Evaluating State-of-the-art Methods for Industrial Anomaly Detection

Master thesis in the department of Civil and Environmental Engineering by Chen, Jinyao
Date of submission: September 23, 2024

1. Review: Prof. Dr. Arjan Kuijper
2. Review: Prof. Dr. Uwe Rüppel
3. Review: Jascha Brötzmann M.Sc.

Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Civil and Environmental Engineering Department
FB 13
The Institute of Numerical Methods and Informatics in Civil Engineering (IIB)

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

Hiermit erkläre ich, Chen, Jinyao, dass ich die vorliegende Arbeit gemäß § 22 Abs. 7 APB der TU Darmstadt selbstständig, ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe mit Ausnahme der zitierten Literatur und anderer in der Arbeit genannter Quellen keine fremden Hilfsmittel benutzt. Die von mir bei der Anfertigung dieser wissenschaftlichen Arbeit wörtlich oder inhaltlich benutzte Literatur und alle anderen Quellen habe ich im Text deutlich gekennzeichnet und gesondert aufgeführt. Dies gilt auch für Quellen oder Hilfsmittel aus dem Internet.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, September 23, 2024

Chen, Jinyao

Contents

1 Introduction **8**
- 1.1 Motivation and Problem Formulation . . . 8
- 1.2 Goal of Thesis and Contribution . . . 9
- 1.3 Structure of Thesis . . . 9

2 Related Work **11**
- 2.1 ResNet . . . 11
- 2.2 Other Benchmark Experiments . . . 12
- 2.3 Anomalib . . . 13
- 2.4 YOLO . . . 15

3 State-of-the-art Topics **17**
- 3.1 Teacher Student . . . 18
- 3.2 One Class Classification . . . 19
- 3.3 Distribution Map . . . 21
- 3.4 Memory Bank . . . 22
- 3.5 Reconstruction-based . . . 22
- 3.6 Few Shot . . . 24
- 3.7 Few Abnormal Samples . . . 24
- 3.8 Noisy AD . . . 25
- 3.9 Continual AD . . . 25
- 3.10 Multi-class Unified . . . 27
- 3.11 Benchmark . . . 27
- 3.12 Zero Shot . . . 27
- 3.13 Less Mentioned Topics . . . 28
  - 3.13.1 RGBD . . . 28
  - 3.13.2 Point Cloud . . . 28
  - 3.13.3 Dataset . . . 29
  - 3.13.4 GPT Related . . . 29
  - 3.13.5 Logical AD . . . 30
  - 3.13.6 3D AD . . . 30
  - 3.13.7 Semi Supervised . . . 31
  - 3.13.8 Self Supervised . . . 31
  - 3.13.9 Supervised AD . . . 31

4 Method **32**
- 4.1 Comparison of Datasets . . . 32
  - 4.1.1 MVTec . . . 33

4.1.2 MVTec3D . . . 35
4.1.3 Btech . . . 35
4.1.4 Kolektor . . . 36
4.1.5 VisA . . . 37
4.2 Metrics . . . 37
4.2.1 Pixel-level and Image-level . . . 38
4.2.2 Area under ROC Curve . . . 39
4.2.3 Per Region Overlap . . . 39
4.2.4 Aarea under the PR curve . . . 39
4.2.5 Forgetting Measure . . . 39
4.3 Evaluated Models . . . 39
4.3.1 Coupled Hypersphere-based Feature Adaptation . . . 40
4.3.2 Conditional Normalizing Flows . . . 42
4.3.3 Cross-Scale-Flows . . . 42
4.3.4 Deep Feature Kernel Density Estimation . . . 43
4.3.5 Probabilistic Modeling of Deep Features . . . 44
4.3.6 Discriminatively Trained Reconstruction Embedding . . . 45
4.3.7 Dual Subspace Re-Projection . . . 46
4.3.8 Localization via 2D Normalizing Flows . . . 47
4.3.9 Anomaly Detection via Adversarial Training . . . 48
4.3.10 Patch Distribution Modeling . . . 50
4.3.11 PatchCore . . . 52
4.3.12 Reverse Distillation . . . 53
4.3.13 Region-Based Kernel Density Estimation . . . 54
4.3.14 Student-Teacher Feature Pyramid Matching . . . 56
4.3.15 U-shaped Normalizing Flow . . . 57
4.3.16 Fast Method For Segmentation . . . 58
4.4 Sample Codes . . . 60
4.4.1 Training and Testing using API . . . 60
4.4.2 Training and Testing using Bash . . . 61
4.4.3 Configuration File . . . 61
4.4.4 Inference . . . 63
4.4.5 Support for Custom Dataset . . . 63

5 Evaluation **66**
5.1 Architecture . . . 66
5.2 Environment Settings . . . 67
5.3 Training Monitoring . . . 69
5.4 Inference Results . . . 71
5.4.1 Models from Anomalib . . . 72
5.4.2 Models from YOLO . . . 74
5.5 Benchmark Results . . . 75
5.5.1 Performance on MVTec . . . 75
5.5.2 Performance on VisA . . . 77
5.5.3 Performance on MVTec3D . . . 80
5.5.4 Performance on Btech . . . 83
5.5.5 Performance on Kolektor . . . 86

5.5.6 MVTec with Multi-class Unified . . . 88
5.5.7 Performance on Custom Dataset . . . 88
5.6 Interactive Visualization . . . 90

6 Conclusion **94**
6.1 Summary . . . 94
6.2 Discussion . . . 95
6.3 Future Work . . . 95
Bibliography . . . 97

7 Appendix **110**
7.1 Explanation of Abbreviation . . . 110
7.2 Code Test . . . 111

Abstract

The importance of anomaly detection has grown considerably in recent years, with applications in a number of fields, including medicine and industry. In the context of industrial manufacturing, anomaly detection represents a pivotal computer vision task, offering the potential to identify defects in a range of products, including those exhibiting surface anomalies, textile defects, and other forms of product surface anomalies. Anomaly detection can be approached through three distinct methodological avenues: unsupervised, semi-supervised and supervised. Researchers face difficulties when assessing disparate models without a unified framework. Moreover, some extant framework environments are challenging to configure. It is therefore imperative that a unified benchmark for image anomaly detection be established in order to bridge the gap between academic research and practical applications. To address this need, we propose the Industrial Anomaly Detection Benchmark Engine (IADBE), which integrates popular datasets. The learning paradigms of Few Shot, Memory Bank, One Shot and other similar approaches are analyzed in terms of their efficiency in training and inference speed. Subsequently, a comprehensive image anomaly detection benchmark platform has been constructed, comprising 22 algorithms on five standard datasets and custom datasets with a uniform setting. Moreover, the proposed IADBE presents a challenge to existing industrial anomaly detection benchmark methods and suggests future research directions. In order to ensure reproducibility and accessibility, the source code has been uploaded to the website: https://github.com/cjy513203427/IADBE.

Kurzfassung

Die Erkennung von Anomalien hat in den letzten Jahren erheblich an Bedeutung gewonnen und findet in einer Reihe von Bereichen Anwendung, darunter in der Medizin und der Industrie. Im Kontext der industriellen Fertigung stellt die Anomalieerkennung eine zentrale Aufgabe der Computer Vision dar, da sie die Möglichkeit bietet, Defekte in einer Reihe von Produkten zu identifizieren, einschließlich solcher, die Oberflächenanomalien, Textildefekte und andere Formen von Produktoberflächenanomalien aufweisen. Die Erkennung von Anomalien kann auf drei verschiedenen methodischen Wegen angegangen werden: Unsupervised, Semi-supervised und Supervised. Forscher stehen vor Schwierigkeiten, wenn sie unterschiedliche Modelle ohne ein einheitliches Framework bewerten wollen. Darüber hinaus sind einige der vorhandenen Framework schwierig zu konfigurieren. Es ist daher zwingend erforderlich, einen einheitlichen Maßstab für die Erkennung von Bildanomalien zu schaffen, um die Kluft zwischen akademischer Forschung und praktischen Anwendungen zu überbrücken. Um diesen Bedarf zu decken, eignent sich die Industrial Anomaly Detection Benchmark Engine (IADBE), da sie populäre Datensätze integriert. Die Lernparadigmen Few Shot, Memory Bank, One Shot und andere ähnliche Ansätze werden im Hinblick auf ihre Effizienz beim Training und ihre Inferenzgeschwindigkeit analysiert. Anschließend wurde eine umfassende Benchmark-Plattform zur Erkennung von Bildanomalien erstellt, die 22 Algorithmen für 5 Standarddatensätze und benutzerdefinierte Datensätze mit einheitlichen Einstellungen umfasst. Darüber hinaus stellt die vorgeschlagene IADBE eine Herausforderung für die bestehenden Benchmark-Methoden zur Erkennung von Anomalien in der Industrie dar und schlägt zukünftige Forschungsrichtungen vor. Um die Reproduzierbarkeit und Zugänglichkeit zu gewährleisten, wurde der Quellcode auf die folgende Website hochgeladen: https://github.com/cjy513203427/IADBE.

1 Introduction

In statistical analysis, an anomaly is defined as a data point that significantly deviates from the majority of other data points in a dataset. Anomaly detection, also known as outlier detection, involves identifying these observations, events, or data points that are inconsistent with the rest of the data. In the realm of contemporary manufacturing and production, the identification of machinery, process, and product faults and defects is a critical aspect of quality control. The severity of a defect can greatly influence the price of a product, and if the defect surpasses a certain threshold, the product may be rejected altogether. With the rapid advancements in deep learning, there is a growing potential to automate and enhance the process of anomaly detection. Specifically, computer vision techniques have shown promise in image anomaly detection (IAD) for industrial applications, such as detecting surface anomalies in products or textile defects. Notwithstanding the advent of novel algorithms and models that demonstrate high levels of accuracy, contemporary research tends to prioritise the advancement of algorithms and the pursuit of superior metrics. Nevertheless, the current body of research gives only limited consideration to the needs of industry. While industrial requirements emphasise system reliability, stability and deployment costs, the evaluation criteria employed in academic research are typically more concerned with model accuracy. This ultimately leads to researchers developing models that perform well in laboratory settings but are often less effective in real industrial environments. This study aims to bridge this gap by evaluating state-of-the-art methods for industrial anomaly detection. Utilizing the open-source Anomalib[1] and YOLOv8[2, 3], we benchmark various algorithms across different levels of supervision (unsupervised versus fully supervised), learning paradigms (few-shot, continual, and noisy label), and efficiency metrics (memory usage and inference speed). The evaluation includes a wide variety of datasets, encompassing audio, video, and image formats, with enhanced support for custom datasets. Metrics such as Area Under Receiver Operating Characteristics (AUROC), Area Under Precision-Recall (AUPR), and Per-Region Overlap (PRO) are adapted to assess the anomaly localization capabilities of these methods. This comprehensive evaluation is designed to provide a unified benchmark for industrial manufacturing, thereby ensuring that our system for anomaly detection is both practically relevant and effectively integrated into real-world applications.

1.1 Motivation and Problem Formulation

Historically, the majority of anomaly detection tasks have been performed by humans, which presents a number of disadvantages. Firstly, human judgment can be subjective, which may result in inconsistencies in the identification of anomalies. Secondly, manual detection is time-consuming, particularly when dealing with large volumes of data. The act of continuous monitoring can lead to fatigue, which in turn may result in a reduction in accuracy over time. Some anomalies are subtle and complex, making them difficult for humans to detect without the use of advanced tools. The recent rapid development of deep learning can therefore be utilized to assist in anomaly detection and to efficiently automate the process. In particular, computer vision methods can be used for image anomaly detection (IAD) in industrial manufacturing applications,

for example, to detect surface anomalies in industrial products or textile defects. In recent years, the field of IAD has witnessed a surge in the development of novel algorithms and models, some of which have demonstrated remarkable accuracy rates. The current research in the field of computer vision is primarily focused on unsupervised learning, with minimal analysis of industry requirements. Such applications frequently necessitate solutions tailored to specific processes, materials, and defect types, which may not be adequately addressed in current research. The implementation of new algorithms into existing manufacturing systems can be complex, and research cannot always address these practical integration issues. Academic research prioritizes certain performance metrics (like accuracy) without considering other important factors for industry, such as speed, robustness, and ease of use. This underscores the necessity for an in-depth analysis of the current situation. Furthermore, a unified benchmark for IAD is urgently needed to bridge the gap between academic research and practical applications.

1.2 Goal of Thesis and Contribution

The objective of this study is to reproduce previous research and to select appropriate models and algorithms for specific industrial automation and decision-making (IAD) problems. It is therefore imperative that a unified benchmark for industrial manufacturing be established without delay. The datasets are benchmarked using the open-source Anomalib[1] library to evaluate the performance of the algorithms, which are tested under a variety of conditions, including different levels of supervision (unsupervised versus fully supervised), learning paradigms (few-shot, continual, and noisy label), and efficiency (memory usage and inference speed). Moreover, the library is compatible with a multitude of data formats, including audio and video in addition to images. Furthermore, it offers enhanced support for custom datasets. The Area Under Receiver Operating Characteristics (AUC), Area Under Precision-Recall (AUPR), and Per Region Overlap (PRO) will be adapted to evaluate anomaly localization capabilities. Anomaly localization capabilities refer to the ability of an algorithm or model to accurately identify and pinpoint the specific locations of anomalies within a given dataset. It is noteworthy that previous research accompanying open-source projects often requires a significant investment of time due to challenges associated with environment deployment. In some instances, projects may not be operational due to versioning issues. However, in this thesis, the solution will be deployed using Docker[4] and Docker Compose[5]. The primary objective is to provide researchers with a readily accessible industrial anomaly detection platform. The platform should be capable of reproducing and identifying previous research, exhibit no software defects, and be straightforward to deploy. A total of 22 models were tested on 5 datasets, with five metrics employed. The datasets used in the experiment conducted by the paper were not the only ones considered; Other industrial datasets and a custom dataset were also utilized. A comprehensive benchmarking analysis was conducted to evaluate the performance of the selected algorithms and to highlight any discrepancies with the official benchmark ranking from Anomaly Detection on MVTec AD[6].

1.3 Structure of Thesis

Chapter 2 presents a review of existing literature pertinent to the thesis. Chapter 3 provides an overview of the state-of-the-art topics in industrial anomaly detection and the relationship between these topics and the thesis presented herewith. Chapter 4 outlines the description and comparison of datasets, the evaluation metrics that were employed, and the models that were trained and tested. Finally, the core sample code is explained. Chapter 5 describes the environment settings, including system architecture, hardware settings, training monitoring, including changes to metrics during the training process, inference results, benchmark results for

several algorithms on several datasets, and custom dataset support. Chapter 6 presents the conclusions and potential avenues for future research.

2 Related Work

This chapter offers an overview of pertinent research on the subject matter of the thesis. The following section presents a description of the most relevant research works, with the aim of illustrating the contribution of the thesis. Section 2.1 introduces the fundamental principles of residual networks, which form the basis for all evaluated models. Section 2.2 describes the other's contribution to benchmark experiments for industrial anomaly detection. Section 2.3 introduces the open source project Anomalib, which forms foundation for our IADBE. Section 2.4 introduces the open source community YOLO[2], which is a highly capable tool for general computer vision tasks.

2.1 ResNet

The Residual Network, or ResNet for short, is a deep convolutional neural network proposed by Kaiming He's team in 2015. ResNet has since been employed extensively in the domains of detection, segmentation, and recognition. The core concept of the Residual Network (ResNet) is to address the issue of gradient vanishing and model degradation in deep network training. This is achieved by introducing the Residual Block (RB), which enhances the performance of the network. In our project, we utilise ResNet18 and ResNet50[7].

We elucidate the manner in which the number of layers is determined, with the example of a 50-layer structure.

$$1(conv1) + 3(conv2\_x) \times 3 + 3(conv3\_x) \times 4 + 3(conv4\_x) \times 6 + 3(conv5\_x) \times 3 + 1(fc) = 50 \tag{2.1}$$

The following table 2.1 presents some residual networks. FLOPs, an acronym for "Floating Point Operations Per Second," is a measure of computer performance. The term "conv2_x" refers to a specific set of convolutional layers within the network. The "conv2" designation indicates that these layers are part of the second convolutional block, and the "_x" signifies that there are multiple layers within this block. It is commonly believed that the greater the depth of a neural network, the more effective it will be[8]. However, when researchers tested this hypothesis by experimenting with deeper neural networks, they found that adding more layers to a deep network did not always result in enhanced performance. In fact, it was observed that the performance of very deep neural networks could be negatively affected by the phenomenon of vanishing gradients. Consequently, it was proposed that adding more layers to a deep neural network should either increase its performance or maintain its current level, but that it should never result in a decrease in performance. In order to achieve this, they proposed the concept of skip connections/residual connections, which allows for the avoidance of loss of information flow[8]. A skip connection is a method of bypassing a layer in a neural network. It takes the activations from an (n-1)th convolution layer and adds them to the convolution output of an (n+1)th layer. This sum is then applied with ReLU, thus skipping the nth layer. The diagram 2.1 below illustrates how a skip connection works. In this diagram, f(x) represents the application of ReLU to x, where x is the output of the convolution operation[7, 8].

layer name	output size	18-layer	50-layer
conv1	112×112	7×7, 64, stride 2	
conv2_x	56×56	3×3 max pool, stride 2	
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax	
FLOPs		1.8×10^9	3.8×10^9

Table 2.1: Architectures for Residual Network. Building blocks are shown in brackets, with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2[7].

It is not immediately apparent how this approach helps. To illustrate, if the nth layer is not learning anything, we will not lose any information. This is because at the (n+1)th layer, we are using the output of the (n-1)th layer as well when we move forward and then applying an activation function to this sum. Consequently, the network is enabled to skip one ReLU activation if the information provided is not useful or if it is zero. In this case, the network will utilise the previous information, thus maintaining consistent performance. If the layers provide significant information, having previous information will boost performance[8].

2.2 Other Benchmark Experiments

The related method to the thesis is that of benchmarking. Benchmarking is the process of evaluating the performance of anomaly detection (AD) methods based on historical behavior or established standards. Anomalib is a comprehensive library designed for the purposes of training, benchmarking, deployment and development of deep learning-based anomaly detection models. The IADBE is primarily founded upon the principles of Anomalib, and the objective is to enhance the capabilities of Anomalib[1]. Another contribution is the introduction of IM-IAD, a complete industrial manufacturing-based AD benchmark featuring 19 algorithms, 7 datasets, and 5 settings. This thesis introduces the evaluation metrics of forgetting from IM-IAD. The structure and design pattern of this thesis are inspired by the aforementioned paper[9]. Furthermore, this paper examines the significance of pretrained feature extractors in unsupervised visual anomaly detection

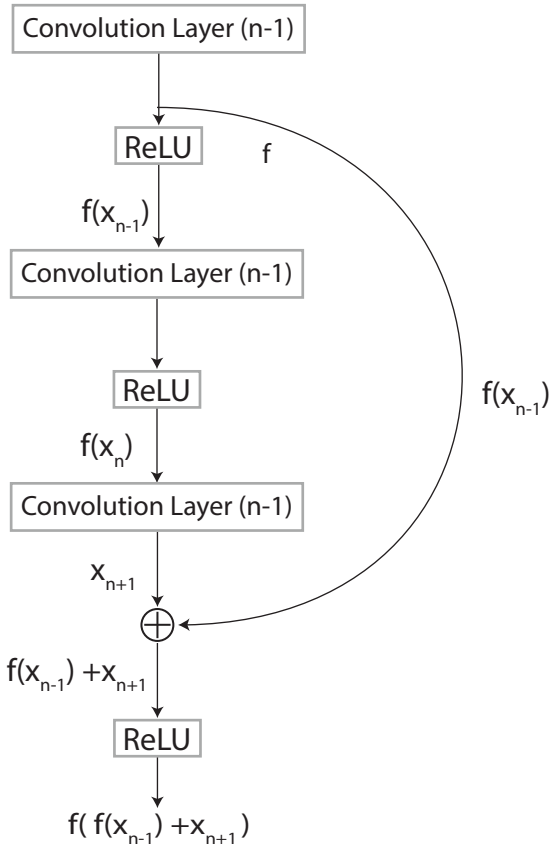

Figure 2.1: The diagram illustrates the functioning of a skip connection. In this context, the notation "f(x)" is employed to represent the application of the Rectified Linear Unit (ReLU) function to the input variable x, which represents the output obtained after the convolution operation has been performed[8].

systems. It highlights that while numerous novel AD methods employ pretrained feature spaces, there has been a paucity of investigation into the influence of the selected feature space on AD performance[10].

2.3 Anomalib

Among the various anomaly detection benchmarking engines, Anomalib is currently the most renowned and user-friendly option. The IADBE (Industrial Anomaly Detection Benchmark Engine) is developed based on an open source community project. Anomalib is a deep learning library that aims to collect state-of-the-art anomaly detection algorithms for benchmarking on both public and private datasets. Anomalib provides several ready-to-use implementations of anomaly detection algorithms described in the recent literature, as well as a set of tools that facilitate the development and implementation of custom models. The library places a strong emphasis on visual anomaly detection, where the objective of the algorithm is to identify and/or localise anomalies within images or videos in a dataset. Anomalib is consistently updated with new algorithms and training/inference extensions.[1].

Anomalib has been developed with a particular emphasis on a modular, scalable, and flexible architectural

approach, with the objective of addressing the diverse requirements of anomaly detection across a range of domains[1, 11].

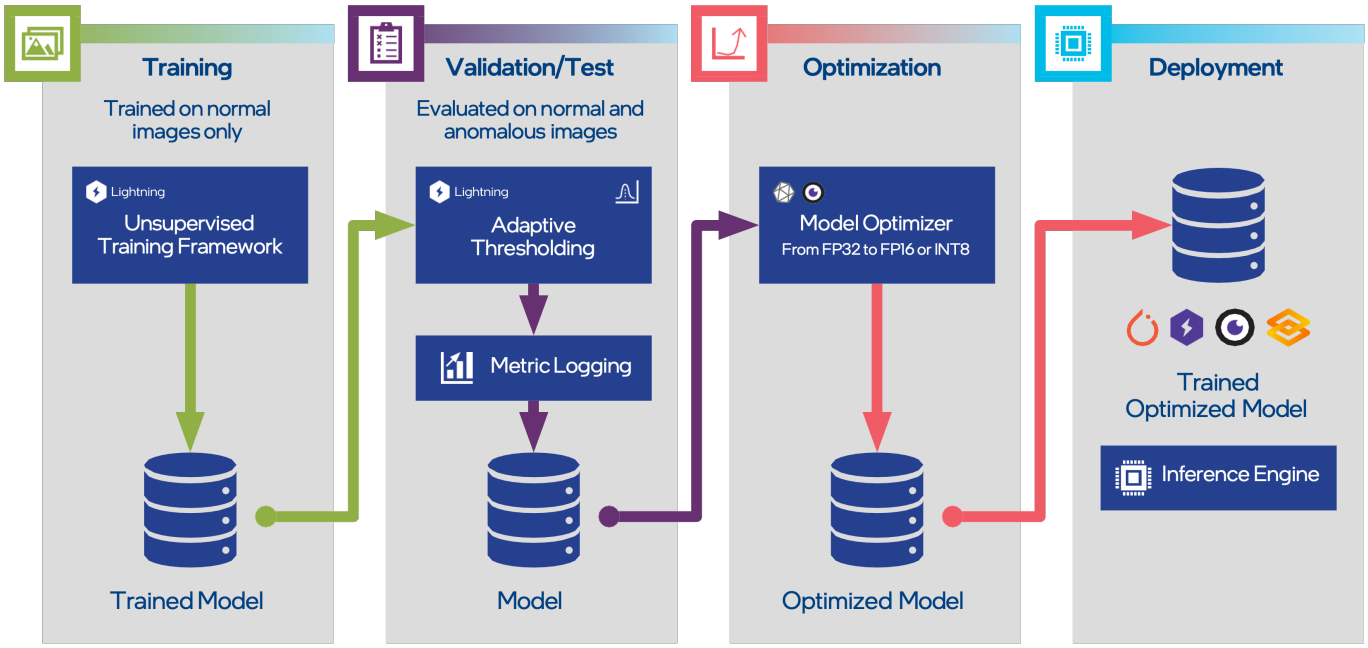

Figure 2.2: High Level Workflow - The diagram depicts the workflow of the anomalib, delineating the constituent components of training, validation/testing, optimization, and deployment[11].

As illustrated in figure 2.2, the high-level workflow of Anomalib is comprised of several key components. The initial phase is the training phase, during which the anomaly detection model learns patterns of normal behavior. However, this step is optional for unsupervised algorithms or zero-shot learning approaches that do not require specific dataset training. Subsequently, the Validation/Test phase is undertaken, whereby the model's performance is validated by determining an appropriate threshold for anomaly detection and testing it against validation or test datasets. Subsequently, the Optimization phase, which is also optional, entails enhancing the model's efficiency through the utilization of tools such as ONNX[12] or OpenVINO[13] for model compression or quantization. Ultimately, the Deployment phase introduces the model into a production environment, facilitating real-time or batch anomaly detection through the employment of diverse tools, including Torch, Lightning, ONNX, OpenVINO, and Gradio[1, 11].

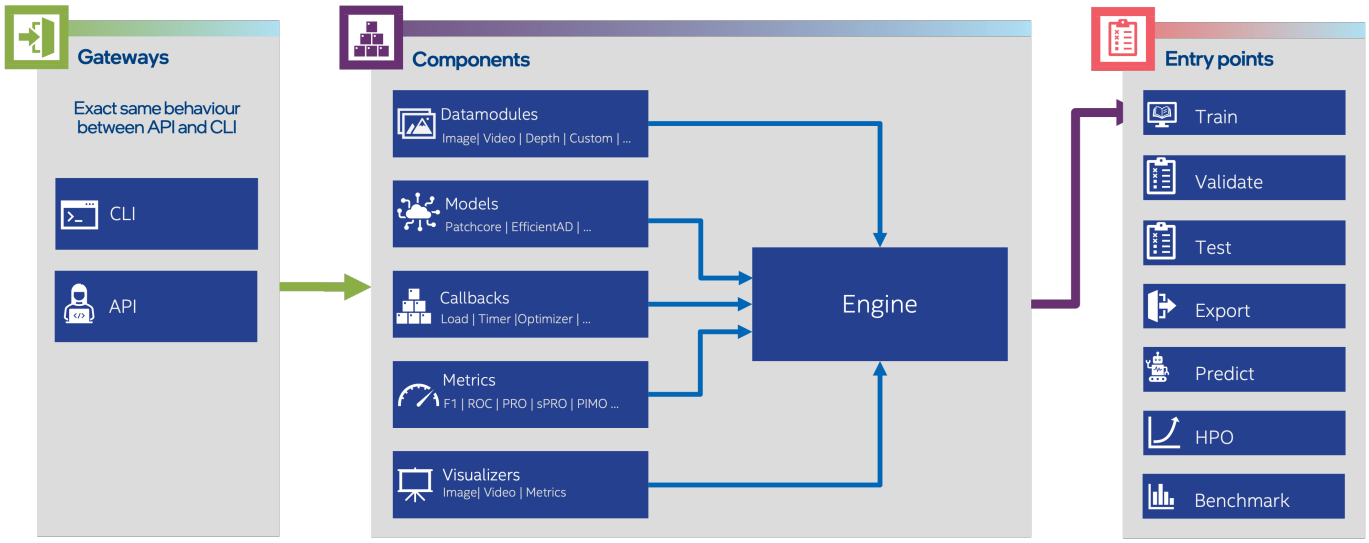

Figure 2.3: High Level Architecture - The diagram depicts the architectural framework of Anomalib, which is comprised of three primary elements namely gateways, components, and entry points[1, 11].

As illustrated in figure 2.3, the high-level architecture of Anomalib is comprised of three principal components. The three main components of the high-level architecture are the API/CLI, the components, and the entry points. The API/CLI provides users with an interface through which they can interact with the library. Anomalib offers both application programming interfaces (APIs) and a command-line interface, which can be used to handle data ingestion, configuration, and result export. It is noteworthy that both the API and CLI offer the same functionality. The Components form the core of Anomalib, comprising essential elements such as Datamodules, Models, Callbacks, Metrics, Visualizers, and the Engine. Finally, the Entry Points include the primary access points for executing various anomaly detection tasks, such as training, validating, testing, exporting, and predicting[1, 11].

2.4 YOLO

The term "YOLO," an acronym for "You Only Look Once," has been coined to describe a specific type of real-time object detection algorithm that is capable of processing images at a high speed[14]. The YOLO series has undergone notable advancements, progressing from YOLOv1 to YOLOv9. YOLOv1[15], which was introduced in 2015, treated object detection as a single regression problem, thereby enabling real-time detection. The YOLOv2 model (2016) introduced two key innovations: batch normalization and anchor boxes. These enhancements led to improved accuracy and speed[16]. YOLOv3 (2018) employed Darknet-53 and multi-scale predictions, thereby enhancing its performance[17]. YOLOv4 (2020) incorporated CSPDarknet53, Mish activation, and SPP (Spatial Pyramid Pooling) for enhanced detection[18]. YOLOv5 (2020), developed by Ultralytics and optimized for PyTorch, offers a range of model sizes to accommodate diverse requirements. YOLOv6 (2022), developed by Meituan, prioritizes efficiency through the incorporation of Rep-Pan and EfficientRep blocks[19]. YOLOv7 (2022) introduced E-ELAN (Extended Efficient Layer Aggregation Network) for enhanced real-time performance[20]. YOLOv8 (2023) further optimized feature extraction and loss functions, achieving notable improvements in accuracy, speed, and robustness[2]. Finally, YOLOv9 (2024) introduced lightweight models like YOLOv9-S, which minimized parameters and computational load while

enhancing accuracy[21]. The YOLO algorithm is primarily based on supervised learning. YOLO supports a range of datasets, including object detection, instance segmentation, pose estimation, classification, and more[22]. Additionally, they offer superior support for video datasets compared to Anomalib. However, anomaly detection is predominantly based on unsupervised learning. Anomalib is more specialized in the field of industrial anomaly detection. The support for anomaly detection datasets is limited. Therefore, it is necessary to utilize anomaly detection datasets as custom datasets.

3 State-of-the-art Topics

This chapter provides a comprehensive overview of the latest developments in industrial anomaly detection. The table 3.1 presents a selection of state-of-the-art anomaly detection papers. Section 3.1 provides an explanation of the Teacher-Student approach, also known as “knowledge distillation”. This method has been demonstrated to be one of the most accurate in our benchmarking. The following section 3.2, entitled “One Class Classification,” offers a detailed description of this technique. In section 3.3, we introduce a method based on probability density functions (PDFs) or other statistical methods. Section 3.4 elucidates the concept of the memory bank. The “Reconstruction-based Method,” detailed in section 3.5, is the most prevalent method utilized in the industrial anomaly detection domain. Section 3.6, designated as “Few Shot,” provides an explanation of the few-shot method. Section 3.7 describes the Few Abnormal Samples method, which is similar to the idea of few-shot. “Noisy AD,” covered in section 3.8, outlines a method for assigning labels to noisy data points to enhance performance, even when the training data contains label noise. Section 3.9 elucidates the operational principles of Continual AD. The “Multi-class Unified” method, discussed in section 3.10, represents a daring endeavor to construct a singular model capable of accommodating a multitude of classes. Section 3.11, “Benchmark,” introduces the benchmark method for industrial anomaly detection, a key research area of this thesis. Section 3.12 introduces the zero-shot method. Additionally, section 3.13 covers other less-used methods, including RGBD, point cloud, dataset, GPT-related, logical AD, 3D AD, semi-supervised, self-supervised, and supervised AD.

Title	Venue	Year	Topic
Anomaly Detection via Reverse Distillation from One-Class Embedding[23]	CVPR	2022	Teacher-Student
Revisiting Reverse Distillation for Anomaly Detection[24]	CVPR	2023	Teacher-Student
SimpleNet: A Simple Network for Image Anomaly Detection and Localization[25]	CVPR	2023	One-Class-Classification
Real-time unsupervised anomaly detection with localization via conditional normalizing flows[26]	WACV	2022	Distribution Map
PyramidFlow: High-Resolution Defect Contrastive Localization using Pyramid Normalizing Flow[27]	CVPR	2023	Distribution Map
Towards total recall in industrial anomaly detection[28]	CVPR	2022	Memory-bank
PNI: Industrial Anomaly Detection using Position and Neighborhood Information[29]	ICCV	2023	Memory-bank
Draem - a discriminatively trained reconstruction embedding for surface anomaly detection[30]	ICCV	2021	Reconstruction-based
DSR: A dual subspace re-projection network for surface anomaly detection[31]	ECCV	2022	Reconstruction-based
Omni-frequency Channel-selection Representations for Unsupervised Anomaly Detection[32]	TIP	2023	Reconstruction-based
Registration based few-shot anomaly detection[33]	ECCV	2022	Few Shot
AnomalyGPT: Detecting Industrial Anomalies using Large Vision-Language Models[34]	AAAI	2024	Few Shot
Catching Both Gray and Black Swans: Open-set Supervised Anomaly Detection[35]	CVPR	2022	Few abnormal samples
Explicit Boundary Guided Semi-Push-Pull Contrastive Learning for Supervised Anomaly Detection[36]	CVPR	2023	Few abnormal samples
Deep one-class classification via interpolated gaussian descriptor[37]	AAAI	2022	Noisy AD
SoftPatch: Unsupervised Anomaly Detection with Noisy Data[38]	NeurIPS	2022	Noisy AD
Inter-Realization Channels: Unsupervised Anomaly Detection Beyond One-Class Classification[39]	ICCV	2023	Noisy AD
Unsupervised Continual Anomaly Detection with Contrastively-learned Prompt[40]	ICCV	2023	Continual AD
A Unified Model for Multi-class Anomaly Detection[41]	NeurIPS	2022	Multi-class unified
Hierarchical Gaussian Mixture Normalizing Flows Modeling for Multi-Class Anomaly Detection	NeurIPS	2023	Multi-class unified
Multimodal Industrial Anomaly Detection via Hybrid Fusion[42]	CVPR	2023	RGBD
Real3D-AD: A Dataset of Point Cloud Anomaly Detection[43]	NeurIPS	2023	Point Cloud
Anomalib: A Deep Learning Library for Anomaly Detection[1]	ICIP	2022	Benchmark
IM-IAD: Industrial Image Anomaly Detection Benchmark in Manufacturing[9]	TCYB	2024	Benchmark
AnoVL: Adapting Vision-Language Models for Unified Zero-shot Anomaly Localization[44]	arxiv	2023	Zero Shot
Segment Any Anomaly without Training via Hybrid Prompt Regularization[45]	arxiv	2023	Zero Shot
UniFormaly: Towards Task-Agnostic Unified Framework for Visual Anomaly Detection[46]	arxiv	2023	Multi-class unified

Table 3.1: This is part of state-of-the-art anomaly detection papers, as referenced in the cited source[47]. The papers are classified according to the venue, year, and topic.

3.1 Teacher Student

Knowledge distillation(KD) means transporting knowledge from a large model to a smaller one, as shown in the figure 3.1. The larger model is significant in that it is capable of providing the complete potential for knowledge, but this capability comes at the computational expense.

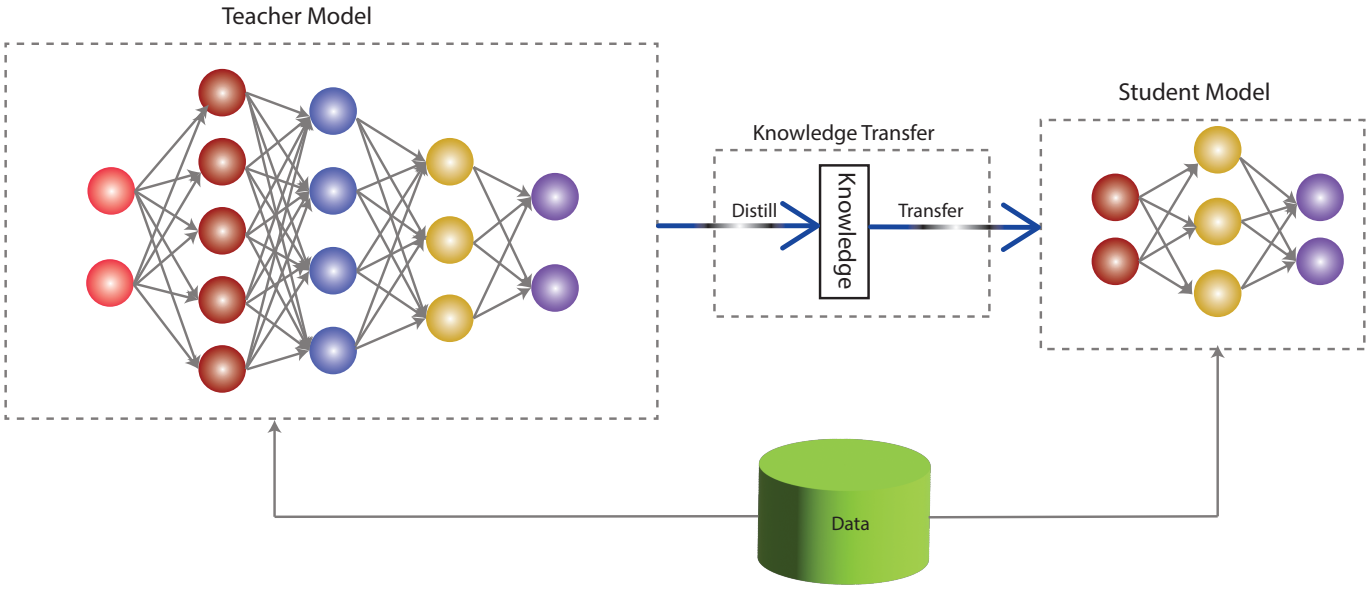

Figure 3.1: Teacher Student - Knowledge distillation means a small **student model** learning to imitate a large **teacher model** and using the teacher's knowledge to achieve similar or better accuracy[48].

Several innovative approaches to knowledge distillation for anomaly detection are discussed. Deng et al. introduced reverse distillation, a novel paradigm aimed at addressing previous shortcomings in KD-based anomaly detection methods and enhancing the responsiveness of the Teacher-Student model to anomalies[23]. Building upon this concept, Tien et al. proposed RD++, which incorporates mechanisms for pseudo anomalies, integration of multiple projection layers, and multi-task learning to improve feature compactness and abnormal alleviation. RD++ operates at a significantly enhanced speed compared to PatchCore and CFA, exhibiting a sixfold and twofold improvement, respectively. Additionally, it exhibits minimal latency when compared to RD[24, 49]. Gu et al. presented a Memory-guided Knowledge Distillation framework to address the "normality forgetting" issue of the student network. This framework employs a normality recall memory to adaptively modulate student features based on both normal and anomalous data[50]. Lastly, Zhang et al. introduced DeSTSeg, a segmentation-guided denoising student-teacher framework, which improves anomaly detection by enabling the generation of discriminative features in anomalous regions and adaptively fusing these features using a segmentation network. Experimental results demonstrate significant performance improvements over previous state-of-the-art methods in various aspects of anomaly detection[51].

3.2 One Class Classification

One-class classification (OCC) is only one class, often referred to "normal" or "inlier" class. As shown in the figure 3.2. In AD area the objective is to build a model that can recognize objects belonging to the normal class, while identifying anomalies as outliers.

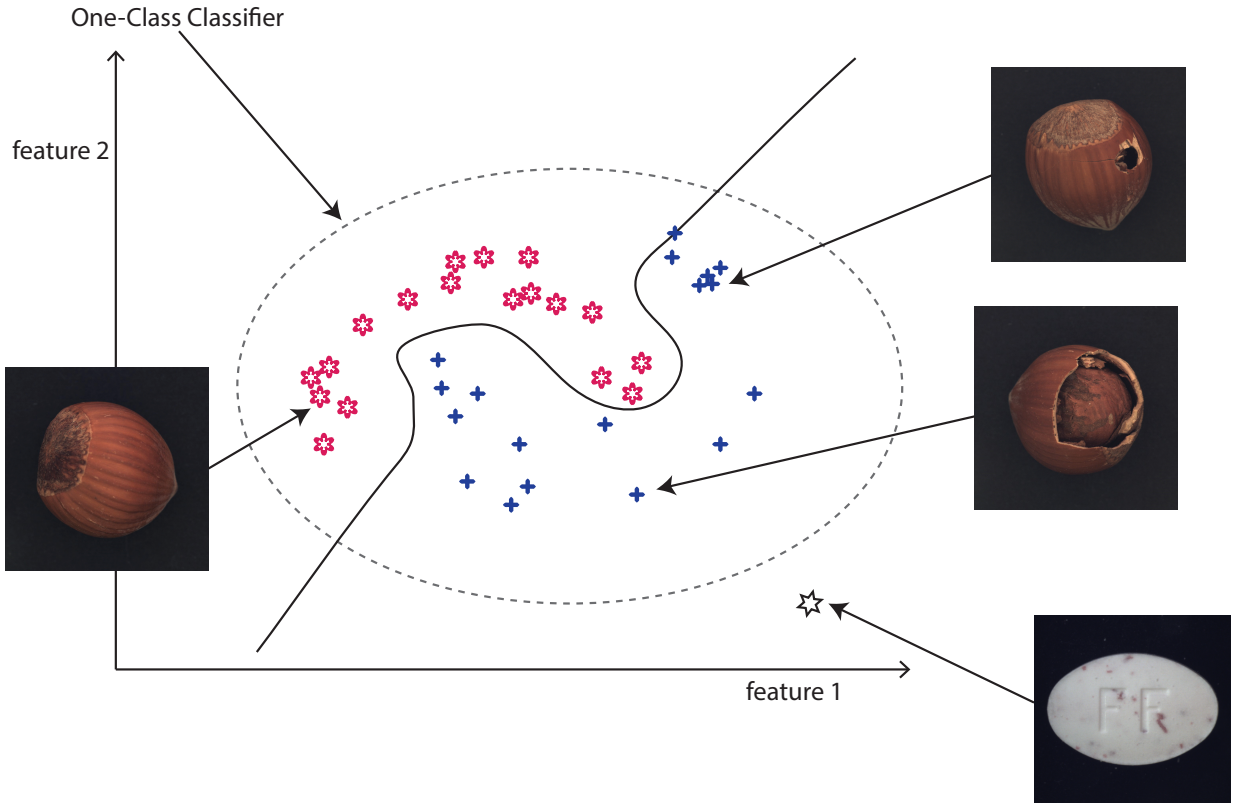

Figure 3.2: One Class Classification - Red stars indicate normal data, blue crosses indicate abnormal data, black star means unrelated data[52].

These papers examines a number of different approaches to anomaly detection in different contexts. Liu et al. propose SimpleNet, a straightforward yet effective method for unsupervised anomaly detection and localization, utilizing One-Class-Classification. SimpleNet comprises several easily trainable neural network modules suitable for industrial scenarios[25]. Cao et al. introduce GNL, a novel approach aimed at mitigating distribution shifts in anomaly detection, thereby enhancing model generalization. This is achieved by narrowing the gap between in-distribution (ID) and out-of-distribution (OOD) data during both the training and inference stages. The study reveals the limitations of combined anomaly detection and OOD generalization methods for this task[53]. In a recent publication, Lv et al. present a one-class classification-based method for accurate and efficient defect inspection without actual defective images. They incorporate an image matching module to improve the consistency of the input images, facilitating feature extraction and anomaly localisation. In addition, a local OCC-based anomaly localisation module and a defect generation module are developed to improve the accuracy of anomaly localisation[54]. Jang et al. propose a novel model for industrial anomaly detection and segmentation that leverages features from the vicinity of the target pixel. The model estimates the nominal distribution of each pixel by considering information from neighbouring pixels, weighted by their similarity to the target pixel. Another estimation of the nominal distribution based on aggregated features is introduced to utilise information from various receptive fields[55]. Grcic et al. tackle the problem of out-of-distribution (OOD) detection by shifting focus from pixels to regions. Their approach aggregates pixel-level evidence into mask-level predictions, thereby increasing the statistical power of anomaly scores. They also demonstrate the benefits of conducting OOD detection before ensembling decisions over specific masks[56].

3.3 Distribution Map

Distribution map is a visualization tool used to represent the distribution of data. It lets us know how data points are distributed in a feature space. Typically, it's based on probability density functions (PDFs) or other statistical methods. The map shows the likelihood of different values occurring in the dataset. We can identify anomalies by comparing new data points to the known distribution. Based on the distribution map we can set thresholds to dertermine which data points are regarded as anomalies.

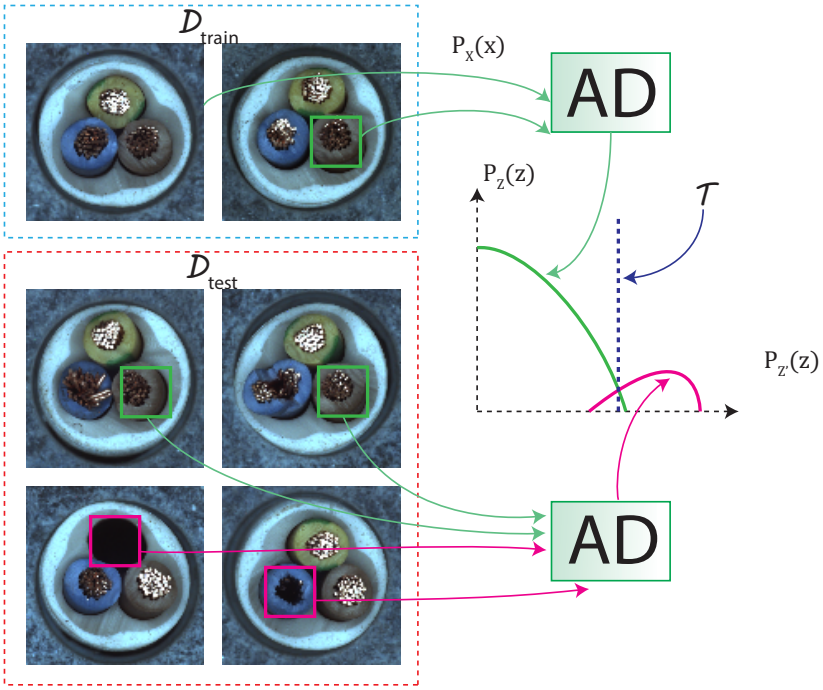

Figure 3.3: Distribution Map - An example[26] of the proposed out-of-distribution (OOD) detector for anomaly localization trained on anomaly-free D_{train} (top row).

As the figure 3.3 shows. Sliced cable images are from the MVTec dataset[6] where the bottom row illustrates D_{test} ground truth masks for anomalies (red) and the middle row shows examples of anomalyfree patches (green). The OOD detector learns the distribution of anomaly-free patches x with $P_X(x)$ density and transforms it into a Gaussian distribution with $P_Z(z)$ density. Threshold τ separates in-distribution patches from the OOD patches with $P_{Z'}(z)$ density. The paper introduces the utilization of a conditional normalizing flow framework to accurately estimate data likelihoods, a task challenging for other generative models. It presents the CFLOW-AD model, which overcomes complexity limitations of existing unsupervised AD models by employing a fully-convolutional translation-equivariant architecture[26]. Additionally, the paper introduces PyramidFlow, the first fully normalizing flow method based on the latent template-based contrastive paradigm. Utilizing pyramid-like normalizing flows and volume normalization, PyramidFlow enables high-resolution defect contrastive localization[27]. In another contribution, the paper proposes improving the density estimation of NF(normalizing flow) for anomaly detection by training NF to dynamically embed given feature distributions into different base distributions[57]. Furthermore, the authors present DS2, a two-stage dense representation learning model for the Image anomaly localization (IAL) task, demonstrating its compatibility with dense IAL tasks and strong transferability to new datasets[58]. Moreover, the paper hypothesizes that attention

modules enhance the performance of state-of-the-art anomaly detection methods in uncontrolled environments. It introduces AttentDifferNet, an unsupervised anomaly detection method based on distribution mappings through normalizing flows, benefiting from attention mechanisms by strategically coupling modular attention blocks to its feature extraction step. AttentDifferNet achieves image-level state-of-the-art performance on InsPLAD-fault, an anomaly detection in-the-wild dataset[59].

3.4 Memory Bank

Memory bank captures important features from known normal instances. By comparing incoming data with the stored memory, anomalies or deviations can be detected. During training, the memory bank learns information about what is considered "normal". At inference time, it assists in identifying abnormal data points. The paper introduces the PatchCore algorithm for cold-start anomaly detection. It leverages knowledge of only nominal examples to detect and segment anomalous data at test time. PatchCore achieves a balance between retaining maximal nominal context at test time and minimal runtime through coreset subsampling. It uses memory banks with locally aware, nominal patch-level feature representations from ImageNet pretrained networks.[28]. Additionally, the paper proposes a new algorithm, Position and Neighborhood Information (PNI), for industrial anomaly detection and localization. PNI accurately estimates the distribution of normal features by incorporating position and neighborhood information. It models position information using accumulated histograms from normal training images and utilizes a multi-layer perceptron network to model the normal feature distribution given neighborhood information. Furthermore, PNI introduces a pixelwise refinement network using synthesized anomaly images to enhance the anomaly map according to the input image[29]. Moreover, the paper introduces ReConPatch(Contrastive Patch Representation) to learn a target-oriented representation space effectively distinguishing anomalies from normal datasets. ReConPatch trains the representation by applying metric learning softly guided by similarity over nominal features[60]. Another contribution is SLAD, a deep anomaly detection method for tabular data. SLAD introduces the concept of scale in tabular data and utilizes new data-driven supervisory signals based on scales. By harnessing this supervision, the model learns inherent regularities and patterns related to the data structure, offering valuable high-level information for identifying anomalies. The paper also discusses the theoretical analysis of ensuring the effectiveness of the created data sample in revealing anomalies and examines the inlier-priority property to support the application of scale learning in anomaly detection[61]. Finally, the paper presents an incremental dimension reduction method to compute the truncated PCA, designed for visual anomaly detection using CNN-based features and optimized for GPUs[62].

3.5 Reconstruction-based

Reconstruction-based methods assume that normal data can be accurately reconstructed, while anomalies are difficult to reconstruct. During training, a model such as an autoencoder is trained using normal data. This model learns how to map input data to a low-dimensional representation (encoding). At test time, the same model attempts to reconstruct new data points. If the reconstruction error is large, it means that the data point is probably an anomaly. As is shown in the figure 3.4.

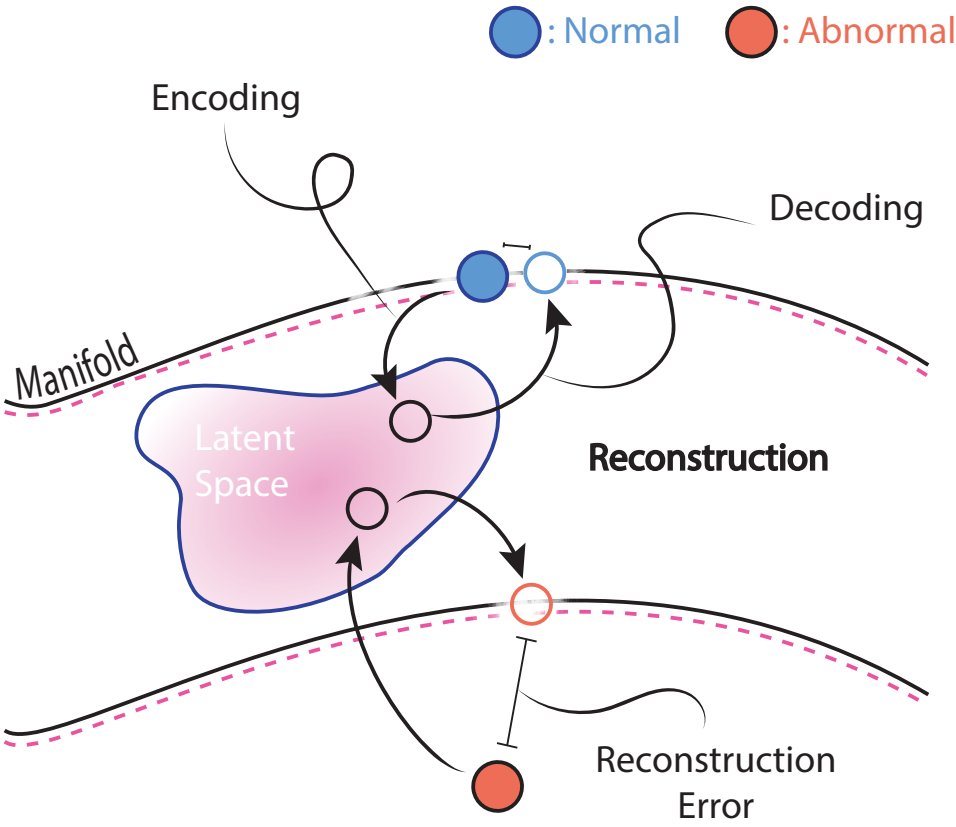

Figure 3.4: Reconstruction Based - Illustrations of reconstruction-based method for unsupervised anomaly detection. The pink gradation dotted line denote the negative log-likelihood and the manifold estimated by the reconstruction model, respectively[63].

DRAEM is presented as a discriminative end-to-end trainable surface anomaly detection and localization method, focusing on reconstruction-based techniques[30]. DSR is proposed as a discriminative surface anomaly detection method based on a dual image reconstruction branch architecture with discretized latent representation[31]. A reconstruction-based OCR-GAN method is introduced for anomaly detection in the frequency domain, utilizing a Frequency Decoupling module for parallel omni-frequency image restorations[32]. RealNet is described as a self-supervised anomaly detection framework integrating Strength-controllable Diffusion Anomaly Synthesis (SDAS), Anomaly-aware Features Selection (AFS), and Reconstruction Residuals Selection (RRS) components[64]. Focus-the-Discrepancy is proposed as a novel anomaly detection framework aiming to detect patch-wise, intra- and inter-discrepancies using I2Correlation instead of traditional self-attention maps[65]. MPDR introduces a method utilizing autoencoders for training Energy-Based Models (EBMs), leveraging the autoencoder's latent space for effective traversal of high-dimensional spaces[66]. A score-based model for anomaly detection is presented, focusing on identifying samples deviating from the normal manifold by leveraging score-based perturbation resilience[63]. SQUID is introduced for unsupervised anomaly detection in radiography images, demonstrating accurate anomaly identification through taxonomizing anatomical structures and recognizing anomalies during inference[67]. A reconstruction-based diversity-measurable anomaly detection framework is proposed, leveraging a Pyramid Deformation module and Information Compression module to enhance anomaly discrimination and reconstruction diversity[68]. DiffAD employs latent diffusion models for anomaly detection, overcoming limitations of reconstruction-based methods by handling structural deformations and avoiding over-generalization[69]. A denoising diffusion model is proposed to

improve reconstruction-based anomaly localization, combining pixel-level and feature-level reconstruction errors as the anomaly score[70]. A conditional diffusion model is introduced for the generation of assigned MURA images. This model is trained on a limited dataset with professional annotations, and it generates realistic images containing the assigned defect classes and locations [71]. SANO is presented as a novel method for unsupervised anomaly localization utilizing log-likelihood gradient magnitude from score-based diffusion models, eliminating the need for reconstruction[72]. DDPMs are explored for unsupervised Out-of-Distribution (OOD) detection, performing reconstructions of inputs with various levels of noise to address reconstruction limitations[73]. HypAD introduces a model for anomaly detection based on hyperbolic uncertainty, allowing for self-adjustment of output certainty to better identify detectable anomaly instances[74]. IRAD addresses domain adaptation in anomaly detection by learning invariant representations between source and target domains and training an anomaly detection model using domain-invariant representations[75].

3.6 Few Shot

Few Shot learning aims to train models using only a small amount of labeled training data. Traditional machine learning often requires large labeled datasets for effective generalization. However, in practical scenarios, obtaining large amount data is hard. This paper introduces an FSAD method that utilizes registration, a task inherently generalizable across categories, as the proxy task. By training a category-agnostic feature registration network with aggregated data from only a few normal samples for each category, the model demonstrates direct generalizability to new categories without requiring re-training or parameter fine-tuning. Anomalies are identified by comparing the registered features of the test image with its corresponding support (normal) images[33, 34]. Another study proposes a novel contrastive learning paradigm called ReContrast for domain-specific unsupervised anomaly detection. ReContrast addresses the transferability of the pre-trained encoder by jointly optimizing all parameters end-to-end, embedding key elements of contrastive learning into the epistemic UAD method to prevent pattern collapse, training instability, and identical shortcut[76]. Furthermore, a novel method called FastRecon is proposed for few-shot anomaly detection based on feature reconstruction. For each query sample, its normal version is constructed with reference to a limited number of normal features. A regression equation with proposed distribution regularization ensures that the reconstructed result maintains high visual similarity with the query sample while preserving the properties of the normal sample[77]. Moreover, leveraging advancements in unsupervised learning, model architectures, training strategies, and feature processing methods, this work proposes a cross-domain meta-learning framework for defect detection and classification tasks[78]. Lastly, the paper presents FewSOME, a new state-of-the-art method in the field of FSAD on normal data samples[79].

3.7 Few Abnormal Samples

Few abnormal samples refer to a situation where there are only a small number of anomalous data points available for training. This study introduces a pioneering approach to tackle the anomaly detection (AD) challenge with limited labelled data. It advocates for the learning of disentangled representations of abnormalities, distinguishing between seen anomalies, pseudo anomalies, and latent residual-based anomalies[35]. Additionally, it proposes a discriminative AD model, referred to as BGAD, which is designed to enhance discriminability and address bias issues[36]. Moreover, it promotes block-wise classifications over pixel-wise segmentation, leveraging a sliding vision transformer with position-constrained residuals for predicting block labels. Through a bagging strategy with Swin Transformers, the method achieves state-of-the-art accuracy on

three prominent AD datasets through a bagging strategy with Swin Transformers. Furthermore, the proposed MixMatch learning scheme, coupled with semi-supervised learning via the SemiREST algorithm, demonstrates remarkable AD performance with significantly reduced annotation requirements, making it more cost-effective while maintaining high accuracy[80].

3.8 Noisy AD

During training, the labels assigned to data points may be noisy. This phenomenon, known as label noise, can negatively impact the performance of anomaly detection models. However, there are ways to enhance their performance even when the training data contains label noise. In their research, the authors introduce an interpolated Gaussian descriptor (IGD) for unsupervised anomaly detection and segmentation. IGD employs a one-class Gaussian anomaly classifier trained with adversarially interpolated training samples, enabling effective normality description based on representative normal samples rather than fringe or anomalous samples[37]. Furthermore, the paper highlights the significance of addressing noisy data problems in unsupervised anomaly detection[38]). Additionally, the authors propose Inter-Realization Channels (InReaCh), a fully unsupervised method for detecting and localizing anomalies[39]. Moreover, the study investigates the accuracy of deep learning methods for detecting cracks in various objects/materials. It highlights the difficulty of accurately annotating cracks, particularly thin ones, which can result in annotation errors that impact the performance of supervised learning methods[81].

3.9 Continual AD

Continual AD aims to maintain model performance as new data arrives. It is acknowledged that real-world scenarios entail dynamic environments where anomalies may emerge gradually, as illustrated in Figure 3.5.

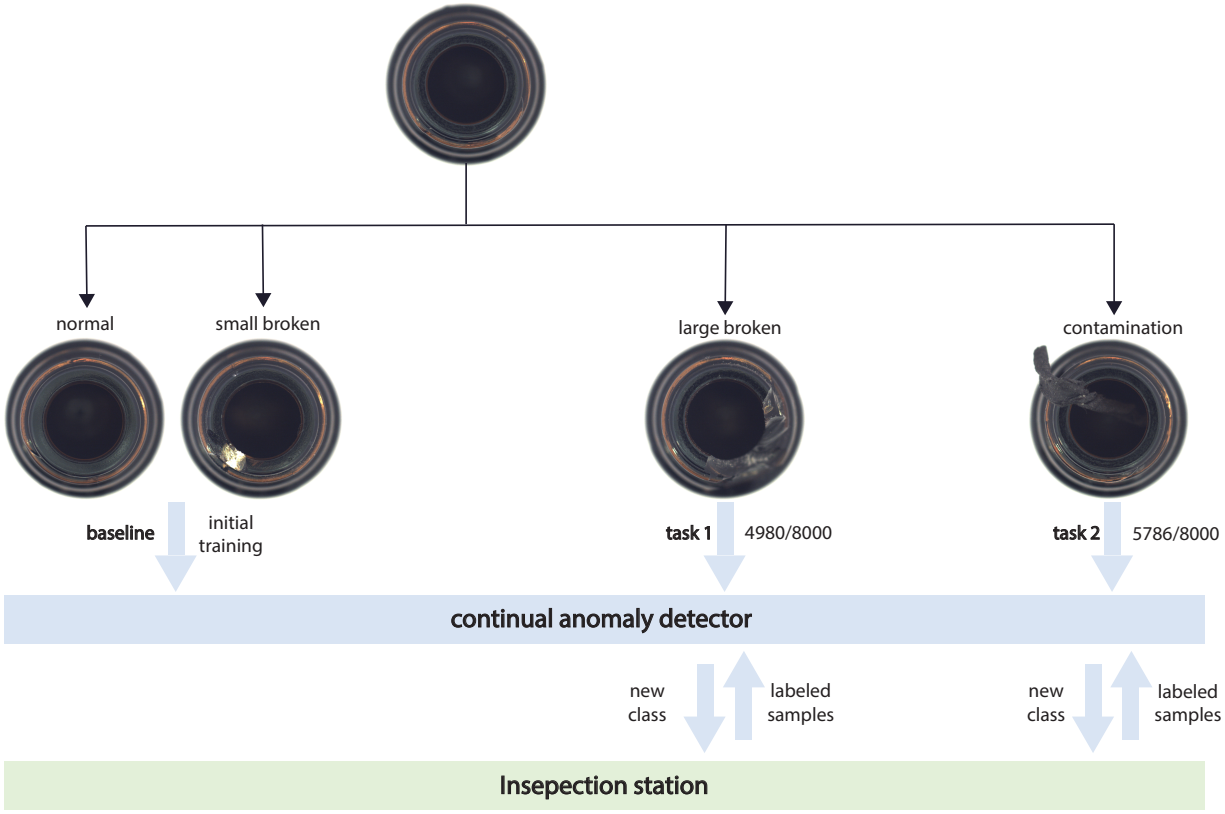

Figure 3.5: Illustration of continual anomaly detection in the MVTec dataset. The top panel shows exemplar images in each batch. The bottom panel shows the ratio of detected samples (the values before "/") among all the samples in the new classes (the values after "/")[82].

The objective of this paper is to investigate the problem of applying continual learning to unsupervised anomaly detection in order to address real-world applications in industrial manufacturing[40]. In this work, the authors propose a unified framework by incorporating continual learning to achieve a newly designed task of continual anomaly detection. In particular, they introduce a novel technique called Distribution of Normal Embeddings, which effectively alleviates catastrophic forgetting by leveraging the feature distribution of normal training samples. Furthermore, DNE can be combined with supervised CL methods to enhance their performance[83]. Futhermore, incremental unified framework effectively integrates a multi-object detection model with object-incremental learning, significantly enhancing the dynamics of defect inspection systems. By leveraging Object-Aware Self-Attention, Semantic Compression Loss, and an updating strategy, they delineate semantic boundaries for objects and minimize interference when incrementing new objects.[84]. Lastly, this paper introduces an online image anomaly detection method called LeMO, which is particularly valuable for industrial applications, such as process manufacturing scenarios. It critically analyses the shortcomings of previous deep feature embedding methods in an online setting and proposes an orthogonal memory initialization method to eliminate the need for pre-collecting a large sample set. Furthermore, they present a straightforward and efficacious intra-class contrast learning-based anomaly detection method that facilitates the online joint optimisation of prototype memory banks and biased features[85].

3.10 Multi-class Unified

Multi-class unified AD addresses the challenge of detecting anomalies across different classes or categories. Instead of training separate models for each class, it aims to set a single model that handles diverse classes simultaneously. This paper addresses the challenge of detecting anomalies across multiple classes or categories by proposing a unified approach called Multi-class Unified Anomaly Detection (UniAD)[41]. Another contribution is the introduction of the Hierarchical Vector Quantized Transformer (HVQ-Trans), a unified model for multi-class unsupervised anomaly detection under the one-for-all setting[86]. Additionally, the authors present a unified anomaly detection framework that leverages self-supervised representations, incorporating techniques such as Back Patch Masking (BPM) and top k-ratio feature matching for task unification in a task-agnostic manner[46]. Furthermore, they propose a diffusion-based framework, DiAD, to address issues in multi-class anomaly detection, namely category and semantic loss. This is achieved through the introduction of the Semantic-Guided network and Spatial-aware Feature Fusion block, which facilitate improved reconstruction of abnormal regions while maintaining semantic information[87]. Lastly, considering practical applications, a unified CNN framework is proposed for localizing anomalies across multiple classes with a single model. This approach incorporates panel-guided anomaly synthesis, dilated channel and spatial attention (DCSA) blocks, and the Diff Neck module, showing improved anomaly reconstruction and localization results[88].

3.11 Benchmark

Benchmarking is the process of evaluating the performance of anomaly detection (AD) methods based on historical behavior or established standards.This paper introduces anomalib, a comprehensive library designed for training, benchmarking, deploying, and developing deep-learning-based anomaly detection models[1]. Another contribution is the introduction of IM-IAD, a complete industrial manufacturing-based AD benchmark featuring 19 algorithms, 7 datasets, and 5 settings (Xie et al.[9]). Furthermore, this paper examines the significance of pretrained feature extractors in unsupervised visual anomaly detection systems. It highlights that while numerous novel AD methods employ pretrained feature spaces, there has been a paucity of investigation into the influence of the selected feature space on AD performance (Heckler et al.[10]).

3.12 Zero Shot

Zero-shot learning enables models to generalise to unseen classes without the need for explicit training examples. In this paper, the authors introduce a novel framework aimed at resolving unified zero-shot anomaly detection and localization[44]. Another study explores the possibility of segmenting any anomaly without further training by leveraging the full power of modern foundation models[45]. Furthermore, a zero-shot approach is proposed that does not necessitate training or fine-tuning on new datasets. This method involves a novel meta-training approach, whereby an off-the-shelf deep AD method is trained on a meta set of interrelated datasets. This approach utilises batch normalisation in every layer and samples from the meta set are either normal samples or anomalies, depending on the context [89]. The limitations of existing anomaly detection methods, which are limited to closed sets, are addressed by the study of Zero-Shot Anomaly Detection (ZSAD). This approach involves training a model on seen classes and testing its ability to detect anomalies in unseen classes without additional training. The proposed PromptAD framework incorporates

a wealth of semantic knowledge from both the abnormality and normality perspectives, utilising natural language text prompts. It refines the representations derived from a fixed CLIP encoder backbone through a dual-branch framework[90]. Another work presents a generic framework for zero-shot anomaly localization, emphasizing the importance of different components and introducing the FCA for patch statistics comparison, which enhances high-fidelity anomaly localization and scales well with large textures[91]. Additionally, a framework is proposed to define normality and anomaly through fine-grained textual definitions and normal reference images for comprehensive anomaly classification and segmentation[92]. Finally, due to the diverse range of industrial products, collecting a large number of training images or training specialized models for each category is often impractical. Thus, designing joint models for zero/few-shot settings is a promising research direction. One study makes slight modifications to the CLIP model by adding extra linear layers, enabling it to perform zero-shot segmentation while also being capable of zero-shot classification[93].

3.13 Less Mentioned Topics

This section presents an overview of less commonly utilized yet significant methodologies pertaining to the domain of industrial anomaly detection. Subsection 3.13.1 posits that RGBD considers not only RGB color but also depth information. In subsection 3.13.2, the concept of a point cloud is introduced, which is employed in a three-dimensional coordinate system. Subsection 3.13.3 discusses the dataset method, emphasizing that a well-ordered and clean dataset is beneficial for training a robust model. Methods related to GPT are covered in subsection 3.13.4. The application of GPT-related methods necessitates a considerable expenditure of computational resources, typically implemented via an interactive chat. It remains unclear whether this is applicable for the automatic detection of large datasets. Subsection 3.13.5 delves into a logical anomaly detection (AD) method, which focuses on identifying violations of logical rules within data. The three-dimensional anomaly detection (3D AD) method, utilized in synthetic three-dimensional datasets, is presented in subsection 3.13.6. A semi-supervised method aimed at enhancing anomaly detection performance by integrating limited labeled information is introduced in subsection 3.13.7. Subsection 3.13.8 explores a Self-Supervised method, which is a subset of unsupervised learning. Finally, subsection 3.13.9 covers Supervised AD, which is seldom used in anomaly detection.

3.13.1 RGBD

RGBD refers to an approach that combines both color (RGB) and depth (D) information for anomaly detection, providing a holistic view of the environment by capturing both color and depth information.In this paper, the authors propose a multimodal industrial anomaly detection method utilizing point clouds and RGB images. Their method leverages multiple memory banks and introduces a hybrid feature fusion scheme to effectively process the multimodal data[42]).

3.13.2 Point Cloud

Point cloud is a discrete set of data points in a three-dimensional coordinate system. It provides a rich representation of three-dimensional environments. In this work, the authors propose a Real3D-AD dataset to investigate high-precision point cloud anomaly detection problems. This is cited in reference to Liu et al.[43].

3.13.3 Dataset

The direction of research in datasets provides the necessary examples for the training and evaluation of models. In this study, the authors present a Synthetic Industrial Parts dataset (SIP-17) designed for the classification of industrial parts in a simulated environment. SIP-17 contains 17 objects from six industrial use cases, including both isolated and assembled parts[94]. Additionally, synthetic data has proven to be a superior pre-training data source across multiple architectures, though it remains burdened by over-labeling issues[95]. Another contribution is the introduction of a pose-agnostic anomaly detection setting with the development of the MAD dataset, the first dataset designed to evaluate pose-agnostic anomaly detection algorithms. Additionally, the authors propose OmniposeAD, an unsupervised anomaly detection method that addresses the challenge of detecting anomalies in various poses of an object, thereby avoiding missed occluded anomaly regions[96]. In the realm of anomaly generation, the authors propose Anomalydiffusion, a novel model that generates anomalous image-mask pairs. This model separates anomaly information into two components: anomaly appearance and location. These are represented by anomaly embedding and spatial embedding in the latent diffusion textual space. An adaptive attention re-weighting mechanism is introduced to enhance the alignment between generated anomaly masks and their corresponding images[97]. Moreover, a comprehensive study on the role of data augmentation in few-shot industrial anomaly detection (IAD) reveals that data augmentation impacts different IAD algorithms based on their underlying principles and assumptions. While some data augmentation methods may improve the performance of specific algorithms, combining them can sometimes degrade accuracy due to unrealistic or inconsistent variations. However, the PatchCore method is an exception, as it focuses on local features of patches and benefits from mixed data augmentation[98]. In another study, a multi-view framework for analyzing PCB components is presented. This end-to-end method improves performance with multi-view images compared to single-view images, and aggregates ensemble results of multi-view data during inference. Furthermore, a semi-automated labelling process is developed to reduce the necessity for manual labelling and ensure consistent annotations for object detection tasks[99]. Additionally, a new glass wool defect dataset named GWD is proposed, which is the first known dataset of its kind. By improving the YOLOv5 algorithm, the study achieves precise localization and classification of glass wool defects[100]. Finally, a method is proposed to address issues in industrial complex defect synthesis, such as the lack of accurate pixel-level annotations, poor diversity, and interference from non-defective information[101].

3.13.4 GPT Related

GPT-related approaches leverage large vision-language models to enhance anomaly detection in industrial scenarios, offering a promising avenue for improving safety and efficiency across various domains. This study delves into the use of GPT-4V(ision), a powerful visual-linguistic model, for tackling anomaly detection tasks in a more generalized manner. The researchers explore GPT-4V's application in multi-modality, multi-domain anomaly detection. The experiments demonstrate that GPT-4V is effective in detecting and explaining both global and fine-grained semantic patterns in zero/one-shot anomaly detection, enabling accurate differentiation between normal and abnormal instances[102]. In another paper, the authors propose a novel large multimodal model called Myriad to achieve industrial anomaly detection tasks. By introducing estimated anomaly maps from pre-trained IAD methods, Myriad incorporates IAD prior knowledge without the need for constructing large-scale IAD datasets. A Vision Expert Tokenizer was designed to encode anomaly maps into expert tokens, which were then fed into large language models[103]. Additionally, another study explores the potential of VQA-oriented GPT-4V(ision) in zero-shot anomaly detection tasks, proposing adapted image and prompt

processing methods for quantitative and qualitative evaluation. The results demonstrate a certain level of effectiveness on popular AD datasets[104].

3.13.5 Logical AD

The field of logical anomaly detection (AD) is concerned with the identification of violations of logical rules within data. Such violations may include quantity discrepancies, arrangement inconsistencies, and composition irregularities. In practical industrial scenarios, the challenge arises from the need to detect anomalies amidst complex normal patterns and to precisely localise anomalous regions of varying sizes. To address this, Guo et al. proposed a novel template-guided hierarchical feature restoration network for anomaly detection. This netowrks recovers anomaly-free features from anomalous features using a bottleneck to filter anomalies and compensating compressed features with templates retrieved from a template bank[105]. Dai et al. introduced an unsupervised anomaly detection framework named GRAD, which generates and reweights dense contrastive patterns to expose a range of local anomaly patterns without requiring scenario-specific priors or external datasets[106]. Kim et al. integrated part segmentation into anomaly detection to identify logical and structural anomalies, proposing a segmentation model that utilizes a few labeled images and logical constraints across normal images, alongside constructing three distinct memory banks[107]. Batzner et al. presented EfficientAD, which sets new standards for detecting and localizing structural and logical anomalies with high computational efficiency, outperforming other methods[108]. Finally, Zhang et al. proposed a dual-student knowledge distillation framework with contextual affinity loss for structural and logical anomaly detection. The framework comprises a local student, which aims to accurately reconstruct low-level features, and a global student, which learns the global context. This enhances the capture of long-range correlations by using both the teacher and student networks during inference[109].

3.13.6 3D AD

3D anomaly detection (3D AD) focuses on identifying and localizing anomalies in three-dimensional data, which presents unique challenges such as geometric structure and domain shifts. Chu et al. have demonstrated that using neural implicit functions to model 3D local shapes significantly aids in detecting detailed irregularities in point clouds, and they provided practical techniques for fusing predictions from different modalities. This shape-guided expert learning framework could advance unsupervised 3D anomaly detection tasks[110]. Chen et al. addressed the challenge of deployment-friendly 3D AD by proposing EasyNet, a neural network that achieves competitive performance without relying on large pretrained models or memory banks[111]. Zavrtanik et al. introduced 3DSR (3D surface anomaly detection method based on dual subspace reprojection), a method capable of detecting 3D anomalies in industrial depth data, enhanced by the novel Depth-aware Discrete Autoencoder (DADA), which separately encodes 3D and RGB data for better representation learning, and a simulated depth generation process for robust representation learning of industrial 3D data[112]. Naumann et al. presented CubeRefine RCNN, which simultaneously detects and reconstructs the shape of intact and damaged parcels from single RGB images. They also introduced Parcel3D, a synthetic dataset for transportation logistics and warehousing applications, which includes intact and damaged parcel images with full 3D annotations[113]. Horwitz et al. conducted an extensive investigation into 3D representations and found that rotation-invariant representations achieved the best performance in 3D anomaly detection, leading to the proposal of BTF, a combination of 3D and color features for enhanced detection accuracy[114].

3.13.7 Semi Supervised

In semi-supervised approaches, a small pool of labelled samples is used alongside a larger set of unlabelled samples. The labelled data is employed to guide the model during training, with the aim of enhancing anomaly detection performance by integrating the limited labelled information. This reduces dependency on extensive labelled datasets. Ren et al. proposed WSDefSegNet, a novel framework for weakly- and semi-supervised polyp segmentation. They introduced a weakly annotated polyp dataset (W-Polyp), created by drawing sketches, which offers an efficient annotation method that minimises manual labour for physicians[115]. In another study, Gaus et al. presented a method for region-based anomaly detection in automated visual surveillance, utilizing both appearance and short-term motion characteristics in infrared (thermal) imagery. This dual approach enhances the detection of anomalies in visual surveillance contexts[116].

3.13.8 Self Supervised

Self-supervised learning, a subset of unsupervised learning, involves the model generating its own supervision signals from the data in order to learn useful representations or features that capture meaningful patterns within the data. In one paper, a self-supervised normalizing flow-based model was proposed, which combines the strengths of normalizing flow models and self-supervised learning. By conditionally optimizing the model to maximize the likelihood of normal features while minimizing synthetic anomaly features, the model more accurately learns the distribution of normal features[117]. Additionally, Baradaran et al. proposed an improved multi-task learning-based video anomaly detection method. This method introduces future semantic segmentation prediction as a novel proxy task for video anomaly detection and combines multiple complementary proxy tasks to better address both appearance and motion anomalies. They also introduced a novel mechanism to enhance motion modeling precision by incorporating context attention[118].

3.13.9 Supervised AD

Supervised anomaly detection (AD) employs labelled data to train models, thereby enhancing the accuracy and reliability of anomaly detection in industrial settings. However, this approach has drawbacks, including the high cost of annotation and poor generalisation to new data. Zhang et al. proposed a novel framework called Prototypical Residual Network (PRN) for anomaly detection and localization. PRN learns residual representations across multi-scale feature maps and within multi-size receptive fields at each scale, allowing for accurate detection and localization of anomalous regions of various sizes, shapes, and numbers. Additionally, they introduced several anomaly generation strategies to expand and diversify the anomalies, further improving the model's robustness and performance[119].

4 Method

The IADBE (Industrial Anomaly Detection Benchmark Engine) platform has been developed based on the open source project Anomalib[1] due to the impracticality of rewriting the underlying code for mathematical formulas or existing models. This allows for a greater allocation of time to be dedicated to hyperparameter optimization and benchmarking. However, it should be noted that anomalib is not without limitations. Some models are not implemented, the support of custom datasets is not optimal, and certain metrics are not introduced. In this study, we enhance the capabilities of the anomalib component within our IADBE (Industrial Anomaly Detection Benchmark Engine). The IADBE project provides researchers with a readily deployable and bug-free project that offers training, testing, and inference support via the application programming interface (API) and command line interface. This chapter outlines the methodology employed by our benchmark engine. Section 4.1 provides a detailed analysis of the MVTec, MVTec3D, Btech, Koelektor, and VisA datasets, offering insights into the distinctive characteristics of each. In Section 4.2, we introduce the metrics utilized, including the Area Under the Receiver Operating Characteristics (AUROC), the Area Under Precision-Recall (AU-PR), and the PRO (Per-Region Overlap) and Forgetting Measure (FM). We then proceed to provide a detailed explanation of each of these metrics. In Section 4.3, we provide a comprehensive explanation of the evaluated algorithms, including a detailed description of their model structure and hyperparameter settings. In Section 4.4, we introduce the core sample codes of our benchmark engine and provide a thorough explanation of their functionality.

4.1 Comparison of Datasets

In order to test the efficiency and accuracy of the models employed, five well-known datasets in the field of industrial image anomaly detection were selected. As the table 4.1 shows, these include MVTec[6], MVTec3D[120], Btech (BeanTech Anomaly Detection)[121], Kolektor[122], visa[123].

Datasets	Sample Number	Categories
MVTec[6]	5354	15
MVTec3D[120]	5312	10
Btech[121]	2830	3
Kolektor[122]	399	1
visa[123]	10821	12

Table 4.1: Comparison of Datasets

4.1.1 MVTec

The MVTec AD dataset is a benchmark for the evaluation of anomaly detection methods, with a particular focus on industrial inspection. It comprises over 5000 high-resolution images, divided into fifteen different object and texture categories. Each category contains a set of defect-free training images and a test set of images with various kinds of defects, as well as images without defects[6]. The exemplar images from the various categories are presented in 4.1. The configuration of the example parameter for the dataset is presented in table 4.2.

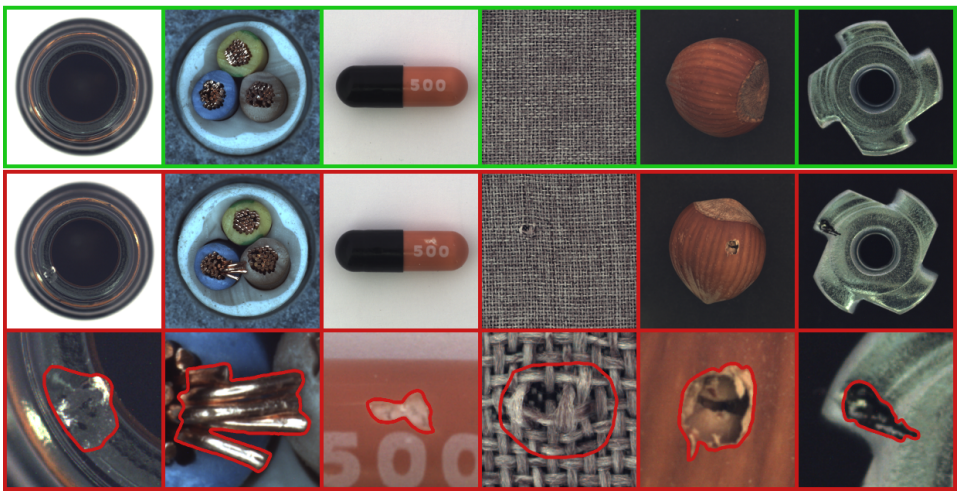

Figure 4.1: The initial two rows comprise categories such as bottle, cable, capsule, carpet, hazelnut, and metal nut, as derived from the MVTec AD dataset [6]. The final row, delineated by a red line, illustrates the segmentation outcome.

Configuration Parameter	Value	Description
root	./datasets/MVTec	The root directory of the dataset
category	bottle	Category of the MVTec datasets (e.g. "bottle" or "cable")
train_batch_size	32	Training batch size. Defaults to “32“
eval_batch_size	32	Test batch size. Defaults to “32“
num_workers	8	Number of workers
task	segmentation	Task type, ’classification’, ’detection’ or ’segmentation’. Defaults to "TaskType.SEGMENTATION"
transform	null	Transforms that should be applied to the input images
train_transform	null	Transforms that should be applied to the input images during training
eval_transform	null	Transforms that should be applied to the input images during evaluation
test_split_mode	from_dir	Setting that determines how the testing subset is obtained
test_split_ratio	0.2	Fraction of images from the train set that will be reserved for testing
val_split_mode	same_as_test	Setting that determines how the validation subset is obtained
val_split_ratio	0.5	Fraction of train or test images that will be reserved for validation
seed	null	Seed which may be set to a fixed value for reproducibility

Table 4.2: The table sets forth the configuration parameters for a dataset employed in a machine learning context, with a particular focus on the MVTec dataset.

As the table 4.2 shows, the root directory is set to "./datasets/MVTec", and the dataset category example is specified as "bottle." The training and evaluation batch sizes are both set to 32, and the number of workers for data loading is 8. The task type is designated as "segmentation". The parameters for data transformations $transform$, $train\_transform$, $eval\_transform$ are currently set to null. The method for creating the test subset is determined by the $test\_split\_mode$ parameter, which is set to $from\_dir$. This results in the reservation of 20% of the training data for testing, with a $test\_split\_ratio$ of 0.2. Similarly, the validation subset is obtained based on the same method as the test subset val_split_mode set to $same\_as\_test$, with 50% of either the training or test data allocated for validation val_split_ratio of 0.5. Additionally, a seed parameter is mentioned for potential use in ensuring reproducibility, although it is currently set to null.

4.1.2 MVTec3D

The MVTec 3D anomaly detection dataset (MVTec 3D-AD) is a comprehensive 3D dataset for the task of unsupervised anomaly detection and localisation. It contains over 4000 high-resolution scans acquired by an industrial 3D sensor. Each of the 10 different object categories comprises a set of defect-free training and validation samples and a test set of samples with various kinds of defects. For each anomalous test sample, precise ground-truth annotations are provided[120]. The exemplar images from the various categories are presented in 4.2.

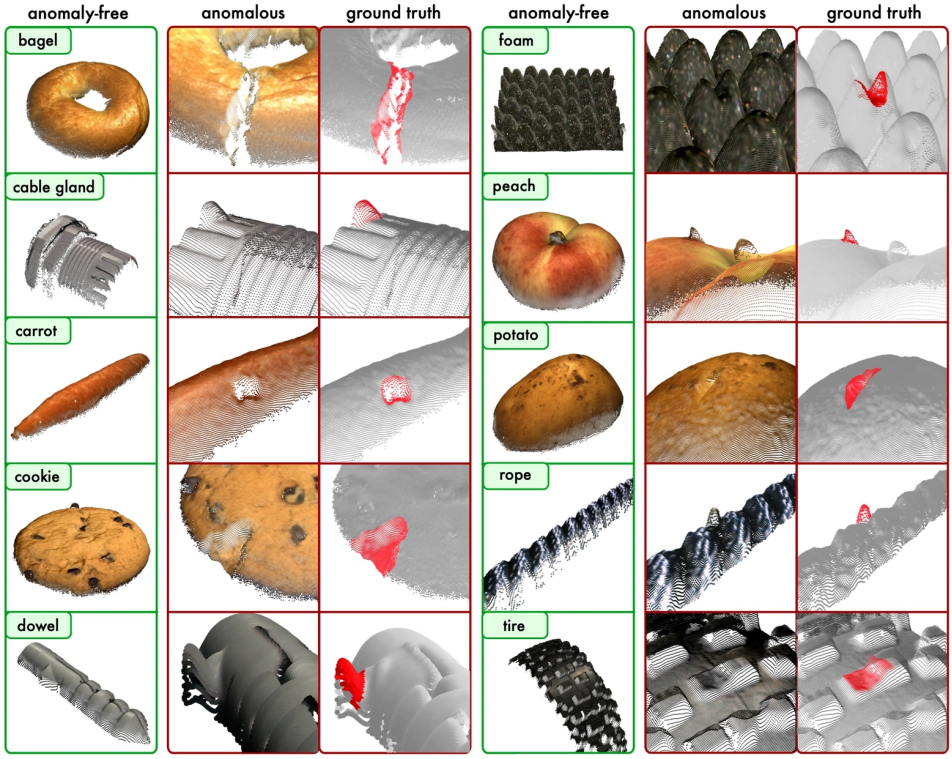

Figure 4.2: The categories presented here, derived from the MVTec3D dataset[120], include bagel, cable gland, carrot, cookie, dowel, foam, peach, potato, rope and tyre. The categories are displayed with anomalous features, anomalous features, ground truth and no anomalous features.

4.1.3 Btech

The BTAD (BeanTech Anomaly Detection) dataset represents a real-world industrial anomaly dataset. It contains a total of 2830 images of three industrial products, which have been subjected to a comprehensive examination to identify any anomalies[121]. The exemplar images from the various categories are presented in 4.3.

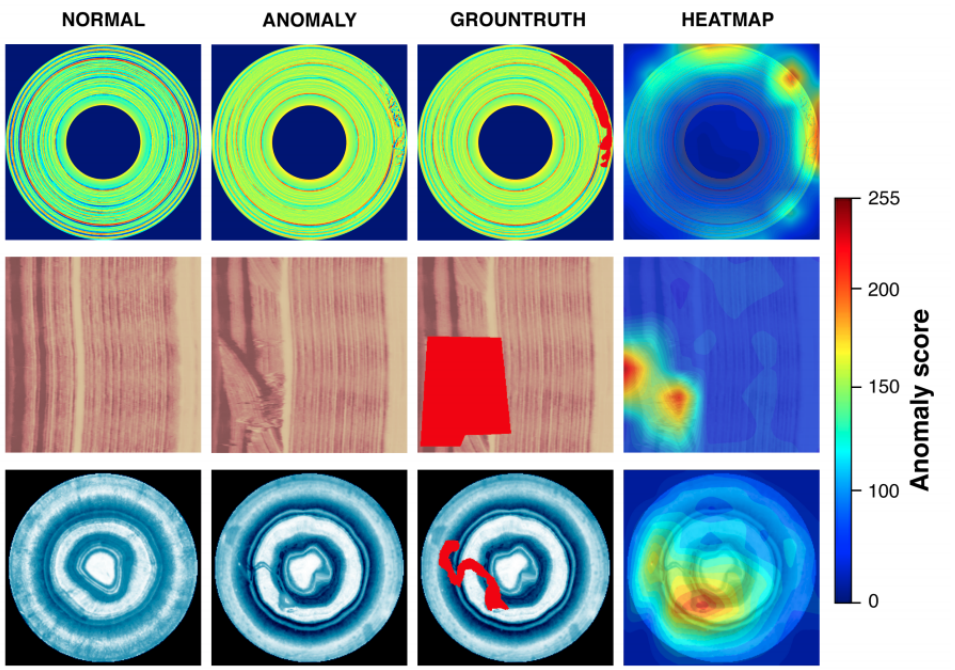

Figure 4.3: The categories in this dataset [121], designated 01, 02, and 03, are presented in a format that includes columns for normal, anomaly, ground truth, and heatmap data.

4.1.4 Kolektor

The dataset is constructed from images of defective production items that were provided and annotated by Kolektor Group[122] As illustrated in the accompanying figure 4.4, the images were captured in a controlled industrial environment in a real-world case. The dataset consists of 399 images at 500x ~1250px in size.

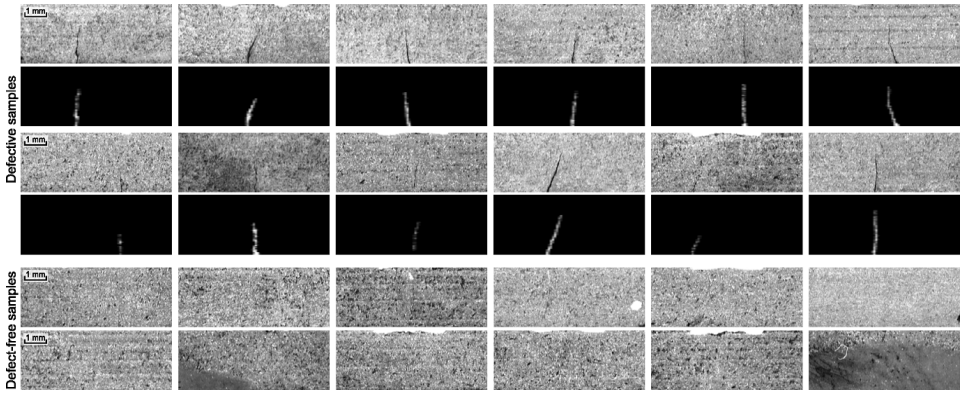

Figure 4.4: The Kolektor dataset [122] is a relatively small collection of data, comprising a single category. The accompanying image illustrates the distinction between defect samples and those that are free of defects.

4.1.5 VisA

The VisA dataset comprises 12 subsets, each corresponding to a different object. The dataset contains 10821 images, of which 9621 are normal and 1200 are anomalous. Four subsets represent different types of printed circuit boards (PCB), which are characterised by their complex structures and the presence of transistors, capacitors, chips, and other components[123]. The exemplar images from the various categories are presented in 4.5.

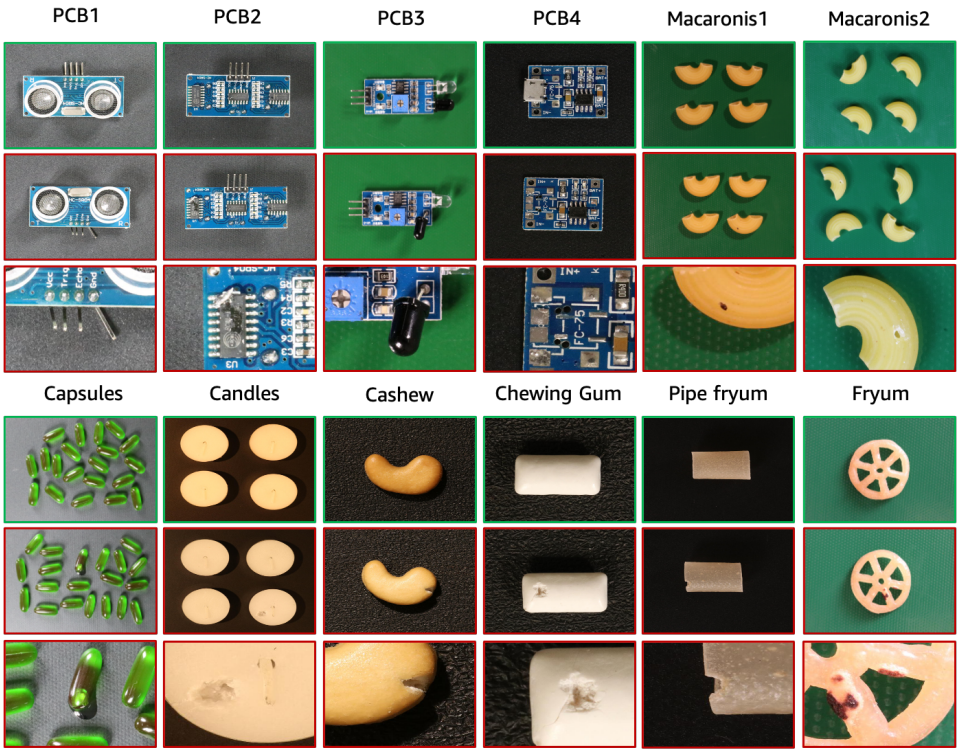

Figure 4.5: The categories in question are as diverse as the four distinct printed circuit boards (PCBs) and the two varieties of macaroni in the VisA dataset [123]. To illustrate, the upper portion of the image depicts the first two rows as typical features, while the final row represents an anomalous feature.

4.2 Metrics

In the evaluation of structural anomalies, metrics such as Area Under the Receiver Operating Characteristics (AUROC), Area Under Precision-Recall (AUPR) and Per-Region Overlap (PRO) are employed to assess the effectiveness of anomaly localisation. Furthermore, the Forgetting Measure (FM)[9] is utilised to evaluate the system's capability to withstand catastrophic forgetting. The Used metrics is shown in table 4.3.

Metrics	Formular	Usage
Precision (P)	$P = TP/(TP + FP)$	True Positive (TP), False Positive (FP)
Recall (R)	$R = TP/(TP + FN)$	False Negative (FN)
True Positive Rate (TPR)	$TPR = TP/(TP + FN)$	Classification
False Positive Rate (FPR)	$FPR = FP/(TN + FP)$	True Negative (TN)
F1 Score	$\frac{2\times P\times R}{P+R}$	Indicates a balance between precision and recall
AUROC	$\int_0^1 (TPR)d(FPR)$	Classification
PRO	$\frac{1}{N}\sum_i \sum_k \frac{P_i \cap C_{i,k}}{C_{i,k}}$	Total ground-truth number (N), Predicted abnormal pixels (P), Defect ground-truth regions (C), Segmentation
AUPR	$\int_0^1 (P)d(R)$	Localization, Segmentation
Forgetting Measure (FM) [9]	$FM_j^k = \underset{l\in\{1,\ldots k-1\}}{max} T_{l,j} - T_{k,j}$	Task (T), Number of tasks (k), Task to be evaluated (j)

Table 4.3: This table summarizes various metrics used in machine learning and their applications. Precision (P) and recall (R) are used to evaluate prediction accuracy and the ability to identify positive instances. True Positive Rate (TPR) and False Positive Rate (FPR) are essential in classification tasks. The F1 score balances precision and recall. AUROC and AUPR assess overall performance in classification and segmentation, respectively. PRO is used for segmentation accuracy, while the Forgetting Measure (FM) evaluates the performance decline across multiple tasks.

4.2.1 Pixel-level and Image-level

In anomaly detection, evaluation metrics can be applied at two different levels: image-level and pixel-level. Image-level metrics, such as AUROC and AUPR, assess whether an entire image is correctly classified as normal or anomalous, providing an overall evaluation of the model's performance in detecting anomalies in whole images. This is useful for tasks where determining the general status of an image is important. Conversely, pixel-level metrics, including pixel-wise AU-ROC, AUPR, and PRO (Per-Region Overlap), evaluate the model's accuracy in identifying the precise locations of anomalies within an image. These metrics are essential for applications requiring precise localisation and segmentation of anomalies, such as detecting the specific regions of defects in manufacturing. Therefore, image-level metrics focus on the overall anomaly detection in images, while pixel-level metrics emphasise detailed anomaly localisation within images. In general, image-level metrics yield superior results compared to pixel-level metrics.

4.2.2 Area under ROC Curve

The ROC curve is a graphical representation of a model's performance at different thresholds. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) as the threshold for classifying anomalies varies. The AUROC is the area under the ROC curve. It quantifies the overall performance of the model across all possible thresholds. The AUROC measures the performance of a model in terms of its ability to discriminate between these two classes. In summary, AUROC provides a concise summary of how well an anomaly detection model distinguishes between normal and anomalous instances.

4.2.3 Per Region Overlap

PRO (Per-Region Overlap) quantifies the overlap between predicted anomaly regions and ground truth anomaly regions, assessing how well an algorithm localizes anomalies at the region level. To calculate PRO, the ground truth is decomposed into connected components. For each component, the intersection between the predicted and ground truth anomaly regions is calculated and normalized by the ground truth area. The weighted average of these normalized values across all regions is then computed. PRO thus offers a comprehensive evaluation of anomaly localization, considering both localization accuracy and the size of detected regions.

4.2.4 Aarea under the PR curve

Precision measures the proportion of true anomalies among those predicted to be anomalies, with a focus on minimising false positives. Recall, also known as sensitivity, quantifies the proportion of true anomalies correctly identified by the model, with the aim of minimising false negatives. The Precision-Recall (PR) curve plots precision against recall as the classification threshold is varied, with each point representing a specific threshold. The area under the PR curve (AUPR) summarises overall performance, making it a valuable metric for evaluating anomaly detection models, particularly in class imbalance scenarios.

4.2.5 Forgetting Measure

The term Forgetting Measure evaluates how well a model retains knowledge over time, especially when learning from a sequence of tasks or data streams. These measures assess the extent to which a model forgets previously learned information as it adapts to new tasks or data[9]. They help researchers and practitioners understand a model's ability to maintain its performance across different tasks, highlighting its capacity to retain and integrate new knowledge without losing previously acquired information.

4.3 Evaluated Models

In this thesis we evaluate a selection of well-known and popular models in the field of industrial anomaly detection, as illustrated in table 4.4.

Topics	Models
Distribution Map	CFLOW[26], CSFlow[124], DFKDE[1], DFM[125], FastFlow[126], RKDE[127]
Reconstruction-based	DRAEM[30], DSR[31], GANomaly[128], FRE[129]
Memory Bank	PaDiM[130], PatchCore[28], CFA[131]
Teacher Student	Reverse Distillation[23], STFPM[132]
One Class Classification	UFLOW[133]

Table 4.4: This table classifies evaluated machine learning models according to their topics. Distribution Map models include CFLOW and FastFlow, while Reconstruction-based models feature DRAEM and GANomaly. Memory Bank models, such as PaDiM and PatchCore, store normal data for comparison. Teacher-Student models involve methods like Reverse Distillation, and One Class Classification includes UFLOW.

The models are evaluated on the following topics: distribution map, reconstruction-based, memory bank, teacher-student and one-class classification. As the table 4.5 shows, the most prevalent task types are segmentation, followed by detection and classification. For segmentation tasks, it includes models such as CFLOW, CSFlow, and DRAEM, among others. Detection tasks feature RKDE, while DFKDE is used for classification tasks.

Tasktype	Models
Segmentation	CFLOW[26], CSFlow[124], DFM[125], FastFlow[126], DRAEM[30], DSR[31], GANomaly[128], PaDiM[130], PatchCore[28], CFA[131], Reverse Distillation[23], STFPM[132], UFLOW[133], FRE[129]
Detection	RKDE[127]
Classification	DFKDE[1]

Table 4.5: This table provides a categorization of machine learning models based on their task types.

Classification involves the labelling of an image or video with specific concepts, thereby answering the question "What is in this image/video?" It assigns categories to the entire image. Detection identifies specific objects within an image, using bounding boxes to indicate their locations. It distinguishes between different objects. Segmentation labels each pixel in an image with specific concepts, thereby providing precise object outlines. This process extends beyond the scope of classification and object detection, whereby the image is divided into pixel groupings and each pixel is assigned a label.

4.3.1 Coupled Hypersphere-based Feature Adaptation

The original implementation is available in this project[134]. The mode type is segmentation and the detailed configuration is presented in table 4.6. Configuration parameters such as the backbone network (wide_resnet50_2), values for gamma_c and gamma_d, and the number of nearest neighbors and hard

negative features. Additionally, it includes training details like the maximum number of epochs (30), radius for the hypersphere search (1.0e-05), patience for early stopping (5), and the mode for monitoring progress ('max'), are listed.

Configuration Parameter	Value	Description
backbone	wide_resnet50_2	Backbone CNN network
gamma_c	1	gamma_c value from the paper
gamma_d	1	gamma_d value from the paper
num_nearest_neighbors	3	Number of nearest neighbors
num_hard_negative_features	3	Number of hard negative features
max_epochs	30	Maximum number of training epochs
radius	1.0e-05	Radius of the hypersphere to search the soft boundary
patience	5	Number of checks with no improvement
mode	max	One of 'min', 'max'. In 'min' mode, training will stop when the quantity monitored has stopped decreasing and in 'max' mode it will stop when the quantity monitored has stopped increasing.

Table 4.6: This table provides settings for the CFA (Coupled Hypersphere-based Feature Adaptation) model. It lists configuration parameters.

Coupled Hypersphere-based Feature Adaptation (CFA) is a method of localising anomalies by using features adapted to the target dataset.

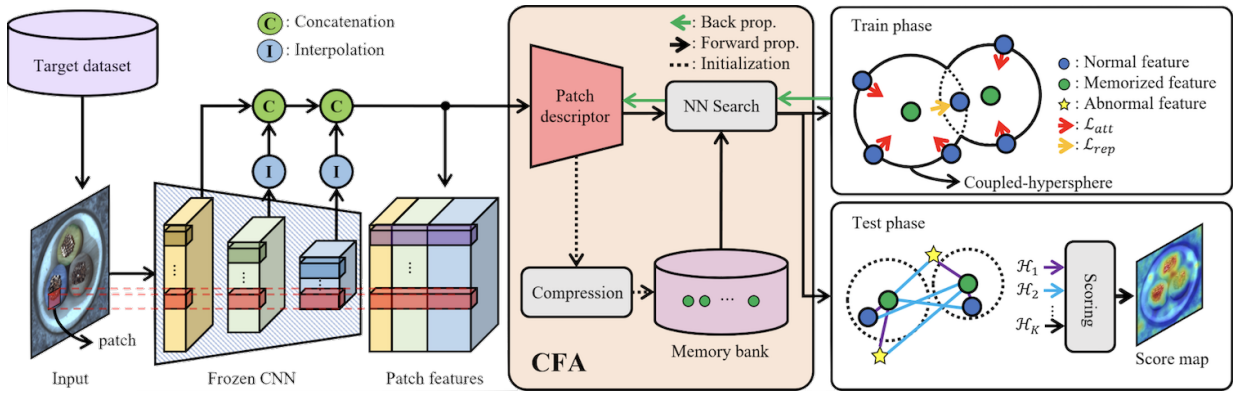

Figure 4.6: Overall structure of CFA[131]

As the figure 4.6 shows, CFA includes a learnable patch descriptor that learns and embeds target-oriented features, and a scalable memory bank that is independent of the size of the target dataset[131]. The application of a patch descriptor and memory bank to a pretrained convolutional neural network (CNN) enables the use of transfer learning, which increases the density of normal features, thus facilitating the distinction between normal and abnormal features[131]. This process is monitored by an early stopping callback with a patience of 5. It is possible to achieve higher metrics by increasing the patience.

4.3.2 Conditional Normalizing Flows

Our code is a modified version of the official repository[135]. The model type is segmentation and the detailed configuration is presented in table 4.7.

Configuration Parameter	Value	Description
backbone	wide_resnet50_2	Backbone CNN network
layers	layer2, layer3, layer4	Layers to extract features from
pre_trained	true	Whether to use pre-trained weights
fiber_batch_size	64	Fiber batch size
decoder	freia-cflow	Decoder architecture
condition_vector	128	Condition vector size
coupling_blocks	8	Number of coupling blocks
clamp_alpha	1.9	Clamping value for the alpha parameter
permute_soft	false	Whether to use soft permutation
lr	0.0001	Learning rate
max_epochs	50	Maximum number of training epochs
patience	2	Number of checks with no improvement

Table 4.7: The table provides an overview of the parameters utilized in the CFLOW model.

The CFLOW (Real-Time Unsupervised Anomaly Detection with Localization via Conditional Normalizing Flows) model is based on a conditional normalising flow framework tailored for anomaly detection with localisation. It features a discriminatively pretrained encoder followed by multi-scale generative decoders[26]. The encoder employs a multi-scale pyramid pooling approach to extract features, capturing both global and local semantic information. The receptive fields of the pooling process expand from top to bottom. The pooled features are then processed by a set of decoders, which explicitly estimate the likelihood of the encoded features. These estimated multi-scale likelihoods are subsequently upsampled to the input size and combined to produce the anomaly map.

4.3.3 Cross-Scale-Flows

The implementation of Anomalib CSFLOW is a modified version of the official repository[136]. The model type is segmentation. The detailed configuration is presented in table 4.8, it encompasses details such as the number of hidden channels in the cross convolution (1024), the number of coupling blocks (4), and the clamp value for the glow layer (3). Additionally, it specifies the number of channels employed in the model (3), the maximum training epochs (240), and the patience value for early stopping (3).

Configuration Parameter	Value	Description
cross_conv_hidden_channels	1024	Number of hidden channels in the cross convolution
n_coupling_blocks	4	Number of coupling blocks in the model
clamp	3	Clamp value for glow layer
num_channels	3	Number of channels in the model
max_epochs	240	Maximum number of training epochs
patience	3	Number of checks with no improvement

Table 4.8: The table provides an overview of the parameters utilized in the CSFLOW model.

The central idea of the CSFLOW (Fully Convolutional Cross-Scale-Flows for Image-based Defect Detection) is to handle fine-grained representations by incorporating both global and local image contexts. This is achieved by extracting features at multiple scales and using a fully convolutional normalizing flow to process these scales jointly. In each cross-scale coupling block, the input tensor is split into two parts across the channel dimension[124].

4.3.4 Deep Feature Kernel Density Estimation

The model type of DFKDE (Deep Feature Kernel Density Estimation) is classification. The detailed configuration is presented in table 4.9, it specifies that features are extracted from the layer 4 of a ResNet18 backbone, which is used as a pre-trained model. Additionally, it indicates the use of PCA with 16 components for dimensionality reduction, the SCALE method for feature scaling, and a maximum of 40,000 training points for fitting the KDE model.

Configuration Parameter	Value	Description
layers	layer4	Layers to extract features from
backbone	resnet18	Pre-trained model backbone
pre_trained	true	Boolean to check whether to use a pre_trained backbone
n_pca_components	16	Number of PCA components
feature_scaling_method	SCALE	Feature scaling method
max_training_points	40000	Number of training points to fit the KDE model

Table 4.9: The table provides the configuration settings for the DFKDE model.

This rapid anomaly classification algorithm comprises a deep feature extraction stage and an anomaly classification stage, which incorporates principal component analysis (PCA) and Gaussian kernel density estimation. Features are extracted by passing images through a ResNet50 backbone, which has been pre-trained on the ImageNet database. The output from the penultimate layer (the average pooling layer) is employed to obtain a semantic feature vector with a fixed length of 2048. In the anomaly classification stage, the aforementioned features are initially reduced to the first 16 principal components. Subsequently, Gaussian Kernel Density Estimation is applied to estimate the probability density of new examples based on the collection of training features obtained during the training phase[1].

4.3.5 Probabilistic Modeling of Deep Features

The model type of DFM (Probabilistic Modeling of Deep Features for Out-of-Distribution and Adversarial Detection) is classification and the detailed configuration is presented in table 4.10. The model employs a ResNet50 as the backbone convolutional neural network (CNN), extracting features from layer 3. The model employs a pre-trained backbone and pools features with a kernel size of 4. Additionally, it applies principal component analysis (PCA) with a retention ratio of 0.97 for dimensionality reduction and utilizes the Frechet score as the scoring type, with the Frechet and negative log-likelihood (NLL) scores as available options.

Configuration Parameter	Value	Description
backbone	resnet50	Backbone CNN network
layer	layer3	Layer to extract features from the backbone CNN
pre_trained	true	Boolean to check whether to use a pre_trained backbone
pooling_kernel_size	4	Kernel size to pool features extracted from the CNN
pca_level	0.97	Ratio from which number of components for PCA are calculated
score_type	fre	Scoring type. Options are ‘fre‘ and ‘nll‘

Table 4.10: This table provides a detailed overview of the configuration parameters utilized in the DFM model.

This fast anomaly classification algorithm comprises a deep feature extraction stage followed by an anomaly classification stage involving PCA and class-conditional Gaussian Density Estimation. The features are extracted by passing images through a ResNet18 backbone, which has been pretrained on ImageNet. The output from the penultimate layer (the average pooling layer) is used to obtain a semantic feature vector with a fixed length of 2048. In the anomaly classification stage, class-conditional PCA transformations and Gaussian Density models are learned. Two types of scores are calculated: The feature-reconstruction scores are calculated to measure the norm of the difference between the high-dimensional pre-image of a reduced dimension feature and the original high-dimensional feature. Additionally, the Negative log-likelihood under the learned density models is calculated. It should be noted that anomaly map generation is supported only

with feature-reconstruction-based scores, while image-level anomaly detection is supported by both score types[125].

4.3.6 Discriminatively Trained Reconstruction Embedding

The model type is segmentation and the detailed configuration is presented in table 4.11. The table includes two beta values (0.1 and 1.0), SSPCAB settings (disabled with a loss weight of 0.1), a maximum of 700 training epochs, an early stopping patience of 20 epochs, and monitoring of the pixel_AUROC metric. Additionally, the anomaly_source_path is set to null, with random noise utilized for anomalies.

Configuration Parameter	Value	Description
beta	0.1, 1.0	List of beta values
enable_sspcab	false	Enable SSPCAB training
sspcab_lambda	0.1	SSPCAB loss weight
anomaly_source_path	null	Path to folder that contains the anomaly source images. Random noise will be used if left empty
patience	20	Number of checks with no improvement
monitor	pixel_AUROC	Metric to monitor during training
max_epochs	700	Maximum number of training epochs

Table 4.11: The table provides a detailed overview of the DRAEM model parameters.

DRAEM (A Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection) is a reconstruction-based algorithm that includes a reconstructive subnetwork and a discriminative subnetwork. DRAEM is trained on simulated anomaly images, which are generated by augmenting normal input images from the training set with random Perlin noise masks extracted from an unrelated source of image data.

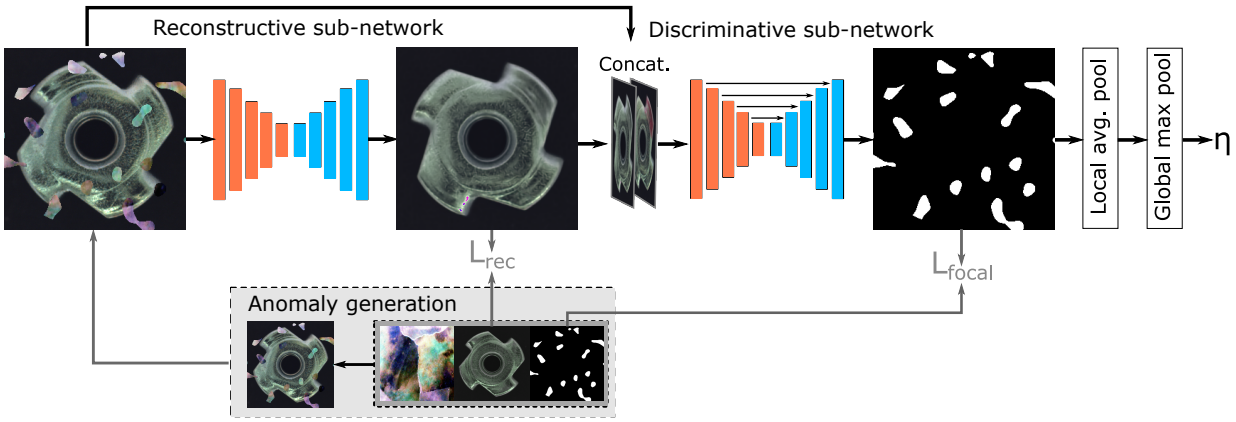

Figure 4.7: The proposed DRAEM employs an anomaly detection process. Initially, anomalous regions are identified and inpainted by the reconstructive subnetwork, which has been trained using the L_{rec} loss function. Subsequently, the output of the reconstructive sub-network and the input image are concatenated and provided as input to the discriminative sub-network. The segmentation network, trained using the Focal loss function L_{focal}[137], localizes the anomalous region and produces an anomaly map. The image-level anomaly score, denoted as η, is obtained from the anomaly score map[30].

As the figure 4.7 shows. The reconstructive subnetwork employs an autoencoder architecture that has been trained to reconstruct the original input images from the augmented images. This is achieved using a combination of L2 loss and Structural Similarity (SSIM) loss. The input to the discriminative subnetwork is a channel-wise concatenation of the augmented input image and the output of the reconstructive subnetwork. The output of the discriminative subnetwork is an anomaly map that contains predicted anomaly scores for each pixel location. The subnetwork is trained using Focal Loss in order to enhance the precision of anomaly detection[30].

4.3.7 Dual Subspace Re-Projection

The model type is segmentation. The detailed configuration is presented in table 4.12, it includes the latent_anomaly_strength, which has been set to 0.2, indicating the intensity of generated anomalies in the latent space. Additionally, the upsampling_train_ratio has been set to 0.7, specifying the ratio of training steps dedicated to the upsampling module. The maximum number of training epochs has been set to 700.

Configuration Parameter	Value	Description
latent_anomaly_strength	0.2	Strength of the generated anomalies in the latent space
upsampling_train_ratio	0.7	Ratio of training steps for the upsampling module
max_epochs	700	Maximum number of training epochs

Table 4.12: The following table enumerates the parameters for the DSR model.

DSR (A Dual Subspace Re-Projection Network for Surface Anomaly Detection) is a quantised-feature-based algorithm comprising an autoencoder with one encoder and two decoders, coupled with an anomaly detection module. DSR learns a codebook of quantised representations on ImageNet, which are then used to encode input images. Furthermore, these quantised representations are employed to sample near-in-distribution anomalies, as they do not necessitate the utilisation of external datasets. Training is conducted in three distinct phases. Initially, the encoder, the "general object decoder," and the codebook are subjected to pretraining on ImageNet. Subsequently, in the second phase, defects are generated at the feature level utilising the codebook on the quantised representations, which are subsequently employed to train the object-specific decoder and the anomaly detection module. In the final phase, the upsampling module is trained on simulated image-level smudges in order to produce more robust anomaly maps[31].

4.3.8 Localization via 2D Normalizing Flows

Fastflow from Anomalib was developed by utilising the torch model, which was implemented in this project[138]. The model type is segmentation and the detailed configuration is presented in table 4.13. This table specifies the utilization of a ResNet18 backbone with a pre-trained option enabled. The model employs eight flow steps, does not utilize exclusively 3x3 convolutions, and calculates hidden variable channels with a ratio of 1.0. The training process is set to a maximum of 500 epochs, with early stopping based on the pixel_AUROC metric and a patience of 3 epochs.

Configuration Parameter	Value	Description
backbone	resnet18	Backbone CNN network
pre_trained	true	Boolean to check whether to use a pre_trained backbone
flow_steps	8	Flow steps
conv3x3_only	false	Use only conv3x3 in fast_flow model
hidden_ratio	1.0	Ratio to calculate hidden var channels
max_epochs	500	Maximum number of training epochs
patience	3	Number of checks with no improvement
monitor	pixel_AUROC	Metric to monitor for early stopping

Table 4.13: The following table presents the configuration settings for the FastFlow model, which is utilized in the context of segmentation tasks.

FastFlow (Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows) is a two-dimensional normalising flow-based probability distribution estimator that can be used as a plug-in module with any deep feature extractor, such as ResNet or vision transformer, for unsupervised anomaly detection and localisation. During the training phase, FastFlow learns to transform the input visual features into a tractable distribution. In the inference phase, it assesses the likelihood of identifying anomalies[126].

4.3.9 Anomaly Detection via Adversarial Training

The model type is classification and the detailed configuration is presented in table 4.14. The table illustrates the batch size of 32, 64 features in CNNs, and a latent vector size of 100. It includes no extra layers, adds a final convolution layer, and sets various weights for adversarial, image regeneration, and latent vector encoder losses. The learning rate is 0.0002 with Adam optimizer parameters beta1 at 0.5 and beta2 at 0.999. Training runs for up to 100 epochs, with early stopping monitored by image_AUROC and a patience of 3 epochs.

Configuration Parameter	Value	Description
batch_size	32	Batch size used during training
n_features	64	Number of features layers in the CNNs
latent_vec_size	100	Size of autoencoder latent vector
extra_layers	0	Number of extra layers for encoder/decoder
add_final_conv_layer	true	Add convolution layer at the end
wadv	1	Weight for adversarial loss
wcon	50	Image regeneration weight
wenc	1	Latent vector encoder weight
lr	0.0002	Learning rate
beta1	0.5	Beta1 parameter for Adam optimizer
beta2	0.999	Beta2 parameter for Adam optimizer
max_epochs	100	Maximum number of training epochs
patience	3	Number of checks with no improvement
monitor	image_AUROC	Metric to monitor for early stopping

Table 4.14: This table outlines GANomaly's configuration.

GANomaly (Semi-Supervised Anomaly Detection via Adversarial Training) employs the conditional GAN approach to train a Generator, which is responsible for producing images of the normal data. This Generator is comprised of an encoder-decoder-encoder architecture, which is used to generate the normal images[128].

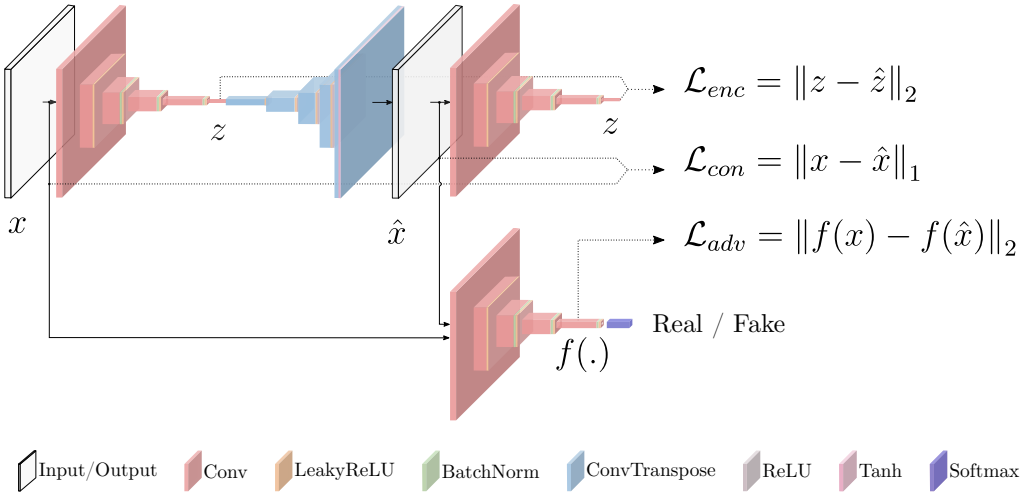

Figure 4.8: Pipeline of the proposed GANomaly for anomaly detection[128]

As the figure 4.8 shows, the distance between the latent vector z and the output vector $\hat{z}$ is minimized during training. The fundamental concept is that, during the inference process, when an anomalous image is fed through the first encoder, the latent vector z will be unable to accurately represent the data. This would result in a suboptimal reconstruction $\hat{x}$, which would then lead to a significantly different $\hat{z}$. The discrepancy between z and $\hat{z}$ serves as the anomaly score[128].

4.3.10 Patch Distribution Modeling

The model type is segmentation and the detailed configuration is presented in table 4.15.

Configuration Parameter	Value	Description
layers	layer1, layer2, layer3	Layers used for feature extraction
backbone	resnet18	Pre-trained model backbone
pre_trained	true	Boolean to check whether to use a pre_trained backbone
n_features	null	Number of features to retain in the dimension reduction step. Default values from the paper are available for: resnet18 (100), wide_resnet50_2 (550). Defaults to “None“.

Table 4.15: This table details the configuration parameters for the PaDiM model. It specifies that features are extracted from layer1, layer2, and layer3 of a ResNet18 backbone, which is pre-trained. The number of features to retain after dimension reduction is not explicitly set (null), with default values provided in the paper for ResNet18 (100) and Wide ResNet50 (550), defaulting to "None" if not specified.

PaDiM (A Patch Distribution Modeling Framework for Anomaly Detection and Localization) is a patch-based algorithm that relies on a pre-trained convolutional neural network (CNN) feature extractor.

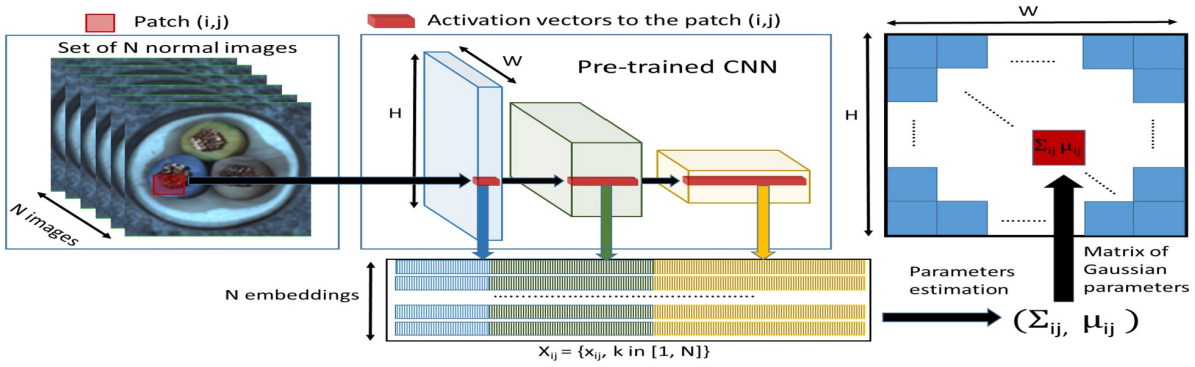

Figure 4.9: For each image patch at position (i, j) in the largest feature map of a convolutional neural network (CNN), PaDiM learns the Gaussian parameters μ_{ij} and Σ_{ij}. These parameters are derived from a set of N training embedding vectors $X_{ij} = x_{ij}^k, k \in [1, N]$, which are computed from N different training images and three different pretrained CNN layers[130].

As the figure 4.9 shows, the image is divided into patches, and embeddings are extracted from each patch using different layers of the feature extractor. Activation vectors from various layers are concatenated to obtain embedding vectors that carry information from different semantic levels and resolutions, which helps encode both fine-grained and global contexts[130]. To address the redundancy in the generated embedding vectors, dimensions are reduced through random selection. A multivariate Gaussian distribution is then generated for each patch embedding across the entire training batch. Consequently, for each patch of the training images, a

distinct multivariate Gaussian distribution is obtained, represented as a matrix of Gaussian parameters. During inference, the Mahalanobis distance is used to score each patch position of the test image. This distance calculation uses the inverse of the covariance matrix computed for the patch during training. The matrix of Mahalanobis distances constitutes the anomaly map, with higher scores indicating anomalous regions.

4.3.11 PatchCore

The model type is segmentation and the detailed configuration is presented in table 4.16. This table depicts a Wide ResNet50_2 as the backbone CNN, with features extracted from layer2 and layer3. The backbone is pre-trained. Key settings include a coreset sampling ratio of 0.1 for subsampling embeddings and 9 nearest neighbors for similarity searches.

Configuration Parameter	Value	Description
backbone	wide_resnet50_2	Backbone CNN network
layers	layer2, layer3	Layers to extract features from the backbone CNN
pre_trained	true	Boolean to check whether to use a pre_trained backbone
coreset_sampling_ratio	0.1	Coreset sampling ratio to subsample embedding
num_neighbors	9	Number of nearest neighbors

Table 4.16: This table outlines the parameters for the PatchCore model.

The PatchCore algorithm is based on the concept that an image can be classified as anomalous if even a single patch within it is anomalous. The input image is tiled into patches, which are then fed into a single pre-trained neural network to extract "mid" level features. In this context, the term "mid-level" refers to the feature extraction layer of the neural network model. This layer was chosen because lower-level features may be too general, while higher-level features may be too specific to the training dataset.

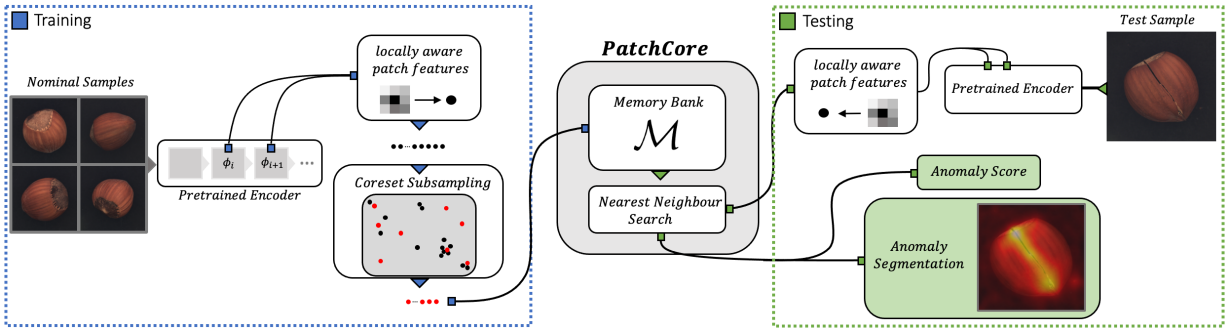

Figure 4.10: This is an overview of PatchCore. Nominal samples are broken down into a memory bank of neighborhood-aware patch-level features. In order to reduce redundancy and inference time, this memory bank is downsampled via greedy coreset subsampling. At test time, images are classified as anomalies if at least one patch is anomalous, and pixel-level anomaly segmentation is generated by scoring each patch feature[28].

During the training phase, as the figure 4.10 shows, the extracted features are stored in a memory bank of neighbourhood-aware patch-level features. During inference, this memory bank is coreset subsampled. Coreset subsampling generates a subset that best approximates the structure of the available set, facilitating approximate solution finding. This subset helps to reduce the search cost associated with nearest neighbour search[28]. The anomaly score is determined as the maximum distance between the test patch in the test patch collection and each respective nearest neighbour. Higher anomaly scores indicate greater anomaly likelihood.

4.3.12 Reverse Distillation

The model type is segmentation and the detailed configuration is presented in table 4.17.

Configuration Parameter	Value	Description
backbone	wide_resnet50_2	Backbone of CNN network
layers	layer1, layer2, layer3	Layers to extract features from the backbone CNN
anomaly_map_mode	ADD	Mode to generate anomaly map
pre_trained	true	Boolean to check whether to use a pre_trained backbone
patience	3	Number of checks with no improvement
check_val_every_n_epoch	200	Frequency of validation checking (in epochs)
max_epochs	200	Maximum number of training epochs

Table 4.17: This table details the parameters for the Reverse Distillation model. It utilizes a Wide ResNet50_2 as the CNN backbone, extracting features from layer1, layer2, and layer3. The anomaly map is generated in ADD mode. The backbone is pre-trained. Training parameters include a patience of 3 epochs, validation checks every 200 epochs, and a maximum of 200 training epochs.

The Reverse Distillation model comprises three networks. As the figure 4.11 shows, a pre-trained feature extractor (E), a one-class bottleneck embedding (OCBE), and a student decoder network (D). The backbone E is a ResNet model pre-trained on the ImageNet dataset. During the forward pass, features from three ResNet blocks are extracted. The aforementioned features are subsequently concatenated and encoded by the multi-scale feature fusion block of OCBE prior to being transmitted to the decoder D. The decoder network is analogous to the feature extractor but in reverse. During the training phase, the outputs from these symmetrical blocks are constrained to be analogous to the corresponding feature extractor layers utilising cosine distance as the loss metric. During testing, a similar step is followed but this time the cosine distance between the feature maps is used to indicate the presence of anomalies. The distance maps from all the three layers are up-sampled to the image size and added (or multiplied) to produce the final feature map. Gaussian blur is applied to the output map to make it smoother. Finally, the anomaly map is generated by applying min-max normalization on the output map[23].

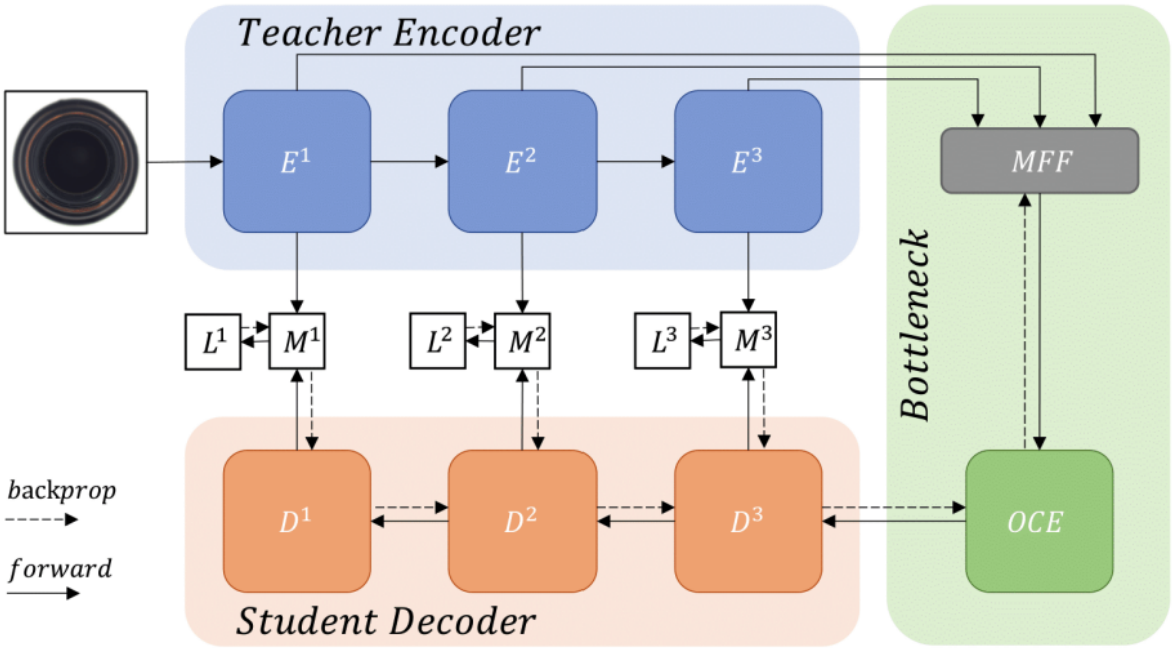

Figure 4.11: This figure provides an overview of a reverse distillation framework for anomaly detection and localization. The model comprises a pre-trained teacher encoder E, a trainable one-class bottleneck embedding module (OCBE), and a student decoder D. A multi-scale feature fusion (MFF) block is employed to integrate low- and high-level features from E and represent them in a compact code through a one-class embedding (OCE) block. During training, the student D is trained to emulate the behaviour of E by minimising the similarity loss L[23].

4.3.13 Region-Based Kernel Density Estimation

The model of RKDE (Region-Based Kernel Density Estimation) type is detection and the detailed configuration is presented in table 4.18. Key settings include the RCNN stage for extracting ROIs, a minimum confidence score of 0.001 for region proposals, a minimum box size of 25 pixels, and an IoU threshold of 0.3 for NMS. It allows a maximum of 100 detections per image. Feature settings include 16 PCA components, the SCALE method for feature scaling, and a maximum of 40,000 training points for the KDE model.

Configuration Parameter	Value	Description
roi_stage	RCNN	Processing stage from which Region of Interest (ROI) are extracted
roi_score_threshold	0.001	Mimumum confidence score for the region proposals
min_box_size	25	Minimum size in pixels for the region proposals
iou_threshold	0.3	Intersection-Over-Union threshold used during NMS
max_detections_per_image	100	Maximum number of region proposals per image
n_pca_components	16	Number of PCA components
feature_scaling_method	SCALE	Scaling method applied to features before passing to KDE
max_training_points	40000	Maximum number of training points to fit the KDE model

Table 4.18: This table outlines the parameters for the RKDE model.

This three-stage anomaly detection framework involves region extraction, feature extraction, and density estimation to classify region proposals as normal or anomalous. Both the region extractor and feature extractor use pre-trained convolutional neural networks, while the density estimation stage employs Kernel Density Estimation (KDE).

In the first stage, region proposals are generated as bounding boxes by passing images through a Faster-RCNN object detector with a ResNet50 backbone, pre-trained on the MS COCO dataset. Depending on the configuration, region proposals are derived either from the final bounding box predictions of the classification heads or from the Region Proposal Network (RPN). Detections labeled as background are discarded, followed by post-processing steps which include discarding small bounding boxes, applying Non-Maximum Suppression (NMS) across all class labels, and removing regions with low confidence scores. Parameters such as the minimum region size, IOU (Intersection-Over-Union) threshold for NMS, and confidence score threshold are configurable.

In the second stage, feature extraction is carried out using a Fast-RCNN model with an AlexNet backbone, trained in a multi-task setting on the MS COCO and Visual Genome datasets. The ROI (Region of Interest) align layer ensures that the feature maps from the convolutional layers are cropped according to the bounding box coordinates obtained during region extraction. The activations from the final shared fully connected layer are used to obtain feature embeddings for each region proposal.

In the final stage, the KDE model estimates the probability density function of the feature space. The KDE model is trained on features extracted from the training images. During inference, features from the regions in test images are evaluated against the KDE model to obtain a density estimation for each region proposal. This density estimate serves as a normality score, which is converted to a normal/anomalous label using Anomalib's thresholding mechanism. Prior to fitting the KDE model, the dimensionality of the feature vectors is reduced using principal component analysis (PCA). The features are then scaled to unit vector length or to the maximum vector length observed in the training set, depending on the configuration.[127].

4.3.14 Student-Teacher Feature Pyramid Matching

The STFPM (Student-Teacher Feature Pyramid Matching for Unsupervised Anomaly Detection) algorithm involves a pre-trained teacher network and a student network with identical architecture. The model type is segmentation and and the detailed configuration is presented in table 4.19. This table specifies using a ResNet18 backbone CNN and extracting features from layers 1, 2, and 3. Training is set to run for a maximum of 100 epochs, with early stopping applied if there is no improvement after 5 checks. The monitored metric for early stopping is the pixel AUROC.

Configuration Parameter	Value	Description
backbone	resnet18	Backbone CNN network
layers	layer1, layer2, layer3	Layers to extract features from the backbone CNN
max_epochs	100	Maximum number of training epochs
patience	5	Number of checks with no improvement
monitor	pixel_AUROC	Metric to monitor for early stopping

Table 4.19: This table outlines the configuration parameters for the STFPM model.

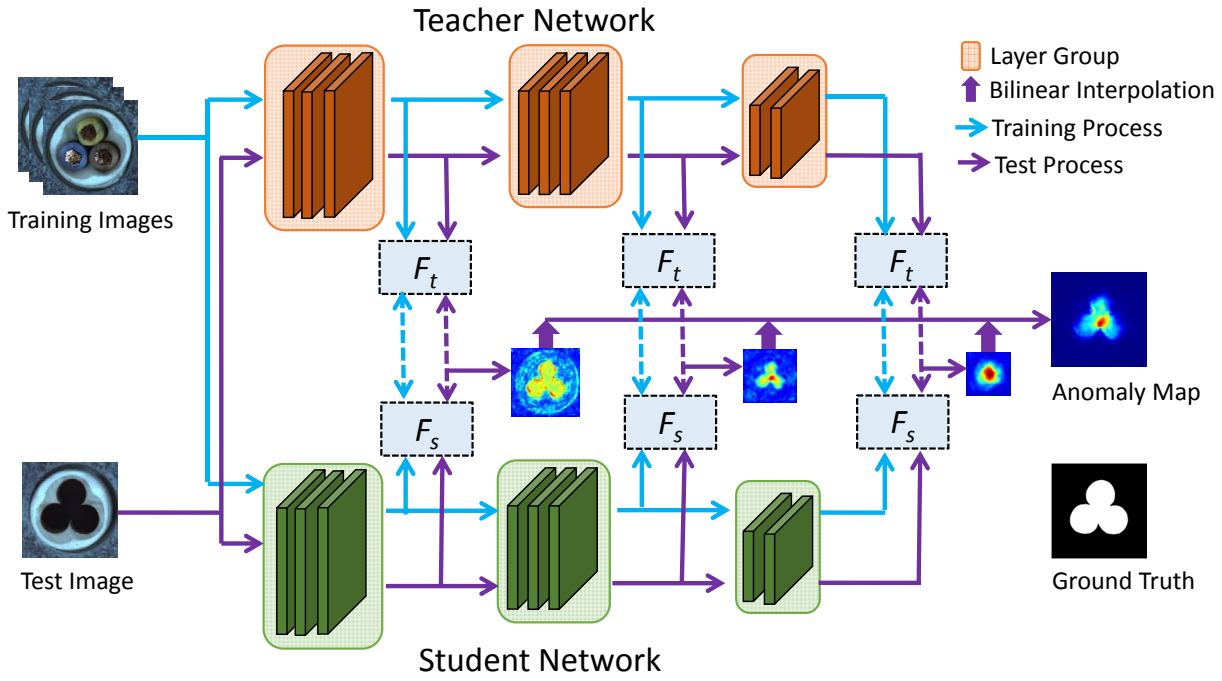

Figure 4.12: The following is a schematic overview of the STFPM model. The feature pyramid of a student network is trained to match with the counterpart of a pre-trained teacher network. A test image (or pixel) is deemed anomalous if the features extracted from the two models differ significantly. The feature pyramid matching enables our method to detect anomalies of various sizes with a single forward pass[132].

As figure 4.12 shows, the student network learns the distribution of anomaly-free images by aligning its features with the corresponding features from the teacher network. To enhance robustness, multi-scale feature matching is employed. This hierarchical feature matching approach enables the student network to acquire a blend of multi-level knowledge from the feature pyramid, thereby facilitating the detection of anomalies of various sizes. During the inference phase, the feature pyramids of the teacher and student networks are compared. A larger difference between the features indicates a higher probability of an anomaly occurring[132].

4.3.15 U-shaped Normalizing Flow

Our code based on project[139]. UFLOW (A U-shaped Normalizing Flow for Anomaly Detection with Unsupervised Threshold) combines modern machine learning techniques with classic statistical detection theory. The model type is segmentation and the detailed configuration is presented in table 4.20. This table specifies 4 flow steps, permute_soft of false, an affine clamp of 2.0, and an affine subnet channels ratio of 1.0. The model has a maximum of 200 training epochs. Logging occurs every 50 steps, and early stopping is applied after 20 checks with no improvement.

Configuration Parameter	Value	Description
flow_steps	4	Number of flow steps
permute_soft	false	Whether to use soft permutation
affine_clamp	2.0	Affine clamp
affine_subnet_channels_ratio	1.0	Affine subnet channels ratio
backbone	mcait	The backbone neural network used in the model
max_epochs	200	Maximum number of training epochs
log_every_n_steps	50	Frequency of logging steps
patience	20	Number of checks with no improvement

Table 4.20: This table details the configuration parameters for the UFLOW model.

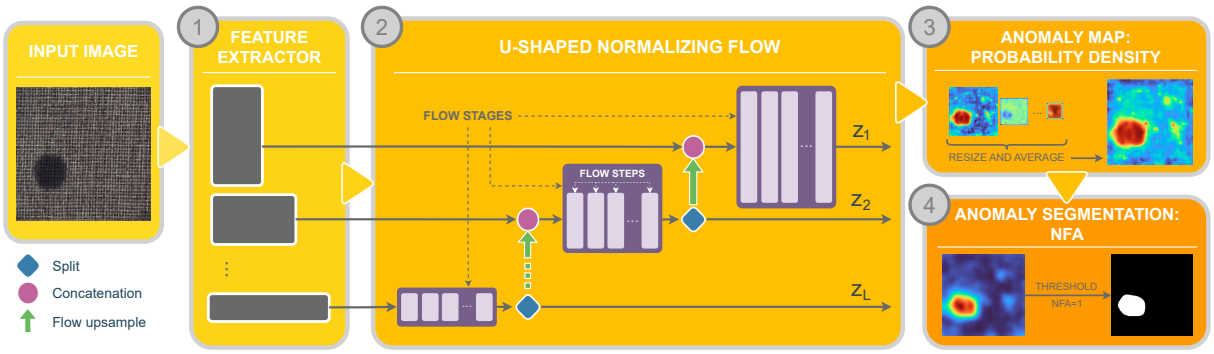

Figure 4.13: The method has three phases: (1) Multi-scale feature extraction using pre-trained image Transformers at different scales. (2) U-shaped Normalizing Flow, adapting the U-like architecture for fully invertible NFs. (3) Anomaly score and segmentation computation[133].

As the figure 4.13 shows, features are extracted using a multi-scale image Transformer architecture. These features are then processed by a U-shaped Normalizing Flow, which establishes the theoretical basis for the final phase. In the final phase, a pixel-level anomaly map is computed, and segmentation is performed using the a contrario framework. This multiple-hypothesis testing strategy allows for the derivation of robust automatic detection thresholds, which are essential for real-world applications where an operational point is necessary[133].

4.3.16 Fast Method For Segmentation

The model type is of the segmentation variety and the detailed configuration is presented in table 4.21. This table includes settings for the backbone CNN (ResNet50) and specifies using layer3 for feature extraction. It mentions whether a pre-trained backbone is used, the kernel size for pooling features, and the dimensions of input and latent features. It also indicates the maximum number of training epochs set to 220. The algorithm for rapid anomaly classification comprises two principal stages[129]: feature extraction and anomaly detection.

In the feature extraction stage, images are processed through a ResNet50 backbone that has been pre-trained on ImageNet. This results in the extraction of a semantic feature vector of length 65536 from an intermediate layer (layer 3 by default). In the anomaly detection stage, a shallow linear autoencoder is trained on these features. The anomaly score is determined by the feature-reconstruction error, calculated as the norm of the difference between the original feature and its reconstruction by the autoencoder. This error tensor is then reshaped and resized to generate an anomaly map that matches the input image dimensions.

Configuration Parameter	Value	Description
backbone	resnet50	Backbone CNN network
layer	layer3	Layer to extract features from the backbone CNN
pre_trained	true	Boolean to check whether to use a pre_trained backbone
pooling_kernel_size	2	Kernel size to pool features extracted from the CNN
input_dim	65536	Dimension of feature at output of layer specified in layer
latent_dim	220	Reduced size of feature after applying dimensionality reduction
max_epochs	220	Maximum number of training epochs

Table 4.21: This table outlines the configuration parameters for the FRE model.

4.4 Sample Codes

This section introduces the core sample codes from IADBE, which utilize Python, Bash, and configuration files with YAML. Subsection 4.4.1 elucidates the code for training and testing via the API. Subsection 4.4.2 presents the script for training and testing with Bash. Subsection 4.4.3 outlines the configuration file for the dataset and model. Subsection 4.4.4 presents an overview of the process of inference. Subsection 4.4.5 illustrates the capacity to incorporate a custom dataset.

4.4.1 Training and Testing using API

```python
from anomalib import TaskType
from anomalib.data import MVTec
from anomalib.engine import Engine
from anomalib.models import Padim

datasets = ['screw', 'pill', 'capsule', 'carpet', 'grid', 'tile', 'wood',
'zipper', 'cable', 'toothbrush', 'transistor', 'metal_nut', 'bottle',
'hazelnut', 'leather']

for dataset in datasets:
    model = Padim()
    datamodule = MVTec(category=dataset, num_workers=0,
                    train_batch_size=256,
                       eval_batch_size=256)
    engine = Engine(pixel_metrics=["AUROC", "PRO"],
    image_metrics=["AUROC", "PRO"], task=TaskType.SEGMENTATION)
    # Training
    engine.fit(model=model, datamodule=datamodule)
    # Testing
    test_results = engine.test(
        model=model,
        datamodule=datamodule,
        ckpt_path=engine.trainer.checkpoint_callback.best_model_path,
    )
```

Listing 4.1: Training and Testing

This Python code snippet 4.1 demonstrates the process of training and testing an anomaly detection model using the anomalib library on multiple datasets. Initially, the code imports the necessary modules, including logging, TaskType, MVTec, Engine, and Padim from anomalib. It then defines a list of 15 datasets related to various categories, such as 'screw', 'pill', 'capsule', etc. For each dataset in the list, a Padim model is instantiated, and an MVTec datamodule is created to handle the dataset-specific configurations with specified batch sizes for training and evaluation. An Engine object is then initialized with metrics for pixel-level and image-level anomaly detection, set to perform segmentation tasks. The model is trained on the respective dataset using the engine.fit method. After training, the model is tested using the engine.test method, which evaluates the model's performance based on the best checkpoint saved during training. The process iterates over each dataset in the list, repeating the training and testing procedures.

4.4.2 Training and Testing using Bash

```
datasets=('screw' 'pill' 'capsule' 'carpet' 'grid' 'tile' 'wood'
'zipper' 'cable'
'toothbrush' 'transistor' 'metal_nut' 'bottle' 'hazelnut' 'leather')
config_file="./configs/models/padim.yaml"

for dataset in "${datasets[@]}"
do
    command = anomalib train --data anomalib.data.MVTec
    --data.category $dataset --config $config_file
    # Execute command
    $command
done
```

Listing 4.2: Training and Testing

The Bash script in Listing 4.2 automates the training process of an anomaly detection model for various datasets using the anomalib library. The script commences by defining a range of dataset categories, including 'screw', 'pill', 'capsule', and so forth. The variable $config\_file$ is then set to point to the YAML configuration file for the padim model. Thereafter, the script iterates over each dataset in the array, constructing a command that employs the anomalib train function with the specified data category and configuration file. For each dataset, the command is executed to train the model on the respective dataset using the given configuration. This script streamlines the process of running multiple training jobs in sequence.

4.4.3 Configuration File

The configuration files are employed for the purpose of Bash training and testing, with the provision of anomalib support. The following paragraph *Dataset* illustrates the configuration of the dataset file. The subsequent paragraph *Model* presents the configuration of the model file. The template configuration files are provided below for reference.

Dataset

```
class_path: anomalib.data.MVTec3D
init_args:
  root: ./datasets/MVTec3D
  category: "bagel"
  train_batch_size: 32
  eval_batch_size: 32
  num_workers: 8
  test_split_mode: from_dir
  test_split_ratio: 0.2
  val_split_mode: same_as_test
  val_split_ratio: 0.5
```

Listing 4.3: MVTec3D Dataset

This configuration file 4.3 establishes the parameters for utilizing the MVTec3D dataset in conjunction with the anomalib library. The class path identifies the dataset class to be utilized, namely anomalib.data.MVTec3D. The $init\_args$ section provides a variety of initialization arguments, including the root directory path, which defines the location of the dataset, and the category specification, which in this case is set to "bagel." The $train\_batch\_size$ and $eval\_batch\_size$ are both set to 32, indicating the number of samples per batch for training and evaluation, respectively. The value of $num\_workers$ is set to 8, which dictates the number of subprocesses for data loading. The $test\_split\_mode$ is set to $from\_dir$, with a $test\_split\_ratio$ of 0.2, signifying that 20% of the data will be utilized for testing purposes. Similarly, the val_split_mode is set to $same\_as\_test$, with a val_split_ratio of 0.5, indicating that half of the test split will be employed for validation.

Model

This configuration file 4.4 defines the setup for training a Reverse Distillation model using the anomalib library.

```
model:
  class_path: anomalib.models.ReverseDistillation
  init_args:
    backbone: wide_resnet50_2
    layers:
      - layer1
      - layer2
      - layer3
    beta1: 0.5
    beta2: 0.999
    anomaly_map_mode: ADD
    pre_trained: true

metrics:
  pixel:
    - AUROC
    - PRO
  image:
    - AUROC
    - PRO

trainer:
  callbacks:
    - class_path: lightning.pytorch.callbacks.EarlyStopping
      init_args:
        patience: 3
        monitor: pixel_AUROC
        mode: max
  check_val_every_n_epoch: 200
  max_epochs: 200
```

Listing 4.4: Reverse Distillation Model

The model section specifies that the ReverseDistillation class from anomalib.models will be used, with a $wide_resnet50\_2$ backbone. The model will utilize three layers: layer1, layer2, and layer3. The parameters beta1 and beta2 are set to 0.5 and 0.999, respectively, likely for the optimizer. The $anomaly\_map\_mode$

is set to ADD, and $pre\_trained$ is set to true to use a pre-trained model. The metrics section indicates that both pixel-level and image-level metrics will be evaluated using AUROC (Area Under the Receiver Operating Characteristic) and PRO (Per-Region Overlap). The trainer section includes settings for training. It defines an early stopping callback using lightning.pytorch.callbacks.EarlyStopping, with a patience of 3 epochs, monitoring the $pixel\_AUROC$ metric in max mode. The model will be validated every 200 epochs, with a maximum of 200 epochs for the entire training process.

4.4.4 Inference

The following Python code snippet 4.5 illustrates the process of making predictions with the aid of a previously initialized data module, a bespoke model, and an engine.

```
predictions = engine.predict(
    datamodule=datamodule,
    model=model,
    ckpt_path="path/to/checkpoint.ckpt",
)
```

Listing 4.5: Inference

The engine.predict function is invoked with three arguments: datamodule, which oversees the data for prediction; model, the pre-trained model to be utilized; and ckpt_path, the path to the checkpoint file that contains the saved model weights. By supplying these arguments to the predict method, the engine is able to generate predictions based on the specified data and model configuration.

4.4.5 Support for Custom Dataset

IADBE offers the option of a custom dataset. The first step is to import the custom dataset into the project and create a custom data configuration file. The system supports two configurations: one with only normal images and one with normal and abnormal images.

```
class_path: anomalib.data.Folder
init_args:
  name: "custom_dataset"
  root: "datasets/Custom_Dataset/hazelnut"
  normal_dir: "train/good"
  abnormal_dir: "test/crack"
  mask_dir: null
  normal_split_ratio: 0.2
  test_split_mode: synthetic
```

Listing 4.6: Configuration with only Normal Images

The following configuration snippet4.6 is provided for the purpose of establishing a bespoke dataset within the IADBE environment. The dataset, designated as $custom\_dataset$, is structured in accordance with the anomalib.data.Folder format. The $root$ variable denotes the root folder of the custom dataset. The variable $normal\_dir$ denotes the directory containing images that are not anomalous, whereas the variable $abnormal\_dir$ points to the directory with images that are anomalous. The value of $normal\_split\_ratio$ has been set to 0.2, which signifies that 20% of the normal images will be utilised for the purposes of validation or

testing. The setting of $test\_split\_mode$ to 'synthetic' denotes the generation of synthetic test cases. The value of $mask\_dir$ has been set to null, indicating that no mask images (used for pixel-wise anomaly detection) are provided.

```
class_path: anomalib.data.Folder
init_args:
  name: "custom_dataset"
  root: "datasets/Custom_Dataset/chest_xray"
  normal_dir: "train/good"
  abnormal_dir: "test/crack"
  normal_test_dir: "test/good"
  normal_split_ratio: 0
  extensions: [".png"]
  image_size: [256, 256]
  train_batch_size: 32
  eval_batch_size: 32
  num_workers: 8
  task: classification
  train_transform: null
  eval_transform: null
  test_split_mode: synthetic
  test_split_ratio: 0.2
  val_split_mode: same_as_test
  val_split_ratio: 0.5
  seed: null
```

Listing 4.7: Configuration with Normal and Abnormal Images

This configuration 4.7 is intended for the establishment of a dataset in IADBE for the purpose of anomaly classification, encompassing both normal and abnormal images. The dataset is designated as $custom\_dataset$ and is stored in the $datasets/Custom\_Dataset/chest\_xray$ directory. The $normal\_dir$ and $abnormal\_dir$ point to the directories in which the training and testing images, respectively, are stored. The directory designated as the $normal\_test\_directory$ is the location where the images representing the normal test set are stored. The image size is set to 256x256 pixels, and both the training and evaluation batch sizes are set to 32. The configuration specifies the use of eight workers for data loading. The test split mode is set to synthetic, and the $test\_split\_ratio$ and val_split_ratio determine the manner in which the dataset is split for testing and validation purposes. This setup is tailored for a classification task, and the $train\_transform$ and $eval\_transform$ fields are left null, indicating that no additional transformations are to be applied to the images.

```
anomalib train --data <path/to/custom_dataset.yaml> \
               --model anomalib.models.Padim
```

Listing 4.8: Train with Custom Dataset

This command 4.8 is employed to train a model using anomalib[1] with a custom dataset configuration. The Anomalib train command initiates the training process. The –data flag indicates the path of the YAML file that contains the dataset configuration. The –model flag specifies the anomaly detection model to be utilized, in this case, Padim (Patch Distribution Modeling), which is a model commonly employed for anomaly detection tasks.

```
anomalib predict --model anomalib.models.Padim \
                 --data <path/to/custom_dataset/custom_image.png> \
                 --ckpt_path <path/to/custom_dataset.ckpt>
```

Listing 4.9: Predict with Custom Dataset

This command 4.9 executes predictions utilizing the Padim anomaly detection model on a custom image. It specifies the model checkpoint (–ckpt_path) to load the trained model and the image file (–data) to be analyzed.

5 Evaluation

The following chapter presents the evaluation of the number of models on a variety of datasets. This section 5.1 offers an overview of the architecture of the IADBE system. Section 5.2 provides an overview of the technologies employed and the corresponding hardware settings. Section 5.3 presents a detailed analysis of the training performance, including the relationship between training speed and different models, as well as the alterations in metrics throughout the training process. Section 5.4 presents the results of inferences made by anomalib and Yolo models. Section 5.5 provides a comprehensive account of the benchmark results obtained by large-scale models on a range of datasets. Section 5.6 showcases our visualization platform.

5.1 Architecture

As illustrated in the figure 5.1, the IADBE system comprises three main components: IADBE, IADBE Server, and IADBE Backend. IADBE represents the system's core, with API and CLI serving as gateways. Datasets, models, and metrics are among the system's most crucial elements. The implementation of models is based on the open-source Anomalib[1] and YOLOv8[2, 17]. The system offers three primary entry points: Train, Test, and Predict.

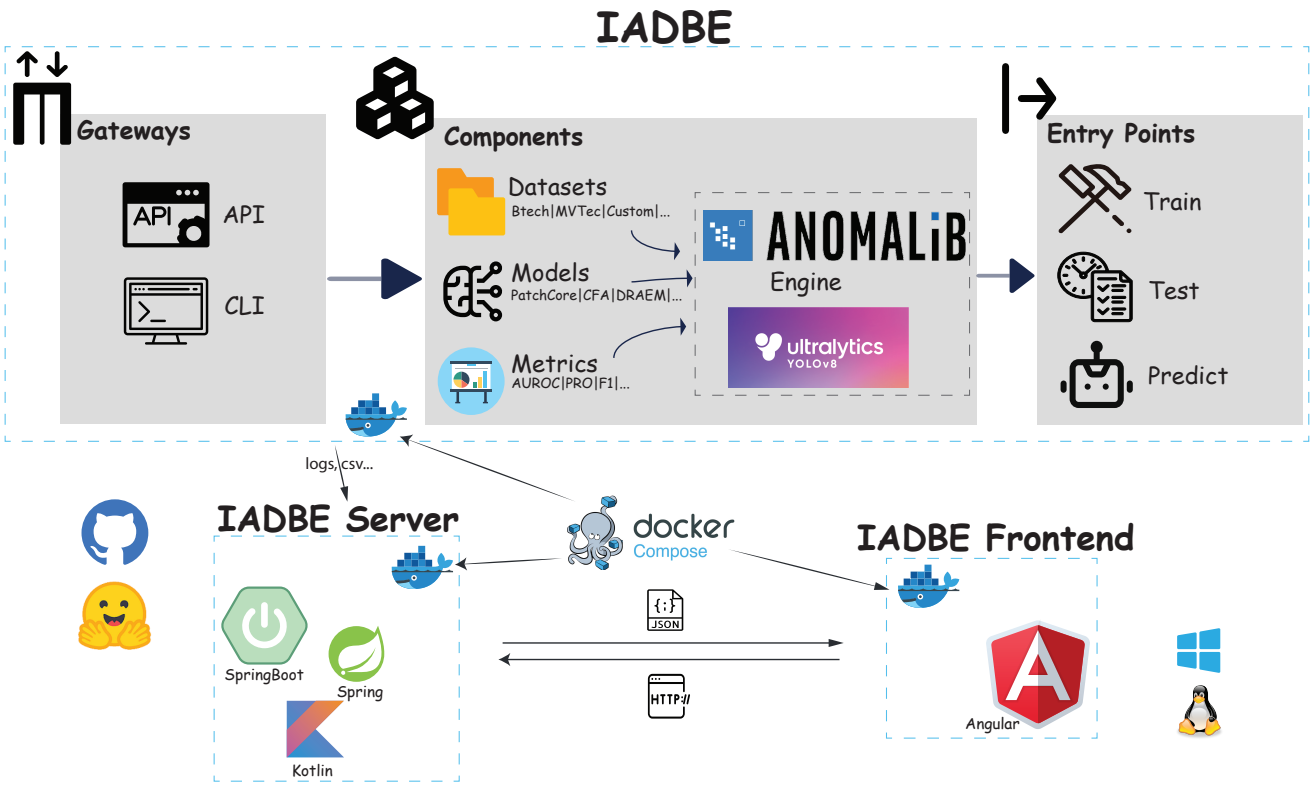

Figure 5.1: Architecture of IADBE System - The diagram illustrates the architectural framework of the IADBE system, delineating the interconnections between its constituent components.

In IADBE Server, the SpringBoot[140] and Kotlin[141] frameworks have been employed. IADBE Server receives training and testing results files from IADBE and is then responsible for file processing, including log processing and CSV processing. In IADBE Frontend, the Angular framework has been utilized. The frontend sends an HTTP request to the server, and the server then returns JSON data to the frontend. The frontend provides visualizations that are both elegantly designed and highly functional.

In order to facilitate the deployment process, we utilise the Docker Compose[5] framework to construct the IADBE, IADBE Server, and IADBE Frontend images collectively, subsequently executing the corresponding images in unison. The version control system employed is GitHub. All relevant software, including IADBE, IADBE Server, and IADBE Frontend, are published on GitHub for reproducibility and authenticity. Custom datasets and pretrained large models are uploaded on Huggingface. The cross-platform functionality has been tested on a range of operating systems, including Windows 11 and 10, as well as Ubuntu 22 and 20. It has been demonstrated that these systems can run the IADBE system without any significant issues.

5.2 Environment Settings

The experiments described in this thesis were conducted using the Python programming language. Python is a platform-independent programming language that provides an ecosystem with established libraries for data

analysis and the use of machine learning algorithms. Furthermore, Jupyter notebooks were employed, which facilitate interactive code execution and data visualization. A Jupyter notebook is a digital notebook that allows the creation of documents containing program code, text for documentation, and data visualizations[142]. The Jupyter notebook was utilized for the tutorial. The pandas[143] and numpy[144] library was employed to manipulate and preprocess the CSV data. Anomalib[1] and YOLOv8[2], which is based on scikit-learn[145], PyTorch[146] libraries, etc., were used to leverage existing ML algorithms. Visualization of data was performed using the Matplotlib[147] library.

Following an extended period of training, a series of training logs have been compiled, comprising a range of performance outcomes pertaining to diverse datasets and categories. These files must now be processed. In order to facilitate this, we have selected SpringBoot[140] with Kotlin[141] as our backend solution. The advantages of Kotlin in the context of logging include type safety, seamless integration with the JVM ecosystem, and enhanced performance. Its strong typing enables the early detection of errors, while its native coroutine support ensures efficient asynchronous logging[141]. When compared to Python[148], Kotlin emerges as a robust choice for JVM-based or high-performance applications.

Subsequently, the JSON data is obtained from the backend. For the frontend, AngularJS[149] integrated with D3[150], Charts[151], and Echarts[152] is selected for its user-friendly and interactive visualization capabilities. AngularJS offers two-way data binding, a comprehensive framework with numerous features, and a mature ecosystem, making it an appropriate choice for enterprise applications[149]. Despite its strengths, it is now considered to have been in use for many years, with Vue[153] and React[154] being preferred for modern development.

Ultimately, the deployment and development of the project are conducted on the Docker platform. This enables researchers to readily replicate the environment and execute the project on a container. The infrastructure comprises three distinct components: IADBE, IADBE Backend, and IADBE Frontend. Consequently, there are three corresponding Dockerfiles. Researchers have the option to build a Docker image independently and subsequently run it. Alternatively, the provided Docker Compose can be employed to construct the three Docker images collectively and manage them in a unified manner[4, 5].

The experiential machine is equipped with a Ryzen Threadripper 3970X and a GeForce RTX 3090 graphics processing unit. The machine is responsible for training, testing, and inferencing tasks. The operating system utilised was Ubuntu 22.04.4 LTS. The second test machine is equipped with an r5 3600 and a GeForce RTX 4060. The operating system in use is Windows 11. This machine is utilized for cross-platform testing and the assessment of the functionality of IADBE on the Windows platform.

5.3 Training Monitoring

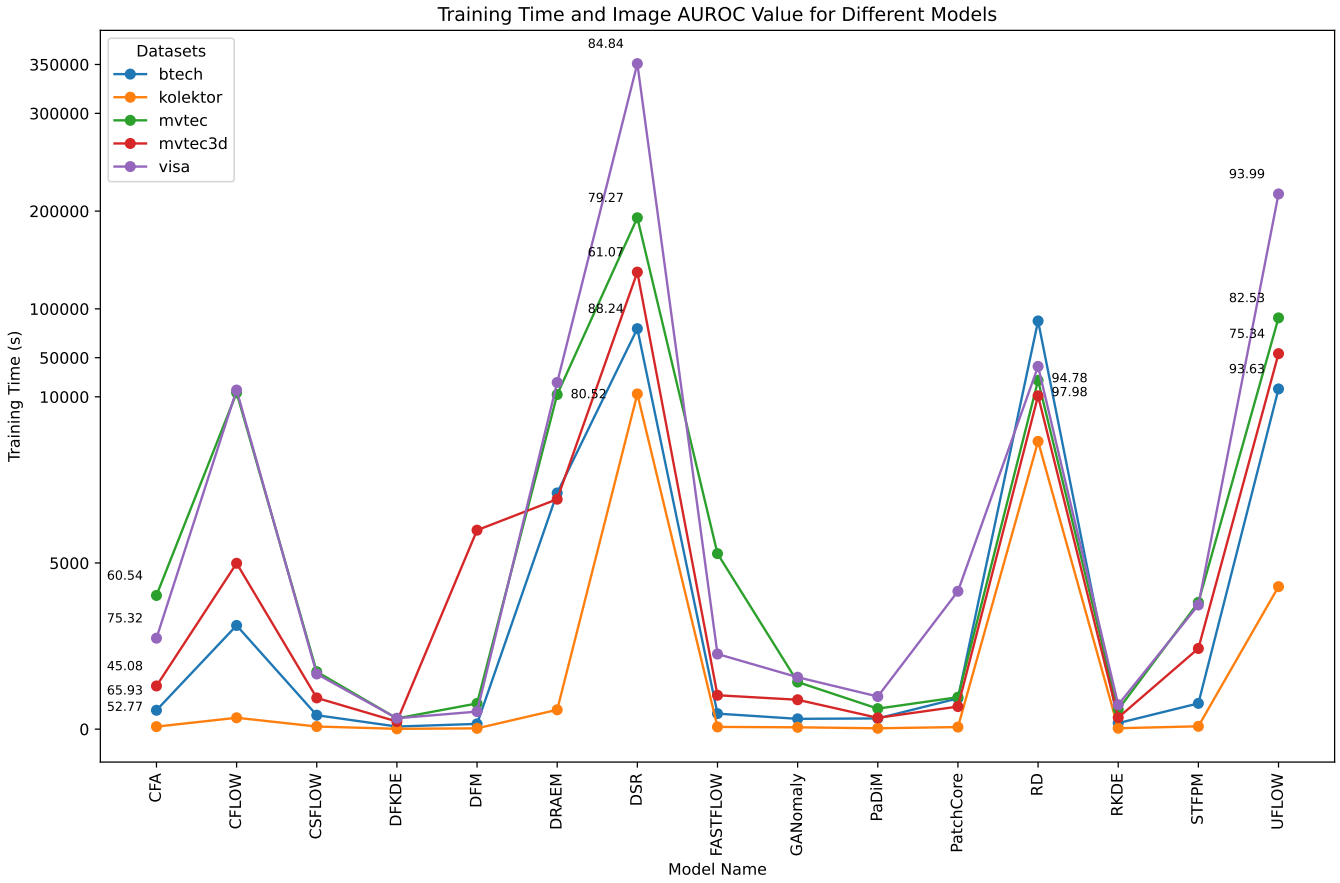

Figure 5.2: The figure illustrates the results of training time of different models. The x-axis depicts the models, while the y-axis illustrates the training time in units of seconds. Partial AUROC values are displayed in proximity to the data points. The diverse colors of the lines indicate the performance of the models on disparate datasets.

This figure 5.2 illustrates the training time required by different models. As detailed in section 4.1, the VisA dataset[123] is the most substantial, followed by MVTec[6], MVTec3D[120], Btech[121], and Kolektor[122]. Consequently, VisA[123] requires the longest training period. Section 4.3 elucidates the disparate complexities of our models, with DSR necessitating the most training time, followed by uflow, RD, and so forth.

Model	BTech Time	BTech AUROC	Kolelktor Time	Kolelktor AUROC	MVTec Time	MVTec AUROC	MVTec3D Time	MVTec3D AUROC	Visa Time	Visa AUROC
CFA	568s	65.93	75s	52.77	4023s	60.54	1299s	45.08	2737s	75.32
CFLOW	3120s	94.06	342s	86.65	14165s	95.71	4991s	76.38	16927s	87.12
CSFLOW	419s	42.53	79s	58.70	1728s	72.74	940s	54.14	1660s	52.46
DFKDE	78s	90.47	10s	67.84	323s	75.72	229s	62.14	327s	72.09
DFM	158s	95.29	24s	92.05	773s	92.80	5987s	58.42	528s	87.21
DRAEM	7106s	85.84	582s	51.92	12291s	78.44	6919s	58.93	24690s	80.52
DSR	79740s	88.24	12954s	57.72	193188s	79.27	137580s	61.07	350887s	84.84
FASTFLOW	467s	92.03	67s	77.36	5282s	83.77	1020s	68.48	2261s	89.11
GANomaly	310s	50.05	55s	56.68	1417s	43.62	884s	46.76	1562s	55.25
PaDiM	322s	95.52	26s	80.25	617s	89.97	340s	73.41	982s	84.19
PatchCore	925s	93.59	62s	86.76	961s	98.91	679s	83.61	4149s	91.66
RD	87622s	95.28	8659s	88.24	26800s	97.98	11021s	95.06	41113s	94.78
RKDE	181s	83.81	27s	64.08	587s	75.43	347s	52.92	734s	67.95
STFPM	775s	92.23	85s	66.24	3815s	72.66	2427s	71.21	3736s	84.95
UFLOW	18066s	93.63	4288s	90.01	90776s	82.53	54160s	75.34	217558s	93.99

Table 5.1: This table presents a comparison of various machine learning models based on their performance metrics across different datasets. It includes columns for execution time and AUROC scores for five datasets.

The table 5.1 provides compensation for the aforementioned figure due to the overlapping of annotations within the figure. The table includes columns for execution time and AUROC (Area Under the Receiver Operating Characteristic Curve) scores for five datasets. The following companies were included in the study: BTech, Kolelktor, MVTec, MVTec3D, and Visa. The models are evaluated in terms of their processing speed (in seconds) and their effectiveness in classification (as measured by AUROC). To illustrate, the DSR requires the longest training period, with a duration of 350,887 seconds (97.47 hours) on the VisA dataset, achieving an image-level AUROC of 84.84. Padim has an accuracy of 95.52 on the Btech dataset with a short training time of 322 seconds (5.37 minutes). The RD model performs well with the highest AUROC scores on several datasets, but at the cost of very high computation times. In contrast, models like DFKDE and PaDiM achieve strong AUROC scores with relatively lower computation times. This table helps to identify the trade-offs between computational efficiency and classification performance across different models.

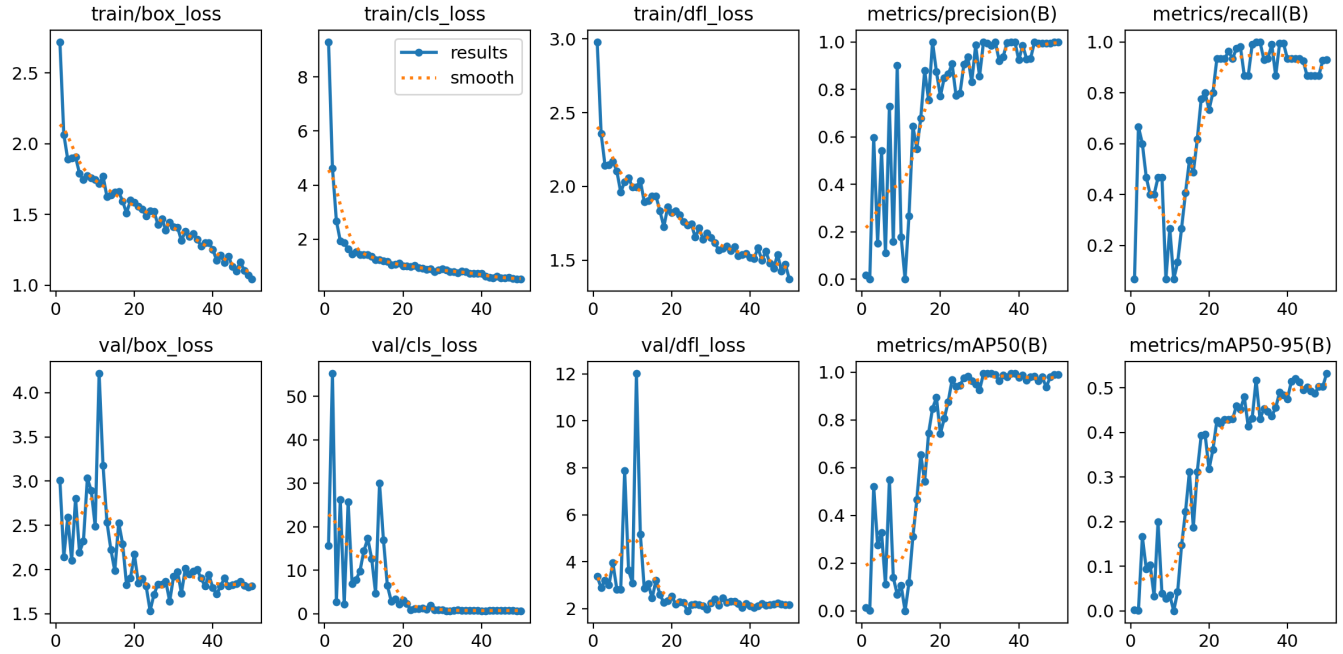

Figure 5.3: The figure illustrates the results of the training process in the custom dataset cardboard.

In YOLOv8, several key metrics and losses are crucial for understanding the model's performance. Box_loss measures the difference between predicted and actual bounding box coordinates, helping the model better localize objects. Cls_loss represents the error in classifying objects, guiding the model to improve its classification accuracy. Dfl_loss (Distribution Focal Loss) enhances the precision of bounding box predictions by refining the coordinate distribution. mAP50 (mean Average Precision at IoU=0.5) assesses the model's overall detection performance with a 50% overlap threshold between predicted and true boxes, while mAP50-95 offers a stricter evaluation by averaging performance across different IoU thresholds from 0.5 to 0.95, providing a more comprehensive measure of accuracy. These metrics collectively help evaluate and optimize YOLOv8 for object detection tasks[2, 155]. This figure 5.3 illustrates the change in metrics in accordance with the increase in epochs. As the number of epochs increases, the train/box_loss, train/cli_loss and train/dfl_loss values decrease. Prior to 20 epochs, the precision value exhibits fluctuations, followed by an upward trend. The recall value initially fluctuates, then decreases, and subsequently increases to a maximum value close to 1, after which it experiences a slight decline. As the number of epochs increased, the val/box_loss, val/cli_loss and val/dfl_loss values exhibited fluctuations prior to epoch 20. In general, a downward trend was observed. Prior to epoch 18, mAP50 and mAP50-95 demonstrated fluctuations, followed by an increase.

5.4 Inference Results

This section 5.4.1 presents illustrative examples of the results of our inference on the MVTec, VisA, MVTec3D, Btech, Kolektor and Cardboard datasets. Section 5.4.2 presents illustrative examples of the results of the YOLO model's inferences on the Cardboard datasets.

5.4.1 Models from Anomalib

The results are presented in five subfigures: the original image, the ground truth, the predicted heat map, the predicted mask, and the final segmentation result. In computer vision tasks such as object detection or segmentation, ground truth typically includes manually annotated labels (e.g., bounding boxes, masks) provided by human annotators. A predicted heatmap is a visual representation that highlights regions of interest in an image. Predicted mask is a binary or multi-class segmentation mask generated by a model.

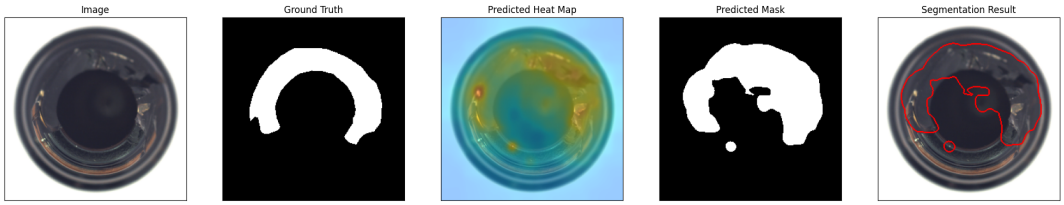

Figure 5.4: The figure illustrates the results of the inference process for the "bottle" category in the MVTec dataset.

As illustrated in figure 5.4, the upper portion of the bottle is fractured. While the model (Padim) has accurately identified the location of the defect, the predicted area is not precise and is larger than the ground truth.

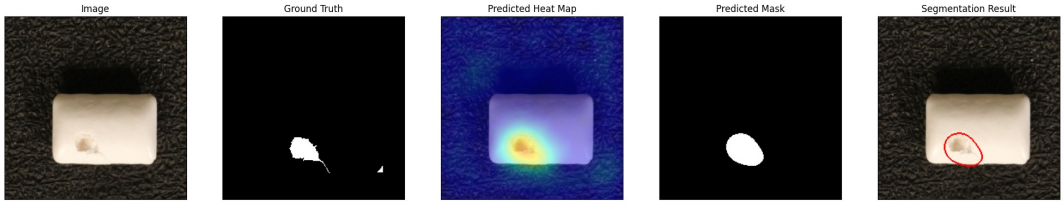

Figure 5.5: The figure illustrates the results of the inference process for the "chewinggum" category in the VisA dataset.

As illustrated in Figure 5.5, the lower left portion of the chewing gum exhibits a defect. The model (PatchCore) has accurately identified the location of the defect and the predicted size is relatively accurate.

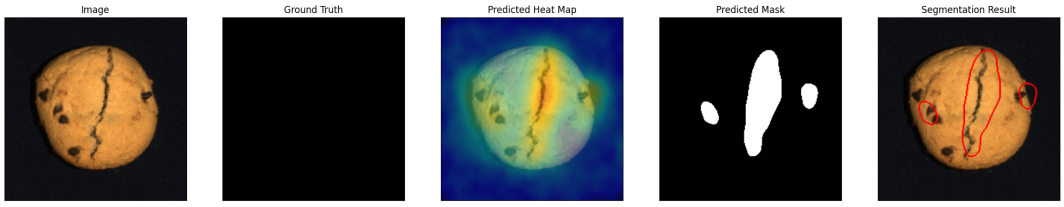

Figure 5.6: The figure illustrates the results of the inference process for the "cookie" category in the MVTec3D dataset.

As illustrated in Figure 5.6, the model (Reverse Distillation) has identified the fissure situated in the central region and the small defect area on the left and right sides.

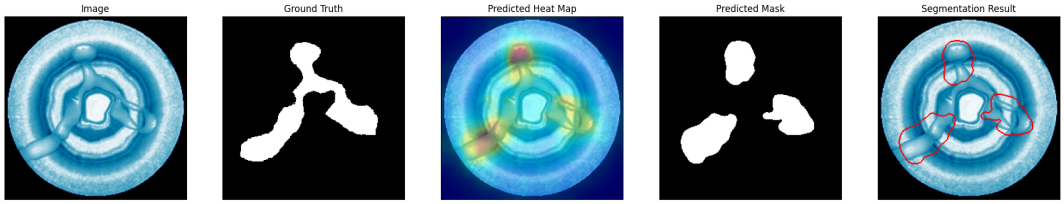

Figure 5.7: The figure illustrates the results of the inference process for the "03" (Cover) category in the Btech dataset.

As illustrated in the figure 5.7, the upper, lower left, and lower right parts of the defect are correctly identified by the model (STFPM). The model has accurately determined the location of the defect and the predicted size is relatively accurate.

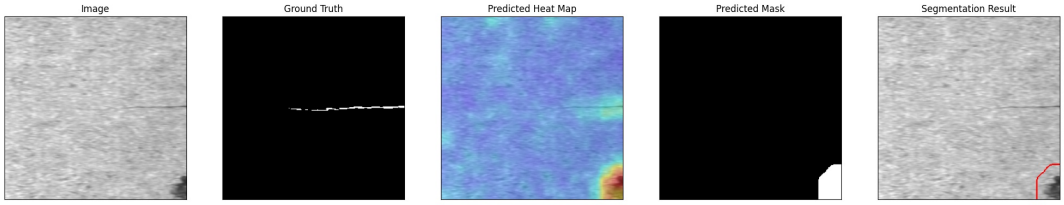

Figure 5.8: The figure illustrates the results of the inference process for Kolektor dataset.

As illustrated in the figure 5.8, the fissure in the central region and the discolouration in the lower right corner have been identified by the model (Padim). The model has accurately determined the location of the defect and the predicted size is accurate.

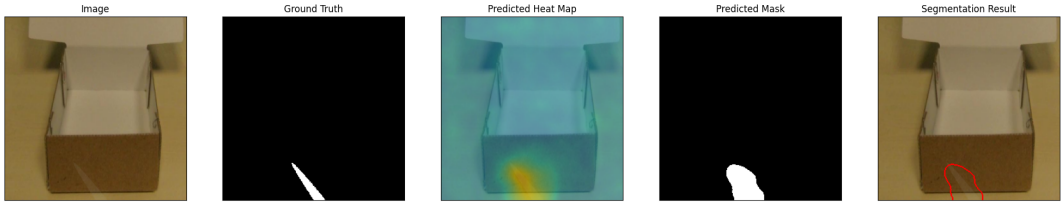

Figure 5.9: This figure shows the results of the inference process for the Cardboard custom dataset.

As illustrated in the figure 5.9, the light area directly below the cardboard is identified with precision by the model (Reverse Distillation).

5.4.2 Models from YOLO

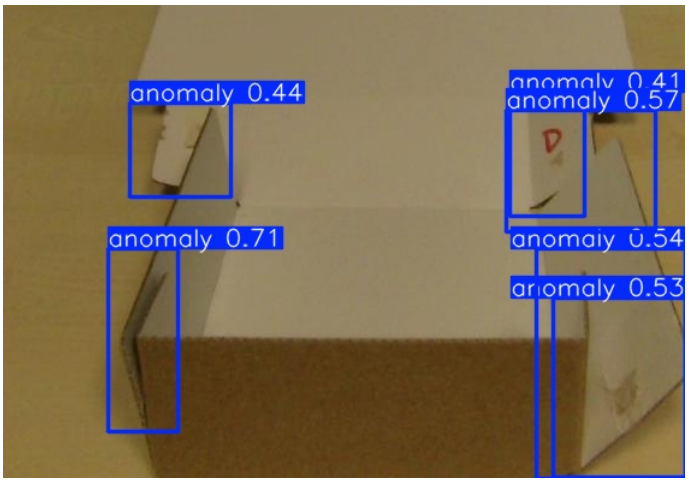

Figure 5.10: This figure shows the results of the inference process for the Cardboard custom dataset with model (YOLOv8m).

As the figure 5.10 shows, the region on lower left exhibits the highest anomaly score of 0.71, which is attributed to the presence of a shadow. The region on the upper right follows with a score of 0.57, which is influenced by the appearance of unexpected red symbols.

In YOLOv8[2], the anomaly score (confidence score) displayed in the blue box subsequent to inference/prediction is calculated in the following manner:

1. Model Output:

   The YOLOv8 model produces a tensor comprising bounding box predictions and class probabilities. This tensor is typically of the form (N, 84, 8400), where N is the batch size, 84 represents the parameters for each prediction box (comprising four bounding box parameters and 80 class probabilities), and 8400 is the total number of feature map points[2, 156].

2. Confidence Score Calculation:

$$ConfidenceScore = Pr(Class_i|Object) \times Pr(Object) \times IoU(pred, truth) \tag{5.1}$$

   Where:

   $Pr(Class_i|Object)$: Probability that the object belongs to class i.

   $Pr(Object)$: Probability that the bounding box contains any object (box confidence).

   $IoU(pred, truth)$: Intersection over Union between the predicted bounding box and the actual ground truth box[157].

3. Post-Processing:

   Subsequent to the calculation of the confidence scores, the model executes a process of non-maximum suppression (NMS), which serves to filter out low-confidence and overlapping boxes.[2, 156, 158].

5.5 Benchmark Results

This section presents a comprehensive account of the benchmark results obtained with 22 models on five standard datasets and one custom dataset. Furthermore, a multi-class unified performance test is conducted with a number of models on the MVTec dataset. The benchmark results were obtained through the use of the experimental machine, designated as GPU 3090.

5.5.1 Performance on MVTec

The results of Image Level AUROC Scores for MVTec dataset are presented in table 5.2 in detail and have been subjected to rigorous testing on the dataset MVTec[6]. In the rkde model, the image-level AUROC value is not available due to the presence of internal bugs in anomalib. Further details regarding these issues can be found in the link[159].

	Screw	Pill	Capsule	Carpet	Grid	Tile	Wood	Zipper	Cable	Toothbrush	Transistor	Metal Nut	Bottle	Hazelnut	Leather	Average	Source
CFA	87.68	93.10	48.17	50.00	84.04	37.50	62.50	40.34	61.96	**98.06**	45.00	39.78	50.00	60.00	50.00	60.54	**99.30**
CFLOW	81.57	92.91	**95.45**	**97.27**	88.47	**100.00**	**99.39**	**97.64**	94.15	94.17	94.88	**100.00**	**100.00**	**99.68**	**100.00**	**95.71**	**96.31**
CSFLOW	41.58	38.09	60.77	**99.16**	78.36	90.44	**95.75**	88.93	67.31	44.72	50.54	74.46	89.92	71.02	**100.00**	72.74	93.50
DFKDE	69.17	64.18	72.04	68.42	46.69	92.50	81.32	88.26	68.67	78.89	81.12	76.20	92.94	77.57	77.79	75.72	77.40
DFM	77.79	**96.89**	92.98	90.93	65.41	**98.67**	**97.81**	**97.51**	94.27	**96.39**	94.79	91.72	**100.00**	**96.82**	**100.00**	92.80	94.30
DRAEM	30.01	74.55	77.02	72.47	80.45	86.80	**95.96**	78.90	63.98	70.42	90.21	93.55	**98.10**	77.50	86.65	78.44	**98.00**
DSR	56.67	70.16	72.72	43.82	**97.08**	80.70	90.75	78.97	76.96	**96.94**	91.04	81.04	86.59	81.64	83.97	79.27	98.20
FASTFLOW	65.85	76.60	69.64	93.38	**96.32**	93.36	**98.16**	72.69	67.62	72.50	89.62	81.33	**99.68**	79.86	**99.90**	83.77	**99.40**
FRE	58.11	83.82	76.59	94.18	64.75	**98.99**	**98.16**	93.91	85.53	**95.00**	90.92	84.12	**99.37**	92.29	**99.93**	87.71	**98.40**
GANomaly	32.79	59.96	26.57	21.71	57.23	54.42	60.88	41.05	52.47	49.17	33.46	26.30	47.78	53.86	36.68	43.62	42.10
PaDiM	78.95	79.95	86.48	**97.99**	87.47	94.55	**97.46**	77.46	85.96	82.50	94.54	**98.34**	**99.52**	88.32	**100.00**	89.97	**96.70**
PatchCore	**98.11**	94.76	**97.85**	**99.12**	**98.08**	**98.81**	**98.77**	**99.21**	**99.10**	**100.00**	**100.00**	**99.80**	**100.00**	**100.00**	**100.00**	**98.91**	**98.10**
RD	**98.03**	**97.63**	**97.93**	**99.36**	**95.49**	**100.00**	**99.39**	**97.16**	**95.45**	91.39	**97.87**	**100.00**	**100.00**	**100.00**	**100.00**	**97.98**	**98.50**
RKDE	50.58	68.77	51.93	-	75.36	67.72	62.54	75.37	85.83	77.17	65.00	90.63	85.07	**100.00**	**100.00**	75.43	-
STFPM	77.62	40.78	60.59	**98.88**	59.23	**97.08**	**98.95**	75.39	91.34	47.50	61.88	40.22	43.97	96.50	**100.00**	72.66	**97.00**
UFLOW	49.19	94.84	56.72	**100.00**	**99.33**	**99.39**	**95.09**	89.73	62.67	64.17	81.46	55.77	**99.21**	90.39	**100.00**	82.53	**98.74**

Table 5.2: Image Level AUROC Scores for MVTec[6] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across 15 different categories from the MVTec dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

PatchCore and RD (Reverse Distillation) are the best performers in the anomaly detection task on the MVTec dataset. PatchCore achieved the highest scores across most categories, with perfect scores (100.00) in cable, toothbrush, transistor, bottle, hazelnut, and leather, which highlights its effectiveness in detecting anomalies across various objects. RD also demonstrated high performance, with perfect results in categories such as bottle, transistor, and hazelnut. Among the moderate performers, CFLOW exhibited high performance in most categories, achieving perfect scores in metal nut, bottle, and leather, but scored lower in the Screw category (81.57). PaDiM generally performed well, with perfect scores in Leather and high scores in tile and transistor. DFM also demonstrated consistent high performance with perfect scores in Bottle and leather. In contrast, GANomaly and DRAEM were the low performers. GANomaly scored the lowest across most categories, particularly in capsule, carpet, and transistor, while DRAEM showed low scores in the screw and cable categories. In the screw category, PatchCore and RD demonstrated significantly higher performance than other methods, with CFA and GANomaly performing the least well. For bottle detection, most methods

performed exceptionally well, indicating that anomaly detection is relatively straightforward in this context. In the Leather category, PatchCore, RD, and CFLOW achieved perfect scores, indicating strong capabilities in texture-based anomaly detection. The tile category exhibited considerable variability in scores, with CFA achieving a relatively low score (37.50) and CFLOW, RD, and PatchCore achieving exceptionally high scores, suggesting that some methods may encounter difficulties in detecting texture-based anomalies. Finally, in the cable category, RD and PatchCore demonstrated superior performance, while GANomaly and CFA exhibited the least efficacy. It is regrettable that the RKDE model on the category Carpet does not yield any results, due to the presence of bugs in anomalib [1]. The last column presents the results obtained from the source paper. There is a notable discrepancy between the stated result (99.30) and the average result (60.54) obtained by our evaluation. However, the results for CSFLOW, DREAM, DSR, FASTFLOW, FRE, PaDiM, STFPM, and UFLOW exhibit a relatively minor discrepancy with our evaluated results. In contrast, CFLOW, DFM, PatchCore, and RD align precisely with our results.

	Screw	Pill	Capsule	Carpet	Grid	Tile	Wood	Zipper	Cable	Toothbrush	Transistor	Metal Nut	Bottle	Hazelnut	Leather	Average
CFA	**98.54**	**98.21**	14.96	22.27	**97.30**	31.83	35.07	23.30	24.63	**98.58**	26.06	35.98	24.52	17.44	29.79	45.23
CFLOW	**97.40**	**97.48**	**98.84**	**99.04**	**97.38**	**96.09**	**95.02**	**97.45**	**96.06**	**98.30**	87.90	**96.65**	**98.58**	**98.72**	**99.54**	**96.96**
CSFLOW	67.61	63.83	79.20	70.34	59.42	67.07	64.05	59.02	64.62	70.36	52.93	67.12	69.03	73.10	66.89	66.31
DFKDE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DFM	90.41	**96.91**	94.24	**95.51**	83.43	91.75	84.55	89.11	**96.06**	**95.95**	**97.79**	94.73	93.88	**97.14**	**95.43**	93.13
DRAEM	65.04	62.63	72.93	55.56	42.77	77.31	70.12	65.49	44.90	69.65	57.17	73.83	74.22	67.57	52.11	63.42
DSR	62.75	43.83	56.51	54.86	68.48	71.69	72.80	60.69	58.96	79.58	57.58	55.36	65.17	73.15	65.55	63.13
FASTFLOW	89.65	92.15	93.43	**97.65**	**97.27**	88.51	94.53	88.90	70.57	90.58	93.54	84.14	**97.05**	93.78	**99.56**	91.42
FRE	93.13	**95.36**	**96.80**	**98.58**	89.74	94.09	89.53	94.18	**95.99**	**98.25**	**98.56**	**97.06**	**97.49**	**98.14**	**98.40**	**95.69**
GANomaly	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PaDiM	**97.85**	**95.14**	**98.46**	**98.99**	94.18	92.74	92.95	**98.20**	**96.98**	**98.67**	**97.56**	**96.57**	**98.42**	**98.02**	**99.02**	**96.92**
PatchCore	**98.91**	**97.67**	**98.76**	**98.87**	**97.96**	94.75	92.94	**97.94**	**98.01**	**98.64**	**97.16**	**98.22**	**98.16**	**98.45**	**98.99**	**97.70**
RD	**99.52**	**97.21**	**98.87**	**99.05**	**99.06**	**95.08**	94.82	**98.44**	**96.72**	**98.85**	90.44	**96.85**	**98.59**	**98.78**	**99.28**	**97.44**
RKDE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
STFPM	22.10	55.29	61.46	**98.61**	78.55	86.76	91.93	88.39	94.38	64.33	62.27	68.92	66.13	**97.09**	**97.90**	75.61
UFLOW	85.49	**97.79**	82.39	**99.40**	**97.86**	**96.04**	92.04	**96.69**	78.14	70.14	86.52	77.19	**97.98**	**98.71**	**99.37**	90.38

Table 5.3: Pixel Level AUROC Scores for MVTec [6] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across 15 different categories from the MVTec dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

The detailed results of Pixel Level AUROC are presented in 5.3 shows. It should be noted that the tasks of DFKDE, GANomaly are classification, whereas RKDE is detection. Consequently, there are no pixel-level AUROC values. PatchCore and RD demonstrated high performance in pixel-level anomaly detection, consistently achieving high scores across all categories, including near-perfect scores for categories such as screw, cable, and leather. Moderate performers, including CFLOW, PaDiM, DFM (Deep Feature Matching), and FASTFLOW, also exhibited strong performance across most categories, with perfect scores in specific categories such as Leather and high scores in others. However, methods such as CSFLOW and DRAEM demonstrated relatively lower performance, with particularly low scores in categories such as Pill and Wood. Specific category observations indicate that high-performing methods demonstrated superior performance in categories such as Screw and Carpet, while categories such as tile exhibited significant variability in performance among different methods.

	Screw	Pill	Capsule	Carpet	Grid	Tile	Wood	Zipper	Cable	Toothbrush	Transistor	Metal Nut	Bottle	Hazelnut	Leather	Average
CFA	40.24	75.15	**100.00**	**100.00**	49.28	**99.99**	**100.00**	**100.00**	99.01	33.04	**100.00**	**100.00**	**100.00**	**100.00**	**99.99**	86.45
CFLOW	26.35	71.89	36.84	82.32	47.11	86.31	81.82	67.94	53.86	32.35	57.84	90.76	84.26	82.56	77.02	65.28
CSFLOW	3.40	73.62	19.70	32.76	43.65	60.71	47.55	69.04	34.64	19.96	23.17	61.62	28.38	56.45	77.36	43.47
DFKDE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DFM	10.06	7.48	16.21	24.63	12.65	72.37	44.84	50.03	30.50	13.54	36.63	38.99	67.90	50.28	25.14	33.42
DRAEM	11.76	43.98	36.83	13.35	4.37	37.52	23.73	18.33	15.26	19.60	32.27	62.74	42.19	56.31	10.19	28.56
DSR	15.21	31.27	18.63	8.89	29.96	22.24	37.75	18.75	20.65	21.06	25.73	54.51	32.74	43.73	20.00	26.74
FASTFLOW	22.78	65.64	43.42	68.76	52.59	60.29	83.90	48.77	45.20	26.01	55.02	74.10	79.29	72.87	83.67	58.82
FRE	15.73	46.38	14.87	57.38	28.18	82.31	58.72	44.08	34.92	17.93	64.24	72.58	69.08	59.99	46.74	47.54
GANomaly	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PaDiM	30.00	80.57	33.36	75.64	41.35	80.18	83.89	70.17	74.36	55.86	72.39	91.40	83.55	69.49	90.14	68.82
PatchCore	39.05	73.99	42.25	66.39	52.92	75.70	61.65	70.79	66.17	21.35	90.73	79.43	81.62	80.82	59.28	64.14
RD	48.35	82.31	49.69	70.49	56.88	87.72	70.25	73.66	60.30	27.36	66.67	91.35	82.73	82.93	68.25	67.93
RKDE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
STFPM	1.57	57.32	23.66	73.19	22.65	64.49	74.15	36.58	60.27	24.96	12.37	92.29	84.00	66.46	69.99	50.93
UFLOW	2.03	69.63	11.97	79.76	41.02	86.69	73.60	60.55	16.37	22.36	50.78	66.00	71.60	67.67	70.85	52.73

Table 5.4: Pixel Level PRO Scores for MVTec[6] dataset categories - The table presents the Per-Region Overlap (PRO) scores for 16 different methods across 15 different categories from the MVTec dataset. Higher PRO scores indicate superior performance in distinguishing between normal and anomalous images.

The detailed results of Pixel Level PRO are presented in table 5.4. The performance analysis of various methods on the MVTec dataset reveals distinct tiers of effectiveness in pixel-level anomaly detection. High performers such as CFA and PaDiM achieved perfect or near-perfect scores across multiple categories, including capsule, carpet, wood, zipper, metal nut, bottle, hazelnut, and leather, indicating their robust and consistent anomaly detection capabilities. Moderate performers, such as CFLOW, RD, and PatchCore, exhibited strong performance in several categories, particularly leather, metal nut, and bottle. However, they also showed variability and moderate scores in others, such as screw and carpet. Low performers, including CSFLOW and DRAEM, struggled with consistency, scoring particularly low in categories like screw, cable, and transistor. This demonstrated significant challenges in handling various anomalies. It is evident from the category-specific observations that while the high-performing methods excel in areas such as carpet and leather, where they achieved perfect scores, there is considerable variability in categories such as tile and cable. This underscores the importance of method selection based on specific application needs.

5.5.2 Performance on VisA

The results of Image Level AUROC Scores for VisA dataset are presented in table 5.5 in detail and have been subjected to rigorous testing on the dataset VisA[123].

	Candle	Capsules	Cashew	Chewinggum	Fryum	Macaroni1	Macaroni2	PCB1	PCB2	PCB3	PCB4	Pipe Fryum	Average
CFA	85.49	12.67	19.12	**96.12**	19.12	88.20	59.00	94.91	93.31	90.27	**99.04**	19.12	75.32
CFLOW	89.24	94.22	**95.60**	**99.66**	80.48	74.59	66.63	87.86	88.12	74.47	**96.50**	**98.12**	87.12
CSFLOW	36.69	55.96	41.39	67.73	39.54	52.00	38.75	63.75	49.53	55.11	75.28	31.83	52.46
DFKDE	85.91	59.15	86.32	82.86	75.94	65.47	45.47	67.19	66.20	69.61	83.19	77.79	72.09
DFM	**95.23**	57.63	**95.86**	**98.46**	**95.22**	76.92	70.17	93.07	86.92	85.45	**97.47**	90.04	87.21
DRAEM	77.33	73.80	89.16	72.54	79.02	77.44	66.49	72.96	90.77	88.99	**95.67**	82.06	80.52
DSR	65.80	77.63	88.80	91.94	83.54	81.14	65.81	**95.65**	**96.31**	94.81	**98.29**	78.35	84.84
FASTFLOW	**95.24**	78.87	89.00	**98.28**	92.82	88.41	75.74	88.93	88.10	85.57	**96.34**	92.00	89.11
FRE	90.80	65.43	89.22	94.22	86.08	70.77	70.22	81.07	79.35	80.97	**97.58**	77.68	81.95
GANomaly	69.67	67.23	80.96	56.78	89.52	64.24	49.48	27.38	31.19	22.16	54.83	49.58	55.25
PaDiM	81.06	64.85	91.30	**97.66**	87.90	83.42	69.71	86.34	84.92	75.91	**95.91**	91.28	84.19
PatchCore	**98.34**	72.48	**96.32**	**99.22**	**95.60**	86.97	71.52	94.20	93.86	93.11	**99.07**	**99.24**	91.66
RD	94.22	87.52	**96.90**	**99.42**	90.62	**96.21**	84.77	**95.79**	**97.02**	**96.51**	**99.89**	**98.46**	94.78
RKDE	73.97	60.90	85.16	64.46	77.98	62.51	46.59	68.51	78.08	74.20	49.55	73.43	67.95
STFPM	76.44	88.17	59.22	91.32	93.80	74.04	89.62	**95.22**	73.44	90.73	92.13	**95.30**	84.95
UFLOW	92.87	87.45	94.94	**99.34**	93.62	**97.80**	78.58	**95.85**	**96.27**	**96.83**	**96.90**	**97.46**	93.99

Table 5.5: Image Level AUROC Scores for VisA[123] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across 12 different categories from the VisA dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

The benchmark results of image level AUROC for the VisA dataset demonstrate the efficacy of various methods in detecting anomalies in images across 12 categories. Notably, PatchCore, RD, and UFLOW exhibit the highest overall performance, with PatchCore achieving the highest average AUROC score of 91.66, closely followed by RD at 94.78, and UFLOW at 93.99. These methods demonstrate consistent performance across most categories, particularly excelling in PCB-related categories and Fryum, with scores often exceeding 95. PatchCore exhibits superior performance in categories like Candle and Cashew, while UFLOW demonstrates robust performance in categories such as Pipe and Fryum. In contrast, methods like GANomaly and RKDE exhibit comparatively lower performance, indicating less effectiveness in anomaly detection. The results indicate that PatchCore, RD, and UFLOW are among the most reliable methods for image-level anomaly detection in the VisA dataset.

	Candle	Capsules	Cashew	Chewinggum	Fryum	Macaroni1	Macaroni2	PCB1	PCB2	PCB3	PCB4	Pipe Fryum	Average
CFA	**98.92**	27.59	90.96	**99.07**	9.93	**99.54**	**97.93**	**99.33**	**97.59**	**98.31**	**98.50**	84.44	83.51
CFLOW	**98.88**	**97.61**	**98.55**	**99.01**	**95.54**	**97.86**	**97.10**	**99.37**	**96.85**	**96.85**	**97.82**	**98.27**	**98.05**
CSFLOW	50.12	62.73	69.37	61.21	64.98	53.68	47.45	66.91	52.07	68.52	61.46	71.78	60.86
DFKDE	-	-	-	-	-	-	-	-	-	-	-	-	-
DFM	94.70	86.39	**97.79**	**95.01**	**95.64**	82.55	76.89	**97.26**	68.03	90.83	94.42	**98.73**	89.85
DRAEM	64.50	47.57	34.86	30.21	50.06	48.08	57.57	44.42	57.54	63.24	79.05	81.09	54.85
DSR	65.63	74.78	82.29	70.97	55.04	70.43	55.11	67.83	67.78	77.37	57.86	68.34	67.15
FASTFLOW	**97.45**	**97.21**	**98.50**	**98.24**	88.20	**97.41**	94.26	**99.46**	**96.82**	**96.67**	**96.79**	**95.77**	**97.07**
FRE	**96.28**	87.90	**98.85**	**97.41**	**96.31**	90.32	87.91	**98.30**	94.00	92.90	**97.06**	**99.30**	94.71
GANomaly	-	-	-	-	-	-	-	-	-	-	-	-	-
PaDiM	**97.77**	92.45	**98.51**	**98.56**	**96.41**	**97.67**	**96.12**	**98.87**	**98.41**	**98.32**	**97.23**	**99.05**	**97.45**
PatchCore	**99.06**	**97.90**	**98.94**	**98.86**	94.49	**97.66**	**96.12**	**99.54**	**97.68**	**98.02**	**98.02**	**98.97**	**97.94**
RD	**99.01**	**99.55**	**95.37**	**98.82**	**96.19**	**99.36**	**99.23**	**99.69**	**98.78**	**99.18**	**98.28**	**98.96**	**98.54**
RKDE	-	-	-	-	-	-	-	-	-	-	-	-	-
STFPM	93.47	**96.88**	92.95	**96.55**	94.36	89.63	**97.88**	**99.36**	94.61	**98.29**	86.40	**97.79**	**95.72**
UFLOW	**99.16**	**99.14**	**99.62**	**99.44**	**96.64**	**99.72**	90.93	**99.74**	**98.83**	**99.00**	**98.33**	**99.30**	**98.32**

Table 5.6: Pixel Level AUROC Scores for VisA[123] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across 12 different categories from the VisA dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

As the table 5.6 shows, the pixel level AUROC scores for the VisA dataset reveal that several methods exhibit exceptional performance in anomaly detection, particularly CFA, CFLOW, PaDiM, PatchCore, RD, and UFLOW, which consistently achieve high scores across most categories. Notably, UFLOW stands out with the highest average score of 98.32, demonstrating superior performance in categories like Cashew, Chewinggum, and Pipe Fryum. Similarly, RD, PatchCore, and PaDiM also perform exceptionally well, with average scores of 98.54, 97.94, and 97.45, respectively. These methods particularly excel in PCB-related categories and Macaroni1, with scores frequently surpassing 98. In contrast, methods such as CSFLOW, DRAEM, and DSR demonstrate significantly lower performance, indicating a discrepancy in their efficacy for pixel-level anomaly detection. Collectively, the outcomes indicate that UFLOW, RD, and PatchCore are among the most dependable methods for fine-grained anomaly detection in the VisA dataset.

	Candle	Capsules	Cashew	Chewinggum	Fryum	Macaroni1	Macaroni2	PCB1	PCB2	PCB3	PCB4	Pipe Fryum	Average
CFA	43.81	0.00	0.00	28.43	0.00	28.60	5.29	10.78	15.50	10.82	36.28	0.00	14.96
CFLOW	33.84	24.14	62.06	34.91	28.97	14.30	7.52	12.50	19.57	6.22	29.28	60.67	27.83
CSFLOW	0.23	1.31	80.21	45.18	71.91	0.44	81.06	43.46	80.71	56.41	78.15	9.49	45.71
DFKDE	-	-	-	-	-	-	-	-	-	-	-	-	-
DFM	1.09	3.36	6.11	18.81	10.59	1.22	0.51	4.31	2.90	5.03	3.71	1.65	4.94
DRAEM	8.97	12.51	17.93	1.92	17.80	7.72	4.84	3.79	4.64	19.33	27.70	47.94	14.59
DSR	8.54	18.66	28.47	9.31	20.27	18.45	3.61	21.67	6.27	17.73	9.94	9.47	14.37
FASTFLOW	19.34	25.89	63.64	48.93	39.36	14.42	6.10	16.65	20.37	13.55	17.17	52.29	28.14
FRE	2.30	4.81	6.99	19.24	13.02	2.91	30.42	8.21	18.42	5.75	17.98	24.78	12.90
GANomaly	-	-	-	-	-	-	-	-	-	-	-	-	-
PaDiM	30.11	15.54	49.10	27.16	28.22	7.52	4.41	29.74	38.05	6.99	24.39	40.63	25.16
PatchCore	41.62	23.44	35.42	23.29	31.71	9.54	10.11	13.26	30.65	6.74	33.23	45.93	25.41
RD	42.79	34.65	51.22	31.30	47.60	18.38	21.95	33.47	23.62	7.45	20.64	57.32	32.53
RKDE	-	-	-	-	-	-	-	-	-	-	-	-	-
STFPM	40.91	43.52	48.15	19.84	49.25	10.05	17.16	19.84	14.73	9.57	14.67	53.44	28.43
UFLOW	44.11	30.63	41.94	24.96	49.15	18.79	17.07	11.09	17.69	9.61	31.25	62.83	29.93

Table 5.7: Pixel Level PRO Scores for VisA[123] dataset categories - The table presents the Per-Region Overlap (PRO) scores for 16 different methods across 12 different categories from the VisA[123] dataset. Higher PRO scores indicate superior performance in distinguishing between normal and anomalous images.

As the table 5.7 shows, the pixel-level PRO scores for the VisA dataset indicate significant variability in the performance of different methods for anomaly detection. Methods such as CSFLOW, RD, and UFLOW show relatively higher average scores, with CSFLOW leading at 45.71, suggesting better effectiveness in distinguishing between normal and anomalous images on a regional basis. CSFLOW performs particularly well in the Cashew and PCB1 categories, achieving scores of 80.21 and 80.71, respectively. However, many methods, including CFA, DFM, and DRAEM, exhibit lower average scores, indicating less effectiveness. For instance, CFA and DFM have average scores of 14.96 and 4.94, respectively. The results suggest that while certain methods like CSFLOW and RD are more reliable for pixel-level anomaly detection, there is still considerable room for improvement, as many methods struggle to consistently achieve high PRO scores across different categories.

5.5.3 Performance on MVTec3D

The results of Image Level AUROC Scores for MVTec3D dataset are presented in table 5.8 in detail and have been subjected to rigorous testing on the dataset MVTec3D[120], the RD model with category "rope and tire" does not yield any image-level AUROC values due to inherent issues in the anomalib.

	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Average
CFA	8.52	53.97	19.61	7.91	46.15	86.94	78.81	56.77	92.07	0.00	45.08
CFLOW	91.43	69.35	73.48	67.06	**98.82**	68.56	70.39	55.73	94.79	74.16	76.38
CSFLOW	53.33	61.03	43.41	48.16	38.31	70.69	56.62	33.45	72.55	63.84	54.14
DFKDE	64.36	71.67	73.25	49.10	72.97	53.87	58.24	52.57	83.11	42.30	62.14
DFM	92.51	66.78	45.38	31.52	45.23	33.19	59.51	65.81	84.83	59.45	58.42
DRAEM	88.02	71.10	56.78	50.87	44.79	14.03	71.12	70.65	59.78	62.16	58.93
DSR	81.87	76.25	71.21	36.51	70.58	53.37	60.14	55.26	48.19	57.29	61.07
FASTFLOW	83.99	66.28	78.09	71.67	67.68	72.62	48.40	40.51	88.27	67.31	68.48
FRE	85.49	59.44	63.22	59.15	77.09	73.87	59.11	46.59	91.53	62.02	72.75
GANomaly	36.78	65.96	50.22	2.84	62.72	47.87	59.00	56.87	40.49	44.83	46.76
PaDiM	**95.71**	73.56	64.83	62.48	89.68	64.50	72.50	53.51	76.36	80.97	73.41
PatchCore	92.72	91.95	86.84	74.48	97.86	72.56	84.94	60.13	94.66	79.95	83.61
RD	**96.18**	90.75	91.44	66.40	**99.33**	83.56	84.43	68.38	-	-	**95.06**
RKDE	51.81	71.43	47.00	60.85	56.32	53.87	53.92	41.90	35.85	56.28	52.92
STFPM	91.74	84.35	64.20	49.38	82.66	73.44	68.65	52.52	93.98	51.13	71.21
UFLOW	**95.40**	88.40	84.34	51.28	**96.89**	62.44	60.49	57.02	**96.24**	60.92	75.34

Table 5.8: Image Level AUROC Scores for MVTec3D[120] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across 10 different categories from the MVTec3D dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

The Image Level AUROC scores for the MVTec3D dataset demonstrate the varying effectiveness of different methods in distinguishing between normal and anomalous images. The RD method stands out with the highest average AUROC score of 95.06, showcasing superior performance across multiple categories. The method achieves remarkable results in categories such as Cable Gland (90.75), Carrot (91.44), and Potato (68.38), with a perfect score for Bagel (96.18) and near-perfect scores for Cookie (99.33) and Tire (98.24). PatchCore also performs exceptionally well, with an average score of 83.61. It excels in particular in categories such as Cable Gland (91.95), Carrot (86.84), and Rope (94.66). Other notable methods include CFLOW and PaDiM, with average scores of 76.38 and 73.41, respectively. CFLOW demonstrates high performance in Bagel (91.43) and Potato (94.79), while PaDiM achieves strong results in Bagel (95.71) and Cable Gland (73.56). Methods like DFM, DRAEM, and RKDE exhibit lower average scores, indicating less effectiveness in anomaly detection within this dataset. For example, DFM has an average score of 58.42, while CFA exhibits the lowest average performance with 45.08. Overall, the benchmark results demonstrate that while some methods, such as RD and PatchCore, are highly effective for certain categories, there is still variability in performance across different types of anomalies. This underscores the need for continued advancement in anomaly detection techniques.

	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Average
CFA	23.18	36.11	30.20	24.75	8.73	**99.82**	**99.09**	**99.29**	**98.98**	34.95	55.51
CFLOW	**95.53**	**98.22**	**98.42**	94.44	**99.70**	**99.90**	**97.79**	**98.37**	**98.80**	**97.52**	**97.87**
CSFLOW	64.73	70.88	75.32	60.59	81.40	76.70	69.86	80.40	62.41	82.90	72.52
DFKDE	-	-	-	-	-	-	-	-	-	-	-
DFM	72.92	89.84	84.44	57.06	59.32	38.87	72.14	51.32	61.05	20.98	60.79
DRAEM	69.74	72.89	35.05	49.03	85.19	21.98	67.83	67.16	60.50	43.56	57.29
DSR	60.65	56.68	72.45	45.68	64.70	64.01	47.89	51.01	55.15	52.04	57.03
FASTFLOW	**98.01**	92.24	87.66	88.10	**97.15**	**99.91**	**97.97**	**97.40**	**97.76**	90.57	94.68
FRE	91.34	93.59	94.35	90.16	**96.15**	**95.41**	**97.26**	**97.32**	**95.61**	89.40	94.06
GANomaly	-	-	-	-	-	-	-	-	-	-	-
PaDiM	**98.35**	**96.26**	**97.23**	**95.25**	**98.94**	**99.77**	**98.96**	**98.04**	94.75	**95.22**	**97.28**
PatchCore	94.89	**98.34**	**98.30**	**96.16**	**99.68**	**99.57**	**98.57**	**98.45**	**97.49**	**99.20**	**98.07**
RD	**98.17**	**99.06**	**99.26**	**96.83**	**99.80**	**99.86**	**99.29**	**99.28**	-	-	**98.94**
RKDE	-	-	-	-	-	-	-	-	-	-	-
STFPM	**97.63**	**97.51**	77.11	**95.23**	**98.88**	**99.76**	**99.04**	**99.43**	**99.12**	**95.71**	**95.94**
UFLOW	**95.17**	**98.70**	**98.13**	87.15	**99.74**	**99.95**	**98.11**	**96.96**	**99.43**	88.23	**96.16**

Table 5.9: Pixel Level AUROC Scores for MVTec3D[120] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across 10 different categories from the MVTec3D dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

As the table 5.9 shows, the Pixel Level AUROC scores for the MVTec3D dataset indicate the performance of various anomaly detection methods across 10 categories. The results demonstrate a clear distinction between the top-performing methods and the rest. Specifically, RD, PatchCore, and CFLOW consistently achieve the highest scores, with RD averaging an impressive 98.94. These methods exhibit superior performance, often exceeding 95 AUROC in multiple categories. For example, the RD method achieved a score of 99.06 in the Cable Gland category, 99.26 in the Carrot category, and 99.80 in the Dowel category. The PatchCore method also demonstrated exceptional performance, with an average score of 98.07. This method excelled in categories such as the Bagel category (94.89), the Cable Gland category (98.34), and the Cookie category (96.16). Similarly, CFLOW maintains a robust performance with an average of 97.87, exhibiting high scores across all categories, including Bagel (95.53) and Cable Gland (98.22). FASTFLOW and UFLOW are also noteworthy, with average scores of 94.68 and 96.16, respectively. In contrast, methods such as CFA, DFM, and DSR exhibit lower average scores, indicating less reliable performance. On the other hand, FASTFLOW demonstrates robustness in categories such as Bagel (98.01) and Foam (99.91), while UFLOW excels in categories like Cable Gland (98.70) and Dowel (99.74). For instance, CFA exhibits a relatively low average score of 55.51, despite exhibiting high performance in the Foam (99.82) and Peach (99.09) categories. The benchmark results indicate that RD, PatchCore, and CFLOW are the leading methods in pixel-level anomaly detection on the MVTec3D dataset, demonstrating consistent and superior performance across most categories.

	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Average
CFA	0.00	0.00	0.00	0.00	99.95	45.36	42.68	20.00	18.66	0.00	22.67
CFLOW	34.00	22.17	19.21	21.65	47.07	65.67	18.21	14.27	25.80	20.63	28.87
CSFLOW	3.66	8.54	76.48	2.24	70.07	37.25	20.37	80.35	39.44	30.23	37.86
DFKDE	-	-	-	-	-	-	-	-	-	-	-
DFM	14.74	33.62	10.44	3.29	11.23	34.69	4.04	6.99	6.90	2.42	13.44
DRAEM	8.07	4.17	5.95	2.70	7.25	1.34	6.08	4.24	8.81	3.34	5.30
DSR	13.04	8.86	6.29	0.70	17.62	17.57	6.50	3.03	3.04	1.98	8.56
FASTFLOW	30.62	30.50	17.77	22.75	25.62	64.30	27.49	6.40	22.38	7.28	25.51
FRE	41.07	4.11	4.93	18.50	26.36	8.33	17.02	2.69	4.04	1.74	12.88
GANomaly	-	-	-	-	-	-	-	-	-	-	-
PaDiM	17.41	42.26	11.42	28.61	32.16	46.11	33.23	54.18	12.65	12.31	29.83
PatchCore	46.07	41.54	19.42	36.18	60.30	59.81	30.05	14.57	2.72	22.48	33.31
RD	38.78	39.68	18.11	34.88	50.54	49.48	30.58	21.72	-	-	35.47
RKDE	-	-	-	-	-	-	-	-	-	-	-
STFPM	33.75	42.29	20.81	34.09	33.28	55.04	16.59	34.61	11.70	14.13	31.03
UFLOW	30.62	30.50	17.77	22.75	25.62	64.30	27.49	6.40	22.38	7.28	25.51

Table 5.10: Pixel Level PRO Scores for MVTec3D[120] dataset categories - The table presents the Per-Region Overlap (PRO) scores for 16 different methods across 10 different categories from the MVTec3D dataset. Higher PRO scores indicate superior performance in distinguishing between normal and anomalous images.

As the table 5.10 shows, the Pixel Level PRO scores for the MVTec3D dataset reveal varied performance across different methods and categories. CFA exhibits very high performance in the Dowel category (99.95), but generally shows low scores across other categories, averaging 22.67. In contrast, methods like PatchCore and RD demonstrate more consistent and robust performance. PatchCore achieves an average score of 33.31, with notable scores in Cable Gland (41.54) and Dowel (60.30). RD follows closely with an average of 35.47, performing well in categories like Cable Gland (39.68) and Dowel (50.54). CSFLOW and FASTFLOW also show competitive performance, particularly in the Foam category, where FASTFLOW scores 64.30. However, many methods, including DFM, DRAEM, and DSR, have lower overall average scores, indicating less reliability in anomaly detection across the dataset. The results highlight PatchCore and RD as leading methods, providing superior performance in distinguishing between normal and anomalous images in several categories.

5.5.4 Performance on Btech

The results of Image Level AUROC Scores for Btech dataset are presented in table 5.11 in detail and have been subjected to rigorous testing on the dataset Btech[121].

	01	02	03	Average
CFA	21.77	76.90	**99.13**	65.93
CFLOW	**97.08**	85.40	**99.70**	94.06
CSFLOW	25.12	73.10	29.38	42.53
DFKDE	93.88	78.40	**99.14**	90.47
DFM	**99.32**	86.57	**99.99**	**95.29**
DRAEM	91.45	80.42	85.66	85.84
DSR	**95.63**	73.75	**95.33**	88.24
FASTFLOW	**99.71**	85.47	90.91	92.03
FRE	**95.14**	78.27	**99.98**	91.13
GANomaly	73.18	35.00	41.97	50.05
PaDiM	**99.71**	87.27	**99.57**	**95.52**
PatchCore	**97.76**	83.03	**99.99**	93.59
RD	**99.61**	86.25	**99.98**	**95.28**
RKDE	92.03	73.78	85.61	83.81
STFPM	**95.43**	81.37	**99.89**	92.23
UFLOW	**98.93**	85.55	**96.42**	93.63

Table 5.11: Image Level AUROC Scores for Btech[121] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across 3 different categories from the Btech dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

The image level AUROC scores for the Btech dataset demonstrate the performance of various methods across three categories. Several methods exhibit exceptional performance, with many achieving AUROC scores near or above 95. Notably, DFM, PaDiM, RD, and PatchCore stand out with the highest average scores of 95.29, 95.52, 95.28, and 93.59, respectively, indicating their robust ability to distinguish between normal and anomalous images. CFA and CSFLOW show relatively lower average performance, with scores of 65.93 and 42.53, respectively. Additionally, methods such as CFLOW, DFKDE, and FASTFLOW also demonstrate satisfactory performance, with average scores of 94.06, 90.47, and 92.03, respectively. In contrast, GANomaly exhibits considerably lower performance, with an average score of 50.05, suggesting that it may be less effective on this particular dataset. The highest individual category scores are frequently at or near perfect (close to 100), which highlights the effectiveness of several methods in specific categories. In summary, PaDiM and DFM emerge as the top performers, consistently delivering high AUROC scores across all categories.

	01	02	03	Average
CFA	51.57	**95.50**	**99.52**	82.20
CFLOW	**95.06**	94.21	**99.64**	**96.30**
CSFLOW	47.61	68.91	51.07	55.86
DFKDE	-	-	-	-
DFM	93.15	94.50	**99.34**	**95.66**
DRAEM	-	-	-	-
DSR	75.69	67.20	55.29	66.06
FASTFLOW	**95.44**	**95.90**	**98.86**	**96.73**
FRE	**95.74**	**95.00**	**98.78**	**96.51**
GANomaly	51.80	60.33	44.39	52.17
PaDiM	**97.13**	**95.68**	**99.61**	**97.47**
PatchCore	**96.56**	**95.39**	**99.46**	**97.14**
RD	**97.39**	**96.76**	**99.72**	**97.96**
RKDE	-	-	-	-
STFPM	94.81	**97.21**	**99.09**	**97.04**
UFLOW	**95.49**	**96.28**	**99.47**	**97.08**

Table 5.12: Pixel Level AUROC Scores for Btech[120] dataset categories - The table presents the Per-Region Overlap (PRO) scores for 16 different methods across 3 different categories from the Btech dataset. Higher PRO scores indicate superior performance in distinguishing between normal and anomalous images.

As the table 5.12 shows, the Pixel Level AUROC Scores for the Btech dataset demonstrate the performance of various anomaly detection methods across three categories. Methods such as CFLOW, DFM, FASTFLOW, PaDiM, PatchCore, RD, and UFLOW exhibit excellent performance with average scores surpassing 95. Notably, RD achieves the highest average score of 97.96, closely followed by PaDiM with 97.47, and PatchCore with 97.14. These methods maintain high scores across all categories, indicating their robustness and reliability. In contrast, CSFLOW and DSR show lower performance with average scores of 55.86 and 66.06, respectively, suggesting they are less effective for this dataset. The absence of scores for DFKDE and GANomaly indicates that these methods were either not evaluated or did not perform adequately. Overall, RD, PaDiM, PatchCore, and FASTFLOW stand out as the top-performing methods in this benchmark, consistently achieving high AUROC scores and demonstrating their effectiveness in distinguishing between normal and anomalous images.

	01	02	03	Average
CFA	0.00	35.44	40.80	25.41
CFLOW	43.96	43.08	47.42	44.82
CSFLOW	**95.66**	34.06	1.50	43.74
DFKDE	-	-	-	-
DFM	21.87	30.24	25.25	25.79
DRAEM	21.53	18.31	1.60	13.81
DSR	36.01	28.92	10.73	25.22
FASTFLOW	34.90	36.57	31.37	34.28
FRE	31.12	37.99	25.72	31.61
GANomaly	-	-	-	-
PaDiM	43.29	39.26	35.46	39.34
PatchCore	32.63	27.29	32.02	30.65
RD	42.14	30.56	41.81	38.17
RKDE	-	-	-	-
STFPM	40.46	38.08	47.99	42.18
UFLOW	32.71	28.18	20.90	27.26

Table 5.13: Pixel Level PRO Scores for Btech[120] dataset categories - The table presents the Per-Region Overlap (PRO) scores for 16 different methods across 3 different categories from the Btech dataset. Higher PRO scores indicate superior performance in distinguishing between normal and anomalous images.

As the table 5.13 shows, the Pixel Level PRO Scores for the Btech dataset reveal notable variability in the performance of different anomaly detection methods across three categories. CSFLOW stands out with a high score of 95.66 for category 01, significantly outperforming other methods. However, it shows a drastic drop in performance for categories 02 and 03. CFLOW and STFPM demonstrate relatively balanced scores across all categories, with CFLOW achieving 43.96, 43.08, and 47.42, and STFPM scoring 40.46, 38.08, and 47.99, respectively. These methods exhibit consistency, albeit at moderate performance levels. Conversely, methods such as DFM, DSR, FASTFLOW, and UFLOW exhibit lower and more variable scores, indicating a lack of reliability. The absence of scores for DFKDE, DRAEM, GANomaly, and RKDE suggests either non-participation in this evaluation or inadequate performance. In summary, CSFLOW demonstrated exceptional performance in one category, while CFLOW and STFPM exhibited the most consistent performance across all categories, indicating their robustness in detecting anomalies within the Btech dataset.

5.5.5 Performance on Kolektor

The results of Image Level AUROC Scores for Kolektor dataset are presented in table 5.14 in detail and have been subjected to rigorous testing on the dataset Kolektor[122].

	CFA	CFLOW	CSFLOW	DFKDE	DFM	DRAEM	DSR	FASTFLOW	FRE	GANomaly	PaDiM	PatchCore	RD	RKDE	STFPM	UFLOW
Kolektor	52.77	86.65	58.70	67.84	92.05	51.92	57.72	77.36	83.13	56.68	80.25	86.76	88.24	64.08	66.24	90.91

Table 5.14: Image Level AUROC Scores for Kolektor[122] dataset categories - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 16 different methods across one category from the Kolektor dataset. Higher AUROC scores indicate superior performance in distinguishing between normal and anomalous images.

The benchmark results for the Kolektor dataset, shown in table 5.14, highlight the performance of 15 different methods in terms of their image level AUROC scores. The UFLOW method achieves the highest score with 90.91, indicating superior performance in distinguishing between normal and anomalous images. CFLOW and PatchCore also perform well, with scores of 86.65 and 86.76, respectively. RD follows closely with an AUROC of 88.24. Other notable performances include PaDiM (80.25) and FASTFLOW (77.36). Methods such as DFM, DRAEM and RKDE are not evaluated, while CFA (52.77) and CSFLOW (58.70) show lower effectiveness compared to the top performers. These results suggest that while several methods show high effectiveness, UFLOW, PatchCore, and RD stand out as the most reliable for image-level anomaly detection in the Kolektor dataset.

	CFA	CFLOW	CSFLOW	DFKDE	DFM	DRAEM	DSR	FASTFLOW	FRE	GANomaly	PaDiM	PatchCore	RD	RKDE	STFPM	UFLOW
Kolektor	56.17	85.41	46.09	-	86.04	53.13	64.76	86.47	84.55	-	84.87	88.30	89.96	-	84.12	97.04

Table 5.15: Pixel Level AUROC Scores for Kolektor[122] dataset categories - The table presents the Per-Region Overlap (PRO) scores for 16 different methods across one category from the Kolektor dataset. Higher PRO scores indicate superior performance in distinguishing between normal and anomalous images.

The benchmark results for the Kolektor dataset, as depicted in Table 5.15, provide an insightful comparison of 15 different methods based on their pixel level AUROC scores at the pixel level. The UFLOW method outperforms all others with an impressive score of 97.04, indicating its superior ability to detect anomalies at a fine-grained level. RD and PatchCore also demonstrate strong performance with scores of 89.96 and 88.30, respectively. FASTFLOW (86.47) and PaDiM (84.87) show competitive results, while DSR (64.76) and CSFLOW (46.09) lag behind. Several methods, including DFKDE, DFM, DRAEM, GANomaly, and RKDE, are not evaluated in this context. Overall, UFLOW, RD, and PatchCore emerge as the most effective methods for pixel-level anomaly detection in the Kolektor dataset, highlighting their robustness in identifying precise defects within images.

	CFA	CFLOW	CSFLOW	DFKDE	DFM	DRAEM	DSR	FASTFLOW	FRE	GANomaly	PaDiM	PatchCore	RD	RKDE	STFPM	UFLOW
Kolektor	0	9.59	81.47	-	8.25	9.24	2.49	9.71	6.83	-	9.28	7.33	7.87	-	10.45	23.22

Table 5.16: Pixel Level PRO Scores for Kolektor[122] dataset categories - The table presents the Per-Region Overlap (PRO) scores for 16 different methods across one category from the Kolektor dataset. Higher PRO scores indicate superior performance in distinguishing between normal and anomalous images.

As the table 5.16 shows, The benchmark results for Pixel Level PRO Scores on the Kolektor[122] dataset reveal significant variation in the performance of different methods. CSFLOW achieves the highest score at 81.47, indicating superior capability in distinguishing between normal and anomalous regions. Other methods,

such as UFLOW (23.22) and CFLOW (9.59), show moderate performance, whereas methods like DSR (2.49), PatchCore (7.33), and RD (7.87) demonstrate lower effectiveness. The diversity in performance underscores the need for careful method selection based on specific requirements and dataset characteristics.

5.5.6 MVTec with Multi-class Unified

Typically, a pretrained model demonstrates expertise within a specific category of dataset. The objective is to ascertain its performance on other categories. To this end, a test is conducted with a unified model for all categories. A model with 100% accuracy for a specific category is selected, or alternatively, a model with relatively good accuracy is chosen. In the event of multiple category models with 100% accuracy, a model is randomly chosen from them.

	Screw	Pill	Capsule	Carpet	Grid	Tile	Wood	Zipper	Cable	Toothbrush	Transistor	Metal Nut	Bottle	Hazelnut	Leather	Average	Chosen
CFA	56.26	93.24	29.84	56.14	50.00	67.21	73.86	50.00	48.37	48.33	50.00	44.70	50.00	92.96	62.23	58.21	Pill
CFLOW	92.45	19.59	12.90	**98.31**	11.99	**99.03**	35.57	30.25	37.44	91.67	87.00	94.62	1.23	6.86	55.84	51.65	Tile
CSFLOW	72.18	57.72	55.05	92.78	58.81	85.19	77.89	56.45	56.20	61.25	35.50	34.41	70.60	77.86	94.46	65.76	Carpet
DFKDE	50	50	50	50	50	92.50	49.17	50	50	50	50	50	50	50	50	52.78	Tile
DFM	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	100.00	50.00	50.00	53.33	Bottle
DRAEM	51.01	39.77	53.77	60.23	51.55	71.25	89.12	61.11	53.17	44.44	43.00	28.25	**97.54**	71.39	77.24	59.52	Bottle
DSR	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	91.67	50.00	50.00	50.00	50.00	50.00	52.78	Toothbrush
FASTFLOW	39.74	40.83	57.68	53.97	59.57	63.56	53.51	58.56	42.71	71.94	51.83	30.45	49.76	51.39	**100.00**	55.03	Leather
FRE	50.00	50.00	50.00	50.00	50.00	50.00	98.77	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	53.25	Wood
GANomaly	46.14	59.71	50	41.05	88.72	45.88	65.79	34.80	49.23	47.50	49.08	48.34	50	55.39	32.78	50.96	Wood
PaDiM	43.80	52.21	50.46	54.82	60.19	74.21	96.84	48.23	49.46	50.00	47.50	50.00	50.00	42.46	**99.93**	58.01	Leather
PatchCore	50.00	43.77	50.00	**96.91**	50.00	88.42	**97.28**	50.00	50.00	50.00	50.00	50.00	50.00	79.86	**100.00**	63.75	Leather
RKDE	43.19	41.13	49.54	56.34	51.13	51.88	51.23	48.44	48.91	51.11	50	50.24	50	85.00	27.82	50.40	Hazelnut
STFPM	50.00	50.00	50.00	**99.16**	76.44	85.28	93.42	50.00	50.00	50.00	50.00	50.00	50.00	50.00	95.31	63.31	Carpet
UFLOW	50.00	50.00	50.00	**100.00**	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	53.33	Carpet

Table 5.17: Image Level AUROC Scores for MVTec [6] dataset categories based on a single unified model - The table presents the area under the receiver operating characteristic curve (AUROC) scores for 15 different methods across 15 different categories from the MVTec dataset.

As demonstrated in table 5.17. A performance test was conducted using a single unified model representing a category from the MVTec dataset. CFLOW exhibited noteworthy performance on the screw and carpet datasets when a pretrained tile model was employed. PatchCore exhibited satisfactory performance on the carpet and wood datasets when pretrained with leather. In general, the low accuracy models, such as GANomaly and CFA 5.2, demonstrated satisfactory results with a single unified model. However, the high accuracy models, including PatchCore, FastFlow, and RD5.2, did not perform as well as anticipated when tested with their own specified models. It is noteworthy that the models from Anomalib that we evaluated were not multi-class unified models. Therefore, it is understandable that they did not achieve optimal results.

5.5.7 Performance on Custom Dataset

The following table illustrates the F1 scores achieved by a range of models on a custom dataset comprising images of cardboard. The F1 score is a metric that balances precision and recall, indicating how well a model identifies relevant instances while minimising errors. The highest score of 96.36 was achieved by YOLOV8, which demonstrated the best performance, while CSFLOW had the lowest score of 0, indicating poor performance. The results demonstrate significant variations in model effectiveness, with YOLO-based models generally outperforming others in identifying cardboard. The superior performance of YOLO compared to

	CFA	CFLOW	CSFLOW	DFM	DRAEM	DSR	FASTFLOW	FRE	PADIM	PATCHCORE	RD	RKDE	STFPM	UFLOW	YOLOV3	YOLOV5	YOLOV6	YOLOV8	YOLOV9	YOLOV10
Cardboard	77.78	70.59	0	69.57	78.26	47.06	59.26	69.23	73.68	85.71	77.78	66.67	76.92	62.50	75.43	90.00	81.57	96.28	96.36	92.43

Table 5.18: Image Level F1Score for custom dataset cardboard - The table presents F1 scores for 11 anomalib models and 6 YOLO models across one category from the cardboard dataset. A high F1 score of 1 indicates optimal precision and recall, whereas a low score indicates suboptimal model performance.

Anomalib models can be attributed to the more efficient architectural design of YOLO, which necessitates the use of finely processed data. The outcomes of YOLO demonstrate the highest level of performance. Nevertheless, the results of Anomalib remain commendable, particularly given the absence of the necessity for annotated data. However, the applicability of YOLO is limited in anomaly detection due to the unavailability of labelled data.

5.6 Interactive Visualization

This section presents our visualization platform, which is implemented based on Angular[149] and Nebular[160]. These figures facilitate a more comprehensive understanding of IADBE in comparison to matplotlib[147] figures.

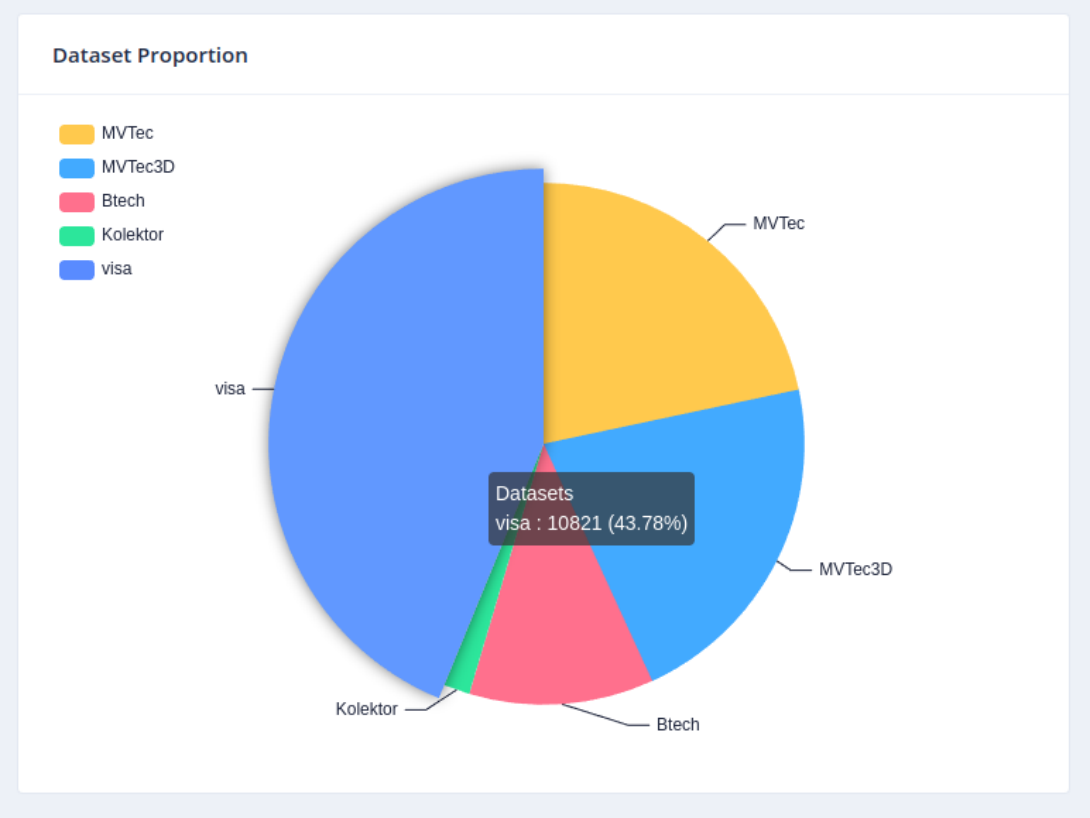

Figure 5.11: Dataset Proportion - The pie chart illustrates the number of images present in each dataset and their proportion within the entire dataset. It is based on the Echarts[152], Angular[149], and Nebular[160] frameworks

As illustrated in figure 5.11. users can interact with the figure. Upon hovering the cursor over the pie, the proportion and image numbers of the dataset are displayed. Additionally, the detailed data can be found in Table 4.1, which provides a comparison of the datasets.

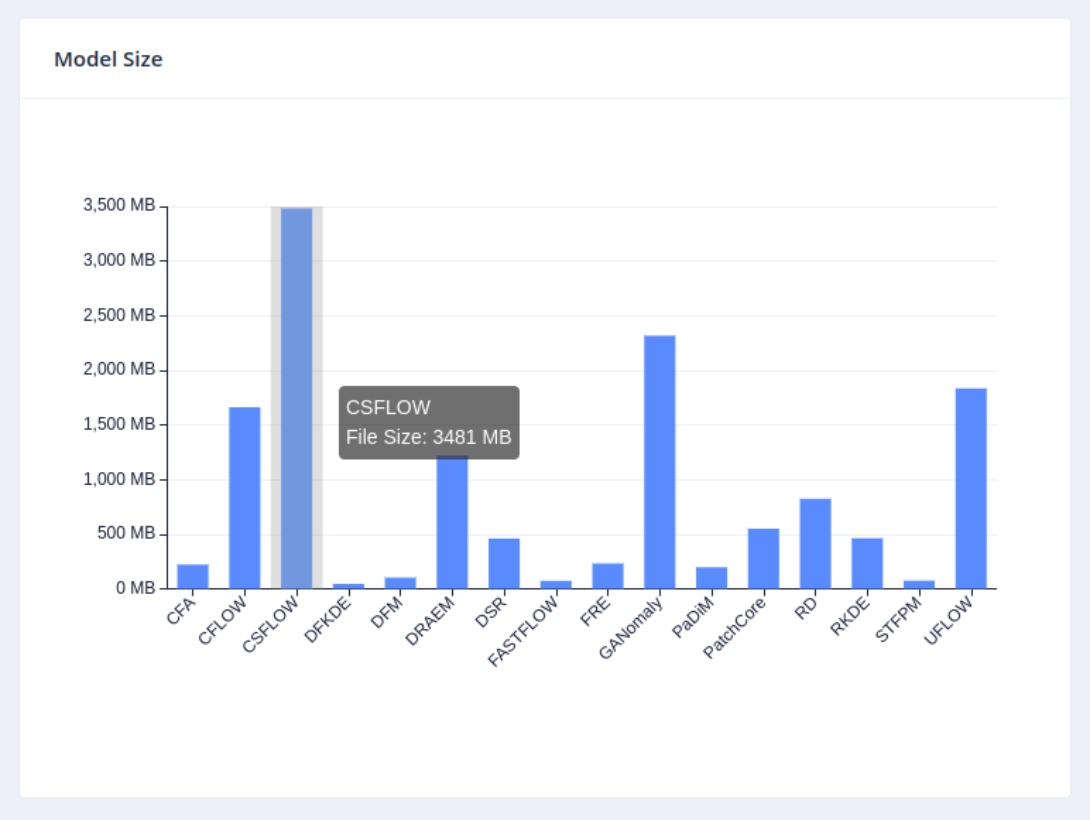

Figure 5.12: Model Size - This bar chart is based on the Echarts[152], Angular[149], and Nebular[160] frameworks

As illustrated in figure 5.12, this table provides an overview of the file size of the model. Upon hovering the cursor over the model, the user can access detailed information regarding the file size and name of the model.

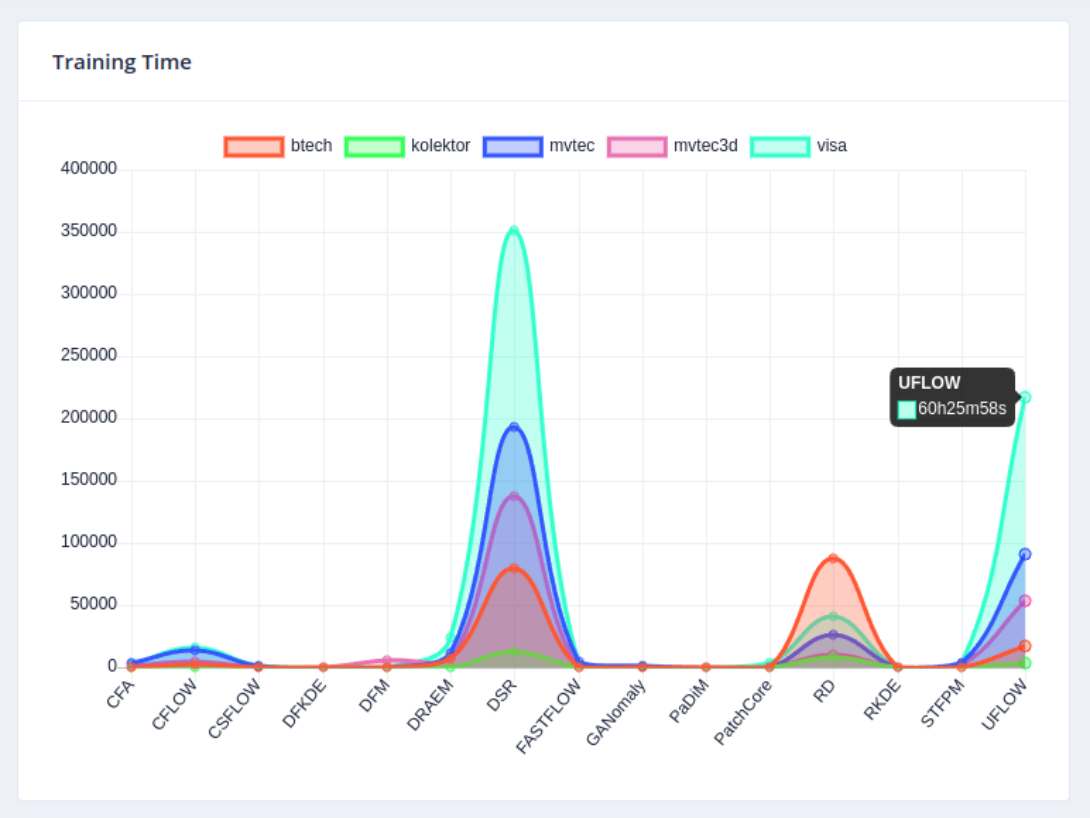

Figure 5.13: Training Time - This line chart is based on the Charts.js[151], Angular[149], and Nebular[160] frameworks

As illustrated in figure 5.13, this table depicts the training time of various models on standard datasets. Upon hovering the cursor over the line chart, the name of the model and the corresponding training time, which has been converted from seconds to hours, is displayed.

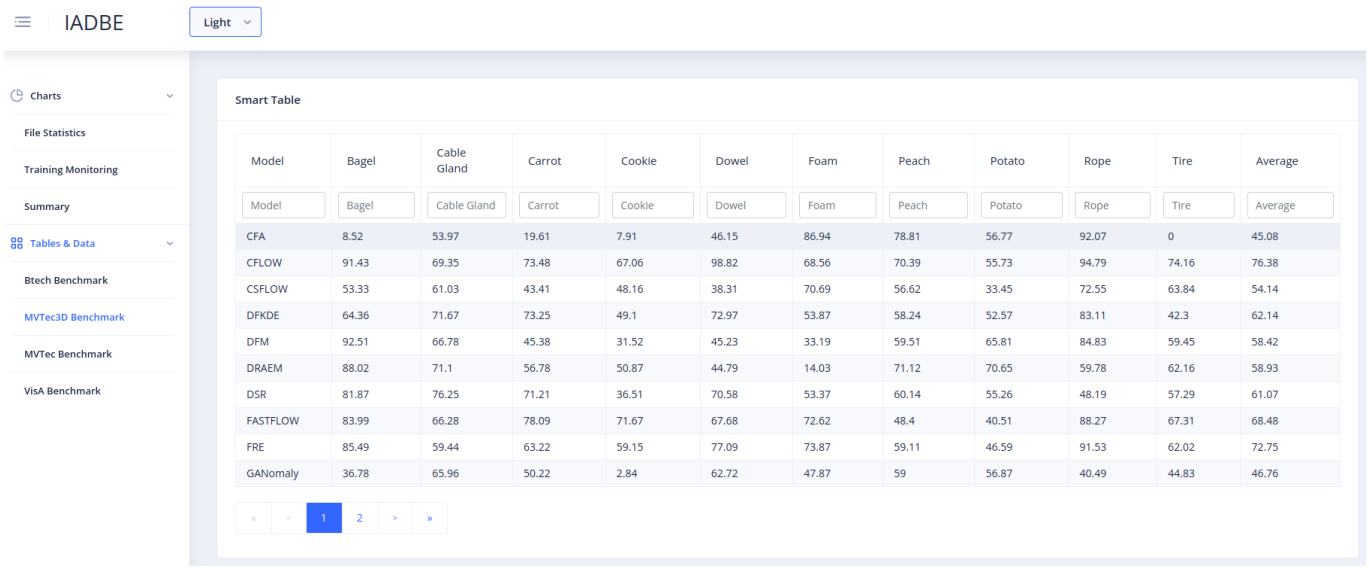

Figure 5.14: Benchmark Table - This smart table is based on Angular[149], and Nebular[160] frameworks

As illustrated in Figure 5.14, this table provides a comprehensive overview of the MVTec3D[120] benchmark results. The input field at the top allows users to search for specific values. The additional results are displayed on the subsequent page. The visualization in use allows for the alteration of the theme, including the option to change the mode from light to dark, and so forth.

6 Conclusion

This chapter provides a summary of the thesis and describes the primary contributions and discussion regarding model performance, comparisons to source papers, and comments on the utilized technologies. Finally, we present potential future work within the scope of this thesis.

6.1 Summary

We introduce IADBE in this paper. The benchmark engine has been designed with the objective of facilitating straightforward deployment by researchers. To date, this is the most comprehensive industrial anomaly detection benchmark engine, comprising 22 popular AD models[1] and 6 YOLO[2] models, 5 renowned industrial datasets, and a number of custom datasets, along with more than 6 metrics. The integration of Anomalib and Ultralytics (YOLOv8)[2] into IADBE enhances the benchmark engine's capacity to address a spectrum of anomaly detection tasks. The benchmark results of over 16 models, comprising both renowned anomaly detection models and YOLO models on more than 6 datasets, are presented in detail and explained.

The analysis of the MVTec[6], VisA[123], MVTec3D[120], Btech[121], and Kolektor[122] datasets reveals several key methods that consistently demonstrate robust performance in anomaly detection, both at the image and pixel levels. PatchCore[28] and RD[23] emerge as the most effective methods across multiple datasets, particularly demonstrating high performance in image-level AUROC and pixel-level AUROC metrics, thereby illustrating their robustness in detecting anomalies across diverse categories. Furthermore, CFLOW[26] demonstrates reliable performance in both metrics, whereas methods such as UFLOW[133] and PaDiM[130] exhibit superior performance in specific datasets, namely VisA and Btech, respectively. Notwithstanding their respective merits, methods such as GANomaly[128], STFPM[132], and CSFLOW[124] display comparatively weaker performance in certain metrics, particularly in the MVTec dataset. However, CSFLOW exhibits a notable improvement in pixel-level PRO in other datasets. Furthermore, CFA[131] and DRAEM[30] also exhibit suboptimal results in pixel-level AUROC, yet demonstrate enhanced performance in pixel-level PRO. The results obtained from the VisA and MVTec3D datasets suggest that while some methods consistently demonstrate high performance, others exhibit variability in their outcomes depending on the type of anomaly and the specific dataset under consideration. The Btech and Kolektor datasets further illustrate the significance of meticulous method selection, as methods like CSFLOW, FASTFLOW[126], and STFPM exhibit strengths in particular categories but not consistently across all metrics. Overall, the results underscore that while some methods like PatchCore, RD, and CFLOW are dependable across diverse scenarios, there is still room for improvement, and selecting the optimal method depends on the specific requirements and characteristics of each dataset.

Moreover, the visualisation platform, which is a supplement of IADBE, is composed of an IADBE backend with Spring Boot[140] and Kotlin[141] and an IADBE frontend with AngularJS[149]. The IADBE backend is utilised for the processing of files, including raw logs and raw tables, among others. The IADBE frontend

provides a sophisticated visualisation for not only AI researchers but also those with a more general interest in the subject matter. In order to facilitate the deployment and development of the project, we utilize the Docker and Docker Compose platforms.[4, 5].

As a supplement to Anomalib, the IADBE provides a comprehensive and meticulous guide to the utilisation of Anomalib[1], which is deficient in accessible and intelligible documentation and tutorials. Our engine in question is entirely reproducible and authentic. The raw training and testing logs are accessible for review in the GitHub repository. As an open-source package, the IADBE is released with the objective of consistently incorporating the most recent, cutting-edge techniques in the field. The platform welcomes contributions from the community, including pull requests and issues. Moreover, both the standard datasets, including MVTec[6], VisA[123], MVTec3D[120], Btech[121], Kolektor[122], and others, as well as the custom datasets, are accessible for download from Huggingface[161]. However, we have also provided pipelines for downloading the standard datasets from IADBE. The models utilized for benchmarking and exceeding 370 GB in size have been made publicly available on Huggingface[161] for the purpose of image inference with a standard dataset.

6.2 Discussion

A notable discrepancy is evident between the stated result (99.30) and the average result (60.54) obtained by our evaluation. However, the results for CSFLOW, DREAM, DSR, FASTFLOW, FRE, PaDiM, STFPM, and UFLOW exhibit a relatively minor discrepancy with our evaluated results. In contrast, CFLOW, DFM, PatchCore, and RD align precisely with our results, indicating that the discrepancy may be attributed to differences in hyperparameter settings or hardware environments. It is important to note that the use of anomalib with identical hyperparameter configurations in API and CLI may yield disparate outcomes. This discrepancy could be attributed to the divergence in implementation between API and CLI.

An increasing number of new models are being published in forthcoming papers. It is necessary to reimplement these anomaly detection models as soon as possible and to test their performance on the given dataset. To access how much we have discrepancy with the method proposed from source paper. The use of additional evaluation metrics for comprehensive assessment is recommended.

It is imperative that the video dataset be supported. At present, the video dataset support provided by Anomalib is unreliable and constrained. While YOLOv8 offers enhanced support for video datasets, its capabilities in anomaly detection datasets remain limited in comparison to those of Anomalib. Yolov8 is primarily based on supervised learning, whereas the models in Anomalib are predominantly based on unsupervised learning. The dataset format of YOLOv8 and Anomalib is markedly disparate. The YOLOv8 dataset necessitates labeling, whereas the Anomalib dataset does not require labeling. Only a select subset of Anomalib models, including "DRAEM, DSR, GANomaly," require masked images, which can be regarded as labeled data for training [1, 2, 3].

6.3 Future Work

At the present time, the inference test is capable of yielding two distinct results. The initial approach is to utilise the IADBE to train a model at the local level. This model, typically represented by a xxx.pt file, is then executed via the Anomalib predict command. This approach is relatively time-consuming. The second approach involves directly downloading a pre-trained model from the our Huggingface models[161] repository

and subsequently executing the Anomalib predict command manually[1]. This approach saves a considerable amount of time that would otherwise be spent on training. However, these two methods are not user-friendly for users who lack the requisite programming knowledge. To address this issue, the existing Gradio can be employed to provide a solution. Gradio has integrated backend and frontend technologies, and with the Python API, an interactive user interface with a simple UI design can be constructed for users[162].

The utilisation of our backend Spring Boot with Kotlin and frontend AngularJS is scalable and flexible. The file processing logic in the backend can be tailored according to specific requirements, and the choice of charts can be made on a granular level. However, this approach may prove to be somewhat complex and, to some extent, less efficient. The existing Gradio has integrated backend and frontend technologies. The Python API enables the construction of an interactive user interface with a user interface design. While Streamlit is more suitable for projects requiring advanced customisation and interactive dashboards with various integration possibilities, Gradio is better suited for simpler, interactive UI-focused applications with quicker development cycles. One of the most notable attributes of Streamlit is its capacity to expedite the development of dashboards. Developers are able to rapidly transform their Python scripts into interactive web applications, negating the necessity for extensive front-end development expertise. This streamlined process facilitates efficient prototyping and iteration, thereby conserving valuable time within the development cycle[162, 163]. Consequently, Streamlit represents an alternative methodology to our SpringBoot backend and AngularJS frontend[140, 149].

With the completion of our benchmark engine, IADBE, we are now in a position to proceed. The backend process files encompass a range of data formats, including raw logs, LaTeX tables, and CSV files, among others. The frontend is responsible for visualizing data returned by the backend. It should be noted, however, that the data displayed on our visualization platform is not updated in real time with the files generated by IADBE. If we were to enable the frontend to control the training and testing phases, the real-time training logs could be displayed on the frontend and stored in the database. This would allow the backend to retrieve the latest information about files in the database and process them. With these additional functionalities, the platform would be accessible to not only AI researchers but also users with limited programming knowledge.

The IADBE evaluated models that demonstrated expertise in specific categories of a dataset. It was determined that the Multi-class Unified AD models, which utilize a single model for anomaly detection across diverse categories within the entire dataset, warranted further assessment.

References

[1] Samet Akcay et al. “Anomalib: A deep learning library for anomaly detection”. In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2022, pp. 1706–1710. URL: https://ieeexplore.ieee.org/abstract/document/9897283.

[2] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. “A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas”. In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1680–1716.

[3] *Ultralytics Documentation*. https://docs.ultralytics.com/. Accessed: 2024-08-17.

[4] *Docker: Accelerated, Containerized Application Development*. https://www.docker.com/. Accessed: 2024-08-17.

[5] Docker. *Docker Compose Documentation*. Accessed: 2024-08-21. 2024. URL: https://docs.docker.com/compose/.

[6] Paul Bergmann et al. “MVTec AD – A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019. URL: https://openaccess.thecvf.com/content_CVPR_2019/html/Bergmann_MVTec_AD_--_A_Comprehensive_Real-World_Dataset_for_Unsupervised_Anomaly_CVPR_2019_paper.html.

[7] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

[8] A. Vidhya. *ResNet: Understand and Implement from Scratch*. https://medium.com/analytics-vidhya/resnet-understand-and-implement-from-scratch-d0eb9725e0db. Accessed: 2024-05-13. 2020.

[9] Guoyang Xie et al. *IM-IAD: Industrial Image Anomaly Detection Benchmark in Manufacturing*. 2024. arXiv: 2301.13359 [cs.CV]. URL: https://ieeexplore.ieee.org/abstract/document/10443076.

[10] Lars Heckler, Rebecca König, and Paul Bergmann. “Exploring the Importance of Pretrained Feature Extractors for Unsupervised Anomaly Detection and Localization”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2917–2926. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00293. URL: https://ieeexplore.ieee.org/document/10208766/ (visited on 03/04/2024).

[11] Anomalib Developers. *SDD: Developer Guide*. https://anomalib.readthedocs.io/en/latest/markdown/guides/developer/sdd.html. Accessed: 2024-07-10. 2024.

[12] ONNX. *ONNX: Open Neural Network Exchange*. Accessed: 2024-08-21. 2024. URL: https://onnx.ai/.

[13] Intel Corporation. *OpenVINO Documentation*. Accessed: 2024-08-21. 2024. URL: https://docs.openvino.ai/2024/index.html.

[14] Baeldung. *Understanding the YOLO Algorithm for Object Detection*. Accessed: 2024-08-21. 2024. URL: https://www.baeldung.com/cs/yolo-algorithm.

[15] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[16] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[17] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[18] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[19] Chuyi Li et al. "Yolov6 v3. 0: A full-scale reloading". In: *arXiv preprint arXiv:2301.05586* (2023).

[20] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 7464–7475.

[21] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. "Yolov9: Learning what you want to learn using programmable gradient information". In: *arXiv preprint arXiv:2402.13616* (2024).

[22] Ultralytics. *Ultralytics Datasets Documentation*. https://docs.ultralytics.com/datasets. Accessed: 21 July 2024. 2024.

[23] Hanqiu Deng and Xingyu Li. "Anomaly Detection via Reverse Distillation From One-Class Embedding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 9737–9746. URL: https://openaccess.thecvf.com/content/CVPR2022/html/Deng_Anomaly_Detection_via_Reverse_Distillation_From_One-Class_Embedding_CVPR_2022_paper.html.

[24] Tran Dinh Tien et al. "Revisiting Reverse Distillation for Anomaly Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 24511–24520. URL: https://openaccess.thecvf.com/content/CVPR2023/html/Tien_Revisiting_Reverse_Distillation_for_Anomaly_Detection_CVPR_2023_paper.html.

[25] Zhikang Liu et al. "SimpleNet: A Simple Network for Image Anomaly Detection and Localization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 20402–20411. URL: https://openaccess.thecvf.com/content/CVPR2023/html/Liu_SimpleNet_A_Simple_Network_for_Image_Anomaly_Detection_and_Localization_CVPR_2023_paper.html.

[26] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. "CFLOW-AD: Real-Time Unsupervised Anomaly Detection With Localization via Conditional Normalizing Flows". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2022, pp. 98–107. URL: https://openaccess.thecvf.com/content/WACV2022/html/Gudovskiy_CFLOW-AD_Real-Time_Unsupervised_Anomaly_Detection_With_Localization_via_Conditional_Normalizing_WACV_2022_paper.html.

[27] Jiarui Lei et al. “PyramidFlow: High-Resolution Defect Contrastive Localization Using Pyramid Normalizing Flow”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 14143–14152. URL: https://openaccess.thecvf.com/content/CVPR2023/html/Lei_PyramidFlow_High-Resolution_Defect_Contrastive_Localization_Using_Pyramid_Normalizing_Flow_CVPR_2023_paper.html.

[28] Karsten Roth et al. *Towards Total Recall in Industrial Anomaly Detection*. 2021. arXiv: 2106.08265 [cs.CV].

[29] Jaehyeok Bae, Jae-Han Lee, and Seyun Kim. “PNI : Industrial Anomaly Detection using Position and Neighborhood Information”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 6373–6383. URL: https://openaccess.thecvf.com/content/ICCV2023/html/Bae_PNI__Industrial_Anomaly_Detection_using_Position_and_Neighborhood_Information_ICCV_2023_paper.html.

[30] Vitjan Zavrtanik, Matej Kristan, and Danijel Skocaj. “DRAEM - A Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 8330–8339. URL: https://openaccess.thecvf.com/content/ICCV2021/html/Zavrtanik_DRAEM_-_A_Discriminatively_Trained_Reconstruction_Embedding_for_Surface_Anomaly_ICCV_2021_paper.html.

[31] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. “DSR – A Dual Subspace Re-Projection Network for Surface Anomaly Detection”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 539–554. ISBN: 978-3-031-19821-2. URL: https://link.springer.com/chapter/10.1007/978-3-031-19821-2_31.

[32] Yufei Liang et al. “Omni-frequency Channel-selection Representations for Unsupervised Anomaly Detection”. In: *arXiv preprint arXiv:2203.00259* (2022). URL: https://ieeexplore.ieee.org/abstract/document/10192551.

[33] Chaoqin Huang et al. “Registration Based Few-Shot Anomaly Detection”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 303–319. ISBN: 978-3-031-20053-3. URL: https://link.springer.com/chapter/10.1007/978-3-031-20053-3_18.

[34] Zhaopeng Gu et al. “AnomalyGPT: Detecting Industrial Anomalies using Large Vision-Language Models”. In: *arXiv preprint arXiv:2308.15366* (2024). URL: https://ojs.aaai.org/index.php/AAAI/article/view/27963.

[35] Choubo Ding, Guansong Pang, and Chunhua Shen. “Catching Both Gray and Black Swans: Open-set Supervised Anomaly Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022. URL: https://openaccess.thecvf.com/content/CVPR2022/html/Ding_Catching_Both_Gray_and_Black_Swans_Open-Set_Supervised_Anomaly_Detection_CVPR_2022_paper.html.

[36] Xincheng Yao et al. “Explicit Boundary Guided Semi-Push-Pull Contrastive Learning for Supervised Anomaly Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 24490–24499. URL: https://openaccess.thecvf.com/content/CVPR2023/html/Yao_Explicit_Boundary_Guided_Semi-Push-Pull_Contrastive_Learning_for_Supervised_Anomaly_Detection_CVPR_2023_paper.html.

[37] Yuanhong Chen et al. “Deep one-class classification via interpolated gaussian descriptor”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. 2022, pp. 383–392. URL: https://ojs.aaai.org/index.php/AAAI/article/view/19915.

[38] Xi Jiang et al. “SoftPatch: Unsupervised Anomaly Detection with Noisy Data”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 15433–15445. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/637a456d89289769ac1ab29617ef7213-Paper-Conference.pdf.

[39] Declan McIntosh and Alexandra Branzan Albu. *Inter-Realization Channels: Unsupervised Anomaly Detection in Images Beyond One-Class Classification*. Oct. 2023. URL: https://openaccess.thecvf.com/content/ICCV2023/html/McIntosh_Inter-Realization_Channels_Unsupervised_Anomaly_Detection_Beyond_One-Class_Classification_ICCV_2023_paper.html.

[40] Jiaqi Liu et al. *Unsupervised Continual Anomaly Detection with Contrastively-learned Prompt*. 2024. arXiv: 2401.01010 [cs.CV].

[41] Zhiyuan You et al. “A Unified Model for Multi-class Anomaly Detection”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 4571–4584. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/1d774c112926348c3e25ea47d87c835b-Paper-Conference.pdf.

[42] Yue Wang et al. “Multimodal Industrial Anomaly Detection via Hybrid Fusion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8032–8041. URL: https://openaccess.thecvf.com/content/CVPR2023/html/Wang_Multimodal_Industrial_Anomaly_Detection_via_Hybrid_Fusion_CVPR_2023_paper.html.

[43] Jiaqi Liu et al. *Real3D-AD: A Dataset of Point Cloud Anomaly Detection*. 2023. arXiv: 2309.13226 [cs.CV].

[44] Hanqiu Deng et al. “AnoVL: Adapting Vision-Language Models for Unified Zero-shot Anomaly Localization”. In: *arXiv preprint arXiv:2308.15939* (2023). URL: https://arxiv.org/abs/2308.15939.

[45] Yunkang Cao et al. “Segment Any Anomaly without Training via Hybrid Prompt Regularization”. In: arXiv:2305.10724 (May 18, 2023). arXiv: 2305.10724[cs]. URL: http://arxiv.org/abs/2305.10724 (visited on 05/19/2023).

[46] Yujin Lee et al. *UniFormaly: Towards Task-Agnostic Unified Framework for Visual Anomaly Detection*. 2023. arXiv: 2307.12540 [cs.CV].

[47] M-3LAB. *Awesome Industrial Anomaly Detection*. https://github.com/M-3LAB/awesome-industrial-anomaly-detection. Accessed: 24 July 2024. 2024.

[48] S. Amit. *Everything You Need to Know About Knowledge Distillation (aka Teacher-Student Model)*. https://amit-s.medium.com/everything-you-need-to-know-about-knowledge-distillation-aka-teacher-student-model-d6ee10fe7276. Accessed: 2024-06-18. 2023.

[49] *Anomalib GitHub Discussions: Issue #2183*. https://github.com/openvinotoolkit/anomalib/discussions/2183. Accessed: 2024-08-17.

[50] Zhihao Gu et al. “Remembering Normality: Memory-guided Knowledge Distillation for Unsupervised Anomaly Detection”. en. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 16355–16363. ISBN: 9798350307184. DOI: 10.1109/ICCV51070.2023.01503. URL: https://ieeexplore.ieee.org/document/10376650/ (visited on 03/01/2024).

[51] Xuan Zhang et al. *DeSTSeg: Segmentation Guided Denoising Student-Teacher for Anomaly Detection*. Mar. 21, 2023. arXiv: 2211.11317[cs]. URL: http://arxiv.org/abs/2211.11317.

[52] David Tax. *One-Class Classification*. https://www.tudelft.nl/ewi/over-de-faculteit/afdelingen/intelligent-systems/pattern-recognition-bioinformatics/pattern-recognition-bioinformatics/people/david-tax/one-class-classification. Accessed: 2024-06-18.

[53] Tri Cao, Jiawen Zhu, and Guansong Pang. *Anomaly Detection under Distribution Shift*. 2023. arXiv: 2303.13845 [cs.CV].

[54] Chengkan Lv et al. "Unsupervised Automatic Defect Inspection based on Image Matching and Local One-class Classification". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4435–4444. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00466. URL: https://ieeexplore.ieee.org/document/10208891/ (visited on 03/01/2024).

[55] JunKyu Jang, Eugene Hwang, and Sung-Hyuk Park. "N-pad : Neighboring Pixel-based Industrial Anomaly Detection". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4365–4374. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00459. URL: https://ieeexplore.ieee.org/document/10208749/ (visited on 03/01/2024).

[56] Matej Grcić, Josip Šarić, and Siniša Šegvić. "On Advantages of Mask-level Recognition for Outlier-aware Segmentation". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2937–2947. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00295. URL: https://ieeexplore.ieee.org/document/10208895/ (visited on 03/05/2024).

[57] Daehyun Kim, Sungyong Baik, and Tae Hyun Kim. "SANFlow: Semantic-Aware Normalizing Flow for Anomaly Detection". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=BqZ70BEtuW.

[58] Haitian He et al. "Learning Transferable Representations for Image Anomaly Localization Using Dense Pretraining". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 1113–1122. URL: https://openaccess.thecvf.com/content/WACV2024/html/He_Learning_Transferable_Representations_for_Image_Anomaly_Localization_Using_Dense_Pretraining_WACV_2024_paper.html.

[59] André Luiz Vieira e Silva et al. "Attention Modules Improve Image-Level Anomaly Detection for Industrial Inspection: A DifferNet Case Study". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 8246–8255. URL: https://openaccess.thecvf.com/content/WACV2024/html/Vieira_e_Silva_Attention_Modules_Improve_Image-Level_Anomaly_Detection_for_Industrial_Inspection_A_WACV_2024_paper.html.

[60] Jeeho Hyun et al. "ReConPatch: Contrastive Patch Representation Learning for Industrial Anomaly Detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 2052–2061. URL: https://openaccess.thecvf.com/content/WACV2024/html/Hyun_ReConPatch_Contrastive_Patch_Representation_Learning_for_Industrial_Anomaly_Detection_WACV_2024_paper.html.

[61] Hongzuo Xu et al. “Fascinating Supervisory Signals and Where to Find Them: Deep Anomaly Detection with Scale Learning”. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 38655–38673. URL: https://proceedings.mlr.press/v202/xu23p.html.

[62] Teng-Yok Lee, Yusuke Nagai, and Akira Minezawa. “Memory-efficient and GPU-oriented visual anomaly detection with incremental dimension reduction”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 1–9. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00292. URL: https://ieeexplore.ieee.org/document/10208330/ (visited on 03/04/2024).

[63] Woosang Shin et al. “Anomaly Detection using Score-based Perturbation Resilience”. en. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 23315–23325. ISBN: 9798350307184. DOI: 10.1109/ICCV51070.2023.02136. URL: https://ieeexplore.ieee.org/document/10376577/ (visited on 03/01/2024).

[64] Ximiao Zhang, Min Xu, and Xiuzhuang Zhou. *RealNet: A Feature Selection Network with Realistic Synthetic Anomaly for Anomaly Detection*. 2024. arXiv: 2403.05897 [cs.CV].

[65] Xincheng Yao et al. “Focus the Discrepancy: Intra- and Inter-Correlation Learning for Image Anomaly Detection”. en. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 6780–6790. ISBN: 9798350307184. DOI: 10.1109/ICCV51070.2023.00626. URL: https://ieeexplore.ieee.org/document/10378024/ (visited on 03/01/2024).

[66] Sangwoong Yoon et al. “Energy-Based Models for Anomaly Detection: A Manifold Diffusion Recovery Approach”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 49445–49466. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/9b6d7202750e8e32cd5270eb7fc131f7-Paper-Conference.pdf.

[67] Tiange Xiang et al. *SQUID: Deep Feature In-Painting for Unsupervised Anomaly Detection*. Mar. 24, 2023. URL: http://arxiv.org/abs/2111.13495 (visited on 03/07/2024).

[68] Wenrui Liu et al. *Diversity-Measurable Anomaly Detection*. Mar. 9, 2023. arXiv: 2303.05047[cs]. URL: http://arxiv.org/abs/2303.05047.

[69] Xinyi Zhang et al. “Unsupervised Surface Anomaly Detection with Diffusion Probabilistic Model”. en. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 6759–6768. ISBN: 9798350307184. DOI: 10.1109/ICCV51070.2023.00624. URL: https://ieeexplore.ieee.org/document/10377534/ (visited on 03/01/2024).

[70] Fanbin Lu et al. “Removing Anomalies as Noises for Industrial Defect Localization”. en. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 16120–16129. ISBN: 9798350307184. DOI: 10.1109/ICCV51070.2023.01481. URL: https://ieeexplore.ieee.org/document/10378530/ (visited on 03/01/2024).

[71] Weizhi Liu et al. “Assigned MURA Defect Generation Based on Diffusion Model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4395–4402. URL: https://openaccess.thecvf.com/content/CVPR2023W/VISION/html/Liu_Assigned_MURA_Defect_Generation_Based_on_Diffusion_Model_CVPRW_2023_paper.html.

[72] Alvaro Gonzalez-Jimenez et al. “SANO: Score-based Diffusion Model for Anomaly Localization in Dermatology”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2988–2994. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00300. URL: https://ieeexplore.ieee.org/document/10208839/ (visited on 03/03/2024).

[73] Mark S. Graham et al. “Denoising diffusion models for out-of-distribution detection”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2948–2957. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00296. URL: https://ieeexplore.ieee.org/document/10208320/ (visited on 03/04/2024).

[74] Alessandro Flaborea et al. “Are we certain it’s anomalous?” In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2897–2907. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00291. URL: https://ieeexplore.ieee.org/document/10208577/ (visited on 03/05/2024).

[75] Ziyi Yang, Iman Soltani, and Eric Darve. “Anomaly Detection with Domain Adaptation”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2958–2967. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00297. URL: https://ieeexplore.ieee.org/document/10208453/ (visited on 03/03/2024).

[76] Jia Guo et al. “ReContrast: Domain-Specific Anomaly Detection via Contrastive Reconstruction”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 10721–10740. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/228b9279ecf9bbafe582406850c57115-Paper-Conference.pdf.

[77] Zheng Fang et al. “FastRecon: Few-shot Industrial Anomaly Detection via Fast Feature Reconstruction”. en. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 17435–17444. ISBN: 9798350307184. DOI: 10.1109/ICCV51070.2023.01603. URL: https://ieeexplore.ieee.org/document/10378185/ (visited on 03/01/2024).

[78] Xian Yeow Lee et al. “XDNet: A Few-Shot Meta-Learning Approach for Cross-Domain Visual Inspection”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4375–4384. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00460. URL: https://ieeexplore.ieee.org/document/10208767/ (visited on 03/01/2024).

[79] Niamh Belton et al. “FewSOME: One-Class Few Shot Anomaly Detection with Siamese Networks”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2978–2987. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00299. URL: https://ieeexplore.ieee.org/document/10208830/ (visited on 03/04/2024).

[80] Hanxi Li et al. *Efficient Anomaly Detection with Budget Annotation Using Semi-Supervised Residual Transformer*. June 6, 2023. arXiv: 2306.03492[cs]. URL: http://arxiv.org/abs/2306.03492 (visited on 03/06/2024).

[81] Liang Xu, Han Zou, and Takayuki Okatani. “How Do Label Errors Affect Thin Crack Detection by DNNs”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4414–4423. URL: https://openaccess.thecvf.com/content/CVPR2023W/VISION/html/Xu_How_Do_Label_Errors_Affect_Thin_Crack_Detection_by_DNNs_CVPRW_2023_paper.html.

[82] Wenbo Sun et al. “A continual learning framework for adaptive defect classification and inspection”. In: *Journal of Quality Technology* 55.5 (2023), pp. 598–614.

[83] Wujin Li et al. “Towards continual adaptation in industrial anomaly detection”. In: *Proceedings of the 30th ACM International Conference on Multimedia*. 2022, pp. 2871–2880. URL: https://dl.acm.org/doi/abs/10.1145/3503161.3548232.

[84] Jiaqi Tang et al. *An Incremental Unified Framework for Small Defect Inspection*. 2024. arXiv: 2312.08917 [cs.CV]. URL: https://arxiv.org/abs/2312.08917.

[85] Han Gao et al. *Towards Total Online Unsupervised Anomaly Detection and Localization in Industrial Vision*. 2023. arXiv: 2305.15652 [cs.CV]. URL: https://arxiv.org/abs/2305.15652.

[86] Ruotian Lu et al. “Hierarchical Vector Quantized Transformer for Multi-class Unsupervised Anomaly Detection”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/1abc87c67cc400a67b869358e627Abstract-Conference.html.

[87] Haoyang He et al. *DiAD: A Diffusion-based Framework for Multi-class Anomaly Detection*. en. arXiv:2312.06607 [cs]. Dec. 2023. URL: http://arxiv.org/abs/2312.06607 (visited on 03/01/2024).

[88] Ying Zhao. “OmniAL: A Unified CNN Framework for Unsupervised Anomaly Localization”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Vancouver, BC, Canada: IEEE, June 2023, pp. 3924–3933. ISBN: 9798350301298. DOI: 10.1109/CVPR52729.2023.00382. URL: https://ieeexplore.ieee.org/document/10204529/ (visited on 03/05/2024).

[89] Aodong Li et al. “Zero-Shot Anomaly Detection via Batch Normalization”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 40963–40993. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/8078e8c3055303a884ffae2d3ea00338-Paper-Conference.pdf.

[90] Yiting Li et al. “PromptAD: Zero-Shot Anomaly Detection Using Text Prompts”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 1093–1102. URL: https://openaccess.thecvf.com/content/WACV2024/html/Li_PromptAD_Zero-Shot_Anomaly_Detection_Using_Text_Prompts_WACV_2024_paper.html.

[91] Andrei-Timotei Ardelean and Tim Weyrich. “High-Fidelity Zero-Shot Texture Anomaly Localization Using Feature Correspondence Analysis”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 1134–1144. URL: https://openaccess.thecvf.com/content/WACV2024/html/Ardelean_High-Fidelity_Zero-Shot_Texture_Anomaly_Localization_Using_Feature_Correspondence_Analysis_WACV_2024_paper.html.

[92] Jongheon Jeong et al. *WinCLIP: Zero-/Few-Shot Anomaly Classification and Segmentation*. Mar. 26, 2023. arXiv: 2303.14814[cs]. URL: http://arxiv.org/abs/2303.14814 (visited on 03/06/2024).

[93] Xuhai Chen, Yue Han, and Jiangning Zhang. *APRIL-GAN: A Zero-/Few-Shot Anomaly Classification and Segmentation Method for CVPR 2023 VAND Workshop Challenge Tracks 1&2: 1st Place on Zero-shot AD and 4th Place on Few-shot AD*. Oct. 11, 2023. arXiv: 2305.17382[cs]. URL: http://arxiv.org/abs/2305.17382 (visited on 03/07/2024).

[94] Xiaomeng Zhu et al. "Towards Sim-to-Real Industrial Parts Classification with Synthetic Dataset". en. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Vancouver, BC, Canada: IEEE, June 2023, pp. 4454–4463. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00468. URL: https://ieeexplore.ieee.org/document/10208421/ (visited on 03/01/2024).

[95] Juraj Fulir et al. "Synthetic Data for Defect Segmentation on Complex Metal Surfaces". en. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Vancouver, BC, Canada: IEEE, June 2023, pp. 4424–4434. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00465. URL: https://ieeexplore.ieee.org/document/10208376/ (visited on 03/01/2024).

[96] Qiang Zhou et al. "PAD: A Dataset and Benchmark for Pose-agnostic Anomaly Detection". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 44558–44571. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/8bc5aef775aacc1650a9790f1428bcea-Paper-Datasets_and_Benchmarks.pdf.

[97] Teng Hu et al. *AnomalyDiffusion: Few-Shot Anomaly Image Generation with Diffusion Model*. en. arXiv:2312.05767 [cs]. Feb. 2024. URL: http://arxiv.org/abs/2312.05767 (visited on 03/01/2024).

[98] Lingrui Zhang et al. "What Makes a Good Data Augmentation for Few-Shot Unsupervised Image Anomaly Detection?" In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4345–4354. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00457. URL: https://ieeexplore.ieee.org/document/10208832/ (visited on 03/01/2024).

[99] Faranak Shamsafar et al. "Leveraging Multi-view Data for Improved Detection Performance: An Industrial Use Case". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4464–4471. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00469. URL: https://ieeexplore.ieee.org/document/10208794/ (visited on 03/01/2024).

[100] Yizhou Jin et al. "Glass Wool Defect Detection Using an Improved YOLOv5". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4385–4394. URL: https://openaccess.thecvf.com/content/CVPR2023W/VISION/html/Jin_Glass_Wool_Defect_Detection_Using_an_Improved_YOLOv5_CVPRW_2023_paper.html.

[101] Jing Wei et al. "Diversified and Multi-Class Controllable Industrial Defect Synthesis for Data Augmentation and Transfer". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4445–4453. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00467. URL: https://ieeexplore.ieee.org/document/10208648/ (visited on 03/01/2024).

[102] Yunkang Cao et al. *Towards Generic Anomaly Detection and Understanding: Large-scale Visual-linguistic Model (GPT-4V) Takes the Lead*. en. arXiv:2311.02782 [cs]. Nov. 2023. URL: http://arxiv.org/abs/2311.02782 (visited on 03/01/2024).

[103] Yuanze Li et al. *Myriad: Large Multimodal Model by Applying Vision Experts for Industrial Anomaly Detection*. en. arXiv:2310.19070 [cs]. Oct. 2023. URL: http://arxiv.org/abs/2310.19070 (visited on 03/01/2024).

[104] Jiangning Zhang et al. *Exploring Grounding Potential of VQA-oriented GPT-4V for Zero-shot Anomaly Detection*. en. arXiv:2311.02612 [cs]. Nov. 2023. URL: http://arxiv.org/abs/2311.02612 (visited on 03/01/2024).

[105] Hewei Guo et al. "Template-guided Hierarchical Feature Restoration for Anomaly Detection". en. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 6424–6435. ISBN: 9798350307184. DOI: 10.1109/ICCV51070.2023.00593. URL: https://ieeexplore.ieee.org/document/10377896/ (visited on 03/01/2024).

[106] Songmin Dai et al. *Generating and Reweighting Dense Contrastive Patterns for Unsupervised Anomaly Detection*. en. arXiv:2312.15911 [cs]. Dec. 2023. URL: http://arxiv.org/abs/2312.15911 (visited on 03/01/2024).

[107] Soopil Kim et al. *Few Shot Part Segmentation Reveals Compositional Logic for Industrial Anomaly Detection*. en. arXiv:2312.13783 [cs]. Dec. 2023. URL: http://arxiv.org/abs/2312.13783 (visited on 03/01/2024).

[108] Kilian Batzner, Lars Heckler, and Rebecca König. "EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 128–138. URL: https://openaccess.thecvf.com/content/WACV2024/html/Batzner_EfficientAD_Accurate_Visual_Anomaly_Detection_at_Millisecond-Level_Latencies_WACV_2024_paper.html.

[109] Jie Zhang, Masanori Suganuma, and Takayuki Okatani. "Contextual Affinity Distillation for Image Anomaly Detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 149–158. URL: https://openaccess.thecvf.com/content/WACV2024/html/Zhang_Contextual_Affinity_Distillation_for_Image_Anomaly_Detection_WACV_2024_paper.html.

[110] Yu-Min Chu et al. "Shape-guided dual-memory learning for 3D anomaly detection". In: (2023). URL: https://openreview.net/forum?id=IkSGn9fcPz.

[111] Ruitao Chen et al. *EasyNet: An Easy Network for 3D Industrial Anomaly Detection*. en. arXiv:2307.13925 [cs]. Aug. 2023. URL: http://arxiv.org/abs/2307.13925 (visited on 03/01/2024).

[112] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. "Cheating Depth: Enhancing 3D Surface Anomaly Detection via Depth Simulation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 2164–2172. URL: https://openaccess.thecvf.com/content/WACV2024/html/Zavrtanik_Cheating_Depth_Enhancing_3D_Surface_Anomaly_Detection_via_Depth_Simulation_WACV_2024_paper.html.

[113] Alexander Naumann et al. "Parcel3D: Shape Reconstruction from Single RGB Images for Applications in Transportation Logistics". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4403–4413. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00463. URL: https://ieeexplore.ieee.org/document/10208450/ (visited on 03/01/2024).

[114] Eliahu Horwitz and Yedid Hoshen. “Back to the Feature: Classical 3D Features are (Almost) All You Need for 3D Anomaly Detection”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2968–2977. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00298. URL: https://ieeexplore.ieee.org/document/10208423/ (visited on 03/05/2024).

[115] Guangyu Ren et al. “Towards Automated Polyp Segmentation Using Weakly- and Semi-Supervised Learning and Deformable Transformers”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 4355–4364. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00458. URL: https://ieeexplore.ieee.org/document/10208981/ (visited on 03/01/2024).

[116] Yona Falinie A. Gaus et al. “Region-based Appearance and Flow Characteristics for Anomaly Detection in Infrared Surveillance Imagery”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2995–3005. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00301. URL: https://ieeexplore.ieee.org/document/10208786/ (visited on 03/03/2024).

[117] Li-Ling Chiu and Shang-Hong Lai. “Self-Supervised Normalizing Flows for Image Anomaly Detection and Localization”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2927–2936. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00294. URL: https://ieeexplore.ieee.org/document/10208652/ (visited on 03/04/2024).

[118] Mohammad Baradaran and Robert Bergevin. “Multi-Task Learning based Video Anomaly Detection with Attention”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 2886–2896. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00290. URL: https://ieeexplore.ieee.org/document/10208994/ (visited on 03/04/2024).

[119] Hui Zhang et al. *Prototypical Residual Networks for Anomaly Detection and Localization*. Apr. 18, 2023. arXiv: 2212.02031[cs]. URL: http://arxiv.org/abs/2212.02031 (visited on 03/06/2024).

[120] Paul Bergmann et al. “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*. SCITEPRESS - Science and Technology Publications, 2022. DOI: 10.5220/0010865000003124. URL: http://dx.doi.org/10.5220/0010865000003124.

[121] Pankaj Mishra et al. “VT-ADL: A Vision Transformer Network for Image Anomaly Detection and Localization”. In: *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. 2021, pp. 01–06. DOI: 10.1109/ISIE45552.2021.9576231.

[122] Domen Tabernik et al. “Segmentation-Based Deep-Learning Approach for Surface-Defect Detection”. In: *Journal of Intelligent Manufacturing* (May 2019). ISSN: 1572-8145. DOI: 10.1007/s10845-019-01476-x. URL: https://link.springer.com/article/10.1007/s10845-019-01476-x.

[123] Yang Zou et al. “SPot-the-Difference Self-supervised Pre-training for Anomaly Detection and Segmentation”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 392–408. ISBN: 978-3-031-20056-4. URL: https://link.springer.com/chapter/10.1007/978-3-031-20056-4_23.

[124] Marco Rudolph et al. “Fully Convolutional Cross-Scale-Flows for Image-Based Defect Detection”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2022, pp. 1088–1097. URL: https://scholar.google.com/scholar?hl=zh-CN&as_sdt=0%2C5&q=Fully+Convolutional+Cross-Scale-Flows+for+Image-Based+Defect+Detection&btnG=.

[125] Nilesh A. Ahuja et al. *Probabilistic Modeling of Deep Features for Out-of-Distribution and Adversarial Detection*. 2019. arXiv: 1909.11786 [stat.ML].

[126] Jiawei Yu et al. *FastFlow: Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows*. 2021. arXiv: 2111.07677 [cs.CV].

[127] Philip Adey et al. “Region Based Anomaly Detection with Real-Time Training and Analysis”. In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 2019, pp. 495–499. DOI: 10.1109/ICMLA.2019.00092.

[128] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. “GANomaly: Semi-supervised Anomaly Detection via Adversarial Training”. In: *Computer Vision – ACCV 2018*. Ed. by C. V. Jawahar et al. Cham: Springer International Publishing, 2019, pp. 622–637. ISBN: 978-3-030-20893-6. URL: https://link.springer.com/chapter/10.1007/978-3-030-20893-6_39.

[129] Ibrahima Ndiour et al. “FRE: A Fast Method For Anomaly Detection And Segmentation”. In: *arXiv preprint arXiv:2211.12650* (2022).

[130] Thomas Defard et al. “PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization”. In: *Pattern Recognition. ICPR International Workshops and Challenges*. Ed. by Alberto Del Bimbo et al. Cham: Springer International Publishing, 2021, pp. 475–489. ISBN: 978-3-030-68799-1. URL: https://link.springer.com/chapter/10.1007/978-3-030-68799-1_35.

[131] Sungwook Lee, Seunghyun Lee, and Byung Cheol Song. “CFA: Coupled-Hypersphere-Based Feature Adaptation for Target-Oriented Anomaly Localization”. In: *IEEE Access* 10 (2022), pp. 78446–78454. DOI: 10.1109/ACCESS.2022.3193699.

[132] Guodong Wang et al. *Student-Teacher Feature Pyramid Matching for Anomaly Detection*. 2021. arXiv: 2103.04257 [cs.CV].

[133] Matías Tailanian, Álvaro Pardo, and Pablo Musé. *U-Flow: A U-shaped Normalizing Flow for Anomaly Detection with Unsupervised Threshold*. 2023. arXiv: 2211.12353 [cs.CV].

[134] Sungwoo Lee. *CFA: A framework for Anomaly Localization*. https://github.com/sungwool/cfa_for_anomaly_localization. Accessed: 21 May 2024. 2023.

[135] Denis Gudovskiy. *CFLOW-AD: Conditional Normalizing Flow for Anomaly Detection*. https://github.com/gudovskiy/cflow-ad. Accessed: 21 May 2024. 2023.

[136] Marco Rudolph. *CS-Flow: Anomalous Image Detection and Localization*. https://github.com/marco-rudolph/cs-flow. Accessed: 21 May 2024. 2023.

[137] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[138] Thierry Gautier. *FastFlow: A Fast Normalizing Flow for Anomaly Detection*. https://github.com/gathierry/FastFlow. Accessed: 21 May 2024. 2023.

[139] Matías Tailanian. *U-Flow: A U-shaped Normalizing Flow for Anomaly Detection*. https://github.com/mtailanian/uflow. Accessed: 21 May 2024. 2023.

[140] *Spring: The Source of Modern Java*. https://spring.io/. Accessed: 2024-08-17.

[141] *Kotlin Programming Language*. https://kotlinlang.org/. Accessed: 2024-08-17.

[142] *Project Jupyter*. https://jupyter.org/. Accessed: 2024-08-17.

[143] *pandas: Python Data Analysis Library*. https://pandas.pydata.org/. Accessed: 2024-08-17.

[144] *NumPy*. https://numpy.org/. Accessed: 2024-08-17.

[145] *scikit-learn: Machine Learning in Python*. https://scikit-learn.org/stable/. Accessed: 2024-08-17.

[146] *PyTorch*. https://pytorch.org/. Accessed: 2024-08-17.

[147] *Matplotlib: Visualization with Python*. https://matplotlib.org/. Accessed: 2024-08-17.

[148] *Python Programming Language*. https://www.python.org/. Accessed: 2024-08-17.

[149] *AngularJS — Superheroic JavaScript MVW Framework*. https://angularjs.org/. Accessed: 2024-08-17.

[150] *D3.js - Data-Driven Documents*. https://d3js.org/. Accessed: 2024-08-17.

[151] *Chart.js: Simple yet flexible JavaScript charting for designers & developers*. https://www.chartjs.org/. Accessed: 2024-08-17.

[152] *Apache ECharts: A Declarative Framework for Rapid Construction of Web-based Visualizations*. https://echarts.apache.org/en/index.html. Accessed: 2024-08-17.

[153] *Vue.js: The Progressive JavaScript Framework*. https://vuejs.org/. Accessed: 2024-08-17.

[154] *React: A JavaScript library for building user interfaces*. https://react.dev/. Accessed: 2024-08-17.

[155] Ultralytics Inc. *YOLO Performance Metrics*. https://docs.ultralytics.com/guides/yolo-performance-metrics/. Accessed: 2024-08-21. 2023.

[156] Ultralytics. *Ultralytics YOLOv8 Documentation: Predict Mode*. Accessed: 2024-07-19. Ultralytics. 2024.

[157] Ultralytics. *Post-Processing and Non-Maximum Suppression in YOLOv8*. https://github.com/ultralytics/ultralytics/issues/4149. Accessed: 2024-07-31. 2023.

[158] Ultralytics. *Ultralytics YOLOv8 Documentation: Performance Metrics*. Accessed: 2024-07-10. Ultralytics. 2024.

[159] OpenVINO Toolkit. *Issue #2111: Error while training stfpm on MVTec-3D-AD dataset*. https://github.com/openvinotoolkit/anomalib/issues/2111. Accessed: 2024-07-13. 2024.

[160] *Nebular: Angular UI Library*. https://akveo.github.io/nebular/. Accessed: 2024-08-17.

[161] *Hugging Face: The Hub for Machine Learning Models*. https://huggingface.co/. Accessed: 2024-08-17.

[162] *Gradio: Create UIs for Your Machine Learning Models*. https://www.gradio.app/. Accessed: 2024-08-17.

[163] *Streamlit: The Fastest Way to Build Data Apps*. https://streamlit.io/. Accessed: 2024-08-17.

7 Appendix

7.1 Explanation of Abbreviation

Table 7.1

Explanation of Abbreviation

shortcut	explanation
AUROC	Area under ROC Curve
PRO	Per Region Overlap
AUPR	Aarea under the PR curve
CFA	Coupled Hypersphere-based Feature Adaptation
CFLOW	Conditional Normalizing Flows
CSFLOW	Cross-Scale-Flows
DFKDE	Deep Feature Kernel Density Estimation
DFM	Probabilistic Modeling of Deep Features for Out-of-Distribution and Adversarial Detection
DRAEM	A Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection
DSR	Dual Subspace Re-Projection
FastFlow	Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows
GANomaly	Semi-Supervised Anomaly Detection via Adversarial Training
PaDiM	A Patch Distribution Modeling
RD	Reverse Distillation
RKDE	Region-Based Kernel Density Estimation
STFPM	Student-Teacher Feature Pyramid Matching
UFLOW	U-shaped Normalizing Flow
FRE	A Fast Method For Anomaly Detection And Segmentation
YOLO	You Only Look Once
IADBE	Industrial Anomaly Detection Benchmark Engine

Note. Abbreviation used in tables and its explanations.

7.2 Code Test

I attempted to deploy the projects from the papers, but encountered some difficulties. Some of the issues were due to differences in hardware, dependencies, and environments. The paper by Deng et al. [23] and Tien et al. [24] both employed a Teacher-Student network. Liu et al. [25] utilized One-Class Classification and required a lot of training time, while Gudovskiy et al. [26] and Lei et al. [27] adopted Distribution Map techniques. Additionally, Huang et al. [33] utilized Few Shot learning, and Ding et al. [35] along with BGAD [36] applied Few Abnormal Samples methods. Moreover, the code provided by Anomalib [1] presents an outstanding open-source anomaly detection framework with an active community.

The paper above and its code can run perfectly on Windows and Linux.

The code from Roth et al. [28] utilized Memory-Bank techniques, while Xisoftpatch [38] implemented Noisy Anomaly Detection methods. Additionally, Zavrtanik et al. [30] employed Reconstruction-based approaches, and Cao et al. [45] utilized Zero-Shot learning. Furthermore, Lee et al. [46] utilized Multi-Class Unified techniques.

The paper above and its code can run only on Linux.

This paper[32], was published in 2022. However, it requires Python 3.7 and an older version of certain packages, such as torch 1.2.1. Despite utilising the identical environment as that employed by the authors, The code contains errors, specifically a ValueError indicating that the multiclass format is not supported. Consequently, the code cannot be executed. The code presented in this paper[31], cannot be executed due to errors in the dependencies listed in the requirements.txt file. The code from the paper[36] can be executed, although the value of 'Pixel-PRO' is -100 and there are issues with the dataset path. The code from this paper[37], is capable of training a model, although it lacks an evaluation result. The paper in question[39], lacks an entrance file (main.py), thereby preventing its testing. This paper[41] and this paper[86] require a Slurm cluster and multiple GPUs, which are not compatible with the available resources on my PC. The code from this paper[44], is operational, although it lacks the necessary components for training. The authors utilised a pre-trained model and provided a local test. The code from this paper[29], can be partially executed, although it is afflicted by a bug of the "FileNotFoundError" variety. The code from the paper[42] is unable to run due to incompatibility issues between the Torch library and the PointNet2 library. The code from this paper[9], is unable to run due to an error message indicating that the device is not available. Additionally, the dataset path is hardcoded with a server IP address. The paper introduces a forgetting measure, yet the formula variables remain opaque. Their repository does not include an implementation of the forgetting measure. In order to run the code from this paper[34], it is necessary to have access to a pre-trained Vicuna model, which can be obtained from a local GPT environment.