314*, 2021. doi: 10.48550/arXiv.2107.02314.

C. Bailer, T. A. Habtegebrial, K. Varanasi, and D. Stricker. Fast Dense Feature Extraction with CNNs that have Pooling or Striding Layers. In *British Machine Vision Conference (BMVC)*, 2017. doi: 10.5244/c.31.101.

S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, et al. Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features. *Scientific Data*, 4(1), 2017. doi: 10.1038/sdata.2017.117.

H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching. In *IJCAI*, pages 659–663, 1977.

C. Baur, B. Wiestler, S. Albarqouni, and N. Navab. Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 161–169. Springer International Publishing, 2019. doi: 10.1007/978-3-030-11723-8_16.

J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *IEEE International Conference on Computer Vision (ICCV)*, pages 9296–9306, 2019. doi: 10.1109/ICCV.2019.00939.

S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li. Review: Deep learning on 3d point clouds. *Remote Sensing*, 12(11), 2020. doi: 10.3390/rs12111729.

W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler. The power of ensembles for active learning in image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9368–9377, 2018. doi: 10.1109/CVPR.2018.00976.

M. Bengs, F. Behrendt, J. Krüger, R. Opfer, and A. Schlaefer. Three-dimensional deep learning with spatial erasing for unsupervised anomaly segmentation in brain MRI. *International Journal of Computer Assisted Radiology and Surgery*, 16, 2021. doi: 10.1007/s11548-021-02451-9.

P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592, 2019a. doi: 10.1109/CVPR.2019.00982.

P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger. Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pages 372–380. INSTICC, SciTePress, 2019b. doi: 10.5220/0007364503720380.

P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. Uninformed Students: Student-Teacher Anomaly Detection With Discriminative Latent Embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4182–4191, 2020. doi: 10.1109/CVPR42600.2020.00424.

P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. *International Journal of Computer Vision*, 129(4):1038–1059, 2021. doi: 10.1007/s11263-020-01400-4.

P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. Beyond Dents and Scratches: Logical Constraints in Unsupervised Anomaly Detection and Localization. *International Journal of Computer Vision*, 130(4):947—-969, 2022a. doi: 10.1007/s11263-022-01578-9.

P. Bergmann, X. Jin, D. Sattlegger, and C. Steger. The MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics*

Theory and Applications - Volume 5: VISAPP, pages 202–213. INSTICC, SciTePress, 2022b. doi: 10.5220/0010865000003124.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, and C. Cadena. Fishyscapes: A Benchmark for Safe Semantic Segmentation in Autonomous Driving. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2403–2412, 2019. doi: 10.1109/ICCVW.2019.00294.

I. Borg and P. Groenen. Modern multidimensional scaling: Theory and applications. *Journal of Educational Measurement*, 40(3):277–280, 2003.

T. Böttger and M. Ulrich. Real-time texture error detection on textured surfaces with compressed sensing. *Pattern Recognition and Image Analysis*, 26(1):88–94, 2016. doi: 10.1134/S1054661816010053.

P. Burlina, N. Joshi, and I.-J. Wang. Where's wally now? deep generative and discriminative embeddings for novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11499–11508, 2019. doi: 10.1109/CVPR.2019.01177.

S. Buschjager, P.-J. Honysz, and K. Morik. Randomized outlier detection with trees. *International Journal of Data Science and Analytics*, 13:1–14, 2022. doi: 10.1007/s41060-020-00238-w.

D. Carrera, F. Manganini, G. Boracchi, and E. Lanzarone. Defect detection in sem images of nanofibrous materials. *IEEE Transactions on Industrial Informatics*, 13(2): 551–561, 2017. doi: 10.1109/TII.2016.2641472.

R. Chalapathy, A. K. Menon, and S. Chawla. Anomaly Detection using One-Class Neural Networks. *arXiv preprint arXiv:1802.06360*, 2018. doi: 10.48550/arXiv.1802.06360.

A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015. doi: 10.48550/arXiv.1512.03012.

R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. doi: 10.1109/CVPR.2017.16.

C. Choy, J. Park, and V. Koltun. Fully Convolutional Geometric Features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 8957–8965, 2019. doi: 10.1109/ICCV.2019.00905.

N. Cohen and Y. Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357v1*, 2020. doi: 10.48550/arXiv.2005.02357.

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016. doi: 10.1109/CVPR.2016.350.

A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443, 2017. doi: 10.1109/CVPR.2017.261.

D. Danon, H. Averbuch-Elor, O. Fried, and D. Cohen-Or. Unsupervised natural image patch learning. *Computational Visual Media*, 5(3):229–237, 2019. doi: 10.1007/s41095-019-0147-y.

H. Deng, T. Birdal, and S. Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *European Conference on Computer Vision (ECCV)*, pages 620–638. Springer International Publishing, 2018a. doi: 10.1007/978-3-030-01228-1_37.

H. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 195–205, 2018b. doi: 10.1109/CVPR.2018.00028.

J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial Feature Learning. *International Conference on Learning Representations (ICLR)*, 2017.

A. Dosovitskiy and T. Brox. Generating Images with Perceptual Similarity Metrics based on Deep Networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

B. Drost, M. Ulrich, P. Bergmann, P. Härtinger, and C. Steger. Introducing MVTec ITODD — A Dataset for 3D Object Recognition in Industry. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2200–2208, 2017. doi: 10.1109/ICCVW.2017.257.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. doi: 10.5555/1953048.2021068.

T. Ehret, A. Davy, J.-M. Morel, and M. Delbracio. Image Anomalies: A Review and Synthesis of Detection Methods. *Journal of Mathematical Imaging and Vision*, 61(5): 710–743, 2019. doi: 10.1007/s10851-019-00885-0.

L. C. Evans. *Partial differential equations.* American Mathematical Society, 2010.

M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. doi: 10.1007/s11263-014-0733-5.

T. Fernando, H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Deep learning for medical anomaly detection – a survey. *ACM Computing Surveys*, 54(7), 2021. doi: 10.1145/3464423.

E. Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989. doi: 10.2307/1403797.

G. Flitton, A. Mouton, and T. P. Breckon. Object classification in 3d baggage security computed tomography imagery using visual codebooks. *Pattern Recognition*, 48(8): 2489—2499, 2015. doi: 10.1016/j.patcog.2015.02.006.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. doi: 10.1109/CVPR.2012.6248074.

D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. Van Den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1705–1714, 2019. doi: 10.1109/ICCV.2019.00179.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. doi: 10.1007/s10710-017-9314-z.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006. doi: 10.1109/CVPR.2006.100.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. doi: 10.48550/arXiv.1606.08415.

D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, and D. Song. A Benchmark for Anomaly Segmentation. *arXiv preprint arXiv:1911.11132v1*, 2019. doi: 10.48550/arXiv.1911.11132.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, chapter 14, pages 237–243. IEEE Press, 2001. doi: 10.1109/9780470544037.ch14.

T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas. BOP Challenge 2020 on 6D Object Localization. *European Conference on Computer Vision Workshops (ECCVW)*, 2020. doi: 10.1007/978-3-030-66096-3_39.

Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11105–11114, 2020. doi: 10.1109/CVPR42600.2020.01112.

Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. Learning Semantic Segmentation of Large-Scale Point Clouds with Random Sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. doi: 10.1109/TPAMI.2021.3083288.

Y. Huang, C. Qiu, Y. Guo, X. Wang, and K. Yuan. Surface defect saliency of magnetic tile. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 612–617, 2018. doi: 10.1109/COASE.2018.8560423.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448—456, 2015.

K. Jnawali, M. R. Arbabshirani, N. Rao, and A. A. Patel. Deep 3D convolution neural network for CT brain hemorrhage classification. In *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 2018. doi: 10.1117/12.2293725.

S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1175–1183, 2017. doi: 10.1109/CVPRW.2017.156.

W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation. In *European Conference on Computer Vision (ECCV)"*, pages 205–220. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46487-9_13.

A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems 30*, pages 5574–5584, 2017.

A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):20, 2019. doi: 10.1186/s42400-019-0038-7.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014.

E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3–4):237—-253, 2000. doi: 10.1007/s007780050006.

S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2656–2666, 2019. doi: 10.1109/CVPR.2019.00277.

H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1), 2009. doi: 10.1145/1497577.1497578.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification With Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, pages 950—-957, San Francisco, CA, USA, 1991. doi: 10.5555/2986916.2987033.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems 30*, pages 6402–6413, 2017.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu. Improving one-class svm for anomaly detection. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 5, pages 3077–3081, 2003. doi: 10.1109/ICMLC.2003.1260106.

W.-X. Li, V. Mahadevan, and N. Vasconcelos. Anomaly Detection and Localization in Crowded Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(1):18–32, 2013. doi: 10.1109/TPAMI.2013.111.

S.-L. Liew, J. M. Anglin, N. W. Banks, et al. A large, open source dataset of stroke anatomical brain images and manual lesion segmentations. *Scientific data*, 5:180011, 2018. doi: 10.1038/sdata.2018.11.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer International Publishing, 2014. doi: 10.1007/978-3-319-10602-1_48.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. doi: 10.1109/ICCV.2017.324.

W. Liu, R. Li, M. Zheng, S. Karanam, Z. Wu, B. Bhanu, R. J. Radke, and O. Camps. Towards visually explaining variational autoencoders. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8639–8648, 2020. doi: 10.1109/CVPR42600.2020.00867.

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi: 10.1109/TIT.1982.1056489.

C. Lu, J. Shi, and J. Jia. Abnormal Event Detection at 150 FPS in MATLAB. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2720–2727, 2013. doi: 10.1109/ICCV.2013.338.

G. Lu and B. Fei. Medical hyperspectral imaging: a review. *Journal of Biomedical Optics*, 19(1):1–24, 2014. doi: 10.1117/1.JBO.19.1.010901.

N. Marchal, C. Moraldo, H. Blum, R. Siegwart, C. Cadena, and A. Gawel. Learning densities in feature space for reliable segmentation of indoor scenes. *IEEE Robotics and Automation Letters*, 5(2):1032–1038, 2020. doi: 10.1109/LRA.2020.2967313.

M. Masana, I. Ruiz, J. Serrat, J. van de Weijer, and A. M. López. Metric learning for novelty and anomaly detection. In *British Machine Vision Conference (BMVC)*. BMVA Press, 2018.

J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, pages 52–59. Springer, 2011. doi: 10.1007/978-3-642-21735-7.

D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. doi: 10.1109/IROS.2015.7353481.

K. G. Mehrotra, C. K. Mohan, and H. Huang. *Anomaly Detection Principles and Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2017. doi: 10.1007/978-3-319-67526-8.

B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015. doi: 10.1109/TMI.2014.2377694.

C. Moenning and N. A. Dodgson. Fast Marching farthest point sampling. In *Eurographics 2003 - Posters*. Eurographics Association, 2003. doi: 10.2312/egp.20031024.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

P. Napoletano, F. Piccoli, and R. Schettini. Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity. *Sensors*, 18(1):209, 2018. doi: 10.3390/s18010209.

T. S. Nazare, R. F. de Mello, and M. A. Ponti. Are pre-trained cnns good feature extractors for anomaly detection in surveillance videos? *arXiv preprint arXiv:1811.08495*, 2018. doi: 10.48550/arXiv.1811.08495.

H. Park, J. Noh, and B. Ham. Learning memory-guided normality for anomaly detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14360–14369, 2020. doi: 10.1109/CVPR42600.2020.01438.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

P. Perera and V. M. Patel. Deep transfer learning for multiple class novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11536–11544, 2019. doi: 10.1109/CVPR.2019.01181.

P. Perera, R. Nallapati, and B. Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2893–2901, 2019. doi: 10.1109/CVPR.2019.00301.

M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014. doi: 10.1016/j.sigpro.2013.12.026.

B. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. doi: 10.1016/0041-5553(64)90137-5.

M. Reif, M. Goldstein, A. Stahl, and T. M. Breuel. Anomaly detection by combining decision trees and parametric densities. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, 2008. doi: 10.1109/ICPR.2008.4761796.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.

A. Roitberg, Z. Al-Halah, and R. Stiefelhagen. Informed democracy: Voting-based novelty detection for action recognition. In *British Machine Vision Conference (BMVC)*. BMVA Press, 2019.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing, 2015. doi: 10.1007/978-3-319-24574-4_28.

M. Rudolph, B. Wandt, and B. Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1906–1915, 2021. doi: 10.1109/WACV48630.2021.00195.

M. Rudolph, T. Wehrbein, B. Rosenhahn, and B. Wandt. Fully convolutional cross-scale-flows for image-based defect detection. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1829–1838, 2022. doi: 10.1109/WACV51458.2022.00189.

L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep One-Class Classification. In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 2018.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette. Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, 172:88–97, 2018. doi: 10.1016/j.cviu.2018.02.006.

M. Sakurada and T. Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, pages 4—-11. Association for Computing Machinery, 2014. doi: 10.1145/2689746.2689747.

B. Saleh, A. Farahdi, and A. Elgammal. Object-Centric Anomaly Detection by Attribute-Based Reasoning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 787–794, 2013. doi: 10.1109/CVPR.2013.107.

M. Salehi, N. Sadjadi, S. Baselizadeh, M. H. Rohban, and H. R. Rabiee. Multiresolution knowledge distillation for anomaly detection. In *IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR), pages 14897–14907, 2021. doi: 10.1109/CVPR46437.2021.01466.

S. Salti, F. Tombari, and L. Di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125: 251–264, 2014. doi: 10.1016/j.cviu.2014.04.011.

T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017. doi: 10.1007/978-3-319-59050-9_12.

T. Schlegl, P. Seeböck, S. Waldstein, G. Langs, and U. Schmidt-Erfurth. f-AnoGAN: Fast Unsupervised Anomaly Detection with Generative Adversarial Networks. *Medical Image Analysis*, 54, 2019. doi: 10.1016/j.media.2019.01.010.

B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computing*, 13(7): 1443—1471, 2001. doi: 10.1162/089976601750264965.

P. Seeböck, J. I. Orlando, T. Schlegl, S. M. Waldstein, H. Bogunović, S. Klimscha, G. Langs, and U. Schmidt-Erfurth. Exploiting epistemic uncertainty of anatomy segmentation for anomaly detection in retinal oct. *IEEE Transactions on Medical Imaging*, 39(1):87–98, 2020. doi: 10.1109/TMI.2019.2919951.

N. Shirodkar, P. Mandrekar, R. S. Mandrekar, R. Sakhalkar, K. Chaman Kumar, and S. Aswale. Credit card fraud detection techniques – a survey. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–7, 2020. doi: 10.1109/ic-ETITE47903.2020.112.

J. Simarro Viana, E. de la Rosa, T. Vande Vyvere, D. Robben, D. M. Sima, and C.-T. P. a. Investigators. Unsupervised 3D Brain Anomaly Detection. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 133–142. Springer International Publishing, 2021. doi: 10.1007/978-3-030-72084-1.

J. Snell, K. Ridgeway, R. Liao, B. D. Roads, M. C. Mozer, and R. S. Zemel. Learning to generate images with perceptual similarity metrics. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4277–4281, 2017. doi: 10.1109/ICIP.2017.8297089.

K. Song and Y. Yan. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science*, 285:858–864, 2013. doi: 10.1016/j.apsusc.2013.09.002.

D. Soukup and T. Pinetz. Reliably Decoding Autoencoders' Latent Spaces for One-Class Learning Image Inspection Scenarios. In *Austrian Association for Pattern Recognition Workshop*. Verlag der Technischen Universität Graz, 2018. doi: 10.3217/978-3-85125-603-1-19.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. doi: 10.5555/2627435.2670313.

C. Steger. Similarity Measures for Occlusion, Clutter, and Illumination Invariant Object Recognition. In *Pattern Recognition*, volume 2191 of *Lecture Notes in Computer Science*, pages 148–154. Springer-Verlag, 2001. doi: 10.1007/3-540-45404-7_20.

C. Steger. Occlusion, Clutter, and Illumination Invariant Object Recognition. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXIV, part 3A, pages 345–350, 2002.

C. Steger, M. Ulrich, and C. Wiedemann. *Machine Vision Algorithms and Applications.* Wiley-VCH, Weinheim, 2nd edition, 2018.

W. Sultani, C. Chen, and M. Shah. Real-World Anomaly Detection in Surveillance Videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6479–6488, 2018. doi: 10.1109/CVPR.2018.00678.

R. Sun, X. Zhu, C. Wu, C. Huang, J. Shi, and L. Ma. Not all areas are equal: Transfer learning for semantic segmentation via hierarchical region selection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4355–4364, 2019. doi: 10.1109/CVPR.2019.00449.

R. Szeliski. *Computer vision algorithms and applications.* Springer, London; New York, 2011. doi: 10.1007/978-1-84882-935-0.

M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, volume 97, pages 6105–6114. PMLR, 2019.

A. Theissler, J. Perez-Velazquez, M. Kettelgerdes, and G. Elger. Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. *Reliability Engineering and System Safety*, 215:107864, 2021. doi: 10.1016/j.ress.2021.107864.

Y. Tian, B. Fan, and F. Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6128–6136, 2017. doi: 10.1109/CVPR.2017.649.

F. Tombari, S. Salti, and L. Di Stefano. Unique Signatures of Histograms for Local Surface Description. In *Proceedings of the 11th European Conference on Computer Vision: Part III*, pages 356—369. Springer-Verlag, 2010. doi: 10.1007/978-3-642-15558-1_26.

J. W. Tukey. *Exploratory data analysis.* Addison-Wesley Series in Behavioral Science. Addison-Wesley, 1977.

D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4105–4113, 2017. doi: 10.1109/CVPR.2017.437.

A. Vasilev, V. Golkov, M. Meissner, I. Lipp, E. Sgarlata, V. Tomassini, D. K. Jones, and D. Cremers. q-space novelty detection with variational autoencoders. In *Computational Diffusion MRI*, pages 113–124. Springer International Publishing, 2020. doi: 10.1007/978-3-030-52893-5_10.

D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *British Machine Vision Conference (BMVC)*, pages 119.1–119.11. BMVA Press, 2016. doi: 10.5244/C.30.119.

A. Veit, M. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 550—-558, 2016.

Z. Wang, E. Simoncelli, and A. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402, 2003. doi: 10.1109/ACSSC.2003.1292216.

Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

M. Wieler and T. Hahn. Weakly Supervised Learning for Industrial Optical Inspection. *29th Annual Symposium of the German Association for Pattern Recognition*, 2007.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. doi: 10.1109/CVPR.2015.7298801.

S. Xia, Z. Xiong, Y. Luo, WeiXu, and G. Zhang. Effectiveness of the euclidean distance in high dimensional spaces. *Optik*, 126(24):5614–5619, 2015. doi: https://doi.org/10.1016/j.ijleo.2015.09.093.

S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *European Conference on Computer Vision (ECCV)"*, pages 574–591. Springer International Publishing, 2020. doi: 10.1007/978-3-030-58580-8_34.

Z. Xie, I. Sato, and M. Sugiyama. A Diffusion Theory For Deep Learning Dynamics: Stochastic Gradient Descent Exponentially Favors Flat Minima. *International Conference on Learning Representations (ICLR)*, 2021.

S. Ying, B. Wang, L. Wang, Q. Li, Y. Zhao, J. Shang, H. Huang, G. Cheng, Z. Yang, and J. Geng. An improved knn-based efficient log anomaly detection method with automatically labeled samples. *ACM Transactions on Knowledge Discovery from Data*, 15(3), 2021. doi: 10.1145/3441448.

F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *International Conference on Learning Representations (ICLR)*, 2016.

F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2633–2642, 2020. doi: 10.1109/CVPR42600.2020.00271.

G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012. doi: 10.1109/TIP.2011.2176743.

H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar. Efficient GAN-Based Anomaly Detection. *arXiv preprint arXiv:1802.06222*, 2018. doi: 10.48550/arXiv.1802.06222.

A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 199–208, 2017. doi: 10.1109/CVPR.2017.29.

S. Zhang, H. Tong, J. Xu, and R. Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):11, 2019. doi: 10.1186/s40649-019-0069-y.

F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021. doi: 10.1109/JPROC.2020.3004555.