CHALMERS
UNIVERSITY OF TECHNOLOGY

Deployment of an Unsupervised Anomaly Detection Model Using Anomalib and PyTorch

Is it feasible on a low-powered edge-device?

Master's thesis in Machine Learning and Anomaly Detection

SOORAJ KUNNATHUPURAKKAL SUBRAMANIAN
LUDVIG HEDIN

Department of Industrial and Materials Science

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's Thesis 2025

Deployment of an Unsupervised Anomaly Detection Model Using Anomalib and PyTorch

Is it feasible on a low-powered edge-device?

SOORAJ KUNNATHUPURAKKAL SUBRAMANIAN, LUDVIG HEDIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Production Systems, Industrial and Materials Science
Chalmers University of Technology
Gothenburg, Sweden 2025

Deployment of an Unsupervised Anomaly Detection Model Using Anomalib and PyTorch
Is it feasible on a low-powered edge-device?
SOORAJ KUNNATHUPURAKKAL SUBRAMANIAN
LUDVIG HEDIN

Supervisor: Siyuan Chen, Production Systems, Industrial and Materials Science
Supervisor: Silvan Marti, Production Systems, Industrial and Materials Science
Examiner: Anders Skoogh, Production Systems, Industrial and Materials Science

Master's Thesis 2025
Production Systems, Industrial and Materials Science
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Input image and output image with an overlaid heatmap highlighting anomalous areas.

Typeset in LaTeX
Gothenburg, Sweden 2025

Deployment of a Unsupervised Anomaly Detection Model Using Anomalib and PyTorch
SOORAJ KUNNATHUPURAKKAL SUBRAMANIAN, LUDVIG HEDIN
Department of Production Systems, Industrial and Materials Science
Chalmers University of Technology

Abstract

The deployment of pre-trained unsupervised anomaly detection models on low-cost and low-powered edge devices, specifically the Raspberry Pi 5, is a promising approach for cost effective and scalable solution for real-time monitoring in production environments. This thesis investigates the plausibility and performance of running such models on the RP5, focusing on their ability to accurately detect anomalies in real-time. This thesis addresses the challenges with hard hardware limitations, software configuration, dataset creation and model performance in an edge environment.

To enable the training and validation of the model a custom dataset consisting of mugs stained with food coloring to act as anomalies. While the model successfully ran on the RP5 the inference results demonstrated a lack in accuracy with false positives and negatives as-well as a cycle time of 2000-3000 ms per image, was deemed to slow for real-time applications. Although the findings suggest that with further optimizations, such reducing the resolution of input data and further developing the inference script, the cycle time could be significantly reduced. As well as improving the accuracy by reducing the prevalence of false positives and negatives. Thus the model could be an effective solution for real-time anomaly detection.

Keywords: anomaly, detection, deep, learning, edge, computing, anomalib, PyTorch, image, OpenVINO

Acknowledgements

We would like to thank our supervisors Siyuan Chen and Silvan Marti for all the help and insightful perspectives they contributed with during this project. It would not be possible without them. We would also like to thank our examiner, Anders Skoogh for making this project possible.

Lastly, I, Ludvig Hedin would like to thank my brother Fabian Hedin who shared his expertise and best practices when it comes to programming and machine learning.

Sooraj Kunnathupurakkal Subramanian & Ludvig Hedin
Gothenburg, January 9, 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AD	Anomaly Detection
AI	Artificial Intelligence
CV	Computer Vision
EC	Edge Computing
QC	Quality Control
RP5	Raspberry Pi 5
WSL	Windows Subsystem for Linux
AUC	Area Under Curve

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

The rapid advancement of Industry 4.0 technologies has brought significant changes in the the manufacturing sector, introducing smart and interconnected systems, emphasizing operational efficiency, precision, and adaptability[8].Among these technologies, edge computing stands out as a transformative approach that processes data closer to its source, reducing latency, bandwidth requirements, and reliance on centralized cloud systems. This localized approach not only enhances real-time operations but also strengthens scalability, reliability, and data security for modern manufacturing environments[9].

In quality control, anomaly detection is a key technique for identifying irregularities that deviate from normal operations, which can indicate defects in products or failures in equipment[10]. However, deploying supervised anomaly detection (AD) models in industrial settings is often impractical due to the high costs and complexities of creating large, labeled datasets with sufficient anomalous examples. This challenge underscores the value of unsupervised anomaly detection models, which require only normal data for training, thereby enabling broader applicability in scenarios where anomalies are rare or diverse. By eliminating the need for extensive labeling, unsupervised models significantly reduce deployment time and costs while streamlining the adoption of machine learning solutions in production environments[11].

Despite advancements in anomaly detection research, a notable gap exists between academic developments and practical deployment in industrial settings. For example, the windshield gluing station at a unnamed and confidential manufacturing firm still depends on manual inspections to ensure adhesive lines are free of defects, a method that is both time-consuming and prone to human error. Transitioning to an automated system leveraging edge computing, a Raspberry Pi, and a camera could address these inefficiencies by enabling real-time, localized anomaly detection.

To address this gap, the study employed a systematic research design. A custom dataset of anomalous and normal images was created to replicate real-world production scenarios. This dataset was utilized to evaluate the performance of a pretrained model deployed on the Raspberry Pi 5, with key metrics such as accuracy, cycle time, and inference reliability analyzed to assess its feasibility for industrial application. These evaluations offer insights into the feasibility of integrating unsu-

pervised anomaly detection into quality control.

This study bridges the gap between research and industrial application by demonstrating a scalable, resource-efficient framework for deploying anomaly detection models on low-powered edge devices. By validating this approach, our research contributes to the broader understanding of real-time anomaly detection under constrained computing environments. From a managerial perspective, the findings offer a practical guidance for industries aiming to enhance productivity, reduce costs, and streamline manufacturing processes through Industry 4.0 technologies. Additionally, this research contributes to the academic understanding of deploying deep learning models on resource-constrained edge devices, demonstrating the balance between computational efficiency and anomaly detection accuracy.

1.2 Aim

This project aims to demonstrate the feasibility of deploying a deep-learning anomaly detection model on a cost-effective and low-power computing device, such as the Raspberry Pi 5 equipped with a camera module. The objective is to enable the system to evaluate images, either captured directly from the attached camera or sourced from a pre-existing dataset to detect the presence of anomalies. A custom dataset containing coffee mugs with stains serving as anomalies will be created to support the development and testing of the model.

1.3 Objectives

The primary objective of this project is to demonstrate the feasibility of deploying a pre-trained anomaly detection model on a low-power computing device like Raspberry Pi 5. This involves successfully loading and running inference with the model, ensuring that the device can handle the computational demands while delivering adequate performance for real-time anomaly detection. The project also aims to investigate the challenges and trade-offs in deploying machine learning models on resource-constrained hardware, such as computational efficiency with accuracy. By addressing these objectives, the project seeks to bridge the gap between theoretical advancement in deep learning and their practical application in embedded systems.

1.4 Research Questions

- RQ1: Can a pre-trained anomaly detection deep learning model be deployed on a low-power edge computer for accurate real-time inference in a production environment?
- RQ2: What metrics and methodologies can be used to effectively evaluate the performance and accuracy of unsupervised anomaly detection models deployed on low-powered edge devices?

1.5 Limitations

- Due to unavailability of original production data, a custom dataset will be created and used for this project.
- The training of models is outside this project's scope. The supervisors supply trained models.
- The scope of this project is confined to testing the Anomalib model on a Raspberry Pi 5. No simulations or real-world production tests will be done.

1.6 Thesis Outline

This thesis is organized into several sections. Following the introduction, the next section provides a comprehensive overview of the foundational concepts related to the project. The methodology and experimental setup are then described, detailing the approach taken to implement and test the system. The results of the experiment are presented next, along with a discussion of their implications. Finally, the thesis concludes with recommendations for future research and applications.

2
Theory

2.1 Industry 4.0

The manufacturing sector is currently experiencing a transformative period known as the fourth industrial revolution, or Industry 4.0. This concept, initially introduced by Kagermann in 2011 [8], represents a significant shift in industrial practices. Industry 4.0 is poised to dramatically enhance manufacturing processes by improving efficiency and reducing costs through the adoption of advanced technologies. Among the most relevant technologies in this revolution are the Internet of Things (IoT), edge computing, artificial intelligence, and unsupervised anomaly detection, all of which are crucial in driving this transformation forward. This report delves into various concepts central to deploying an anomaly detection model. Anomaly detection is a technique vital for the improvement and maintaining efficiency and quality control in the modern manufacturing environment.

In industry 4.0, where connectivity and data-driven operations are paramount, Iot emerges as a crucial element. Iot was initially introduced in supply chain management in 1999[12] and then the concept was developed and widely adapted to other fields such as healthcare, transport, and manufacturing. The IoT represents a concept in manufacturing where machines, equipment, and sensors are connected by the internet to gather, communicate, and make decisions based on collected data autonomously. The integration of sensors, software, and network connectivity in industrial assets drives this. Enabling control through a central control system. IoT has the potential to revolutionize manufacturing by enhancing production efficiency, reducing machine downtime, enabling predictive maintenance through data analysis, providing real-time insight into production, and improving customer experience. In manufacturing, advances in wireless communication, data analytics, edge computing, and cloud computing enable IoT in manufacturing, allowing real-time decision-making[13].

2.2 Quality control

Quality control (QC) or management is a fundamental aspect of the contemporary world[14]. It ensures the conformity of both the process and the product to the customer's requirement [15]. In general, QC can be defined as a system that maintains a desired level of quality by continuously monitoring product characteristics and implementing corrective actions in case of deviation from standards are identified[16]. The concept of QC gained prominence in the 19th century, with relevant research and applications emerging during that period. In the era of the 21st century, despite advancement organizations continue to face challenges in consistently achieving quality goals[14] [17].

Historically, QC has involved a combination of manual inspection, statistical sampling techniques, and strict adherence to quality dimensions such as performance, reliability, features, durability, conformance, aesthetics, and serviceability, each with its interpretations[18][19]. Traditional QC had focused more on identifying product defects after production, typically through the keen eyes of human inspectors or simple automated machines. Even though these methods have proven their worth over time, they are time-consuming, labor-intensive, and prone to errors especially when dealing with large-scale production and complex manufacturing environments. A defect that goes unnoticed may lead to significant losses like production delay, need for rework, increased cost, and damage to company reputation[17][16]. In the modern world, with the introduction of significant technologies and automation, the phase of quality control has changed with the introduction of more proactive methods[20]. The present state is discussed more in the next section. Figure 2.1 and 2.2 represent a human visual inspection on an electronic device in triplets as a strategy to reduce escapes and a visual inspection system created by the company Instrumental on an assembly line[21].

Figure 2.1: Human visual inspectors inspect an electronic device

Figure 2.2: Instrumental's AI visual inspection system on an assembly line

2.2.1 Evolution of quality control

As manufacturing evolves with Industry 4.0, the methods for ensuring quality have also advanced, with the introduction of smart technologies, automation, and data-driven processes revolutionizing the quality control landscape[20]. Rather than relying on post-production checks modern quality control systems now integrate real-time monitoring of production processes, allowing manufacturers to detect and address defects during production rather than if it has occurred[22]. These automated quality control systems use technologies such as sensors, cameras, and IoT devices to continuously monitor product attributes and detect anomalies in real-time [23]. Furthermore, data analytics and machine learning models enable more predictive quality control, allowing manufacturers to forecast potential issues and take preventive actions before defective products are created[24]. This shift towards real-time, predictive quality control is crucial for maintaining the high standards expected in today's highly competitive global market[25].

2.2.2 Role of anomaly detection in quality control

The contribution of anomaly detection is crucial in modern quality control, especially in the context of Industry 4.0 technology. It refers to the identification of abnormal patterns or data points that deviate from expected behavior [2]. In Manufacturing, these anomalies could indicate potential imperfections in products and disruption in the production process. These defects demand immediate attention to maintain high-quality standards[26].

One of the key advantages of anomaly detection is its ability to perform real-time monitoring. By continuously analyzing data from sensors and machining during the production process, the anomaly system can immediately identify any deviation from normal behaviour[20]. As mentioned earlier, this allows manufacturing to immediately look into the potential defect and address it or equipment failure before they negatively impact production or product quality[22]. Anomaly is particularly important in predictive maintenance, where it helps to identify signs of wear and tear or equipment disruption before the issue escalates. By such early detection

of anomalies, manufacturers can schedule maintenance or repair in advance. This helps in improving properties like reducing downtime and avoiding breakdowns that may affect the quality and performance of the production line[23].

Furthermore, machine learning models enhance anomaly detection capabilities by learning from historical data and improving their accuracy over time. These models can understand the difference between normal variation which are not very important compared to genuine defects from historical data making them important in a complex environment with variable production processes. Overall, integrating anomaly detection in quality control helps to enable post-production checks proactive, real-time monitoring and predictive interventions compared to traditional anomaly detection and significantly improves the efficiency and reliability of quality control in manufacturing[25].Figure 2.3 shows the workflow of machine learning used to detect anomalies. It illustrates data is gathered from the IoT devices and stored and a machine learning model is used for anomaly detection. The performance evaluation is the final step to validate the model.

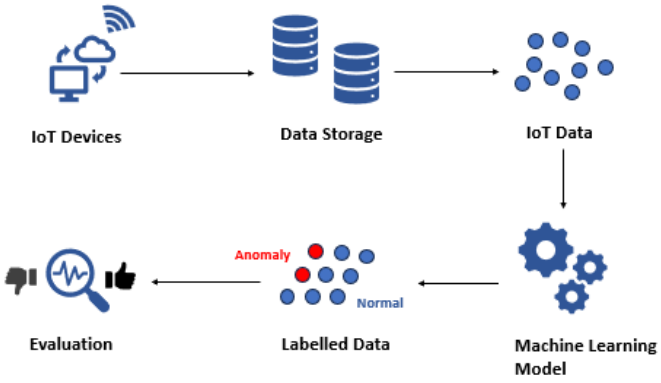


Figure 2.3: Machine learning workflow for anomaly detection

2.2.3 Deep learning in anomaly detection for quality control

Deep learning is a powerful tool in anomaly detection for quality control. Compared to conventional machine learning algorithms with limited ability to process natural data in raw form, deep learning models use a complex structure of algorithms that can automatically learn features from complex and unstructured datasets such as documents, images, and text[27]. It is a sophisticated and complex evolution of machine learning algorithms[28]. Deep learning uses a back-propagation algorithm to understand the changes in the internal parameters used to compute the representation in each layer from the previous layer[29].

In manufacturing, data from sensors, machines, and IoT devices are complex and multi-dimensional. Deep learning models, such as convolutional neural networks

(CNNs) and recurrent neural networks (RNNs), can handle high dimensional data by capturing subtle patterns and complex relationships between data, leading to highly accurate detection of defects in real-time[29]. In the same context, many scenarios involve identifying anomalies in data without predefined labels such as a manufacturing environment where abnormal data might be rare and unpredictable. Deep learning is capable of unsupervised learning using techniques, such as autoencoders and variation autoencoders (VAEs). These techniques can be trained to reconstruct normal data patterns, having any significant deviation from learned patterns as anomalies[30].

2.3 Edge computing

Edge computing involves bringing data production and processing closer to its source, reducing reliance on centralized cloud infrastructure [9]. This approach enhances cloud services by providing efficient computing resources nearer to the end user, which results in reduced latency, lower bandwidth usage, and improved real-time processing capabilities. Additionally, edge computing offers high scalability, reliability, fault tolerance, and increased privacy and security.

In the realm of Industry 4.0, utilizing edge computing for machine learning-based anomaly detection is particularly crucial. Machine learning techniques enable the analysis of high-speed data and real-time streams, facilitating the identification of patterns and predictions in the IoT domain. Deep learning, in particular, excels over traditional methods by uncovering features and patterns that may not be discernible to human observers [10]. This capability is essential for advancing anomaly detection and optimizing the effectiveness of Industry 4.0 technologies.

The relationship between edge computing and IoT is deep, as IoT enables the connection and communication of various devices, across manufacturing setups generating vast amounts of data at the edge of networks. This is where edge computing has a crucial role compared to cloud computing as it is sometimes inefficient in data processing when the data is produced at the network edge. The figure below illustrates edge devices' capability to execute cloud-based complex computational tasks locally. The edge infrastructure can handle various operations, including computational offloading, data storage, caching, and processing information locally, and also plays a crucial role in managing and distributing service requests and deliveries between cloud and end-users. Given the critical nature of these functions in the network ecosystems, the system must be engineered to efficiently meet demanding service requirements while ensuring reliability, robust security measures, and privacy protection for users[1].

2.3.1 Raspberry Pi as an edge device

According to the thesis objective, deploying an anomaly detection model for real-time monitoring to revolutionize manufacturing quality control, Raspberry Pi 5 was chosen by our team as the edge computing device. As the requirement of localized

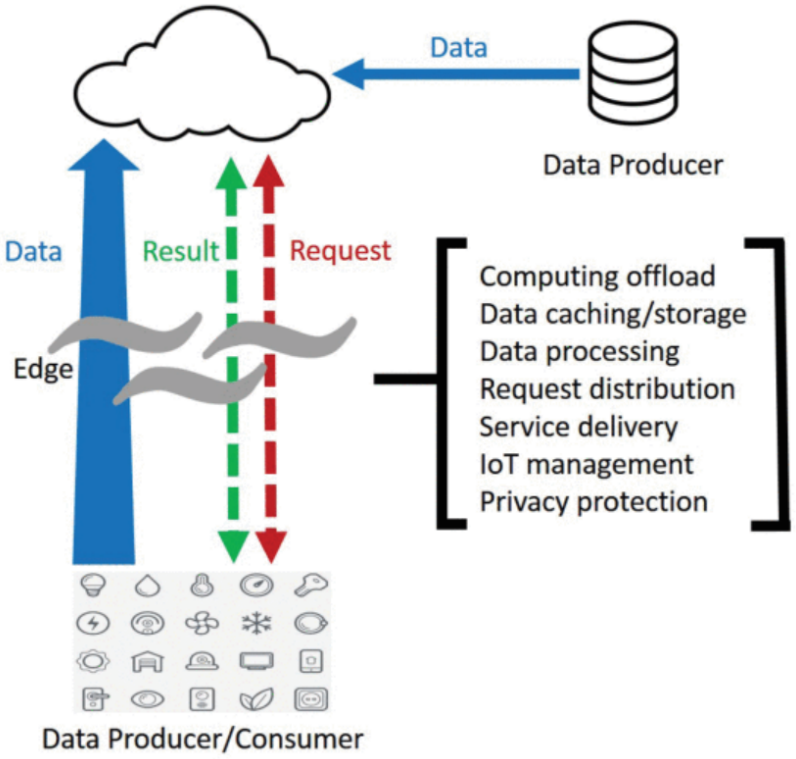


Figure 2.4: Edge Computing paradigm[1]

data processing and quick response time for anomaly detection, RP5 came up as an ideal solution due to its computational power, ease of integration with IoT systems, and affordability to handle edge computing for this deployment process within a constrained environment[31].

Figure 2.5: Raspberry Pi 5

Raspberry Pi 5 offers several new features, making it highly suitable for edge computing in industrial applications. It is the first time we have a full-sized RP computer

using silicon built in-house for Raspberry Pi. It is powered by a powerful processor of 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with cryptographic extension, 512KB per-core L2 caches, and a 2MB shared L3 cache, offering robust processing capabilities. It is twice as fast as its predecessor. The VideoCore VII GPU supports OpenGL ES 3.1 and Vulkan 1.2, providing efficient graphics rendering and dual 4Kp60 HDMI® display output. Additionally, it supports 4Kp60 HEVC video decoding, enhancing its video processing capabilities. In terms of connectivity, it features dual-band 802.11ac Wi-Fi® and Bluetooth 5.0 / Bluetooth Low Energy (BLE). The High-speed microSD card interface supports SDR104 mode, and it features two USB 2.0 ports. The device also offers Gigabit Ethernet with PoE+ support (with an additional PoE+ HAT), dual 4 lane MIPI camera/display transceiver, and a PCIe 2.0x1 interface for fast connection between pheripherals. Additionally, the RP5 comes with a real time clock, a power button, and maintains compatibility with the Raspberry Pi standard 40-pim GPIO header. Also, there are other official accessories to get the best possible performance like an active cooler to make sure it runs smoothly while working at its extreme capabilities and a clip-together case with a built-in fan. All the accessories used are mentioned in the table 3.1. The company also assures the product life time at least until 2036[32].

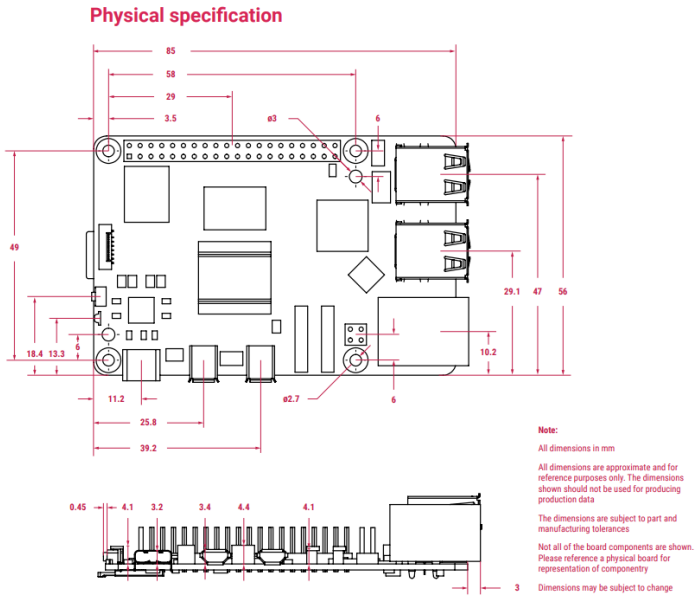


Figure 2.6: Raspberry Pi 5 physical specification

There are several reasons for selecting RP5 as the edge computing device for this project, one of the main reasons was its ability to significantly reduce latency[33]. As required by the project, processing data locally at the edge, RP 5 can ensure the necessary time to identify and respond to anomalies will be minimized, with its processor's ability to run deep learning models. This is particularly important in a time-sensitive environment like industrial monitoring, where delay is not acceptable since it can cause equipment and production losses. Hence edge computing with RP5 can allow fast processing and real-time decision making.

Another crucial factor is energy efficiency. As we require a sustainable era, considering a low power consumption edge computing device is very important in industrial operation. Without depending upon supercomputers to deploy deep learning models that consume high energy, instead utilizing an RP5 with low power consumption is Ideal for long-term deployment in environments that may have power constraints and require continuous monitoring, such as remote industrial locations. Additionally, this local processing capability reduces the dependency on cloud-based systems for any kind of analysis, thereby ensuring uninterrupted operations and saving bandwidth[33].

2.4 Anomaly detection

Anomaly detection is a technique used to identify patterns or instances that deviate from the norm, such as outliers or unusual events that deviate from expected patterns in data. These anomalies can be represented as manufacturing defects, abnormal health conditions in medical monitoring, fraudulent transactions in financial systems, or unauthorized access attempts in cybersecurity, where a user logs in from an unusual geographic location that differs from their normal usage patterns[2][34][35].

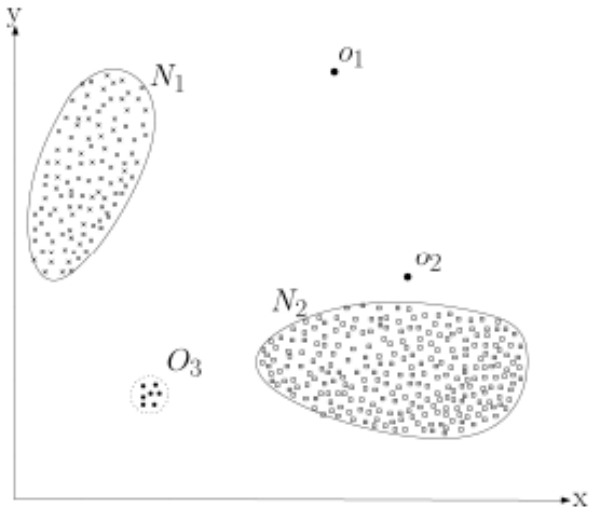


Figure 2.7: Simple anomalies in a two-dimensional data[2]

To understand anomaly detection, it is important to know what an anomaly is. An anomaly is a outcome that deviates from the normal, what is expected. However, the exact requirements for identifying anomalies vary depending on the situation[3]. Anomalies in a simple two-dimensional dataset are illustrated in figure 2.7, where the regions N1 and N2 are normal regions with the most observation concentrated. Regions O1, O2, and O3 are far from the normal areas, hence considered as anomalies[2].

In the industrial IoT context, these anomalies can be equipment malfunctioning or operational or production quality issues. Detecting such anomalies is crucial across

various industries, as they have significant impacts. Due to its increasing demand and applications in multiple sectors, it is a long-standing research area back from the 1960s[26]. Some related notable contributions have been made in different fields such as the manufacturing industry[36] [11] , health monitoring systems [37] [38], intrusion detection systems [39] [40], and many more.

2.4.1 Nature of input data in anomaly detection

In anomaly detection, there is various approaches to anomaly identification based upon the nature and structure of input data.The input data typically consist of data instances or records, described by a set of attributes where each instance can be univariate (having one attribute) or multivariate (having multiple attributes). Accordingly data can be broadly classified into several categories like, record data, sequence data, spatial data and graph data. Record data is most commonly used, where each data instance is independent of each others, for example credit card transactions or medical records. Sequence data, here data points are in specific order such as time-series data, for example sensor data in industrial process. Anomaly detection in such data require methods to account the temporary relationships between the data Spacial data instances are related to each other based on their physical location, for example climate measurements from different locations. Here anomaly detection focus on identifying irregular patterns that might span across regions. Graph data involves vertices connected edges as relationship between data points, for example social network analysis and network traffic data. The complexity of data like high dimensionality relationships among instances and more determines the choice of anomaly detection[2].

2.4.2 Types of anomalies

Anomalies can be categorized in various perspectives according to different situations. For example, figure 2.7, explores anomalies as data points. They are generally categorized into three main types: data point-based anomalies, Context-based anomalies, and pattern-based anomalies[3].

Data point-based anomalies refer to individual data points that deviate from the normal majority but are not outliers. Anomalies are unexpected and point to something uncommon or rare within the dataset, whereas outliers are expected due to random or systematic errors. For example, In density and tensile strength representation of sampled screws as shown in figure 2.8[3]. The samples accepted are represented in a solid box created by the intersection of both sets of dotted lines. Data points outside the box are identified as anomalies. Related work,[41]. On the other hand, context-based anomalies seem normal at first but are abnormal in certain situations. The context is defined by attributes such as time and location. For example, the price for flight tickets or spending during holidays may be high, but if the same behavior during other times of year is unusual, would be flagged as an anomaly. Context-based anomalies are frequently found in time-series data, figure 2.9 shows a temperature of 35°F might be normal in winter (at t_1) but abnormal

in summer (at t_2) at the same location.

Some related works[42][43]. Pattern-based anomalies are trends or patterns that deviate from historical data. For example, in time series data of a workplace network monitoring, expected traffic patterns are built over time. If the employee suddenly uploads or downloads a large volume of data that deviates from normal behavior, is considered as a pattern-based anomaly[3].The figure 2.10 shows an illustration of major steps in pattern-based anomaly detection method in mixed-type time series. Note the 1st and 3th window, and the 2nd and 4th windows match the same set of patterns extracted from both the time series and the event log. The 5th window is anomalous because of the co-occurrence of a peak with a green event[44].

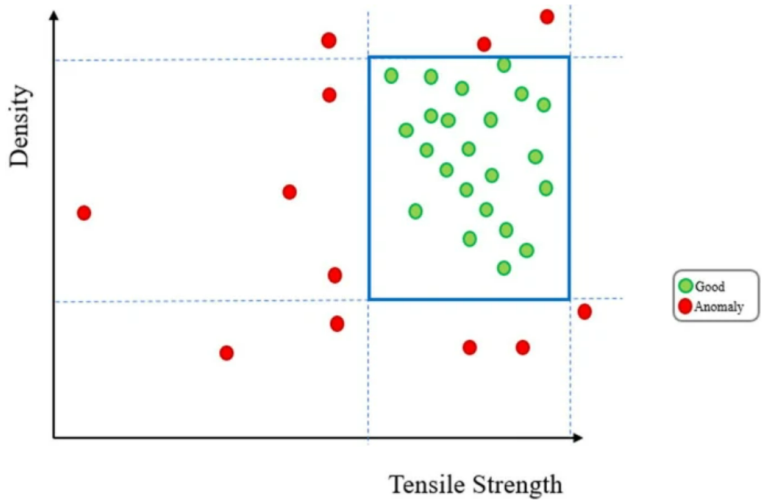


Figure 2.8: Data point based anomaly[3]

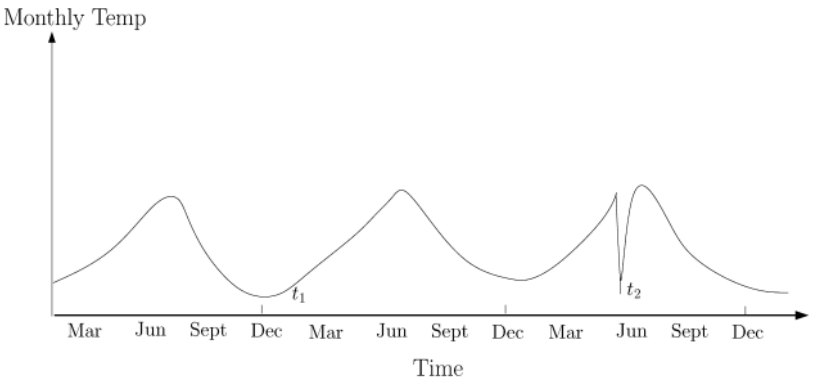


Figure 2.9: Temperature time-series that shows the monthly temperature of an area over few years[2]

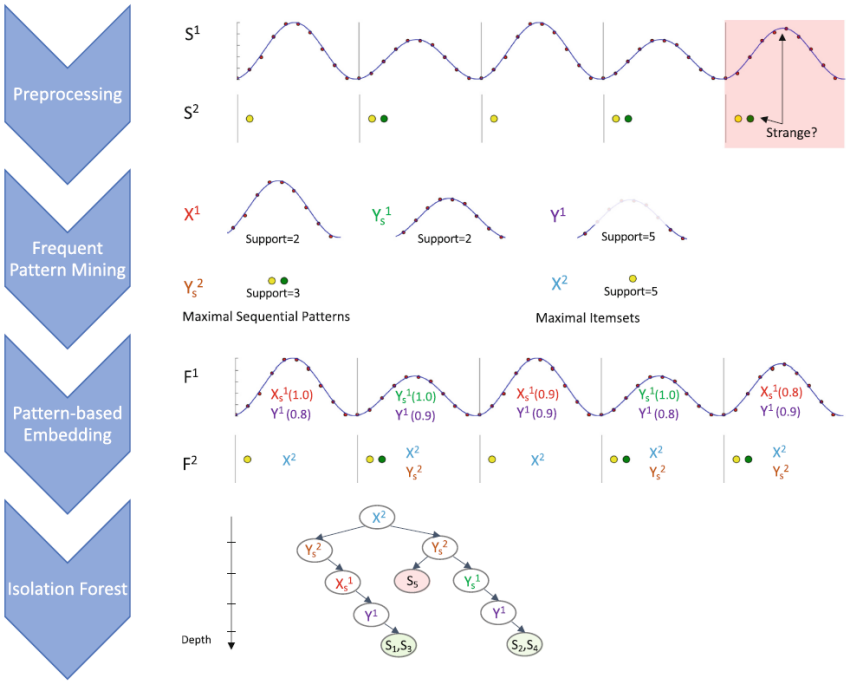


Figure 2.10: Pattern-based anomaly detection in mixed-type time series that shows the monthly temperature of an area over a few years[4]

2.4.3 Data labels and output of anomaly detection

In anomaly detection, the availability and accuracy of data labels associated with each instance determine whether it is normal (instance exhibiting expected behavior) or anomalous (Instance showing deviations from normal behavior) and the method to be used for detection. data labeling often requires domain experts and it may not include all possible anomalous scenarios.

According to the classifications of data labels, There are two main approaches when it comes to anomaly detection, supervised and unsupervised. In supervised anomaly detection a dataset containing both normal images and labeled anomalous images is used to train the model. In unsupervised anomaly detection, the model is trained on a dataset containing only normal images[11]. Supervised anomaly detection is generally more accurate than unsupervised but it requires a substantial amount of labeled data which is not always available. In contrast unsupervised anomaly detection is generally less accurate but it does not rely on labeled data. It identifies anomalies based on patterns and distributions inherent to the data, assuming that anomalies are rare relative to normal samples. Unsupervised anomaly detection is more flexible and easier to implement, specifically in situations where anomalous samples are hard to come by, at the cost of higher sensitivity to data quality and being less precise[2].

Anomaly detection systems typically produce one of two types of outputs, binary labels or anomaly score. In binary labels, each instance is assigned a label as normal or anomalous. This simple output is very helpful in taking clear cut decisions.

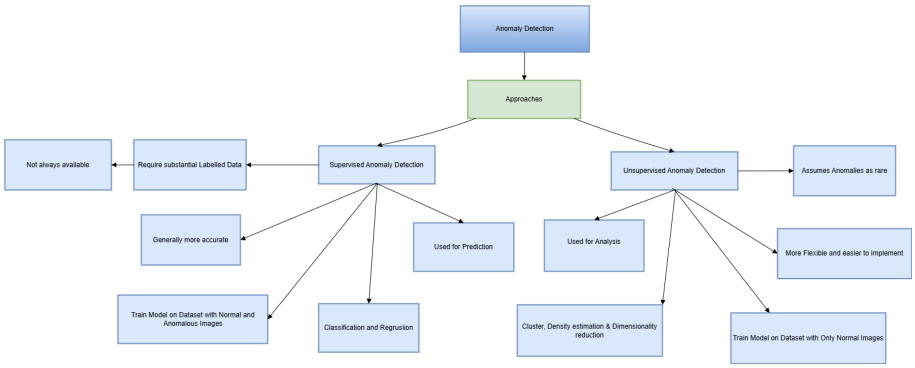


Figure 2.11: Supervised vs unsupervised anomaly detection

For example, automatic shutdowns in manufacturing. On other hand, system assign anomaly score to each instance. the score determines the degree to which the instance is considered anomaly. A threshold can be decided for further investigation. Anomaly detection systems can be tailored for specific need of application by combining labeled data and assigning appropriate output.

2.4.4 Challenges

An anomaly detection is a simple approach to identifying data patterns representing normal regions and flagging any deviations as potential anomalies. However, there are several challenges associated with complicating this process. Defining a normal and abnormal range is always challenging, as these patterns overlap. Anomalies that occur near boundaries may be misclassified as normal and vice versa. This makes it hard to define clear boundaries. Additionally, models created using past historical data may lack future accurate predictions, as anomaly patterns can vary significantly across different domains. What is considered abnormal in one field can be regarded as normal in another. Furthermore, a major challenge is the lack of labeled data and the presence of noise in datasets, which makes it harder to distinguish real anomalies from unwanted deviations [2].

The labels associated with data are often done manually by experts and require considerable effort in creating accurate and representative labeled datasets for training. Hence the process is expensive. Generally, it is hard to get a labeled data set of anomalies with all possible behavior compared to a normal labeled data set. As the anomalies are dynamic, new anomalies may appear. [2][45]

2.5 Deep learning approaches in anomaly detection

There has was always a desire to create a machine that thinks. Since the development of programmable computers, the vision has expanded to create intelligent systems that can perform intricate tasks autonomously. Today, with the contribution of artificial intelligence (AI) the vision is to make intelligent machines and

software to automate complex processes to ease life-like labor routines in manufacturing, make diagnoses in medicine, understand speech and images, and support scientific research[27].

In the context of manufacturing, this thesis is about solving a practical quality control issue by deploying a deep learning model on an edge device near the production system. This AI deep learning approach reduces the need for human operators to provide the knowledge upfront, instead allowing the machine to learn complex data patterns autonomously.

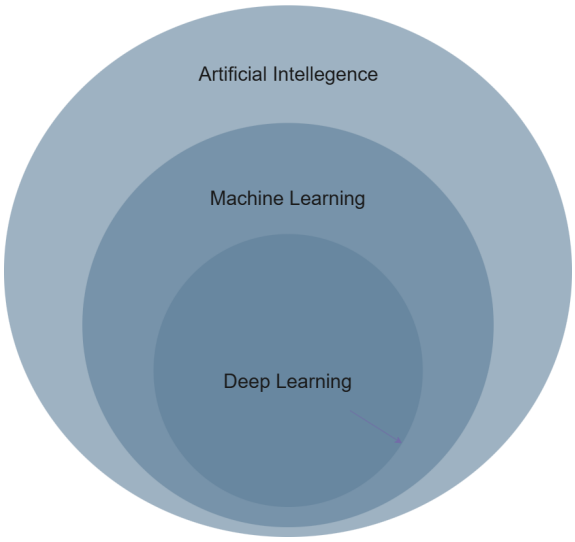


Figure 2.12: Representation of relation Between AI, ML, and DL

Deep learning, a subset of machine learning based on artificial neural networks (ANN), as shown in figure 2.13, has become a preferred choice for anomaly detection due to its ability to model complex patterns in high-dimensional data. Figure 2.13 shows a simple artificial neural network where the left layer is the input layer, while the layer on the far right is the output layer. Between them are hidden layers, which are termed 'hidden' because their values are not directly visible in the training data. These hidden layers contain calculated values that enable the network to perform its functions. When a network has two or more hidden layers, it is typically considered a deep neural network[28].
Unlike traditional methods of anomaly detection that focus more on statistical approaches or rule-based systems, which are limited by the need for extensive domain knowledge and the inability to change to complex and developing patterns. Deep learning anomaly detection can enable the automatic representation learning of patterns directly from complex raw data, end-to-end optimization, and intricate relation learning making it highly efficient and pushing the boundaries of conventional anomaly detection methods. This means deep learning can automatically detect complex and evolving patterns, making it highly adaptable to the dynamic environments of modern manufacturing where data from sensors, cameras, and other IoT devices is high[26].

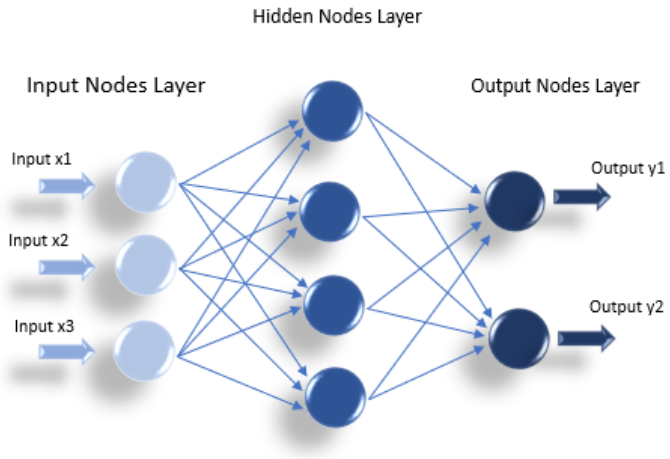


Figure 2.13: A simple artificial neural network (ANN)

One of the main strengths of deep learning is its ability to understand complex patterns in data. This capability can be viewed from two perspectives: first, deep learning's layered structure enables the system to autonomously learn the optimal way to represent data, without needing detailed rules from humans. Second, each layer of the network works as a step in a process, where the system builds up its understanding, layer by layer. As it goes deeper, it can refer back to earlier steps, refining its knowledge with each layer making the representation more powerful and contextually aware. This multilayer architecture makes deep learning highly effective giving a unique ability to capture complex relationships and high-dimensional data, which is especially useful for detecting anomalies in real-time on production lines[27]. There are many deep learning architectures used in anomaly detection, which will be discussed in detail in the next section. The figure 2.14 demonstrates the workflow of a Recurrent Neural Networks (RNNs) model which is a powerful tool for anomaly detection in time-series data. The model consists of an input layer, hidden layers that manage internal state and memory, and an output layer. The hidden layers learn time-dependent features from historical data, allowing the network to capture normal behavior patterns. Once trained, the model classifies new inputs as normal or anomalous, alerting operators when data deviates from expected patterns[46].

2.5.1 Architecture of deep learning models for anomaly detection

The models commonly used in anomaly detection are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Autoencoders (AEs), and Generative Adversarial Networks (GANs). They have proven highly effective and leverage their unique structure to uncover patterns in different data types from different fields.

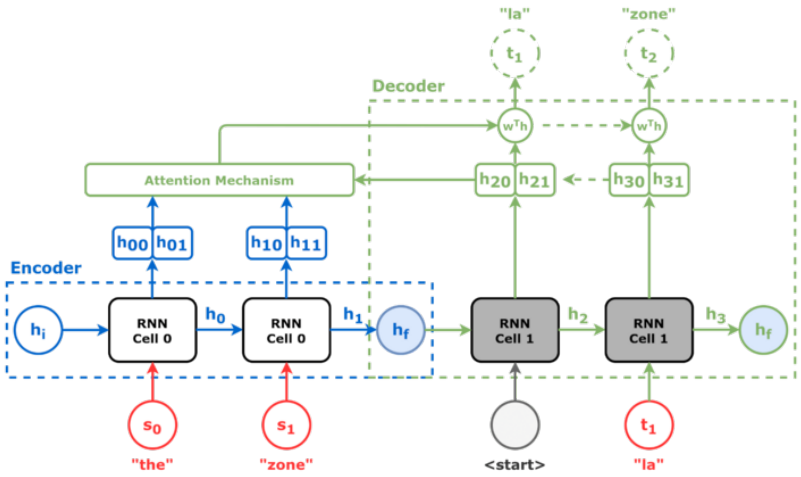


Figure 2.14: Workflow of an RNN-based anomaly detection model.[5]

CNN is a suitable choice of neural network for spatial data like visual imagery. Its ability to detect complex hidden features and patterns such as edge, shape, and texture from high dimensional complex data using convolutional layers enabled its use in image classification, object detection, speech-related applications, and facial recognition. Beyond this, CNN has been applied to audio and video data and is currently an active area of research[6][47].

RNNs are particularly suited for sequential data, where the order of the data points matters and where patterns develop over time, hence ideal for time series analysis, natural language processing (NLP), and speech recognition. RNNs' internal memory can retain information from previous historical data allowing the network to capture patterns in sequences over time, making them suitable for detecting irregularities in data from Iot sensors or monitoring equipment for failure signals[48]. The workflow of RNNs is illustrated in the figure 2.14. However, traditionally RNNs struggled with capturing the context as time steps increased, a limitation that has been addressed with Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs). They were designed to capture longer dependencies in sequential data, where LSTMs comprise a memory cell to retain information and GRUs use a set of control flow of information instead of separate memory cell[47][6].

Autoencoders (AEs) are models used for unsupervised learning. They consist of two main parts, an encoder that compresses the input data into lower dimensional representations and a decoder that reconstructs the data from the compressed form, closer to the original inputs[30]. Through this process, AEs effectively learn essential functions and patterns in the data. Figure 2.15 illustrates several variants of autoencoder architectures proposed to handle various data types. The choice of autoencoder architecture depends on the nature of the data, where convolutional networks are preferred for image data sets and Long short-term memory (LSTM) is preferred for sequential data. An effective AE architecture considered in anomaly detection is a combination of CNN as an encoder and a multilayer LSTM network as a decoder for reconstructing images. The use of various combined models

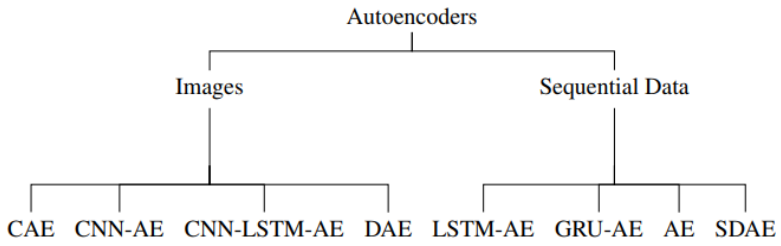


Figure 2.15: Autoencoder architectures for anomaly detection[6].

as shown in Figure 2.15 (acronyms used in figure, AE: Autoencoders[49], LSTM: Long Short Term Memory Network[50], SDAE: Stacked Denoising Autoencoder[51], DAE: Denoising Autoencoder[51], GRU: Gated Recurrent Unit[52], CNN: Convolutional Neural Network[53], CNN-LSTM-AE: Convolutional Long Short Term Memory Autoencoder[54], CAE: Convolutional Autoencoders[55]), can eliminate the need for preparing hand-crafted features and facilitates the model to work directly with raw data simplifying anomaly detection. Auto-encoders are often used for anomaly detection, image de-noising with Denoising Autoencoders (DAE), and data pre-processing where anomalies are rare and do not have distinct patterns[6].

AE-based anomaly detection is a deviation-based anomaly detection method. The autoencoder is only trained with normal instances. After training, the encoder can reconstruct normal data without any error but will fail to do so with anomaly data as the autoencoder has not encountered anomalies. Here reconstruction errors are considered as the anomaly score. The higher reconstruction error, data points are considered an anomaly. Figure 2.16 shows the algorithm using reconstruction errors of autoencoders. Variational encoders (VAEs) are an extension of traditional autoencoders, commonly used in generating new data, like images, that are similar to the training data[30].

INPUT: Normal dataset X, Anomalous dataset $x^{(i)}$ $i = 1, \cdots, N$, threshold α
OUTPUT: reconstruction error $\|x - \hat{x}\|$

$\phi, \theta \leftarrow$ train a autoencoder using the normal dataset X
for i=1 **to** N **do**
  $reconstruction\ error(i) = \|x^{(i)} - g_\theta(f_\phi(x^{(i)}))\|$
  if $reconstruction\ error(i) > \alpha$ **then**
    $x^{(i)}$ is an anomaly
  else
    $x^{(i)}$ is not an anomaly
  end if
end for

Figure 2.16: Autoencoder-based anomaly detection algorithm

GANs framework is composed of two neural network models: a generative model

and a discriminative model. The generator attempts to create realistic data samples (such as images), while the discriminator evaluates them, distinguishing real data from fake. These models generate new synthetic data similar to the training data, improving the generator's ability to create high-quality, realistic data over time, while the discriminator learns to better differentiate real from synthetic. GANS are popular in applications like image generation, style transfer, and even data augmentation for training other models[56].

2.5.2 Anomalib

Anomalib[57] is an open-source library designed to facilitate and streamline the development, training, and deployment of anomaly detection models. It provides a comprehensive framework for implementing anomaly detection in various contexts, particularly in production environments where identifying deviations from normal patterns is critical. The library accommodates both supervised and unsupervised approaches, offering flexibility in model selection depending on the problem at hand. It supports several techniques, including statistical methods, machine learning, and deep learning, making it adaptable for a wide range of applications such as quality control, predictive maintenance, and process monitoring.

Popular tools such as PyTorch, scikit-learn, Keras, OpenCV, NumPy, and Pandas are also integrated, allowing users to leverage familiar libraries within the anomaly detection pipeline. Anomalib also includes pre-trained models and pipelines for specific anomaly detection tasks, accelerating the time to deployment. The library aims to simplify and standardize the anomaly detection process, offering detailed documentation[58] and a modular architecture to reduce the complexities inherent in the field. Additionally, it provides utilities for data prepossessing, model evaluation, and visualization, enabling a seamless end-to-end workflow [59][60].

For benchmarking the anomalib repository[60] has an extensive library of benchmarking data of all the models included in the library. Each model is benchmarked on the the same dataset, the MVTec AD dataset[61]. For each model there is benchmarking data in the form of image-level AUC, pixel-level AUC and image F1 score for every category in the MVTec AD dataset. For both the ResNet-18 and ResNet-50 variations of the models.

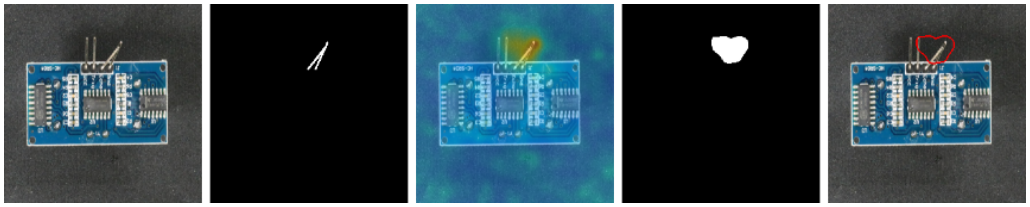


Figure 2.17: Example of anomalib detecting anomalies[7]

2.6 Datasets

2.6.1 The role of datasets in anomaly detection

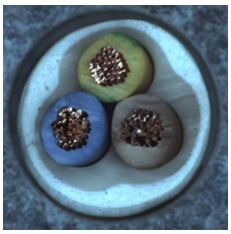
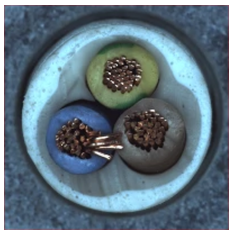
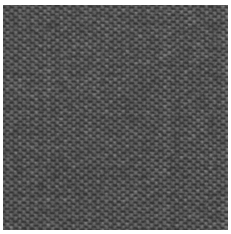
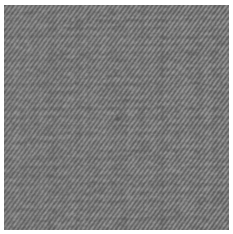
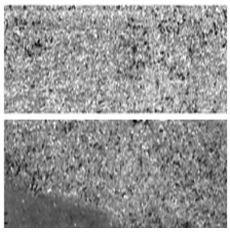
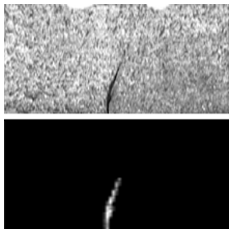
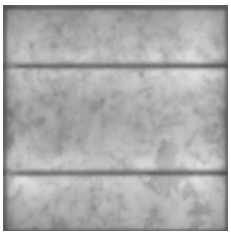
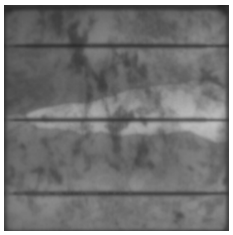
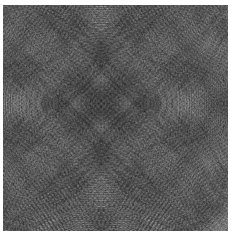
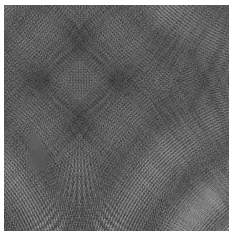
In order to determine the effectiveness of an AD model it has to be tested and evaluated. This is done using appropriate datasets. In this context a dataset is a set of images that contains both normal and anomalous images. Datasets serve as a benchmarking tool for comparing the performance(speed and accuracy) of various models. This helps with fine tuning models for optimal performance.

2.6.2 Challenges with existing anomaly detection datasets

Currently, there are several popular defect detection datasets that can be used for anomaly detection, including the widely-used MVTec AD for industrial defect detection and the AITEX dataset for fabric anomalies (Author, Year). These datasets each come with their own limitations. Together, they provide a wide range of industrial images and defects, but they do not adequately reflect our specific case or fulfill our specific requirements[69].

Given the need for real-time processing of images supplied by the camera module, it is essential to create our own dataset, as the model must be trained on the same objects on which it will be applied. Our dataset aims to emulate a production environment, currently achieved by using stains on mugs made with food coloring to represent the anomalies. For initial testing and development, we can utilize publicly available datasets such as MVTec AD. However, to effectively address our first research question, "Can a pre-trained anomaly detection deep learning model be deployed on a low-power edge computer for accurate real-time data in a production environment?" we must create our own dataset. There are also additional benefits to using our own dataset, such as complete control over its contents and enhanced security[70].

Table 2.1: Comparison of Datasets for Anomaly Detection

Dataset Name	Type	Normal	Anomaly
MVTec AD[61]	Industrial Defects		
AITEX FABRIC[62]	Fabric Anomalies		
SDD[63]	Industrial Defects		
ELPV[64][65][66][67]	Solar Panel Defects		
Optical[68]	Artificial Defects		
Mugs Dataset	Custom Dataset		

3
Methodology

This chapter will outline the process used to deploy a pre-trained Anomalib model on a Raspberry Pi 5 for real-time anomaly detection on a low-powered edge device. This chapter focuses on hardware setup, software environment setup, model deployment, data handling and evaluation.

3.1 Acknowledgment of AI Usage

Throughout the development of this thesis, artificial intelligence (AI) tools were utilized as part of the writing process. Specifically, AI was used to iteratively refine drafts of sections and paragraphs. The role of AI was to adjust the tone, structure and formality to align with academic standards, while ensuring the original meaning and content was preserved. This approach maintained the integrity of the research and writing, while benefiting from AI's capabilities to enhance clarity and consistency in the presentation of ideas.

3.2 Pre-study

The catalyst for this project was a need identified by the manufacturing industry for an automated quality control solution for one of their automated gluing stations. This station uses an industrial robot to apply a glue line to attach windshields to cars. Currently, quality control at this station is performed manually by a human operator. Automating this process could yield cost and productivity benefits.

Before the model could be deployed on the edge device, it had to undergo training and export. This task was conducted by the supervisors using a machine specifically designed for training machine learning models, as the Raspberry Pi 5 lacks the computational power required for this step. The hardware used for training included an NVIDIA RTX 4090 GPU, an Intel i9-13900K CPU, and 128 GB of RAM. The model architecture selected was PaDiM, chosen, for its proven capability to identify anomalies in industrial settings[71]. Figure 3.1 illustrates the architecture of the PaDiM framework for anomaly detection, adapted from the original implementation[72]. The figure has been modified to include the mug dataset used in this project. This adaptation demonstrates how the framework utilized a pre-trained CNN to extract activation vectors, which are used for modeling Gaussian parameters and identifying anomalies within the dataset.

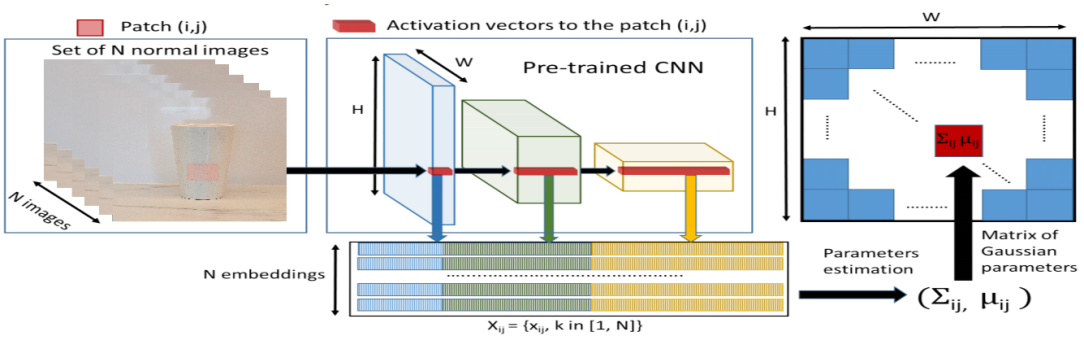


Figure 3.1: Architecture of the PaDiM framework for anomaly detection. For each patch (i, j) in the feature map, the Gaussian parameters μ_{ij} and Σ_{ij} are computed from the set of N training embedding vectors $X_{ij} = \{x_{ij}^k, k \in [[1, N]]\}$, obtained from N different images and three different pre-trained CNN layers.

During the initial stages of the project, validation and development were performed using the MVTec AD dataset. This dataset was selected due to its proven performance in industrial anomaly detection. Additionally, MVTec AD provides readily available benchmarking data for the PaDiM architecture, including image-level and pixel-level AUC scores. This enabled direct comparison of the project's results against established benchmarks, ensuring the model's performance was on the right track. By leveraging this robust dataset, potential issues related to dataset quality were mitigated, simplifying early development and offering a strong foundation for evaluating the effectiveness of the model.

3.3 Raspberry Pi 5 setup

In this section, we outline the hardware and software setup and configuration of RP5, which serves as the deployment and testing platform for AD models. The following components and configurations were utilized to create a functional and efficient work environment for this project.

3.3.1 Hardware configuration

The setup utilizes a Raspberry 5, which acts as a central processing unit for the project. It was selected for its balance between processing power, energy efficiency, size, and cost-effectiveness, making it an ideal platform for embedded applications. The device features the Broadcom BCM2712 quad-core Arm Cortex A76 processor @ 2.4GHz, with RAM variants up to 8GB, allowing it to meet the computational needs of real-time anomaly detection. Ubuntu was installed on Raspberry 5, providing a well-featured Linux environment for running deep learning frameworks.
In order to capture real-time images for anomaly detection, the Raspberry camera module 3 is integrated into the RP5 system. The connection between them is facilitated with a compatible camera cable with a specification of Mini FPC 22-pin to FPC-15 pin, 200mm in length ensuring stable and reliable connectivity.

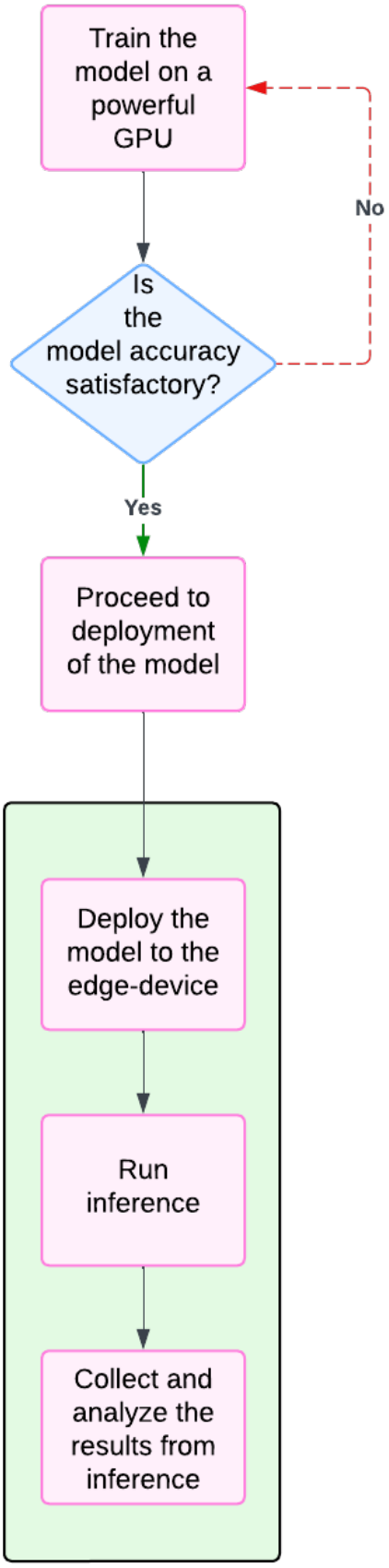


Figure 3.2: Flowchart of the model deployment. The green area marks the scope of this thesis.

3.3.2 Software configuration

Remote access and management of RP5 are critical for the ongoing development and deployment process of the anomaly detection model. Physically accessing the RP5 every time an update or adjustment is needed would be inefficient and impractical, especially if the system is is deployed in a location that is not easily available. Remote access allows for flexible and continuous management, reducing downtime and enabling faster response to any issues that may arise during deployment.
To facilitate secure and reliable remote connectivity, the setup incorporates ZeroTier, a networking tool that creates a virtual network that connects devices and services with each other, anywhere in the world[79]. This virtual network provides encrypted access to the RP5, ensuring that all the interaction with the device remain private and secure. Through ZeroTier, the RP5 can be accessed remotely from any location with an internet connection, allowing for constant monitoring and development as required. Additionally, a Remote Desktop Connection is configured, providing full

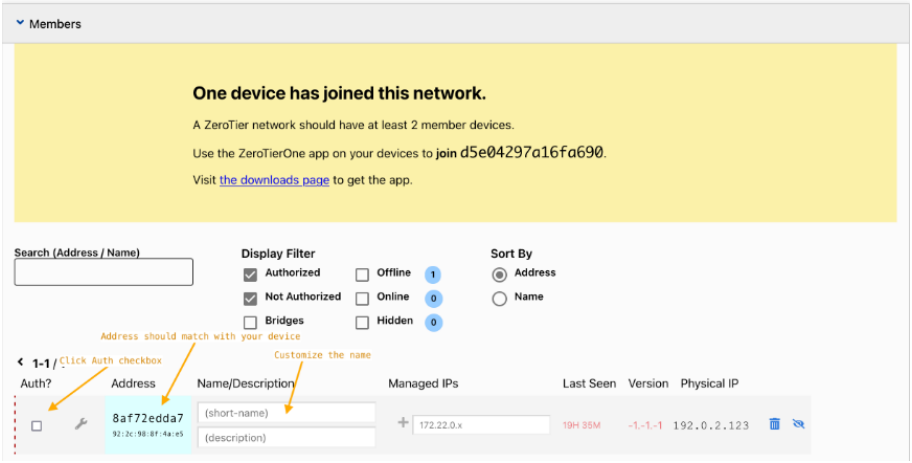


Figure 3.3: ZeroTier network interface for remote access configuration

graphical access to the RP5 desktop environment, as shown in figure 3.4 . This feature simplifies system management and enhances the user experience by allowing direct interaction with the RP5 graphical interface as if operating locally.
The development environment for the RP5 is centered around visual studio code, which is a powerful and versatile code editor. Visual studio code is connected to the RP5 via SSH (secure shell), a protocol that facilitates secure communication over the network. Figure 3.5 illustrates the architecture of visual studio code connected to to a remote machine (in this case, the RP5) through an SSH tunnel. In this setup, Vs Code on the local operating system communicates with the VS Code server running on the remote RP5[80]. This setup is critical for configuring the dependencies for Anomalib and PyTorch on the Raspberry Pi. Docker was used to create an isolated environment and to streamline deployment.

3.3.3 Model preparation

A pre-trained anomaly detection model was provided with the weights in the .ckpt file format and a model configuration in the .yaml format. The model architecture

Figure 3.4: Raspberry Pi Desktop Interface via Remote Desktop Connection

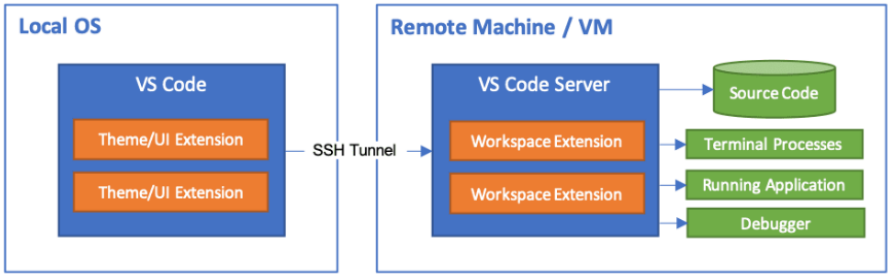


Figure 3.5: The architecture of Visual Studio Code (VS Code) connected to a remote machine through an SSH tunnel

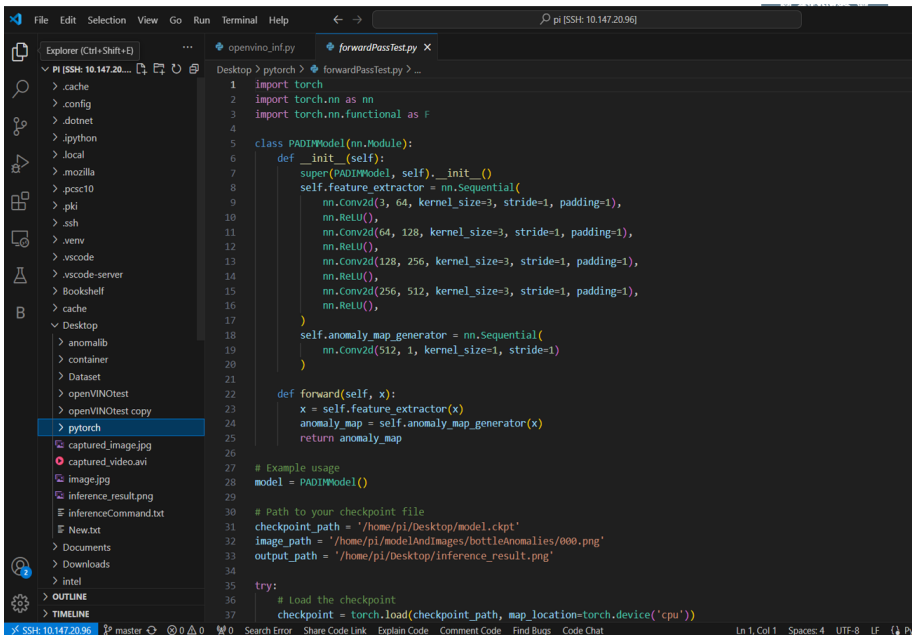


Figure 3.6: Visual Studio Code Interface Connected to Raspberry Pi via SSH

of the supplied model is PaDiM. The model was trained on the mugs dataset, shown in 2.1. The model was prepared for inference by developing a compatible inference script and ensuring that all model files are correctly loaded.

3.3.4 Data acquisition and processing

Data can both be acquired in real-time using the onboard camera module of the RP5 using the openCV library. Or by reading pre-loaded images from memory. The images used are preprocessed in the inference script, where they are resized and normalized to fit the input criteria of the model and to optimize the speed and accuracy of inference.

To create a visual representation of the anomaly data a heatmap was used. The heatmap visualizes the areas of an image where anomalies are detected. It is created during the model's inference process by comparing the features of the input image to those from the normal dataset used during training. Regions with higher anomaly scores are shown in warm colors like red, while normal areas appear in cooler colors like blue. This overlay on the original image helps highlight where anomalies are likely present. The heatmap generation was handled using PyTorch functions from the Anomalib library, which calculate the anomaly scores and map them to a color gradient for easy interpretation.

3.4 Creation of dataset

In order to validate and train the anomaly detection model a dataset is needed. One was created containing a total of 300 images, out of which 240 are normal images while 60 are anomalous. Out of these 240 normal images 160 are reserved for training, while the remaining 80 are used for evaluation of the model. The 60 anomalous images are divided evenly into 3 categories of anomalies, small, medium and large. Each category of anomaly being allocated 20 images.

The item selected for the dataset was a white mug with no handle. To create the anomalies a diluted green food coloring solution was created, which was then painted on to the mugs using a small brush. The different categories of anomalies are categorized by the size and number of food coloring spots on them. Photographing of the mugs was done with a tripod to keep the camera fixed. The camera used was a mobile phone, as at the time of dataset creation there were driver compatibility issues with the camera module on the Raspberry Pi. After fixing the camera on the tripod, a spot was marked where the mugs would be placed to get keep the dataset as consistent as possible.

All the images were taken at a aspect ratio of 1:1 and after the images had been taken they were all scaled down to the size of 900x900 and renamed to make them easier to navigate using python scripts A.1 A.2. Examples of dataset images can be found at A.1.

3.5 Software development

The software development process for this project focused on creating an environment to facilitate development and anomaly detection testing on a low-power edge device, the RP5. Initially, Visual Studio Code (VS Code) was employed as the primary Integrated Development Environment (IDE), but later transitioned to Cursor IDE due to its AI integration features, which improved productivity and facilitated more efficient debugging.

To streamline development on the RP5, a SSH connection was established for remote access, allowing for direct control of the device without physical access. This setup minimizes downtime and ensuring efficient development cycles. More details on the remote connection setup are covered in 3.3.2.

A Docker container was created on the RP5 to establish a consistent development environment. Docker was chosen to encapsulate dependencies, manage versions, and simplify deployments. This container-based setup provided several advantages such as environment consistency, which ensured all development and deployment steps operated under the same settings. Minimizing cross-platform issues and simplified dependency management since complex dependencies for libraries like PyTorch and Anomalib are handled within a contained environment, and portability and scalability, which facilitated testing across different systems by creating a portable environment that can be easily replicated.

The development of the anomaly detection inference script began with a review of PyTorch, Anomalib, and OpenVINO documentation to gain an understanding of their functionalities and integration requirements. Using a sample inference script from the Anomalib repository[57] as a base, modifications were iteratively applied to tailor the code to the specific needs of this project.

Debugging involved a systematic approach utilizing online resources, GitHub issue threads, and built-in AI tools within Cursor IDE. This iterative process helped address common issues, refine the code structure, and incorporate best practices recommended by the PyTorch and Anomalib communities. The development process is illustrated in a flowchart in 3.7.

3.6 Model evaluation

To evaluate the performance of the anomaly detection model when running it on the RP5, the benchmarking metrics provided in the Anomalib repository [7] for the MVTec AD dataset. These metrics include Image AUC, Pixel AUC and F1 score. Each one of these metrics provides a different perspective on the model's performance, both for global anomaly detection and precise localization.

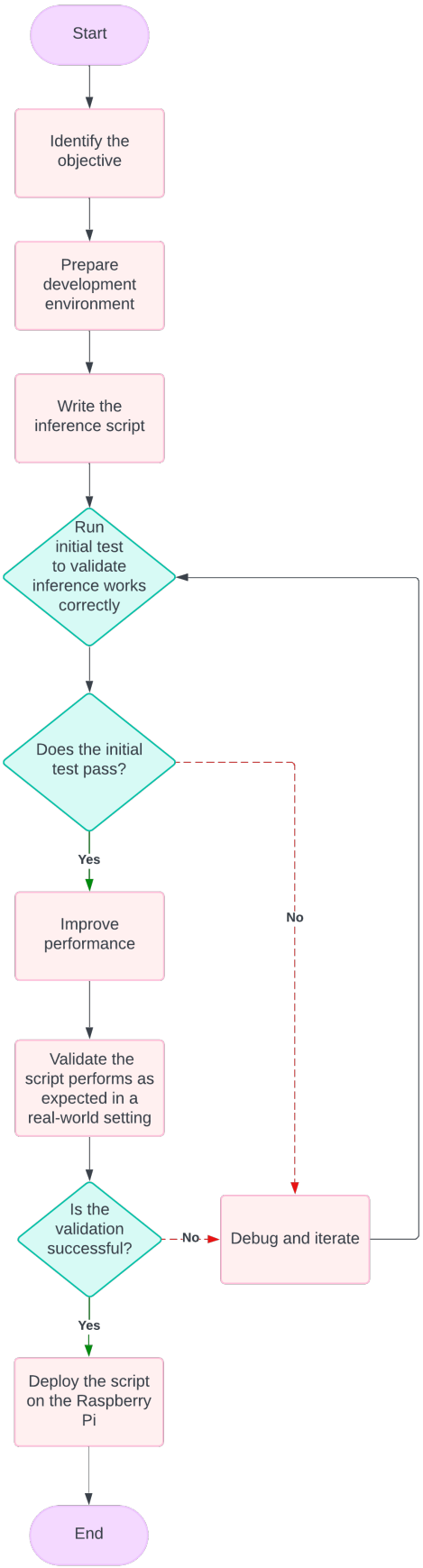


Figure 3.7: Flowchart of the development process

The Anomalib repository provides a range of pre-computed benchmarking data for a range of different anomaly detection models, such as PaDiM and PatchCore on the widely used MVTec AD dataset. These benchmarks serve as a reference when evaluating our models performance on the mugs dataset. By comparing the performance of the model on the mugs dataset with these metrics we get a standardized comparison between the deployed model and existing approaches in the space.

Table 3.1: Raspberry Pi 5 Hardware

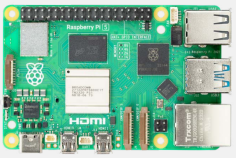
Hardware Name	Specification	Figure
Raspberry Pi 5 [31]	Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with cryptography extensions, 512KB per-core L2 caches and a 2MB shared L3 cache	
Raspberry Pi Camera Module 3[73][74]	Back-illuminated, stacked CMOS 12-megapixel Sony IMX708 image sensor, Dimensions: 25 × 24 × 11.5mm	
Raspberry Pi 5 Camera cable[75][76]	Camera cable mini FPC 22-pin to FPC 15-pin 200mm	
Raspberry Pi Power Supply[77]	Power:27W, Connector: USB-C, Average active efficiency: 89.0 percentage	
Raspberry Pi Active Cooler[78]	5V DC supplied via four-pin fan header, Pulse width modulation control with tachometer	
Raspberry Pi Case	Integrated, temperature-controlled cooling fan that connects to the fan connector on Raspberry Pi 5, 12mm × 17mm × 4mm heatsink with self-adhesive pad improves heat transfer from the processor	

Table 3.2: PaDiM model performance metrics on MVTec AD from Anomalib repository [7]

Category	Image-Level AUC	Pixel-Level AUC	Image F1 Score
ResNet-18			
Carpet	0.945	0.984	0.930
Grid	0.857	0.918	0.893
Leather	0.982	0.994	0.984
Tile	0.950	0.934	0.934
Wood	0.976	0.947	0.952
Bottle	0.994	0.983	0.976
Cable	0.844	0.965	0.858
Capsule	0.901	0.984	0.960
Hazelnut	0.750	0.978	0.836
Metal Nut	0.961	0.970	0.974
Pill	0.863	0.957	0.932
Screw	0.759	0.978	0.879
Toothbrush	0.889	0.988	0.923
Transistor	0.920	0.968	0.796
Zipper	0.780	0.979	0.915
Average	0.891	0.968	0.916
Wide ResNet-50			
Carpet	0.995	0.991	0.989
Grid	0.942	0.970	0.930
Leather	1.000	0.993	1.000
Tile	0.974	0.955	0.960
Wood	0.993	0.957	0.983
Bottle	0.999	0.985	0.992
Cable	0.878	0.970	0.856
Capsule	0.927	0.988	0.982
Hazelnut	0.964	0.985	0.937
Metal Nut	0.989	0.982	0.978
Pill	0.939	0.966	0.946
Screw	0.845	0.988	0.895
Toothbrush	0.942	0.991	0.952
Transistor	0.976	0.976	0.914
Zipper	0.882	0.986	0.947
Average	0.950	0.979	0.951

4
Results

4.1 Dataset

The dataset was designed to meet specific requirements, consisting of a minimum of 300 images, with at least 240 labeled as normal, 60 representing various levels of anomalies. These anomalies were further categorized into three groups, small, medium and large, with 20 images allocated to each category. The finalized dataset exceeded these specifications, including a total of 340 images of mugs. Of these, 262 are classified as normal, with 60 reserved for evaluation and the rest to be used for training. The anomalous subset contains a total of 74 images, distributed as 22 small, 28 medium, and 24 larger anomalies. Each image in the dataset has the resolution of 900x900 pixels. File names are sequentially numbered and include the classification details in the name for ease of navigation of the dataset.

4.2 Inference

The current implementation of the inference script, using the provided model on the public MVTec dataset, demonstrates partial success but also some notable limitations. For example, in the MVTec bottle and custom mugs dataset, the resulting heatmap fails to highlight the anomalous regions of the images correctly. Areas without anomalies are mistakenly marked as anomalies, while the actual anomalies remain undetected. Conversely, for the MVTec toothbrush dataset, the model does correctly identify the anomaly but tends to mark a larger surrounding area as anomalous, which is not accurate. While the inference process functions to some extent, it is not sufficiently reliable for use in a production environment, as it struggles with both false positives and false negatives.

Regarding performance, the inference script operates with a cycle time of approximately 2000-3000 per 900x900 image, though the images are down-scaled to 256x256 for faster processing. This processing may be a consideration for real-time applications where faster inference is needed. Generally real-time is seen as a cycle time of about 50 ms.

The images below display the heatmap overlays generated by running inference with the script developed in this project. These overlays highlight areas identified as anomalous by the model during testing. While the model successfully detected and highlighted some anomalies, instances of false positives (normal areas marked

Figure 4.1: Normal sample

Figure 4.2: Small anomaly

Figure 4.3: Medium anomaly

Figure 4.4: Large anomaly

as anomalous) and false negatives (anomalies not detected) were observed. These results demonstrate the effectiveness of the script in visualizing model predictions while also identifying areas where further optimization is needed to improve accuracy and reliability in real-world applications.

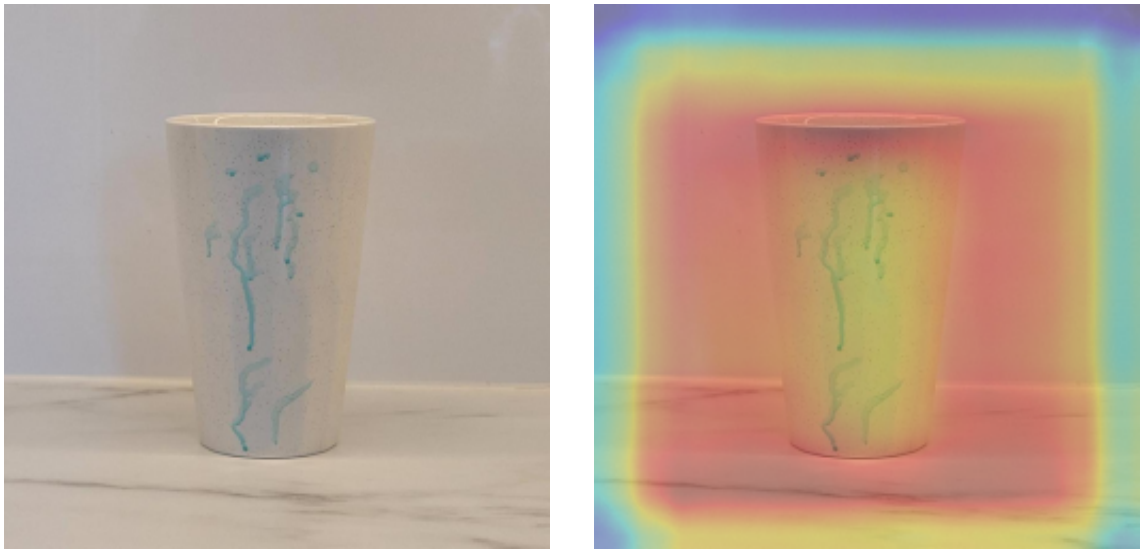


Figure 4.5: Example result of inference on an image from the custom mugs dataset

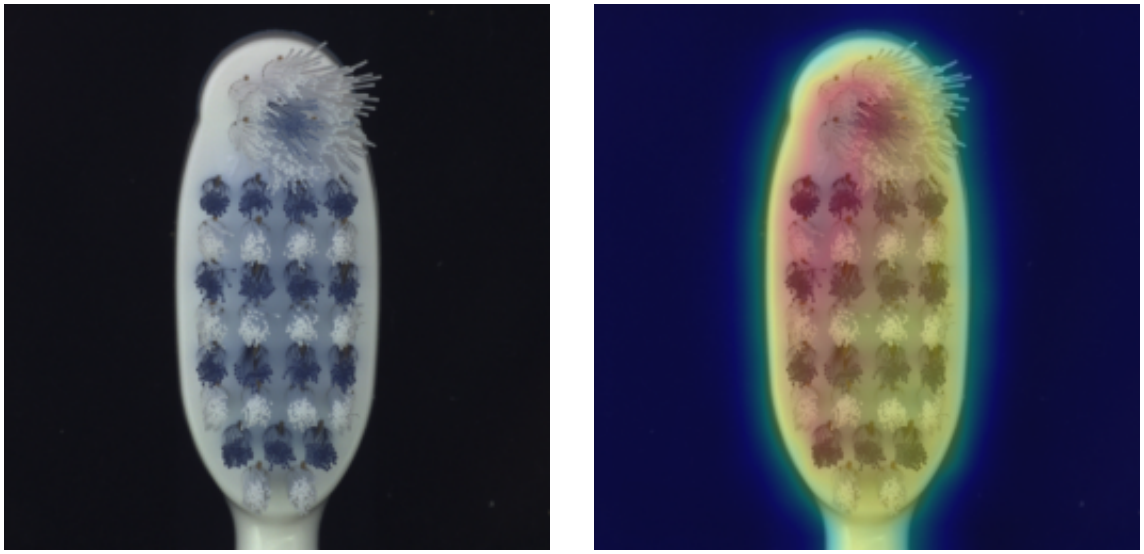


Figure 4.6: Example result of inference on an image from MVTec AD toothbrush dataset

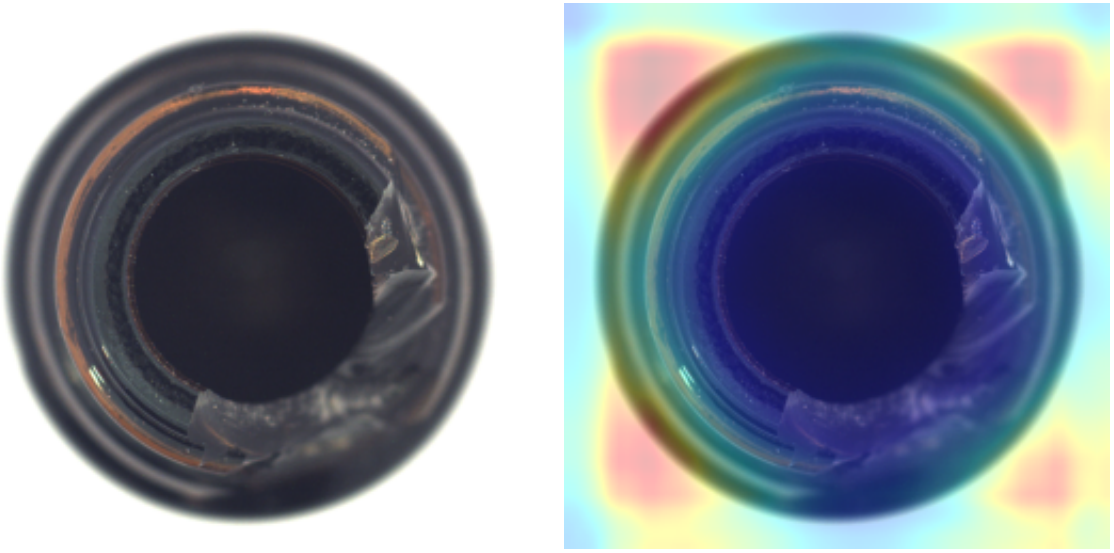


Figure 4.7: Example result of inference on an image from MVTec AD bottle dataset

5
Discussion

5.1 Inference development and results

The inference results did not meet initial expectations. However, achieving functional inference was a significant milestone, given the substantial challenges encountered throughout the project. Early efforts to the inference script faced constant obstacles. Attempts were made using WSL and Docker on laptops, as well as directly on the RP5, both within a Docker container and without one, but all attempts were unsuccessful. During this time of development, additional issues arose with the RP5, including a complete system freeze that required a full operation system re-installation. This completely wiped the memory RP5, but luckily backups of the development were saved to GitHub. Though some time still had to be spent setting up the RP5 again.

One of the significant challenges encountered during the initial setup of the RP5 was the installation of a 32-bit operating system instead of the required 64-bit version. This mismatch likely contributed to the persistent errors experienced during the setup of both the Docker container and the execution of the inference script. After the RP5 froze, the technician who initially set it up was contacted for assistance and it was decided that a re-installation of the OS was required. Unfortunately, during the re-installation process, a 32-bit OS was mistakenly installed. This oversight significantly delayed development, as inference could not be ran on a 32-bit OS and the issue was only identified in the later stages of the project. As we had mistakenly only investigated the physical architecture of the RP5 which was 64-bit, and not the version of the OS.

However, once the issue had been detected and the correct 64-bit OS was installed, development could continue. Although under tight time constrains due to the advanced stage of the project. Within a few days of the correct OS being installed more progress had been made than in the previous months and the inference script was successfully executed for the first time. Even though the output was faulty is, it showed that inference could be run on a low-powered edge-device, and it constituted a significant breakthrough in the project. This directly addresses RQ1 of whether a pre-trained anomaly detection model can be deployed on a low-power edge device, as it demonstrated feasibility despite current limitations. The likely cause of the faulty output of the model is suspected to be a mismatch between the parameters and settings used in the inference script relative to the training setup. However,

other potential issues that have not been ruled out include, dataset reprocessing discrepancies and heatmap generation.

In hindsight, implementing the entire workflow, from training to deployment, may have been advantageous, rather than focusing solely on deployment. This approach would reduce the risk of mismatches between the settings and parameters used during training and deployment, as it allows for the use of identical files and configurations throughout the process. While this would likely involve additional effort, the time invested could be recouped by removing the need for communication with the model's creator. Furthermore, adopting a comprehensive workflow would provide a deeper understanding of the entire process, from model training to deployment, offering insights into the model's performance and the challenges encountered at each stage.

The inference time of 2000-3000 ms per image can likely be improved through optimization of the inference script. One approach to enhance performance would be to scale down the images used by the model, which would reduce the cycle time at the cost of some accuracy. Identifying the optimal balance between these two factors: processing speed and resolution could lead to a reduction in cycle time. They would have to be tuned to the specific use case. The current inference script is biased toward higher resolution and accuracy. Additionally, the heatmap generation process, which consumes valuable processing power, could be eliminated or reduced in resolution if the model were to be deployed in a production environment. The output could be streamlined down to a simple binary response indicating the presence or absence of an anomaly to optimize the model for production use further. It is worth nothing that we did not proceed with optimization due to time constraints. Our primary focus was to ensure the model functioned correctly, as optimizing prematurely could have resulted in unnecessary rework if the baseline functionality was not robust.

The evaluation of the anomaly detection model on the RP5 leveraged standardized benchmarking metrics, such as image AUC, pixel AUC, and F1 score, which are widely used in the anomaly detection space[61]. These metrics combined provide a holistic picture of the model's performance. Assessing both global AD accuracy and localization abilities. By using these metrics a direct comparison can be made between the performance of the deployed model and it's results on the mugs dataset and the benchmarks for a similar PaDiM model on the MVTec AD dataset. This ensured a standardized and reliable evaluation strategy.

Image AUC shows the model's ability to distinguish between normal and anomalous images, while pixel AUC shows the model's ability to localize the anomalies in the images. The F1 score, combining the prominence of the rate of false negatives and false positives offers a holistic view of the detection effectiveness. Together these metrics addressed RQ2 by demonstrating an effective method to evaluate the effectiveness of an anomaly detection model.

However, this particular deployment case revealed some qualitative output from the inference results in a heatmap that stood out as inconsistent. The visual analysis clearly showed that anomalies were not accurately localized or detected, and when compared to the ground truth as well. Given this observation, performing further detailed evaluations such as AUC, pixel AUC, and F1 score would not have added meaningful insight. The visual evidence already demonstrated the limitations of the model.

It is important to clarify here that the decision to not focus on these metrics was not a lack of due diligence, but a determined decision to focus on the descriptive diagnosis phase at this stage. This allowed the research to focus on identifying the persistence factors contributing to model under performance, such as data characteristics, model parameters, or inference script setup that needed to be improved to refine the generated model. However, future iterations study of this study should include the standardized metrics evaluations more thoroughly once significant improvements in the performance of the inference results are observed.

5.2 Creation of dataset

The mugs dataset was developed to overcome challenges related to using data from a manufacturing firm's gluing station, mainly due to the confidentiality of production floor data.[9]. Despite this restriction, the project's workflow, spanning dataset creation to deployment remains unchanged. Ensuring that the value and outcomes of the project are not diminished by the choice of dataset. The alternative mugs dataset still allows for achieving the project's objectives and answering the research questions. Additionally, to determine whether a low-powered edge device can perform accurate real-time inference, real-time testing was required. This testing requires a dataset of items that we have physical access to. Thus it was decided to create a dataset of mugs with stains of food coloring acting as anomalies. This was essential for training and testing of the anomaly detection model.

One key consideration in this study was the choice to either use publicly available datasets such as MVTec AD, or create a custom dataset tailored to the requirements of the project. There are some considerations to take in to account when making this choice. Public datasets offer several advantages, such as standardization and extensive benchmarking data on a diverse selection of anomalies, which are valuable for model testing and validation. However custom datasets can be customized to fit the needs of the project.

The custom mugs dataset, while requiring additional effort to create, was essential for validating that real-time anomaly detection performed as intended. This validation would not have been feasible with a public dataset, as recreating the scenes depicted in public images and feeding them into the Raspberry Pi 5's camera module was impractical. Over the course of this project, the strengths of both approaches were leveraged. Public datasets, such as MVTec AD, were utilized during the initial development phases to benchmark the model and streamline early testing.

In the later stages, the custom dataset became crucial for achieving the project's objectives, enabling accurate evaluation under conditions that closely mirrored the intended deployment environment.

5.3 Software development

During the initial phases of the project, prior to receiving the RP5, we set up a virtual machine on a laptop using WSL2 and Docker to emulate the RP5 environment. This approach was intended to accelerate development by leveraging the superior processing power of a laptop compared to the RP5.

Once the RP5 was available, and it was configured for direct development using SSH, enabling work from any location. However, around the same time, issues arose with with the WSL2 and Docker setup on the laptops, which were later on identified as host machine permissions conflicts. Despite the efforts to resolve these problems, they remained at the time. Therefore development was transitioned to being done directly on the RP5, which was already operational, in order to avoid downtime in the development process. Although this shift resulted in a performance reduction during certain tasks, such as Docker container builds and inference script execution, these tasks constituted a small portion of the overall development time which mostly consisted of research, writing of code, and debugging. Additionally, working directly on the RP5 proved valuable, since inference testing on the target device was an essential part of the project.

5.4 Edge computing

Edge computing was a foundational requirement of this project, as the primary objective was to evaluate the feasibility of deploying a pre-trained anomaly detection model on a low-power edge device, like the RP5. By processing data locally, edge computing offers a more stable and lower latency, a critical factor for real-time anomaly detection in production environments. The ability to detect and act on anomalies instantly is critical in manufacturing, where delays can result in costly downtime.
[9] Another advantage was enhanced data privacy and security. Deploying the model on the RP5 eliminated the need to transmit any sensitive data over a network, reducing the risks of breaches. A real-world example of this value is a manufacturing firm's gluing station, which served as the catalyst for this project, where production data cannot be shared externally. Furthermore, edge computing is more robust when it comes to network stability since it can operate independently of internet connectivity, ensuring a more consistent performance where the network is unreliable.

The modularity and scalability of edge devices such as the RP5 make it possible to replicate the setup across multiple production lines or facilities, in a cost-efficient manner, which aligns with the broader vision of this project. The low power

consumption of the RP5 also adds to it suitability for deployment in a resource-constrained case[1].

However, edge computing does come with downsides. The limited computational resources available of the RP5 require optimization to achieve an acceptable performance level. For instance, the cycle time accomplished in this project of 2000-3000 ms per image was sufficient for proof-of-concept but fell short of the requirements for real-time speeds[9], is generally seen as 20 Hz, or a cycle time of 50 ms. However this limitation could be mitigated by using more powerful edge devices, but this would erode the benefits of low-cost and power-efficient.

Edge computing was central to this project as it directly addressed the research objectives. It enables the deployment of a cost-effective, energy-efficient anomaly detection system that can meet the requirements for real-time performance while ensuring data privacy and security. These characteristics cement its relevance for industrial applications requiring localized and reliable data processing.

5.5 Anomaly Detection in Industry 4.0

In this context, anomaly detection plays a crucial role in ensuring reliability, efficiency, and sustainability in industrial settings. These systems can contribute significantly to key areas such as quality control[36], predictive maintenance[81], and smart manufacturing[82].

In a production environment, the anomaly detection model can be applied to identify defective components, irregularities in products, and deviations in production processes. Real-time analysis of images will enable the immediate detection of anomalies, preventing substandard products from advancing through the supply chain. Similarly, AD plays a crucial role in predictive maintenance. By identifying early signs of equipment failure from deviation in sensor or image data, the maintenance team can proactively address potential issues before they result in costly downtime or unplanned interruption.

The mugs dataset utilized in the study as discussed in 3.4 serves as a proof of concept for AD systems in industrial applications. Similar principles and methodology can be extended to real-world industrial cases. To apply the model to a real-world use case, the same workflow and software utilized during training and deployment can be employed. The primary modification required is the use of a dataset tailored to the specific application during the training and evaluation phases. The model could subsequently be deployed for tasks such as inspecting surface defects on products within manufacturing industries [36], electronic components inspections [83], and more. These insights using AI and IoT-enabled systems for autonomous quality assurance and optimizing production workflow align closely with Industry 4.0. This highlights how such technologies can significantly improve industrial processes through enhanced reliability and predictive capabilities, reduction of human errors in quality control, and real-time AD for rapid interventions. Future work should val-

idate these models against real-world industrial datasets to bridge the gap between experimental research and practical deployment. Further, integrating the model into IoT enabled manufacturing environment would offer a fully automated AD solution in the manufacturing process.

6
Conclusion

This thesis set out to investigate the deployment and evaluation of pre-trained unsupervised anomaly detection models on low-powered edge devices, with a focus on the RP5. The research explored whether such models could perform accurate real-time inference of image data, a important requirement for practical application in production environments.

The result of this thesis was that partial success was achieved. But there are challenges that must be addressed for deployment in applications that require real-time inference. Inference was successfully run on a RP5, which proved capable of running the model, although there were limitations both in accuracy and speed. False positives and negatives were noted in the anomaly detection process. These issues are likely the cause of mismatches in model parameters, settings and inference script setup. Although the progress made during the project has shown that successfully running a AD model on a edge-device such as the RP5 is plausible.

The study highlighted challenges associated with hardware configuration and software setup on the RP5. A central issue was the installation of an incorrect 32-bit operating system instead of the correct 64-bit which significantly delayed progress. After the correct operating system was installed, the inference script could be run for the first time, marking a breakthrough in the project.

In terms of dataset creation, the decision to use mugs with food coloring stains as anomalies, was due to the unavailability of real production data. This decision did not detract from the project's objectives. This dataset provided suitable for testing and training the model, ensuring that the project could proceed despite the confidentiality constraints.

Looking ahead for future work, several steps could improve and build on the outcomes of the research. Spending more time on the inference script, refining, debugging, tuning and optimizing could make it fulfill the goals of the project. Both when it comes to reduction the amount of false positives and false negatives as well as reducing the cycle time. A more comprehensive workflow encompassing the entire pipeline from training to deployment could reduce the risk for mismatches between the settings and parameters used when training the model and when deploying it. It could also increase the feasibility of rapidly testing different models. Another potential direction for further research is exploring other hardware solutions for edge-devices.

Bibliography

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, nr. 5, pgs. 637–646, DOI: . 10.1109/JIOT.2016.2579198

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, nr. 3, pgs. 1–58, 2009.

[3] S. K. Adari and S. Alla, *Introduction to Anomaly Detection*. Berkeley, CA: Apress, 2024, DOI: . 10.1007/979-8-8688-0008-5_1. [Online]. Available: https://doi.org/10.1007/979-8-8688-0008-5_1

[4] M. Ahmed and A. N. Mahmood, "Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection," *Annals of Data Science*, vol. 2, nr. 1, pgs. 111–130, 2015.

[5] D. V. Godoy, "Dl-visuals," 2024, accessed: 2024-12-19. [Online]. Available: https://github.com/dvgodoy/dl-visuals/

[6] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[7] O. Toolkit, "Anomalib: A deep learning library for anomaly detection," https://github.com/openvinotoolkit/anomalib, 2021, accessed: 2024-10-09.

[8] H. Kagermann, W.-D. Lukas, and W. Wahlster, "Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution," *VDI nachrichten*, vol. 13, nr. 1, pgs. 2–3, 2011.

[9] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino, "Edge computing: current trends, research challenges and future directions," *Computing*, vol. 103, nr. 5, pgs. 993–1023, 2021.

[10] B. Sharma, L. Sharma, and C. Lal, "Anomaly detection techniques using deep learning in iot: A survey," i *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2019, pgs. 146–149.

[11] H. Shen, B. Wei, and Y. Ma, "Unsupervised anomaly detection for manufacturing product images by significant feature space distance measurement," *Mechanical Systems and Signal Processing*, vol. 212, pg. 111328, DOI: . https://doi.org/10.1016/j.ymssp.2024.111328. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327024002267

[12] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, nr. 7, pgs. 97–114, 2009.

[13] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, nr. 5, pgs. 164–173, 2015.

[14] A. Gunasekaran, N. Subramanian, and W. T. E. Ngai, "Quality management in the 21st century enterprises: Research pathway towards industry 4.0," *International Journal of Production Economics*, vol. 207, pgs. 125–129, DOI: . https://doi.org/10.1016/j.ijpe.2018.09.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092552731830375X

[15] T. A. Ryabchik, E. E. Smirnova, M. I. Lukashova, and H. Haydar, "Manufacturing processes quality control as a main factor of performance enhancement in industrial management," i *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2019, pgs. 1463–1466.

[16] A. Mitra, *Fundamentals of quality control and improvement*. John Wiley & Sons, 2016.

[17] A. V. Feigenbaum, "Total quality control," *New York*, pg. 12, 1991.

[18] D. A. Garvin, "Competing on the eight dimensions of quality," 1987.

[19] A. Parasuraman, V. A. Zeithaml, and L. L. Berry, "A conceptual model of service quality and its implications for future research," *Journal of marketing*, vol. 49, nr. 4, pgs. 41–50, 1985.

[20] D. Mourtzis, A. Vlachou, and V. Zogopoulos, "Cloud-based augmented reality remote maintenance through shop-floor monitoring: a product-service system approach," *Journal of Manufacturing Science and Engineering*, vol. 139, nr. 6, pg. 061011, 2017.

[21] "Manual inspection vs. ai inspection with instrumental." Instrumental. [Online]. Available: https://instrumental.com/build-better-handbook/machine-vision-vs-manual-inspection-vs-instrumental, Retrieved: 2024-10-22.

[22] A. Gunasekaran and E. W. Ngai, "The future of operations management: an outlook and analysis," *International Journal of Production Economics*, vol. 135, nr. 2, pgs. 687–701, 2012.

[23] A. Kusiak, "Smart manufacturing must embrace big data," *Nature*, vol. 544, nr. 7648, pgs. 23–25, 2017.

[24] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing letters*, vol. 3, pgs. 18–23, 2015.

[25] D. Mourtzis, "Simulation in the design and operation of manufacturing systems: state of the art and new trends," *International Journal of Production Research*, vol. 58, nr. 7, pgs. 1927–1949, 2020.

[26] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, nr. 2, DOI: . 10.1145/3439950. [Online]. Available: https://doi.org/10.1145/3439950

[27] I. Goodfellow, "Deep learning," 2016.

[28] Arne Wolfewicz, "Deep learning vs. machine learning," September 30, 2024, accessed: October 10, 2024. [Online]. Available: https://levity.ai/blog/difference-machine-learning-deep-learning#:~:text=Machine%20Learning%20means%20computers%20learning,documents%2C%20images%2C%20and%20text.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, nr. 7553, pgs. 436–444, 2015.

[30] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special lecture on IE*, vol. 2, nr. 1, pgs. 1–18, 2015.

[31] Raspberry Pi Foundation, "Raspberry pi 5," 2024, accessed: October 4, 2024. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/

[32] Raspberry Pi Ltd, "Raspberry pi 5," 2024, accessed: October 4, 2024. [Online]. Available: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf

[33] M. Fezari and A. Al-Dahoud, "Raspberry pi 5: The new raspberry pi family with more computation power and ai integration," 2023.

[34] L. Khan and M. Hayat, "A survey of data mining techniques for intrusion detection," *International Journal of Computer Applications*, vol. 90, nr. 16, pgs. 1–6, 2014.

[35] Y. Zhou and Y. Pei, "Anomaly detection based on machine learning for cybersecurity," *Journal of Computer Networks and Communications*, vol. 2020, 2020.

[36] M. Haselmann, D. P. Gruber, and P. Tabatabai, "Anomaly detection using deep learning based image completion," i *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pgs. 1237–1242.

[37] S. Mahfuz, H. Isah, F. Zulkernine, and P. Nicholls, "Detecting irregular patterns in iot streaming data for fall detection," i *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pgs. 588–594.

[38] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, "Iot healthcare analytics: The importance of anomaly detection," i *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 2016, pgs. 994–997.

[39] C. Zhang *et al.*, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, nr. 01, pgs. 1409–1416, DOI: . 10.1609/aaai.v33i01.33011409. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/3942

[40] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the internet of things," *Sensors*, vol. 19, nr. 9, DOI: . 10.3390/s19091977. [Online]. Available: https://www.mdpi.com/1424-8220/19/9/1977

[41] P. J. Rousseeuw and M. Hubert, "Anomaly detection by robust statistics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, nr. 2, pg. e1236, 2018.

[42] M. A. Hayes and M. A. Capretz, "Contextual anomaly detection in big sensor data," i *2014 IEEE International Congress on Big Data*, 2014, pgs. 64–71.

[43] S. Shekhar, C.-T. Lu, and P. Zhang, "Detecting graph-based spatial outliers," *Intelligent Data Analysis*, vol. 6, nr. 5, pgs. 451–468, 2002.

[44] L. Feremans, V. Vercruyssen, B. Cule, W. Meert, and B. Goethals, "Pattern-based anomaly detection in mixed-type time series," i *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019,*

Würzburg, Germany, September 16–20, 2019, Proceedings, Part I. Springer, 2020, pgs. 240–256.

[45] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the internet of things," *Sensors*, vol. 19, nr. 9, pg. 1977, 2019.

[46] XenonStack, "Time series deep learning: Techniques and applications," https://www.xenonstack.com/blog/time-series-deep-learning, 2024, november,2024.

[47] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," i *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pgs. 1595–1598.

[48] P. Malhotra *et al.*, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[49] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pgs. 84–96, 2014.

[50] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[51] P. Vincent *et al.*, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, nr. 12, 2010.

[52] K. Cho, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[54] K. N. Haque, M. A. Yousuf, and R. Rana, "Image denoising and restoration with cnn-lstm encoder decoder with direct attention," *arXiv preprint arXiv:1801.05141*, 2018.

[55] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," i *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*. Springer, 2011, pgs. 52–59.

[56] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[57] O. Toolkit, "Anomalib github repository," https://github.com/openvinotoolkit/anomalib, n.d., accessed: September 12, 2024.

[58] O. Toolkit, "Anomalib documentation," https://anomalib.readthedocs.io/, n.d., accessed: September 12, 2024.

[59] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pgs. 19–31, 2016.

[60] S. Akcay *et al.*, "Anomalib: A deep learning library for anomaly detection," i *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, pgs. 1706–1710.

[61] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsuper-

vised anomaly detection," *International Journal of Computer Vision*, vol. 129, nr. 4, pgs. 1038–1059, DOI: . 10.1007/s11263-020-01400-4

[62] J. Silvestre-Blanes, T. Albero-Albero, I. Miralles, R. Pérez-Llorens, and J. Moreno, "Afid: a public fabric image database for defect detection," *AUTEX Research Journal*, vol. ahead-of-print, nr. No. 4, pgs. 1–10, DOI: . 10.2478/aut-2019-0035. [Online]. Available: https://content.sciendo.com/view/journals/aut/ahead-of-print/article-10.2478-aut-2019-0035.xml

[63] D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj, "Segmentation-Based Deep-Learning Approach for Surface-Defect Detection," *Journal of Intelligent Manufacturing*, DOI: . 10.1007/s10845-019-01476-x

[64] Z. M. Ahmad and M. B. Ahmad, "Elpv dataset: A large-scale dataset for solar panel defect detection," https://github.com/zae-bayern/elpv-dataset, 2018, accessed: 2024-10-09.

[65] C. Buerhop-Lutz *et al.*, "A benchmark for visual identification of defective solar cells in electroluminescence imagery," i *European PV Solar Energy Conference and Exhibition (EU PVSEC)*, 2018.

[66] S. Deitsch *et al.*, "Automatic classification of defective photovoltaic module cells in electroluminescence images," *Solar Energy*, vol. 185, pgs. 455–468, DOI: . 10.1016/j.solener.2019.02.067

[67] S. Deitsch *et al.*, "Segmentation of photovoltaic module cells in uncalibrated electroluminescence images," vol. 32, DOI: . 10.1007/s00138-021-01191-9

[68] M. Wieler, T. Hahn, and F. A. Hamprecht, "Weakly supervised learning for industrial optical inspection," [Dataset]. Retrieved from: https://hci.iwr.uni-heidelberg.de/content/weakly-supervised-learning-industrial-optical-inspection, 2007, accessed: 2024-10-09.

[69] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," i *International conference on information processing in medical imaging*. Springer, 2017, pgs. 146–157.

[70] D. Sculley *et al.*, "Hidden technical debt in machine learning systems," *Advances in neural information processing systems*, vol. 28, 2015.

[71] S. Chen, S. Bandaru, S. Marti, E. T. Bekar, and A. Skoogh, "Empirical comparison of unsupervised deep learning models for anomaly detection in images of sheet metal glue lines," *Available at SSRN 5019971*, 2024.

[72] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "Padim: a patch distribution modeling framework for anomaly detection and localization," i *International Conference on Pattern Recognition*. Springer, 2021, pgs. 475–489.

[73] "Raspberry pi camera module 3." Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/products/camera-module-3/, Retrieved: 2024-11-8.

[74] "Raspberry pi camera module 3 image." Electrokit Sweden AB. [Online]. Available: https://www.electrokit.com/raspberry-pi-kameramodul-3-12mp-75, Retrieved: 2024-11-8.

[75] "Raspberry pi camera cable." Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/products/camera-cable/, Retrieved: 2024-11-8.

[76] "Raspberry pi camera cable image." Electrokit Sweden AB. [Online]. Available: https://www.electrokit.com/raspberry-pi-kamerakabel-fpc-0.5mm-200mm?src=raspberrypi, Retrieved: 2024-11-8.

[77] "Raspberry pi 27w usb-c power supply." Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/products/27w-power-supply/, Retrieved: 2024-11-8.

[78] "Raspberry pi active cooler." Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/products/active-cooler/, Retrieved: 2024-11-8.

[79] "Zerotier documentaion." ZeroTier, Inc. [Online]. Available: https://docs.zerotier.com/start/, Retrieved: 2024-11-08.

[80] "Remote development using ssh." Visual Studio Code. [Online]. Available: https://code.visualstudio.com/docs/remote/ssh, Retrieved: 2024-11-08.

[81] O. Serradilla, E. Zugasti, J. Ramirez de Okariz, J. Rodriguez, and U. Zurutuza, "Adaptable and explainable predictive maintenance: Semi-supervised deep learning for anomaly detection and diagnosis in press machine data," *Applied Sciences*, vol. 11, nr. 16, pg. 7376, 2021.

[82] A. Jaramillo-Alcazar, J. Govea, and W. Villegas-Ch, "Anomaly detection in a smart industrial machinery plant using iot and machine learning," *Sensors*, vol. 23, nr. 19, pg. 8286, 2023.

[83] E. Weiss, "Advancements in electronic component assembly: Real-time ai-driven inspection techniques." *Electronics (2079-9292)*, vol. 13, nr. 18, 2024.

A

Appendix 1

Listing A.1: Python Script for Resizing Images

```
1
2  import os
3  from PIL import Image
4
5  # Define the path to the dataset
6  dataset_path = '/home/pi/Desktop/Dataset'
7
8  # Define the target size
9  target_size = (900, 900)
10
11 # Function to resize an image
12 def resize_image(image_path, target_size):
13     with Image.open(image_path) as img:
14         # Resize the image
15         img = img.resize(target_size, Image.ANTIALIAS)
16         # Save it back to the same path
17         img.save(image_path)
18
19 # Walk through all directories and files in the dataset path
20 for root, dirs, files in os.walk(dataset_path):
21     for file in files:
22         # Create the full path to the file
23         file_path = os.path.join(root, file)
24         # Check if the file is an image
25         if file.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp',
               '.gif', '.tiff')):
26             try:
27                 # Resize the image
28                 resize_image(file_path, target_size)
29                 print(f"Resized: {file_path}")
30             except Exception as e:
31                 print(f"Failed to resize {file_path}: {e}")
```

Listing A.2: Python Script for Renaming Images

```
1
2  import os
3
4  # Define the root directory of the dataset
5  root_dir = '/home/pi/Desktop/Dataset'
6
7  # Define valid image extensions
8  valid_extensions = ('.jpg', '.jpeg', '.png', '.bmp', '.tiff', '.gif
       ')
9
10 # Loop through all the subdirectories in the root directory
11 for subdir, dirs, files in os.walk(root_dir):
12     # Get the folder name
13     folder_name = os.path.basename(subdir)
14
15     # Filter and sort files to only include images
16     images = [f for f in files if f.lower().endswith(
           valid_extensions)]
17     images.sort()
18
19     # Loop through each image in the folder
20     for i, filename in enumerate(images):
21         # Define the new filename with folder name and zero-padded
               index
22         new_name = f"{folder_name}{i:03d}{os.path.splitext(filename
               )[1]}"
23
24         # Get the full old and new paths
25         old_path = os.path.join(subdir, filename)
26         new_path = os.path.join(subdir, new_name)
27
28         # Rename the file
29         os.rename(old_path, new_path)
30         print(f"Renamed: {old_path} -> {new_path}")
```

Normal sample

Small anomaly

Medium anomaly

Large anomaly

Figure A.1: Examples of dataset images. Complete dataset can be found here[**?**].

Figure A.2: Raspberry Pi 5 with camera module

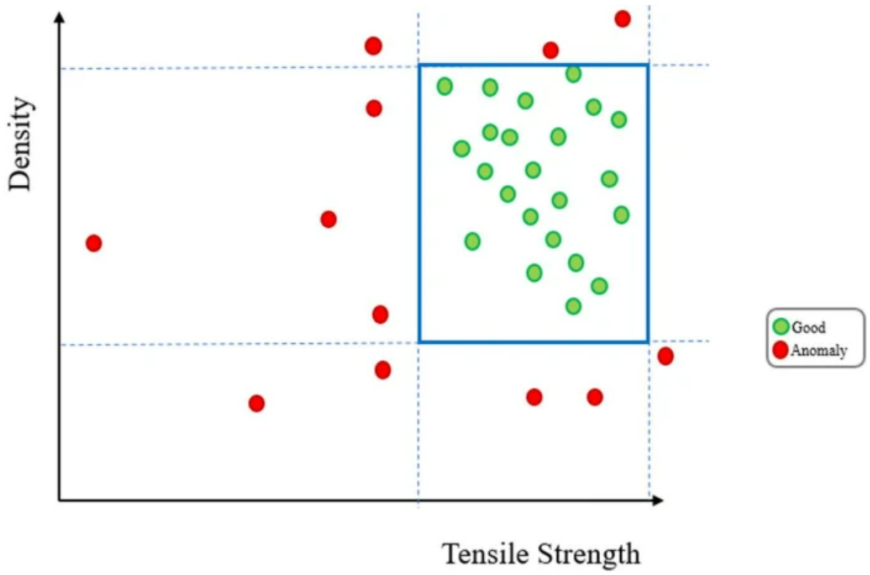


Figure A.3: Example of data points identified as "good" and "anomalies"

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY