



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ÚSTAV AUTOMATIZACE A INFORMATIKY

VISUAL ANOMALY DETECTION OF INTAKE MODULE PRE-FILTERS USING UNSUPERVISED DEEP LEARNING

VIZUÁLNÍ DETEKCE ANOMÁLIÍ PŘEDFILTRŮ SACÍCH MODULŮ HLUBOKÝM UČENÍM BEZ UČITELE

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Josef Hruška

SUPERVISOR

VEDOUCÍ PRÁCE

doc. Ing. Pavel Škrabánek, Ph.D.

BRNO 2025

Assignment Bachelor's Thesis

Institut: Institute of Automation and Computer Science
Student: **Josef Hruška**
Degree program: Fundamentals of Mechanical Engineering
Branch: Fundamentals of Mechanical Engineering
Supervisor: **doc. Ing. Pavel Škrabánek, Ph.D.**
Academic year: 2024/25

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Bachelor's Thesis:

Visual Anomaly Detection of Intake Module Pre-filters Using Unsupervised Deep Learning

Brief Description:

Visual inspection of complex objects is a rapidly developing field with a range of real-world applications. This progress is enabled by the availability of powerful hardware and related advancements in the development of neural network topologies designed for image data processing. Modern neural network architectures can detect anomalies in image data with high accuracy, even when capturing complex objects in intricate environments. It can therefore be assumed that they will achieve similar precision in the visual inspection of geometrically complex plastic parts, such as intake module pre-filters with integrated grids in automobiles, which remove large particles from the incoming air.

Bachelor's Thesis goals:

The student will conduct a literature review mapping the issues of anomaly detection in image data using unsupervised deep learning. Furthermore, the student will study the topic of data augmentation. The student will then design and implement one or more neural network topologies for anomaly detection in plastic modules. For training and evaluation of these topologies, the student will create and annotate a sufficiently large and representative dataset. Subsequently, the student will evaluate the quality of the outputs of the proposed topologies on this dataset.

Recommended bibliography:

LIU, Jiaqi; XIE, Guoyang; WANG, Jinbao; LI, Shangnian; WANG, Chengjie et al. Deep Industrial Image Anomaly Detection: A Survey. Online. Machine Intelligence Research. 2024, roč. 21, č. 1, s. 104-135. ISSN 2731-538X. Dostupné z: <https://doi.org/10.1007/s11633-023-1459-z>. [cit. 2024-10-17].

HAYASHI, Toshitaka; CIMR, Dalibor; FUJITA, Hamido a CIMLER, Richard. Critical Review for One-class Classification: recent advances and the reality behind them. Online. 2024, roč. 2024, č. 1, s. 45. Dostupné z: <https://arxiv.org/abs/2404.17931>. [cit. 2024-10-17].

Deadline for submission Bachelor's Thesis is given by the Schedule of the Academic year 2024/25

In Brno,

L. S.

doc. Ing. Zdeněk Hadaš, Ph.D.
Director of the Institute

doc. Ing. Jiří Hlinka, Ph.D.
FME dean

ABSTRACT

Unsupervised deep learning methods can detect anomalies by learning patterns exclusively from normal, non-anomalous data. In industry, where anomalous data are rare, this approach is particularly promising for visual defect detection on manufactured parts. This thesis provides an overview of image anomaly detection methods using unsupervised deep learning, with the primary goal of developing a custom solution for detecting defects on a mass-produced plastic pre-filter. To ensure robust training and evaluation of the models, an extensive dataset of both normal and anomalous pre-filter images is collected. The anomalous images are then annotated to create binary masks for pixel-level evaluation. Two models are developed. The first model is an author's custom autoencoder based on direct image reconstruction, and second is a modified feature reconstruction model built on an existing architecture. The accuracy of the models is evaluated both at the image classification and at the pixel level. The best performing model is implemented for further experimental testing.

ABSTRAKT

Metody hlubokého učení bez učitele dokážou detekovat anomálie učním se vzorců výhradně z normálních, neanomálních dat. V průmyslu, kde jsou anomální data vzácná, má tento přístup potenciál zejména při vizuální detekci vad na vyrobených dílech. Tato práce se zaměřuje na souhrn metod pro detekci anomálií obrazu pomocí hlubokého učení bez učitele, s hlavním cílem vyvinout vlastní řešení pro detekci vad na sériově vyráběném plastovém předfiltru. Pro zajištění robustního trénování a hodnocení modelů je shromážděna rozsáhlá datová sada, obsahující normální i anomální snímky předfiltru. Snímky vad jsou dále ručně anotovány s cílem vytvořit binární masky k vyhodnocení přesnosti na úrovni pixelů. Vyvinuty jsou dva modely. První model je autorův vlastní autoenkodér, založen na přímé rekonstrukci obrazu, a druhý model upravuje existující architekturu založenou na rekonstrukci příznaků. Přesnost modelů je posouzena na úrovni klasifikace obrázků i pixelů. Model s nejlepšími výsledky je implementován pro další experimentální testování.

KEYWORDS

Unsupervised deep learning, visual quality inspection, image anomaly detection

KLÍČOVÁ SLOVA

Hluboké učení bez učitele, vizuální kontrola kvality, obrazová detekce anomálií



INSTITUTE OF AUTOMATION
AND COMPUTER SCIENCE



2025

BIBLIOGRAPHIC CITATION

HRUŠKA, Josef. *Visual Anomaly Detection of Intake Module Pre-filters Using Unsupervised Deep Learning*. Brno, 2025. Available at: <https://www.vut.cz/studenti/zav-prace/detail/165885>. Bachelor's Thesis. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Supervised by doc. Ing. Pavel Škrabánek, Ph.D.

ROZŠÍŘENÝ ABSTRAKT

Úvod

Vizuální systémy v průmyslu se neustále vyvíjejí a významně přispívají k automatizaci mnoha výrobních procesů. Nejčastěji nacházejí uplatnění při kontrole přítomnosti součástek na sestavě nebo při měření vzdáleností. Tyto úlohy lze poměrně snadno automatizovat pomocí konvenčních algoritmů založených na exaktních pravidlech.

Vedle těchto běžných úkolů však existují i výrazně složitější problémy strojového vidění, jejichž plná automatizace je obtížná. Takové úkoly jsou ve většině případů stále řešeny lidským operátorem. Typickým příkladem je vizuální kontrola komplexních dílů nebo sestav na konci výrobního procesu. Díl může být označen jako dobrý, pouze, pokud splňuje veškeré požadované náležitosti a je tedy způsobilý k zabalení a dalšímu zpracování. Posouzení kvality dílu je však komplikované zejména kvůli kosmetickým vadám, které mohou mít různorodý tvar, velikost a nacházet se kdekoli na díle. Závažnější funkční vady jsou obvykle odhaleny dříve ve výrobním procesu, zatímco drobné vizuální nedostatky často zůstávají neodhaleny až do vizuální kontroly na konci procesu. Pro automatizaci tohoto druhu kontroly je třeba využít přístupy hlubokého učení. Velmi vhodnou metodou je učení bez učitele, kdy je model trénován pouze na datech bez vad – v průmyslové praxi totiž často není k dispozici dostatečné množství vadných snímků potřebných pro učení s učitelem. Taková řešení již existují i ve formě komerčních systémů, které využívají kolaborativní roboty a výkonné průmyslové počítače. Tato zařízení dosahují vysoké přesnosti a flexibility. Jejich nevýhodou však zůstává vysoká pořizovací cena, která může zkomplikovat návratnost investice v rozumném čase.

V reakci na tento fakt se tato práce zaměřuje na vývoj jednoduššího a dostupnějšího řešení. Nejprve představuje teoretické základy průmyslových vision systémů a hlubokého učení, následované rešerší aktuálních přístupů k detekci vizuálních anomálií metodami učení bez učitele. Hlavní část je věnována návrhu experimentálního systému pro detekci povrchových vad plastových předfiltrů pomocí hlubokých neuronových sítí. Cílem je vytvořit a model schopný detekovat anomálie na horní ploše předfiltru bez nutnosti trénovat model i na snímcích defektů.

Popis řešení

Prvním krokem je nastavení kamery, světelného boxu a definice metodiky sběru dat. Snímky jsou pořizovány přímo v provozu. Autor vybíral jednotlivé díly z výrobní linky, prováděl manuální vizuální inspekci a označoval je jako dobré nebo vadné. Díly bez defektů byly snímány pětkrát v různých natočeních a náklonech pro zvýšení variability. Vadné díly, vzhledem k jejich nižšímu počtu, byly snímány sedmkrát. Každý díl byl zachycen v

pěti různých úrovních přiblížení. Cílem bylo vytvořit dvě varianty datasetu: Dataset-A (obsahující většinou celý díl i s okrajovými úchyty) a Dataset-B (zaměřený na střední část, kde se vady nejčastěji vyskytují).

Po sesbírání datasetu následovalo předzpracování snímků testovací sady. Zahrnovalo adaptivní ekvalizaci histogramu a další metody pro zvýraznění vizuálních vad. Trénovací sada byla rozšířena pomocí augmentace, jako je rotace, zrcadlení, šum a úprava kontrastu. Malá část trénovacího datasetu byla upravena přesnou technikou zvýraznění vad shodnou s testovací sadou. Tím se zvýšila robustnost modelu – model se setkal i s malou částí dat, které byly vizuální úpravou podobné testovací sadě. Dále byly vadné snímky ručně anotovány a byly vytvořeny binární masky vyznačující oblasti s anomáliemi.

Nakonec byly navrženy a natrénovány dva modely v prostředí PyTorch. Oba využívaly princip rekonstrukce, kdy je model trénován na normálních datech a na základě rozdílu mezi vstupem a rekonstrukcí odhaduje přítomnost vady. První model je autorem vyvinutý autoenkodér využívající mechanismy attention a reziduální spojení, který rekonstruoval přímo RGB obraz. Druhý model je převzatý z již existujícího řešení, ale jeho části jsou nově modifikovány. Tento model rekonstruuje příznaky (features) extrahované předtrénovanou sítí EfficientNet-B6.

Výsledky a závěr

Oba modely byly vyhodnoceny z hlediska klasifikační přesnosti (rozdělení na dobré a vadné snímky) a přesnosti lokalizace anomálií na úrovni pixelů, přičemž výsledky byly posouzeny samostatně pro každou datovou sadu. Oba modely byly trénovány i testovány na totožných datech. K vyhodnocení přesnosti modelů byla použita metrika obsahu pod *Receiver Operating Characteristic* (ROC) křivkou, jinak také *Area Under Receiver Operating Characteristic Curve* (AUROC).

První model dosáhl nízké klasifikační úspěšnosti, AUROC 0.61 pro Dataset-A a 0.63 pro Dataset-B. Přestože pixelová přesnost byla pro Dataset-A na první pohled vysoká (AUROC 0.85), analýza ukázala, že model dobře rekonstruoval pouze pozadí snímků a celý díl označoval jako vadný. U Datasetu-B, kde je díl blíže, a pozadí zabírá daleko menší část fotky, dosáhl model podstatně nižší výkonnosti AUROC 0,58. To jen potvrdilo předchozí analýzu. Autorem vyvinutý autoenkodér se tedy ukázal jako nedostatečně přesný a nevhodný pro implementaci. Jeho neúspěch lze pravděpodobně připsat slabé generalizaci způsobené přílišnou komplexitou dat a nedostatečně přizpůsobenou chybovou funkcí při tréninku.

Druhý model, využívající extrahované příznaky z předtrénované konvoluční neuronové sítě EfficientNet-B6, poskytl výrazně lepší výsledky. Byly testovány varianty s výstupními mapami příznaků velikosti 128×128 a 170×170 . Model s větší výstupní mapou podal lepší výsledky na úrovni klasifikace celých snímků (AUROC 0,86 pro Dataset-A

a 0,92 pro Dataset-B). Model s menší mapou byl ale lepší v lokalizaci anomálie na úrovni pixelů (AUROC 0,89 pro Dataset-A a 0,82 pro Dataset-B). Tento výsledek může na první pohled působit neintuitivně, avšak existuje pravděpodobné vysvětlení. Model s menší velikostí výstupní mapy generoval prostorově hladší a méně detailní rekonstrukce, čímž potlačil jemné odchylky v obraze. Pravděpodobně, se tak méně zaměřoval na drobné vizuální detaily což je často i náhodný šum nebo nepodstatné obrazové prvky. Díky tomuto pravděpodobně model dosáhl vyšších hodnot přesnosti na úrovni pixelů. Tato výhoda modelu s menší výstupní mapou příznaků však byla vykoupena ztrátou citlivosti na velmi malé nebo nenápadné vady. To se následně projevilo v mírně nižší přesnosti klasifikace na úrovni celých snímků.

Model s mapou 170×170 byl zvolen pro finální implementaci, protože dosáhl nejvyšší přesnosti klasifikace celých snímků, což je pro průmysl nejpodstatnější. Implementace je úspěšnou demonstrací potenciálu hlubokého učení bez učitele pro detekci anomálií na vizuálních datech.

AUTHOR'S DECLARATION

I hereby declare that this bachelor thesis was prepared as an original author's work under the supervision of doc. Ing. Pavel Škrabánek, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

As the author of the mentioned work, I further declare that in connection with the creation of this work, I have not infringed on the copyrights of third parties, especially I have not interfered in an unauthorized manner with the personal copyrights of others, and I am fully aware of the consequences of violating the provisions of Section 11 and subsequent sections of Copyright Act No. 121/2000 Coll., including possible criminal consequences.

In Brno, 20. 5. 2025

.....

Josef Hruška

ACKNOWLEDGEMENT

I would like to express my gratitude to doc. Ing. Pavel Škrabánek, Ph.D. for his valuable advice and methodological guidance of the work. I also thank MANN+HUMMEL for providing the necessary hardware for the tasks associated with the practical part of the bachelor's thesis. Specifically, I would like to name the following colleagues from the manufacturing engineering department: Ing. Lucie Svobodová for her assistance in ordering essential accessories, Ing. Miroslav Čada for overall support in setting up the workplace, Ondřej Jarkovský and Jaroslav Kosmák for their support in obtaining the plastic parts that are the subject of detection, Ing. Miloš Obůrka for motivation to write this thesis even though the results were uncertain. Finally, I thank Jaroslav Brabenec and Bohuslav Hladký for approving the entire project.

CONTENTS

1	INTRODUCTION	13
2	VISUAL INSPECTION IN THE INDUSTRY	14
2.1	Automatability of Industrial Visual Inspections	14
2.2	Lighting	16
2.3	Types of Machine Vision Systems	16
2.4	Image Enhancement and Preprocessing Algorithms	18
2.5	Feature Extraction and Analysis Algorithms	20
2.5.1	Edge Detection	21
2.5.2	Connected Components Analysis	22
2.5.3	Hough Transform	22
2.5.4	SIFT and SURF	23
2.6	Industrial Vision Tools and Applications	24
2.7	Industrial Vision Tools for Complex Inspection Scenarios	26
3	DEEP LEARNING FOR VISUAL TASKS	27
3.1	Deep Learning Fundamentals	27
3.2	Convolutional Neural Networks	33
3.3	Key Pretrained CNNs and Transfer Learning	34
3.4	Vision Transformers	36
3.5	Data Augmentation	37
4	IMAGE ANOMALY DETECTION WITH UNSUPERVISED LEARNING	39
4.1	Reconstruction Based Methods	39
4.1.1	Autoencoders and Variational Autoencoders	40
4.1.2	Generative Adversarial Networks	41
4.1.3	Latent Space Estimation with PCA	42
4.2	Feature Embedding Methods	42
4.2.1	Distribution Map	43
4.2.2	Student-Teacher Methods	44
4.2.3	One-Class Classification	44
4.2.4	Memory Bank Methods	45
4.3	Evaluation Metrics for Image Anomaly Detection	46
5	EXPERIMENTAL PART	48
5.1	Pre-filter Component	48
5.2	Experimental Setup	50
5.3	Data Acquisition	56
5.4	Data Preprocessing	59
5.5	Development of Models	61

5.5.1	Methodology	62
5.5.2	Baseline Autoencoder with Attention	63
5.5.3	Deep Feature Reconstruction Autoencoder	65
5.6	Final Results	69
5.6.1	Results of Baseline Autoencoder with Attention	69
5.6.2	Results of the Deep Feature Reconstruction Autoencoder	71
6	CONCLUSION	76
	BIBLIOGRAPHY	77
	SYMBOLS AND ABBREVIATIONS	83
	LIST OF FIGURES	85
	LIST OF TABLES	87
	LIST OF APPENDICES	88
A	Attachment	89

1 INTRODUCTION

Quality control is a crucial part of production. Even as manufacturing becomes more advanced and automated each year, perfect accuracy remains out of reach, as some parts will always deviate from tolerances. Detecting and removing these parts is necessary to prevent customers from receiving faulty products.

In mass production, image-based quality control usually targets straightforward tasks, such as verifying the presence of components or measuring distances. For example, it may check whether a rubber gasket is fit in a housing or if a screw is placed correctly. These tasks usually rely on rule-based algorithms which are highly accurate for their specific purpose but also highly sensitive to environmental conditions and any deviations from the specific task.

However, some inspection tasks are more complex and difficult to address using rule-based methods. One such task is the final quality inspection of an assembly at the end of the production line. This step is often difficult to automate and remains largely dependent on human operators. Operators must identify any functional or visual anomalies, often working long shifts, which makes the task prone to fatigue.

For automating these tasks, deep learning offers significant potential due to its generalization capabilities. Unsupervised learning is particularly useful because it does not require defective samples for training. There are commercial solutions for automated anomaly detection using powerful computing centers and collaborative robots, but they may suffer from a slow return on investment.

This thesis aims to address the challenge by developing a more cost-effective solution that will demonstrate the potential of unsupervised deep learning. The main objective is to create an unsupervised image anomaly detection solution for a mass-produced plastic pre-filter part, evaluate its performance, and implement the best-performing model for experimental testing. The thesis first introduces the necessary theoretical background in industrial visual inspection. Then it presents deep learning and neural networks introduction, with a particular focus on convolutional neural networks. Special attention is then given to unsupervised image anomaly detection methods. In addition, data augmentation and evaluation metrics are discussed.

The experimental part describes the development of an unsupervised image anomaly detection system that identifies surface anomalies on plastic pre-filter parts. First, it presents an experimental setup for part imaging, followed by data acquisition and pre-processing pipeline. The development of two unsupervised anomaly detection models is then described. These models are trained and tested on two datasets: one showing the part as a whole or nearly whole, and the other focusing only on the central area, where anomalies typically occur.

2 VISUAL INSPECTION IN THE INDUSTRY

This chapter presents various use cases of visual inspection and their automation in industry. Special attention is paid to human inspection processes and their inherent limitations. The chapter then provides a thorough description of machine vision systems, covering both the hardware involved and the software algorithms that drive these systems.

2.1 Automatability of Industrial Visual Inspections

Visual inspection tasks can be divided into routine and challenging categories based on their potential for automation. Many routine tasks are already highly automated, surpassing human performance. However, some tasks remain difficult to automate, especially when using only conventional rule-based algorithms.

Routine Tasks Suitable for Automation

Routine visual inspection tasks are easily automatable and have been widely implemented in industry for many years. The most significant ones are: detection of presence and absence, measurements of various distances and angles, scanning of bar codes or DataMatrix codes, *Optical Character Recognition* (OCR) and *Optical Character Verification* (OCV).

The presence and absence inspection tasks are used to ensure that all required components or features are present on a product before it continues down the production line. Common examples include: checking the presence of electronic components on *Printed Circuit Board* (PCB), verifying that screws or washers are installed to secure parts, and confirming the application of adhesives in assembly processes. [1, 2]

Measurement of a distance involves determining the distance between two points or objects within an image. They typically measure part dimensions and compare them with pre-defined tolerances. For example, Zhang *et al.* [3] demonstrated nearly perfect accuracy in measuring the dimensions of the spring clamp. This shows that measurement tasks can be reliably automated. [1, 4]

Barcode and DataMatrix code reading are also highly automated due to the availability of specialized reading devices. These devices are engineered for the fast and reliable detection of codes and printed text. That makes identification and verification tasks straightforward and robust [5]. For example, Cognex readers scan DataMatrix codes with an error rate of less than 1 in 10 million scanned characters [6].

OCR and OCV identify and interpret text in images, with a key difference. OCR reads text to trigger another action, while OCV is used to verify the quality of text against a known standard [4]. The initial OCR tasks used techniques such as template matching, where the pre-defined template was aligned and compared to the input [7]. Although precise under controlled conditions, these methods struggled with font variability, noise,

and lighting changes, leading to a degradation of accuracy [8]. With the rise of machine learning, approaches became much more robust to changing fonts and surrounding conditions; an example is Tesseract 4.0 [9]. Those approaches benefit from a substantial amount of annotated training data available, such as public text datasets (e.g. ICDAR) [10].

The algorithms and mechanics behind the automation of these tasks are explained in further sections on image algorithms 2.4 and 2.5.

Challenging Scenarios for Automation

In today's production, some tasks remain very difficult to automate. The main challenges include variable lighting or background conditions, detection of subtle defects, and inspection of final assemblies. The final assemblies are often complex in shape and contain many individual parts. Each of these parts can present unexpected defects, located anywhere, with no specific *Region of Interest* (ROI) for camera to check.

Such tasks are generally unsolvable by conventional rule-based algorithms or traditional machine learning methods. Instead, they require deep learning approaches. For example, [11] demonstrates the successful use of deep learning for detecting micro-level cracks. Article [12] demonstrates the deep learning approach in fabric defect detection.

Human Inspection

Previously mentioned challenging tasks are still carried out by humans in most cases. With human operators come unavoidable limitations. Human visual inspection in industrial settings is widely accepted to have an accuracy rate of approximately 80%. It is strongly influenced by factors such as fatigue, lighting conditions, and duration of the shift. Study from Muraoka [13] indicate a 10-15% decline in accuracy, after just one hour of continuous inspection due to cognitive exhaustion. The time constraints of inspection also play a significant role. As Judi E. [14] suggests, there is a time paradox in visual inspection. It means that longer inspection times tend to increase false alarms due to over-scrutiny, while shorter inspection times lead to more missed defects from rushed decisions.

Visual inspection is a mentally demanding task that requires significant effort from the inspectors. As inspection time increases both, correct detections and errors increase, while confidence of the inspector declines. Paradoxically, those who make more false alarms experience a higher subjective workload and distress, but also report a lower engagement and greater worry. [14]

In conclusion, these findings clearly demonstrate that human visual inspection is labor intensive and psychologically demanding. Its performance is limited by fatigue, time pressure, and other factors. The accuracy achieved by human inspectors is not particularly high and decreases further under prolonged or stressful conditions. Given these limitations, there is a compelling need to advance the automation of such inspection tasks to improve reliability, consistency, and overall efficiency.

2.2 Lighting

Before diving into cameras and algorithms, lighting needs to be discussed as it is an essential component of every machine vision system. Lighting selection is critical for optimizing image contrast and feature extraction. Red lighting (600–700 nm) is frequently used because of its alignment with the spectral sensitivity peaks of camera sensors, particularly in monochrome systems [15]. There are few essential lighting configurations (see Fig. 1):

- **Bright field forward lighting:** Characterized by direct illumination at (45–90°) angles, it provides high contrast for surface features, but introduces specular reflections on glossy surfaces.
- **Dark field forward lighting:** Using oblique angles (10–30°), it captures diffuse scattering from texture variations, making it ideal for detecting scratches.
- **Coaxial forward lighting:** Directs light through a beam splitter along the optical axis of the camera, eliminating reflections from flat, reflective surfaces.
- **Scattering forward lighting with dome structure:** Generates uniform and diffuse illumination by reflecting light off a hemispherical surface, mimicking overcast daylight. This method minimizes directional reflections; see Fig. 2.
- **Back-lighting:** Positioned behind translucent or opaque objects, it produces high-contrast silhouettes for dimensional gauging or internal inspection of transparent materials.

Each technique balances trade-offs between contrast, reflection management, and geometric compatibility. The selection of the appropriate lighting method directly affects the detectability of defects. [15, 16]

2.3 Types of Machine Vision Systems

Machine vision systems consist of hardware and software components. Hardware typically includes a camera, an illumination system, and a computing device for data processing. The software includes camera configuration, image acquisition tools, and algorithms designed to perform specific tasks such as measurement. Machine vision systems can be classified according to their system architecture and processing capabilities. This section distinguishes three principal categories: readers and sensors, smart cameras, and computer-based camera systems. Each of these categories is represented by flagship products from Keyence and Cognex, two of the most established brands in high-volume manufacturing. Their systems are valued for their robust integration capabilities, intuitive user interfaces, and flexibility.

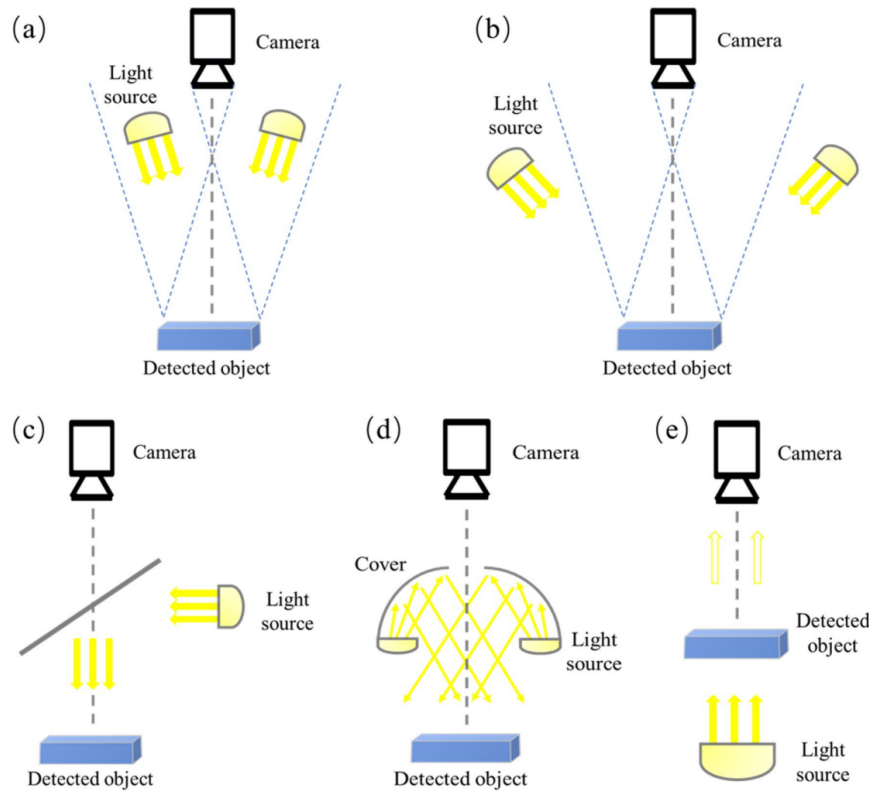


Fig. 1: Schematic diagram of typical illumination modes: a) Bright field forward lighting, b) Dark field forward lighting, c) Coaxial forward lighting, d) Scattering forward lighting of dome structure, e) Back lighting [16]

Readers and Smart Sensors

These systems integrate a sensor, optical components, and basic processing capabilities into a compact unit designed for simple inspection tasks. Typical applications include barcode reading, OCR, and presence or absence detection.

For example, Keyence IV-4 is a descent choice for presence/absence checks, OCR, simple measurements, counting, and alignment [17]. It is capable of covering most of the ordinary tasks in machine vision. Integration into production environments is straightforward, with the ability to transmit results via digital signals (24V) or EtherNet/IP tags to *Programmable Logic Controller* (PLC). Similarly for readers, the Cognex DataMan series is widely recognized for its high-speed barcode and 2D code reading capabilities [18].

Smart Cameras

Smart cameras integrate a camera and a more powerful processor within a single housing, allowing for more advanced inspection tasks to be performed. Such tasks can be pattern matching, precise dimensional gauging, and defect detection with the help of artificial in-



Fig. 2: Comparison of reflective surface illumination: ordinary bright field forward lighting (left) and a lighting with dome structure (right) [15]

telligence.¹ They also offer higher resolutions and optical zoom capabilities. They process images internally and communicate the results directly to PLCs or factory networks. The Keyence VS series is a great example of a flexible high-accuracy inspection camera that also supports AI-based defect detection. [19]

Computer-Based Camera Systems

In these systems, image capture is separated from processing. Cameras transmit raw image data to an external computer or vision controller, where advanced algorithms and deep learning models are executed. This architecture supports complex, multi-camera applications. Keyence CV-X system allows for the centralized processing of multiple cameras and advanced inspection tasks [20].

Although not a complete camera system, Cognex VisionPro is also highly relevant. As standalone software, it can be used with any sufficiently powerful industrial computer and camera setup. It enables complex AI-based defect detection with dedicated deep unsupervised learning capabilities [21].

This computer-based approach is widely used by smaller providers that dedicate their products to complicated inspections, custom solutions, or research applications. Such systems are particularly advantageous when inspection scenarios require an integration of deep learning. [22]

2.4 Image Enhancement and Preprocessing Algorithms

Preprocessing techniques standardize the input data to improve subsequent analysis. These algorithms suppress noise, enhance contrast, and correct geometric properties. There are a lot of preprocessing algorithms, but the focus will be turned towards the most relevant ones for this thesis.

¹ This level of artificial intelligence, often referred to as “edge learning” in the industry, uses simple pretrained models with minimal training [19], offering general-purpose defect detection but is not meant for very specific challenging applications.

Filtering Techniques

Filtering is generally used to enhance the details of the image. Filters can blur or sharpen the entire image and can also smooth out or highlight just the edges that the image contains. They can also mitigate noise in the image. There are two main types of filters: spatial and frequency-based. [23]

Spatial filtering operates directly on pixel neighborhoods, analyzing neighboring pixels. Examples of spatial filters include: Gaussian filters for noise reduction and median filters for salt-and-pepper noise removal. [23, 24]

The Gaussian spatial filter is a widely used linear smoothing filter, especially for noise reduction and preprocessing before edge detection. It convolves the image with a kernel shaped like a bell curve, assigning higher weights to pixels near the center and lower weights to those further away. The equation (1) presents two-dimensional gaussian filter.

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{(-\frac{x^2+y^2}{2\sigma^2})} \quad (1)$$

where:

- x, y are spatial coordinates, measured in pixels from kernel center,
- σ is standard deviation, which controls blur intensity,
- $\frac{1}{2\pi\sigma^2}$ is normalization factor, that ensures the filter preserves image brightness during convolution.

Frequency domain filtering modifies images by operating on their spectral representations, offering precise control over spatial frequency components. This approach begins with applying the Fourier transform to convert an image from the spatial domain to the frequency domain. Here, pixel intensity variations are expressed as sinusoidal waves. After these frequency-based modifications, the inverse Fourier transform reconstructs the processed image back into the spatial domain. [25, 23]

Types of frequency filters include low-pass filters, which blur images by suppressing high frequencies, and high-pass filters, which enhance edges by blocking low frequencies. Both filter types can be implemented using Gaussian or Butterworth designs [23, 24]. For example, a Gaussian low-pass filter in the frequency domain is defined in the following equation (2):

$$P(u, v) = e^{-F^2(u, v)/2\sigma^2} \quad (2)$$

where:

- u, v are frequency coordinates, measured from the origin of the frequency spectrum,
- $F(u, v)$ represents the distance from origin in frequency space, typically calculated as $F(u, v) = \sqrt{(u - u_0)^2 + (v - v_0)^2}$ where u_0, v_0 is the center of the frequency spectrum,

- σ is standard deviation, which controls the cutoff frequency; smaller values create narrower filters that pass/block fewer frequencies.

Contrast Enhancement

Histogram equalization is a fundamental technique for enhancing image contrast. It works by redistributing the intensity values to use the full dynamic range. The transformation applies the same mapping function to every pixel, regardless of its position in the image. This approach works well for images with uniform lighting conditions. However, the results for images with varying brightness regions are not satisfactory as this global approach cannot adapt to local contrast requirements.

When processing images with varying luminance levels in different regions, the *Contrast Limited Adaptive Histogram Equalization* (CLAHE) offers better performance. It addresses this limitation by performing localized contrast enhancement. Unlike global approaches that create noticeable boundary artifacts, CLAHE uses a more sophisticated technique. The "Adaptive" in the abbreviation means it divides the image into certain number of square non-overlapping patches and then computes histogram equalization functions for these patches. Afterwards, it smoothly interpolates these functions between neighboring patches. This interpolation uses bilinear blending based on the relative position of a pixel within its patch and effectively eliminates harsh transitions. The "contrast limited" aspect of CLAHE prevents over-amplification of noise by setting a maximum slope for the transformation function. [24]

Image Resizing and Interpolation Methods

When resizing images or zooming in, interpolation is necessary to estimate pixel values at new positions, minimizing visual artifacts. The three most common interpolation techniques for raster images are nearest-neighbor, bilinear, and bicubic interpolation. These methods are standard for image resizing and differ in how they calculate pixel values at non-integer coordinates, that is, locations that do not correspond exactly to the original pixel grid. [23]

The nearest neighbor simply assigns the value of the closest pixel, resulting in a fast but blocky output. Bilinear interpolation computes a weighted average of the four nearest pixels, providing smoother results. Bicubic interpolation uses 16 surrounding pixels for even smoother and sharper images, at a higher computational cost. The comparison of those three interpolation methods can be seen in Fig. 3. [23]

2.5 Feature Extraction and Analysis Algorithms

Following algorithms are core components of all machine vision systems in the industry. These algorithms enable machines to interpret and process visual data by identifying

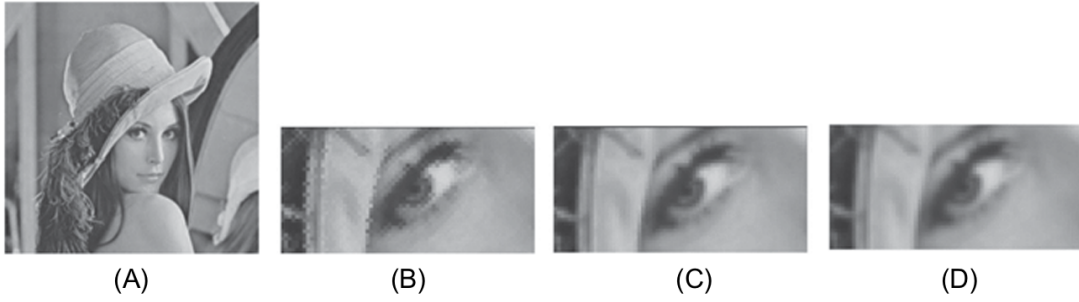


Fig. 3: Comparison of interpolation methods: (A) Original image, (B) Nearest-neighbor interpolation, (C) Bilinear interpolation, and (D) Bicubic interpolation. [23]

meaningful patterns, structures, and boundaries within images. Since this thesis focuses primarily on the quality inspection of 2D images, only algorithms applicable to two-dimensional data will be discussed.

2.5.1 Edge Detection

Edge detection is a fundamental technique in image processing that identifies boundaries and significant transitions in the intensity of the pixels. It is crucial for tasks such as segmentation, object recognition, or measurements. By detecting edges, the volume of data to process is reduced, while preserving essential structural information. The most relevant types of kernel operator for edge detection are discussed in more detail.

Prewitt and Sobel Operators

The Prewitt and Sobel operators detect edges by computing horizontal and vertical gradients using 3×3 convolution kernels (G_x for horizontal gradient – vertical edges and G_y for vertical gradient – horizontal edges). For Prewitt, $a = 1$ and for Sobel $a = 2$ for the matrices in equation (3).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -a & 0 & a \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -a & -1 \\ 0 & 0 & 0 \\ 1 & a & 1 \end{bmatrix} \quad (3)$$

The Prewitt operator uses simple integer-valued masks and is computationally less expensive. It is effective for detecting edges in images with good contrast and little noise, but is more sensitive to noise and may produce thicker edges. The Prewitt is often chosen for its simplicity and speed. [25, 26]

The Sobel operator modifies the Prewitt kernels by giving more weight to the central pixels, which improves noise suppression and edge localization. As a result, Sobel tends to find more edges and makes them more visible compared to Prewitt, especially in noisy or low-contrast images. As a trade-off it is more computationally demanding than Prewitt. [25, 26]

Canny Edge Detector

The Canny edge detector is a multistage process. It involves first smoothing the image from noise using filters such as the Gaussian from equation (1). Afterwards, the Sobel and Prewitt methods are applied, and gradients are computed. The edges are then suppressed into just one pixel lines [25]. The actual and potential edges are then determined by use of double threshold.² The last step is edge tracking by hysteresis: edges that are not connected with strong bonds are suppressed. This algorithm generally outperforms plain Prewitt and Sobel, but again comes at an increased computational cost. [26]

2.5.2 Connected Components Analysis

Some preprocessing algorithms typically output a single region that may contain multiple distinct objects. To separate these individual objects, connected component analysis is applied. This process identifies groups of pixels that are connected according to a specific adjacency criterion. Then it labels each distinct object as a separate component. It is important to define when two pixels are considered connected:

- **4-connectivity:** Two pixels are connected if they share a common edge. That is, one pixel is directly above, below, left, or right of the other. Thus, each pixel has up to four neighbors.
- **8-connectivity:** This also includes diagonal neighbors, so pixels that share a corner are also considered connected. Each pixel can have up to eight neighbors.

The choice between 4- and 8-connectivity affects the shape and number of connected components detected. As Steger [25] suggests, using the same connectivity definition for both the foreground and background regions can lead to counterintuitive results. Such results include merged background components around rotated foreground lines or misclassifications of holes. These inconsistencies occur because symmetric connectivity rules cannot tell adjacent regions apart from overlapping ones under geometric transformations. To resolve this, opposite connectivity for foreground and background must be applied to maintain logical separation of regions.

2.5.3 Hough Transform

The Hough transform, originally proposed by Hough (1962), is a robust feature extraction technique designed to identify parametric shapes such as lines, circles, and ellipses. It operates on edge-detected images (e.g., preprocessed using the Canny detector) by transforming edge points from Cartesian coordinates into a parameterized voting space. This approach effectively translates shape detection into a voting problem, where edge pixels

² Thresholding is a technique that converts a grayscale image into a binary image by assigning pixels to foreground or background based on intensity. Global thresholding uses a single threshold value for the whole image, while adaptive thresholding determines thresholds for smaller regions. [23]

collaboratively "elect" the most likely shapes. The votes are tallied in the accumulator cells, and the peaks in this accumulator array correspond to the most probable shapes. Its industrial utility is unequivocal; see Fig. 4, where the Hough transform is used to detect the circular shape of the capacitor. [25] [24]

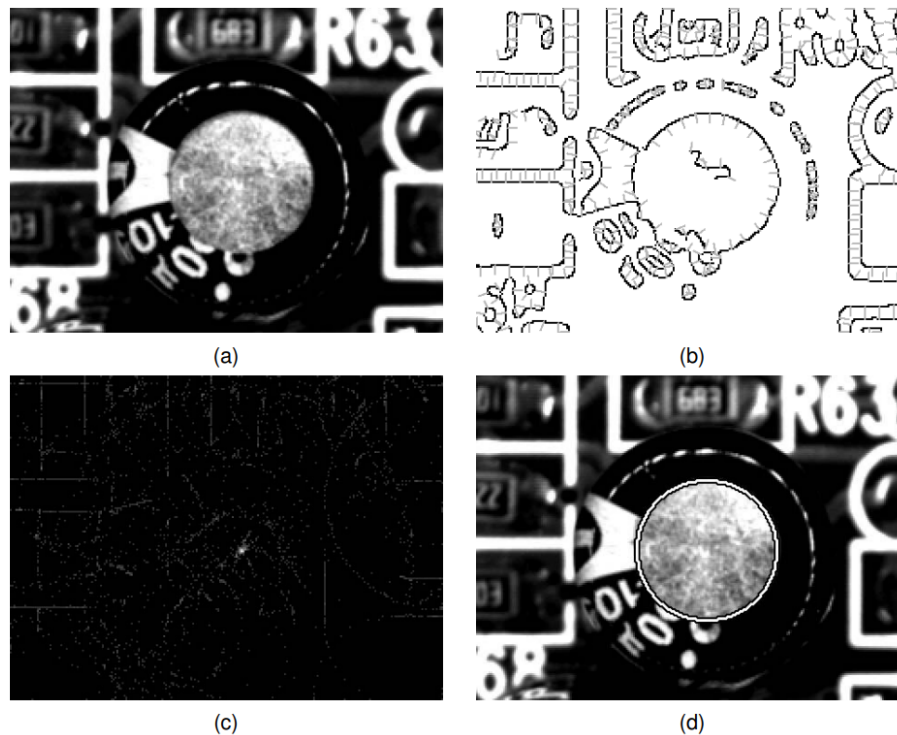


Fig. 4: Hough transform: (a) Original image of capacitor, (b) Image after Canny edge detection, (c) Hough accumulator array, (d) Circle obtained with Hough transform and later thresholding. [25]

2.5.4 SIFT and SURF

Scale-Invariant Feature Transform (SIFT) and *Speeded-Up Robust Features* (SURF) are also feature detection algorithms. SIFT identifies distinctive key points invariant to scale, rotation, and lighting by analyzing gradients in Gaussian-blurred images. Then it generates 128-dimensional descriptors. To do this, a 16×16 window around each keypoint is divided into 16 subblocks of 4×4 size. For each subblock, an 8-bin orientation histogram is created, resulting in $4 \times 4 \times 8 = 128$ bin values that form the feature vector. This ensures robustness to viewpoint and illumination changes. [27]

SURF, optimized for speed, replaces gradient computations with integral images. Integral images use precomputed tables that allow rapid summation of pixel intensities over larger regions. This allows SURF to generate 64-dimensional descriptors efficiently, trading slight precision for faster real-time performance. [28]

Both algorithms excel in cluttered environments. For example SIFT ensures precise defect detection in PCBs via fine-grained keypoints [29]. SURF on the other hand, accelerates robotic pick-and-place operations [30].

2.6 Industrial Vision Tools and Applications

Modern machine vision systems make advanced algorithms accessible through intuitive graphical interfaces. Tools such as edge extraction, blob analysis, template matching, and color analysis are presented in this section. Those industrial tools allow even non-experts to configure reliable inspections. The following sections describe practical use cases that show how these technologies enable automated visual inspection in various industrial settings.

Edge Extraction and Measurements

These tools, as the name suggests, are derived from edge detection algorithms. They are used primarily for boundary detection in applications such as object measurement and alignment. They allow subpixel accuracy, making them suitable for high-precision measurement tasks. Operators typically configure parameters such as ROI boundaries, intensity thresholds, and maximum allowable edge discontinuities. A common industrial application example can be the inspection of weld seam continuity in automotive manufacturing, where even minor defects can compromise structural integrity. [31]

Blob Analysis and Pixel Counting

Blob analysis tools are based on connected component analysis. It involves extracting features from groups of connected pixels (referred to as "blobs") that share the same logical state. The key features calculated for each blob include area (the number of pixels), center of gravity, and orientation. A typical application is in pharmaceutical packaging, where the blob analysis can be used to count the number of pills in a blister pack. If a circular pill is missing, the tool will detect fewer circular blobs than expected. [4, 32]

To set up such inspections, the operator typically defines the ROI, the required number of blobs, or the minimum pixel count for each blob. The detected blobs and their centers of gravity are illustrated in Fig. 5. Although blob analysis is computationally efficient and suitable for rapid inspections, it generally provides only coarse measurements (e.g., with a precision of about 3 pixels), in contrast to the subpixel precision achievable with edge detection methods [31].

Pattern and Template Matching

The pattern and template matching tools identify objects in an image by comparing regions with a predefined template. This process involves computing similarity metrics

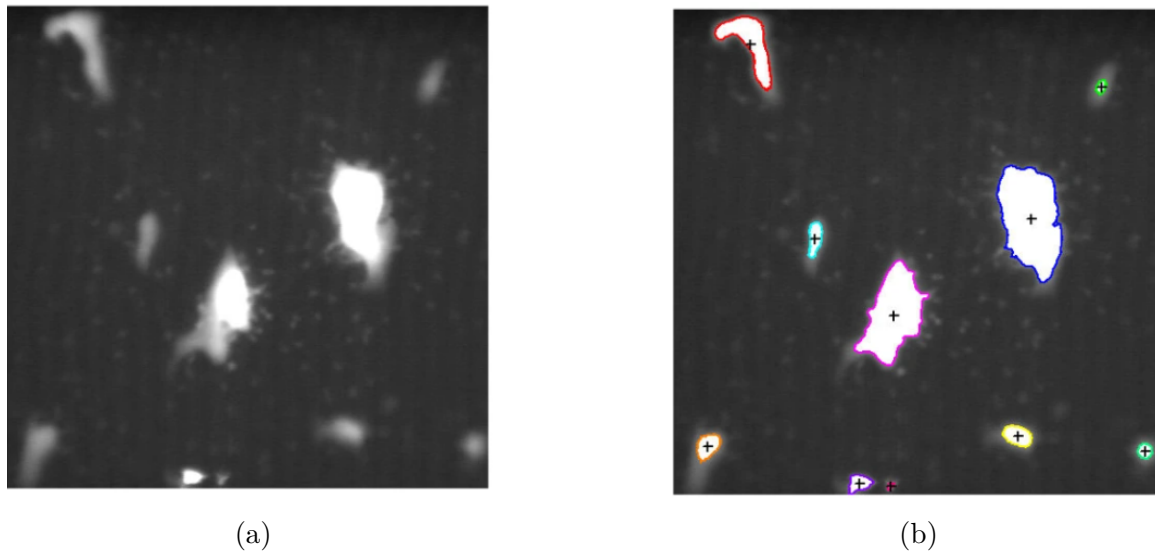


Fig. 5: (a) Original image, (b) Image with extracted blobs and calculated center points. [32]

between the template and the image in all potential positions, rotations, and scales of the template [25]. When the similarity score exceeds a defined threshold, the system registers a successful match.³ These tools are used for the determination of part orientation (e.g., in robotic guidance systems) and quality inspection, where deviations from the reference template indicate defects.

Key algorithms in this domain include SIFT and SURF, which are particularly valued for their robustness to scale and rotation variations. Operators typically configure only the reference template and acceptable tolerance thresholds (e.g., angular deviation limits) [4]. Historically, template matching formed the backbone of OCR systems, where alignment techniques matched characters to stored glyph templates [7]. However, this approach has been largely replaced by neural network-based methods trained on diverse datasets [9].

Color Analysis

Color analysis tools focus on verifying color consistency. These tools check whether colors match predefined standards, which is critical for quality control in products such as packaging, textiles, or painted surfaces. Rather than relying on basic RGB color models, these systems utilize HSV (Hue, Saturation, Value) to separate color information from brightness. That makes the analysis more robust under varying lighting conditions [25].

The pivotal role in color matching plays a K-means clustering algorithm. This unsupervised machine learning algorithm partitions unlabeled color data into K clusters of

³ Similarity score is commonly calculated using Gray-Value-Based Template Matching or normalized cross-correlation. [25]

similar hues, allowing the identification of dominant colors within an image. For example, a 24-bit RGB image (with over 16 million possible colors) can be reduced to only 8 representative clusters, significantly simplifying computations. [24]

2.7 Industrial Vision Tools for Complex Inspection Scenarios

Although artificial intelligence and deep learning in visual quality inspection will be thoroughly discussed in the next chapters, it is necessary to briefly introduce the overall structure of these solutions here.

The previously discussed task of visual quality inspection of the final assembly faces many challenges, such as variability in the position and appearance of defects. To address this problem, deep learning models are employed.



Fig. 6: Deep learning-based solution for final assembly quality inspection with collaborative robot [22]

First, the model must be trained on relevant data. If enough defect data is available, supervised learning is used, training on both good and defective images of the parts. If there is not a sufficient amount of defective data, only good images are used for training via unsupervised learning. After training and testing the model, it is loaded onto an edge device, which then performs inference [24].

A typical deployment involves a collaborative robot that can move the camera around the part, capturing all possible angles. At the same time, it does not need to be shielded from humans, as it is designed to be safe for such tasks. An example of a collaborative robot equipped with a camera inspecting a final assembly is shown in Fig. 6.

3 DEEP LEARNING FOR VISUAL TASKS

In this chapter, the general basics of deep learning are presented. The principles of *Convolutional Neural Network* (CNN) are then discussed, along with the most popular pretrained models. The chapter also briefly explains vision transformers. Finally, data augmentation techniques are introduced as a means to improve model robustness and performance.

3.1 Deep Learning Fundamentals

Deep learning approaches try to mimic the neural activity of the human brain by using architectures called artificial neural networks. An artificial neural network is made up of simple units called neurons. Each neuron computes a weighted sum of its inputs and adds a bias. The result is then passed through an activation function, which introduces non-linearity into the model; see equation (4).

$$y = f \left(\sum_i x_i w_i + b \right) \quad (4)$$

where:

- y is output of the neuron (also called the activation),
- f is activation function (e.g. ReLU, sigmoid, tanh),
- x_i is input from neuron i of the previous layer,
- w_i is weight associated with input x_i ,
- b is bias term.

These neurons are grouped into layers, and the layers are stacked sequentially. There is an input layer, an output layer, and all the layers in between are referred to as hidden layers. A single neuron that takes the entire previous layer as input is called a perceptron. This basic unit represents the simplest neural network. An example of a perceptron is shown in Fig. 7. [33]

A simplest neural network composed of multiple layers of perceptrons is called a *Multilayer Perceptron* (MLP). It is a feedforward network, meaning that information flows from the input layer through one or more hidden layers to the output layer. It has fully connected layers, which means that each neuron in a layer is connected to every neuron in the next layer by trainable weights. Such networks can approximate complex functions by stacking many simple neuron units. This allows them to solve both linear and non-linear tasks. Common applications include linear regression to predict continuous values and logistic regression for classification problems. [33]

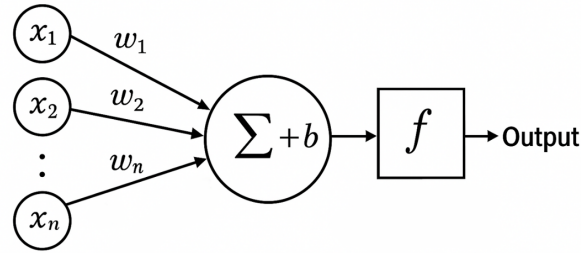


Fig. 7: Perceptron: a neuron that computes a weighted sum of inputs, adds bias b , and applies an activation function f .

Recurrent neural networks present a different approach to information flow – they introduce feedback connections. This allows the model to process sequential data, such as time series or language [33].

Training a neural network involves adjusting its weights and biases to minimize a loss function using a given dataset. Typically, the data is divided into three parts: a training set used to learn the model parameters, a validation set used for hyperparameter tuning, and a test set used for the final evaluation. The goal of training is to produce a model that generalizes well to new unseen data. In order to avoid biased or misleading results, the validation and test data must remain separate from the training process. [34]

Learning approaches are generally categorized as supervised, unsupervised, semi-supervised, and reinforcement learning. Supervised learning uses datasets with multiple labeled classes. Unsupervised learning, in contrast, uses unlabeled data and must discover underlying patterns without external guidance. Semi-supervised learning combines a small amount of labeled data with a larger pool of unlabeled data. Reinforcement learning is based on a reward system that encourages desirable behavior through interaction with the environment. [33]

Activation Functions

Activation functions introduce non-linearity into neural networks. The importance of non-linearity is well explained in [33]. It shows that if you only use the identity activation function $f(x) = x$ in each layer, the architecture will be reduced to a single layer network. Therefore, to solve non-linear problems, activation functions are needed. Several activation functions are commonly used in practice, each with specific advantages depending on the task and network architecture. One of the earliest activation functions is the *sigmoid* function, which maps the input values x to the interval $(0, 1)$; see equation (5). Due to its probabilistic interpretation, it is used in the output layer of binary classification models. However, its tendency to saturate for large absolute values of x results in vanishing

gradients, particularly problematic in deep networks. [34]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (5)$$

A related function is *hyperbolic tangent*, or *tanh*; see equation (6). It maps the input values to the range $(-1, 1)$ and is zero centered. This often leads to faster convergence during training compared to the sigmoid. However, like sigmoid, tanh also suffers from saturation for large values of x , as the derivative approaches zero. [33]

$$\tanh(x) = 2 \cdot \sigma(2x) - 1 = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (6)$$

To mitigate these issues, modern deep networks typically use the *Rectified Linear Unit* (ReLU); see equation (7). ReLU is non-saturating for $x > 0$, which preserves gradient flow and makes optimization more efficient. It also introduces sparsity by setting negative activations to zero. ReLU is the primary choice for CNN hidden layer architectures [33].

$$\text{ReLU}(x) = \max(0, x) \quad (7)$$

When there are a lot of negative activations, the pure *ReLU* architecture could suffer from most neurons "dying" (becoming 0). Although this is part of the learning process, too many dead neurons can hinder the training process. Therefore, *Leaky ReLU* was proposed; see equation (8). It provides non-zero gradients for $x < 0$. [33]

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0, \\ \alpha x, & \text{if } x < 0, \end{cases} \quad (8)$$

where α is a small positive constant (e.g., $\alpha = 0.01$). Another modern alternative is *Swish*; see equation (9), proposed by researchers at Google [35]. Swish is smooth and non-monotonic, and has been shown to outperform ReLU in certain deep architectures by maintaining small gradients even for negative inputs [35].

$$\text{Swish}(x) = x \cdot \sigma(x) \quad (9)$$

where $\sigma(x)$ is the sigmoid function. Finally, in the output layer of classification tasks that involve more than two classes, the *softmax* function is commonly used; see equation (10). It converts a vector of raw scores x_i into a probability distribution over classes.

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (10)$$

where:

- x_i is the logit or input score (before activation) for class i ,
- K is the total number of classes,
- j indexes over all classes from 1 to K .

Loss Functions

A loss function (or cost function) measures the error between the desired output y and the actual output a of the neural network. It is an essential component, which tells the neural network how it is performing, guiding its optimization. The loss function is derived when performing a gradient descent; more details in the following sections.

The most common loss function used for regression problems is *Mean Squared Error* (MSE); see equation (11), which takes the quadratic difference between the desired and the actual value. It penalizes larger errors more severely [36].

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (y_i - a_i)^2 \quad (11)$$

where y_i is the ground truth, a_i is the prediction, and n is the number of samples. MSE is sensitive to outliers and ideal when large errors are especially undesirable. It is also common to use it for reconstruction of images [37]. *Mean Absolute Error* (MAE), expressed in equation (12), offers a robust alternative for regression tasks, where outliers should not overly influence learning. It is more robust to outliers, but may result in slower convergence [36].

$$\mathcal{L}_{\text{MAE}} = \frac{1}{n} \sum_{i=1}^n |y_i - a_i| \quad (12)$$

Cross-Entropy Loss (Log-loss), expressed in equation (13), is the standard for multi-class classification with softmax outputs.

$$\mathcal{L}_{\text{CE}} = - \sum_{i=1}^C y_i \log(a_i) \quad (13)$$

Here, C is the number of classes, y_i is the one (encoded ground truth), and a_i is the predicted probability [36]. *Binary Cross-Entropy* is used for binary classification problems, see following equation (14).

$$\mathcal{L}_{\text{BCE}} = -(y \log(a) + (1 - y) \log(1 - a)) \quad (14)$$

Structural Similarity Index Measure (SSIM) is the custom similarity metric proposed by Bergmann *et al.* [38]. It has a perceptual nature, which means that it evaluates visual similarity based on how humans perceive images. It takes into account luminance, contrast, and structural information, not just numeric differences between pixels. It is particularly useful when reconstructing RGB or grayscale images directly. Later in [38], it was shown that SSIM generally improves the performance of image reconstruction-based methods (explained in 4).

There is also a modern approach to the design of loss functions known as *Composite Loss*, which consists of a weighted average of two distinct loss functions; see equation (15). The goal of this method is to combine the strengths of different metrics and compensate for their individual weaknesses. For instance, in a study [39], researchers employed a custom

loss function that combined MSE and SSIM with equal weights. This combination was found to enhance the quality of the reconstruction.

$$\mathcal{L}_{\text{Composite}} = \alpha \mathcal{L}_A + (1 - \alpha) \mathcal{L}_B \quad (15)$$

where $\mathcal{L}_A$ and $\mathcal{L}_B$ are loss functions and α is a weighting factor between 0 and 1.

Gradient Descent and Backpropagation

Training a neural network is achieved by adjusting its weights and biases so that it performs well on a specific task. This optimization is driven by a loss function. The lower the loss, the better the network performs.

To minimize the loss, an optimization algorithm called gradient descent is used, and all optimizers mentioned later are built upon the gradient descent. The core idea behind it is to compute the gradient (i.e., the direction of the steepest increase) of the loss function with respect to the model parameters. Then those parameters are updated in the opposite direction, moving "downhill" on the loss surface. This process is repeated over many iterations until the network converges to a set of parameters that produce a low error. [33]

A variant called *Stochastic Gradient Descent* (SGD) is used more frequently for learning because it allows faster and more memory-efficient training. Instead of calculating the gradient using the entire dataset (which is computationally irrational), SGD estimates the gradient from a small, randomly chosen mini-batch. Although noisier, this estimate is typically good enough and has the added benefit of helping the model escape shallow local minima. [33]

To efficiently compute those gradients for deep networks with many layers, we use the backpropagation algorithm. Backpropagation applies the chain rule from calculus to propagate the error from the output layer back through the network, layer by layer. It calculates how much each parameter contributed to the final error, allowing all weights and biases to be updated accordingly. [33]

In summary, gradient descent tells the network how to adjust its parameters, and backpropagation provides the information it needs to do so efficiently. Together, they form the backbone of deep learning model training.

Optimizers

In deep learning optimization, few key optimizers are used for most of the tasks. As mentioned previously, they all derive from the gradient descent.

SGD with Momentum improves stochastic gradient descent by retaining few previous updates, simulating inertia. It accelerates convergence, and also helps to escape local minima. **Adaptive Gradient Descent (AdaGrad)** introduces adaptive learning rates adjusted by cumulative squared gradients. It excels with sparse data, but suffers

from the "dead neuron problem" in deep networks, when the learning rate becomes too small. **Root Mean Square Propagation (RMSProp)** addresses AdaGrad's limitation by using exponential moving averages of squared gradients (instead of cumulative squared gradients). It combines momentum-like behavior with adaptive learning rates. Despite tackling AdaGrad's limitation successfully, convergence is usually slower, compared to other optimizers. Finally, **Adaptive Moment Estimation (Adam)** unifies momentum and RMSProp concepts. This hybrid approach achieves computational efficiency, minimal memory requirements, and robust performance across diverse tasks. It is also very robust to hyperparameter selection. This makes Adam default choice for many tasks. [40]

Overfitting and Underfitting

Overfitting and underfitting represent two fundamental failure modes in machine learning. An underfitting model lacks sufficient complexity to capture the underlying structure of the data. It results in high training error and correspondingly poor performance on unseen data. In contrast, an overfitting model is excessively complex, achieving a very low training error by memorizing the training data but failing to generalize – high error on test data. Underfitting can typically be addressed by increasing the capacity of the model, such as adding more neurons or layers. Mitigating overfitting is more challenging. Although reducing the complexity of the model is an approach, it may compromise the representational power of the model. More effective strategies include expanding the dataset and increasing the diversity of data through augmentation. Advanced techniques, such as regularization, deliberately penalize the loss function or randomly deactivate neurons during training to improve generalization (see the following section for details on regularization methods). The concepts of underfitting and overfitting can be intuitively illustrated by considering function regression through data points, as depicted in Fig. 8. [34]

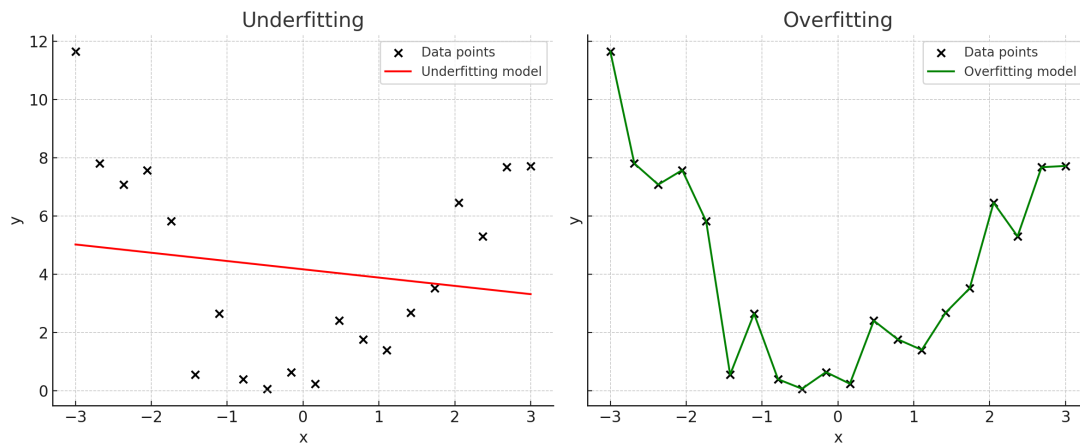


Fig. 8: Example of underfitting and overfitting in regression task

Regularization

Regularization techniques are used to prevent overfitting by penalizing the complexity of the model. A common approach is to add a norm penalty on the weights to the loss. For example, regularization L_2 (weight decay) adds the sum of squared weights to the cost, encouraging smaller weights. L_1 regularization adds the sum of absolute weights, which can drive many weights to zero and induce sparsity. Another popular technique is dropout, where a fraction of hidden units is randomly “dropped” (zeroed) on each training pass. Dropout forces the network to learn redundant representations and reduces the dependence on any single neuron. These methods help a network to generalize better and make it less likely to overfit. [34]

3.2 Convolutional Neural Networks

CNNs are a specialized class of deep neural networks designed to process data with a grid-like topology, such as two-dimensional images. Unlike fully connected networks, which treat input features as independent, CNNs leverage the strong spatial dependencies in image data by applying convolutional layers. These layers use small, trainable filters (kernels) to perform local dot-product operations across the input volume, capturing edges or different patterns. With the depth of the network, the complexity of the captured features increases. The input to a CNN is typically a three-dimensional volume consisting of width, height, and depth (e.g., RGB channels), where neighboring pixels often share similar values. By learning local features and maintaining spatial hierarchies, CNNs are highly effective for image classification, segmentation, and detection tasks. They usually incorporate three types of layers: convolutional layers, pooling layers, and at the end fully connected layers. While CNNs are most commonly associated with image data, their applicability extends to other types of grid-structured data, including time series. However, they excel most with images and have become fundamental for modern computer vision. [34, 33]

Convolutional Layers

A convolutional layer consists of a set of learnable filters (also called kernels) that are significantly smaller in spatial size than the input, but also cover the input depth (RGB image). For example, a first layer filter might be $5 \times 5 \times 3$ for the RGB input. Each filter is convolved across the width and height of the input, producing a 2D activation map of responses. By having many such filters, the layer produces a stack of feature maps – the data grow in depth. The convolution shares weights spatially, so the same filter is applied at every location. This means that the network can detect certain feature anywhere. This ability is called translation equivariance. [33]

The useful special-type convolutions include strided convolutions that can be used to downsample spatial resolution (skipping positions). There are also dilated convolutions that insert gaps between filter elements to increase the receptive field without downsampling. [34]

Pooling Layers

Pooling layers are often inserted after convolutional layers to progressively reduce spatial size. The reason is mainly the reduction of parameters and the improved generalization capability of CNN. The most common are max-pooling and average-pooling. Max pooling partitions the input into non-overlapping regions (e.g. 2×2 blocks) and outputs the maximum value in each block, whereas average pooling outputs the mean. The pooling makes the representation smaller and more manageable. It also controls overfitting by making the network focus on the most significant features. [34, 33]

Fully Connected Layers

Toward the end of a CNN, one or more fully connected layers are typically used for classification. After several rounds of convolution and pooling, the final feature maps are flattened and fed into those layers. For instance, in a 10-class image classification task, the last fully connected layer might produce a $1 \times 1 \times 10$ score vector (one score per class). Each neuron in these final layers is connected to all activations in the previous layer and computes a weighted sum plus bias. The output of the final fully connected layer is passed through an activation (such as softmax) to produce class probabilities. [34]

Attention Mechanisms

Attention mechanisms in deep learning allow models to selectively focus on the most relevant parts of the input. It improves their ability to capture the most important features and spatial information. Channel attention adaptively recalibrates channel-wise feature responses by explicitly modeling dependencies between channels. The Squeeze-and-Excitation Networks paper [41] introduced this approach. The spatial attention mechanism, in contrast, focuses on identifying where the most informative features are located within a feature map.

Building on this, the *Convolutional Block Attention Module* (CBAM) paper [42] proposed a lightweight module that sequentially applies both channel and spatial attention, further improving performance.

3.3 Key Pretrained CNNs and Transfer Learning

Pretrained CNN architectures are widely used as feature extractors or baselines. Below are several influential models, each marked by a key innovation. They all excelled in

a widely recognized database of data called ImageNet. They performed exceptionally well in the annual organized competition called ImageNet Large-Scale Visual Recognition Challenge [43]. The competition has a few categories, so each year there are few winners, not just one.

Pretrained CNNs are all used for the so-called transfer learning. This means that although they were initially developed and trained for the ImageNet classification task, they are reused as the starting point for another model on a different but related task. This allows faster training and improved performance with less data.

VGG (Simonyan and Zisserman 2014 [44]): a very deep network using only 3×3 convolutions. VGG demonstrated that simply increasing the depth of the network (16-19 layers) with small filters can significantly improve the accuracy on ImageNet. VGG achieved first place in category classification plus localization of the ImageNet 2014 competition, showing that very deep architectures generalize well. Its simplicity (repeated convolutional layers then pooling layers) and publicly released weights make VGG popular for transfer learning. However, it is very parametrically heavy. [43]

ResNet (He *et al.* 2015 [45]): Introduced the residual learning framework to allow for extremely deep networks. ResNet adds skip connections that bypass one or more layers, allowing layers to learn residual functions with reference to their inputs. This innovation allows very deep networks to be optimized, which would otherwise be impossible. He *et al.* trained ResNets on up to 152 layers ($8\times$ deeper than VGG) and achieved a top-5 error on ImageNet. ResNet won the ImageNet 2015 competition and paved the way for new approaches. [43]

Inception (Szegedy *et al.* 2014 [46]): Also known as GoogLeNet, uses specialized Inception modules to increase network width and depth while controlling computation. Each Inception module applies filters of multiple sizes in parallel (e.g. 1×1 , 3×3 , 5×5) and concatenates the results. This multiscale processing “lets the network decide” which filter size is best. The original Inception (22-layer GoogLeNet) won the ImageNet 2014 competition in a few categories. Later versions (Inception-v4, Inception-ResNet) further improved performance by factorizing convolutions and adding residual connections. [43]

EfficientNet (Tan and Le 2019 [47]): Proposed a compound model scaling method. Instead of arbitrarily making the models bigger, EfficientNet uniformly scales all dimensions (depth, width, and image resolution) using a compound coefficient. They used a neural network approach to find the optimal baseline architecture of their new CNN. EfficientNets achieved state-of-the-art accuracy with far fewer parameters. For example, EfficientNet-B7 reaches 84.3% top-1 accuracy on ImageNet while being smaller and faster than previous CNNs. EfficientNet models are used for their high accuracy and parameter efficiency.

3.4 Vision Transformers

The Transformer architecture was first introduced in the landmark paper *Attention Is All You Need* by Vaswani *et al.* [48]. Unlike previous models based on convolutions, the transformer relies entirely on a self-attention mechanism to model dependencies within sequences. Its encoder-decoder structure, equipped with multihead self-attention and positional encodings, demonstrated significant advances in natural language processing tasks. A major strength of the Transformer lies in its ability to capture long-range relationships without regard to sequence position.

Inspired by this success in language modeling, researchers extended the transformer framework to computer vision, leading to the development of *Vision Transformer* (ViT). It was first introduced in ground-breaking paper *An Image is only worth 16×16 Words* by Dosovitskiy *et al.* [49]. Vision Transformers adapt the original transformer architecture to operate on image data by dividing an image into fixed-size non-overlapping patches. Then the transformer flattens each patch and projects it into an embedding space. These patch embeddings are then fed into a transformer encoder. A learnable classification token is then added to the sequence together with the positional encoding to preserve the spatial layout (ViTs are not translation equivariant like CNNs) [49]. This patch-wise treatment allows ViTs to capture global context and long-range dependencies from the very first layer. The structure is depicted in Fig. 9.

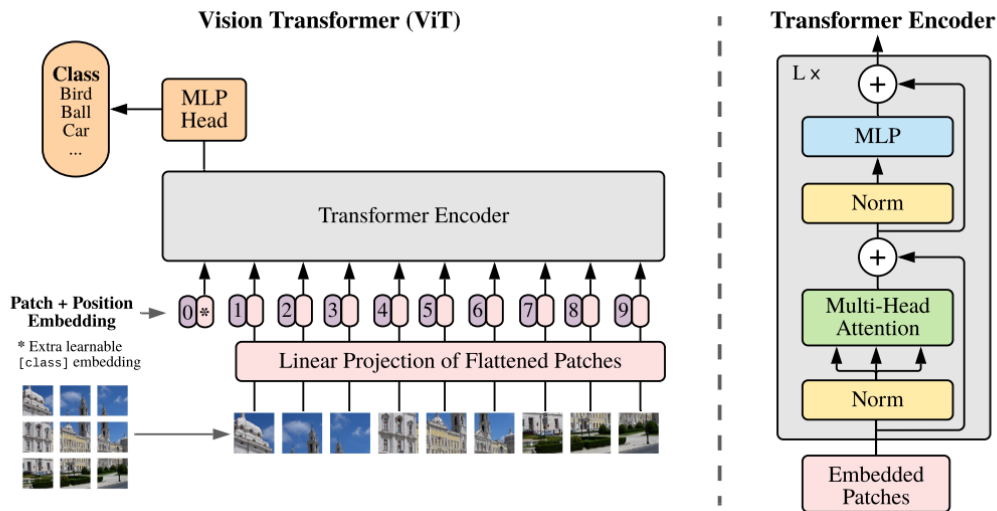


Fig. 9: Architecture of a vision transformer [49]

One of the key strengths of ViTs is their scalability with respect to data and model size. When pretrained on large-scale datasets, ViTs have demonstrated superior performance over many convolutional architectures on standard classification benchmarks [49]. The attention mechanism provides a global receptive field from the very first layer. This allows

the model to learn complex interactions between distant regions of the input image. This global context modeling contrasts with CNNs, which typically rely on deep stacking and hierarchical feature aggregation to achieve similar receptive fields.

However, this generalization capacity comes at a cost. Vision transformers lack spatial inductive biases, as discussed previously. These biases make CNNs particularly efficient at learning from limited data. In contrast, ViTs must learn such priors from the data itself, making them significantly more data-hungry and less effective without large-scale pre-training. Furthermore, the self-attention mechanism in transformers scales quadratically with the number of input tokens, introducing computational overhead. [48]

In conclusion, in data-rich settings, where extensive computational resources are available, ViTs offer superior performance compared to CNNs. However, in resource-constrained scenarios with moderate data amount, CNNs remain more practical due to their efficiency and built-in inductive biases. [49, 48].

3.5 Data Augmentation

Data augmentation is the process of artificially expanding a training dataset by applying label-preserving transformations to existing images. The goal is to increase the size and diversity of the data available for training. In this way, the model is more likely to achieve better generalization [50]. In practice, data augmentation addresses the problem of limited training data and helps mitigate overfitting. This approach was pivotal in early deep learning breakthroughs. For instance, the AlexNet CNN [51] used extensive augmentation (random crops, flips, and color shifts), effectively expanding the dataset by the factor of 2048 and reducing the classification error by more than 1% [51]. Augmentation also acts as a form of regularization, allowing models to achieve a lower validation error and even lower testing error, as real-world data could be less challenging than augmented training data at the end of the day [50].

Traditional augmentation methods include geometric operations such as rotation, flipping, cropping, and scaling. For example, horizontal flips and small-angle rotations are particularly effective in natural image datasets, as they simulate real-world perspective changes [51]. Photometric augmentations, such as color jitter or brightness and contrast adjustments, allow for robust performance of the model under varying lighting conditions.

Modern techniques like CutMix [52] paste regions of one image onto another, encouraging the model to focus on multiple discriminative features. This leads to measurable improvements in classification accuracy in datasets such as CIFAR-100 [52]. Another notable advancement in the field is AutoAugment [53], which automates the search for optimal augmentation policies using reinforcement learning. By discovering effective augmentation strategies directly from the data, AutoAugment was shown to outperform manually designed policies.

However, the benefits of data augmentation are not universal. When transformations distort essential semantic content, they may harm rather than help. For example, rotating digits can turn an '6' into a '9', or modifying colors might misrepresent the true class of an object if the hue is a key identifier [50]. In high-resource settings with large and diverse datasets, augmentation offers insignificant returns and may unnecessarily increase training time.

4 IMAGE ANOMALY DETECTION WITH UNSUPERVISED LEARNING

After establishing the theoretical foundations of deep learning, the thesis now turns to unsupervised approaches for image anomaly detection. The core of the chapter presents unsupervised visual anomaly detection techniques, and the end is dedicated to metrics for evaluating image anomaly detection models.

To contextualize the role of unsupervised learning, it is important to compare the strengths and limitations of unsupervised methods with supervised learning. In general, supervised methods achieve higher overall classification accuracy when distinguishing between normal and anomalous parts, but only if sufficient labeled data are available. These methods benefit from explicit exposure to defective samples during training because they can learn specific representations of anomalies. However, this strength can become a limitation. Supervised models are restricted by the characteristics of their training labels. Therefore, they may struggle to classify anomalous cases with different semantic properties than those seen during training. [37]

Unsupervised anomaly detection, on the contrary, relies only on modeling the distribution of normal data. By learning what makes a "normal" part, the model is able to flag any significant deviation from that learned distribution. This approach offers a superior generalization to the unknown or rare types of defects. This is a significant advantage that is particularly valuable in real-world scenarios, where possible anomalies are endless and unpredictable. Furthermore, unsupervised approaches require only normal samples during training, which addresses a key industrial limitation of an insufficient amount of labeled anomalous data. This makes unsupervised methods more scalable and accessible for industrial deployment. However, in scenarios with extensive and well-labeled anomaly datasets, unsupervised methods typically reach lower precision than supervised approaches. [37]

4.1 Reconstruction Based Methods

First category of these methods is based on reconstruction. The core assumption is that a neural network trained just on normal images will not be able to reconstruct anomalous inputs well.

These reconstruction-based methods discussed in this chapter include: ordinary *Autoencoder* (AE), *Variational Autoencoder* (VAE) and *Generative Adversarial Network* (GAN). The models learn compact representations of normal samples and attempt to reconstruct the input from a latent encoding. Latent encoding means that spatial information from the input is reduced to a compact representation of features with a larger

depth dimension. The *Principal Component Analysis* (PCA) will also be discussed as it is essential to estimate the dimension of the latent space.

According to Lie *et al.* [37], reconstruction-based methods generally achieve higher pixel-level accuracy than embedding-based alternatives. This is because of their ability to directly compare individual pixels. However, at the image level, where semantic understanding and robustness are essential, pure image reconstruction-based approaches struggle. They perform much worse than feature embedding approaches, as they fail to generate high semantic context. [37]

4.1.1 Autoencoders and Variational Autoencoders

AE and VAE are the most prevalent architectures for the anomaly detection task. Their intuitive functionality makes them the first choice for such task. [37]

The architecture consists of two primary components: an encoder and a decoder. The encoder compresses the input image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, where H is the image height, W is the image width, C is the number of color channels, into a latent representation $\mathbf{z} \in \mathbb{R}^d$, where d is the dimensionality of the latent space. The dimension of the latent space usually satisfies $d \ll H \times W \times C$. The decoder then attempts to reconstruct the original image from this latent vector, producing an output $\hat{\mathbf{x}} \approx \mathbf{x}$. The network is trained by minimizing the reconstruction loss that measures the discrepancy between the input and its reconstruction. For grayscale or color image data, the most commonly used loss is the MSE. [54]

Variational Autoencoders extend this concept by imposing a probabilistic structure on the latent space. Instead of encoding the input as a single deterministic vector, the encoder outputs parameters of a Gaussian distribution. During training, a latent vector is sampled from this distribution. The loss function for VAEs consists of two terms and is expressed in equation (16).

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\|\mathbf{x} - \hat{\mathbf{x}}\|^2] + D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})), \quad (16)$$

where the first term is the reconstruction loss (MSE) and the second is the Kullback-Leibler divergence that regularizes the approximate distribution $q(\mathbf{z}|\mathbf{x})$, learned by the encoder network to follow a known prior distribution $p(\mathbf{z})$, typically a standard normal distribution [54].¹

Several studies have demonstrated the effectiveness of AE and VAE-based models for visual anomaly detection. For instance, [38] showed that a simple convolutional autoencoder achieved strong pixel-wise anomaly localization across multiple classes in the MVTec dataset [56], with specially designed loss function [37].

¹ The Kullback–Leibler (KL) divergence is a fundamental concept in information theory that quantifies how one probability distribution diverges from a second, reference distribution; it was first introduced by Solomon Kullback and Richard A. Leibler in their seminal 1951 paper [55].

However, pure image reconstruction methods usually reconstruct anomalous patterns too well, leading to insufficient anomaly detection. To improve performance, researchers have proposed variants such as feature autoencoders [57], which have proved to be much more precise in image level classification than raw image reconstruction approaches [37]. The feature reconstruction approach is called *Deep Feature Reconstruction* (DFR) in the [57] article and will be very important in the practical part of the thesis. It is built upon the same principle as an ordinary autoencoder, but reconstructs features extracted by a large pretrained network.

4.1.2 Generative Adversarial Networks

GANs are composed of two competing neural networks: a generator G and a discriminator D , trained simultaneously in adversarial settings. The generator learns to map latent vectors from a prior distribution to the data space, producing synthetic samples $G(\mathbf{z})$. Those samples are intended to resemble real samples as much as possible. The discriminator is trained to distinguish real data $\mathbf{x}$ from generated data $G(\mathbf{z})$. The training setting is for better understanding depicted in Fig. 10. The generator and discriminator gradually become more capable. In the end, the discriminator should not be ideally able to distinguish the generated data from the real data. [58]

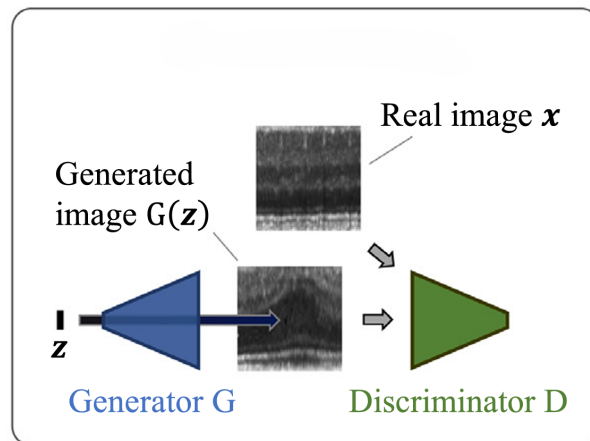


Fig. 10: Training setting of a GAN network [58]

At test time, the generator reconstructs input images and anomalies are detected by deviations from the reconstruction error of the generator and also from the discriminator feedback. The discriminator evaluates both the original and the reconstructed images. This way, it provides complementary signals to identify subtle anomalies that reconstruction alone might miss. This dual mechanism improves robustness compared to autoencoders, which are based only on reconstruction error. [37, 58]

GAN-based approaches frequently suffer from instability during training. However, when trained with stability, methods such as f-AnoGAN [58] have reported strong results on a medical datasets.

4.1.3 Latent Space Estimation with PCA

PCA is not a reconstruction-based method for anomaly detection, but rather a classical statistical method used to reduce dimensionality while preserving as much variance as possible in the data. It is presented among those reconstruction-based methods as it can be useful for dimensionality estimation.

Given a dataset with many correlated features, PCA transforms the data into a new coordinate system where the axes (principal components) are uncorrelated. They are ordered by the amount of data variance they capture. [34]

The method works by computing the eigenvectors and eigenvalues of the covariance matrix of the input data.² The eigenvectors define the directions of maximum variance and the eigenvalues indicate the magnitude of variance captured by each component. By selecting the top k components, PCA can reduce the dimensionality of the data from n to k dimensions while retaining the most important information. This can be very useful when computing the optimal latent dimension of an autoencoder to preserve enough data variance. [34]

4.2 Feature Embedding Methods

Unlike reconstruction-based methods that rely on pixel-wise comparison between input and output images, feature embedding methods operate in a deep feature space. These approaches extract intermediate representations from the convolutional layers of pretrained networks and use these high-level features to detect anomalies. The core idea is that normal images will be mapped to a consistent region in the feature space, while anomalies will appear far from this region. The techniques used by those approaches include distance metrics, density estimation, or representation regression. Approaches based on memory banks or probabilistic models fall within this category too. [37]

According to Lie *et al.* [37], embedding-based methods consistently outperform reconstruction-based models in image-level anomaly detection tasks. This is due to their ability to leverage pretrained backbones and capture high-level semantic information. However, they may underperform at pixel-level as they lack direct pixel comparisons.

² An eigenvector, when transformed by the matrix $\mathbb{A}$, does not change direction – only its length changes. The eigenvalue provides how much the eigenvector is stretched or compressed during this transformation.

4.2.1 Distribution Map

Distribution-based methods rely on learning a transformation that maps normal image features into a well-defined probability distribution, most commonly a multivariate Gaussian (bell-shaped curve). These approaches use a strong pretrained backbone to extract representative features from normal images. A mapping module then transforms these features to fit the target distribution; see Fig. 11 for better understanding. During inference, if the features of a test image significantly deviate from this learned distribution, the image is considered anomalous. [37]

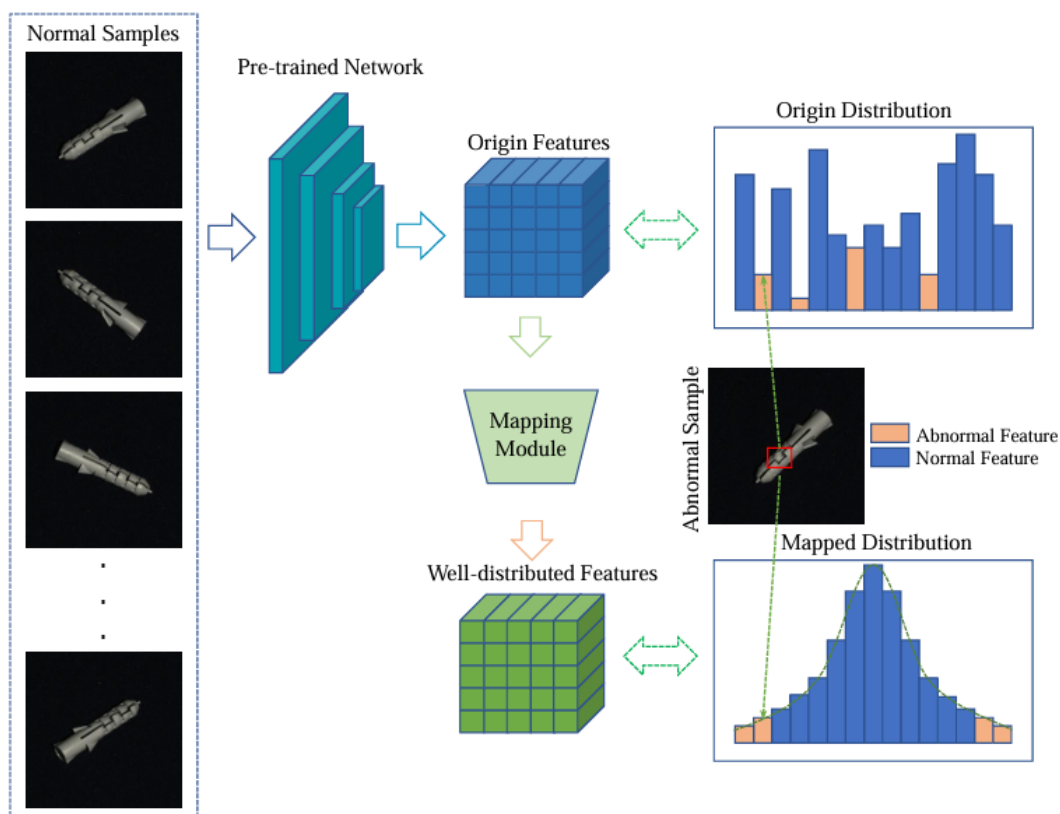


Fig. 11: Architecture of distribution-map based models [37]

Among these methods, normalizing flows have become particularly popular because of their strong ability to model complex distributions. Compared to one-class classification methods, which focus on separating normal from abnormal features via decision boundaries, distribution-based methods emphasize modeling the feature distribution directly. As a result, they can assign anomaly scores based on the likelihood that a sample is under the learned distribution. Another notable method in this category is FastFlow, which uses a 2D normalization flow applied directly to convolutional feature maps [37].

Distribution maps have the advantage of providing exact and tractable likelihoods. They are based directly on the underlying statistics of normal features. They have proven

to be a robust tool in modern unsupervised image anomaly detection – Fastflow reached the top 3 performance at the image level in the anomaly detection survey [37].

4.2.2 Student-Teacher Methods

This novel approach proposed by Bergmann *et al.* [59] is based on a teacher network, typically a robust pre-trained feature extractor, and a student network that attempts to mimic the teacher’s outputs given only normal data. The underlying principle is that a student model is trained exclusively on normal data relevant to the task. It will therefore struggle to reproduce the teacher’s output when presented with anomalous data. This discrepancy can be quantified and used as an anomaly score (see Fig. 12).

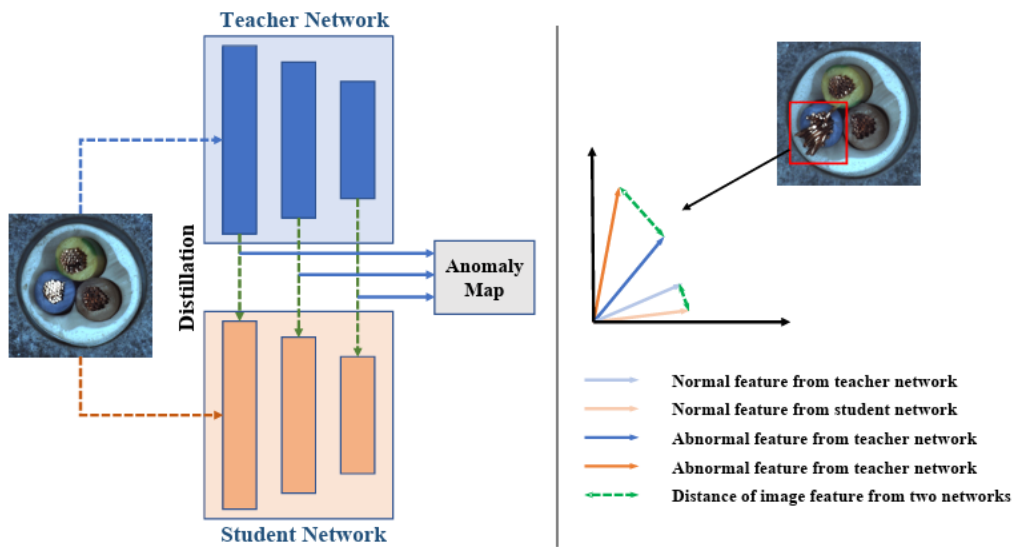


Fig. 12: Architecture of Student-teacher model [37]

The seminal paper by Bergmann *et al.* introduced a student-teacher anomaly detection framework in which the student network is trained to regress the internal representations of a WideResNet teacher [59]. They showed that the deviation between the teacher and the student predictions is a very effective anomaly score, achieving high performance on the MVTec dataset [56]. Later improvements include multi-layer supervision or feature pyramids that improve the precision of anomaly maps. [37]

Student-teacher methods generally perform well on both pixel-level and image-level anomaly detection benchmarks. According to the survey [37], these approaches strike a balance between semantic representation and pixel precision.

4.2.3 One-Class Classification

The detection of anomalies can also be formulated as a *One-class Classification* (OCC) problem. A common strategy is to enclose normal samples within a minimal bounding

volume in a feature space, typically modeled as a hypersphere; see Fig. 13. This algorithm is called *Support Vector Data Description* (SVDD). [37]

At test time, any sample that lies outside the boundary of this learned hypersphere is considered anomalous. This method also heavily relies on artificially generated anomalous images that improve the hypersphere boundary. This does not break any rules for the task to be unsupervised because the anomalous images are generated from normal data. The problem is when artificially generated negative samples do not capture representative defect characteristics. When that happens, the performance of the model degrades significantly. [37]

Despite the conceptual simplicity of one-class classification methods, they are inherently limited by the challenge of modeling abnormalities without real examples. The survey [37] reports that OCC models generally underperform in image-level classification tasks compared to other architectures mentioned.

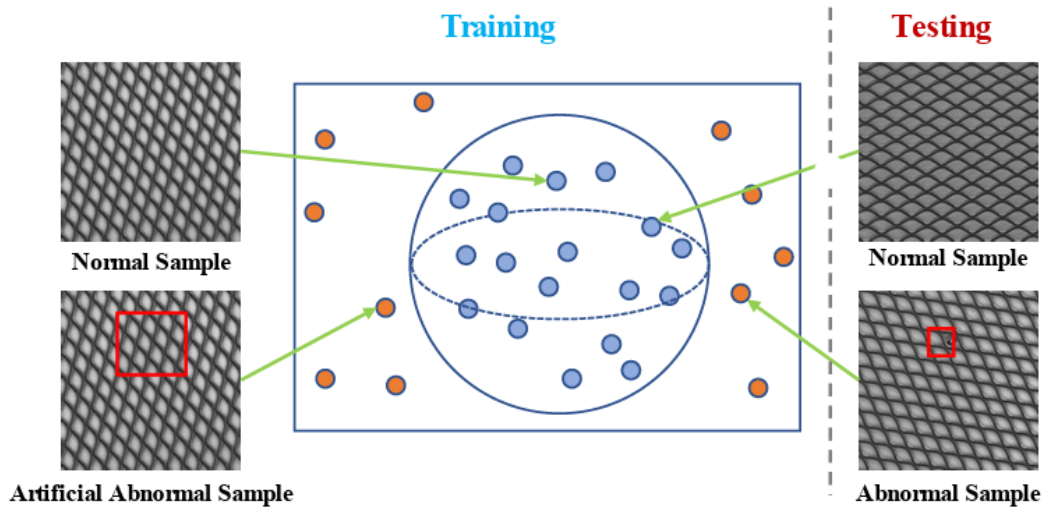


Fig. 13: Architecture of one-class classification models [37]

4.2.4 Memory Bank Methods

Memory-based approaches operate on the principle that feature representations of normal samples can be stored and later used to measure the similarity of unseen test samples. These methods construct a *memory bank* by collecting deep feature vectors in the form of patches. Those feature vectors are again extracted from the intermediate layers of the convolutional backbone. During inference, the model compares each local patch of the input image with the most similar feature in the memory bank. The anomalies are then identified by computing the distance between the extracted test features and their nearest neighbors in the memory bank (see Fig. 14). The farther the distance, the more likely the patch is anomalous. The main underlying concept is well-known K-nearest neighbors algorithm. [60, 37]

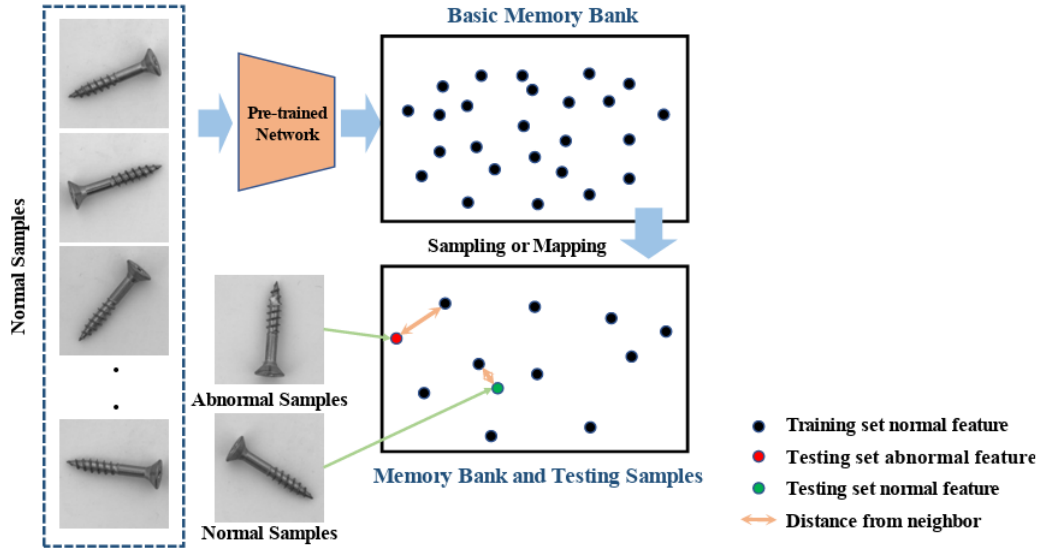


Fig. 14: Architecture of memory bank based methods [37]

This technique avoids the need for a dedicated training phase, making it very time efficient and computationally modest. However, it can be intensive for memory-constrained systems, as patches of a large dataset can take up a significant amount of space. One could argue that this approach is not a deep learning approach, as it incorporates just a basic machine learning algorithm (K-Nearest neighbors). However, that would be wrong, as this approach still relies on pretrained CNN backbones for feature extraction. [37]

One of the most prominent implementations is PatchCore [60]. It achieved state-of-the-art performance across multiple MVTec dataset benchmarks [56]. It was particularly effective in the pixel-level detection, as it is very locally oriented thanks to patches. It also performed exceptionally at the image level due to effective similarity-based scoring mechanisms. [37]

4.3 Evaluation Metrics for Image Anomaly Detection

Evaluating the performance of image anomaly detection models requires a combination of metrics that measure both image-level classification and pixel-level segmentation accuracy. These metrics offer insights into the model's ability to detect, localize, and generalize anomalies effectively.

Image-Level Metrics

Image-level detection is commonly framed as a binary classification task. A confusion matrix summarizes predictions *True Positives* (TP), *False Positives* (FP), *True Negatives* (TN), *False Negatives* (FN). From these values, several core metrics are derived. They are expressed by equations (17) – (20).

- **Accuracy:** Measures overall correctness.

$$\text{Accuracy} = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \quad (17)$$

- **Precision:** Indicates how many predicted anomalies are correct.

$$\text{Precision} = \frac{|TP|}{|TP| + |FP|} \quad (18)$$

- **Recall:** Measures how many actual anomalies are detected.

$$\text{Recall} = \frac{|TP|}{|TP| + |FN|} \quad (19)$$

- **F₁ Score:** Harmonic mean of precision and recall.

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (20)$$

These metrics are useful under different conditions. Precision is valuable when false positives are costly, while recall is critical when missing anomalies must be avoided. The F₁ score provides a balanced view. [61]

ROC Curve and AUROC

The ROC curve plots the True Positive Rate (Recall) against the False Positive Rate as the classification threshold varies. The area under curve AUROC summarizes this curve in equation (21).

$$\text{AUROC} = \int_0^1 \text{TPR}(\text{FPR}) d(\text{FPR}) \quad (21)$$

AUROC reflects the ability of the model to distinguish a true anomaly above a normal sample. A value of 1.0 indicates a perfect classifier and 0.5 indicates random guessing. AUROC is particularly useful for imbalanced datasets, which are common in anomaly detection. [61]

Pixel-Level Metrics

Pixel-level metrics assess anomaly localization. A simple approach uses pixel-wise AUROC, treating each pixel as a binary classifier. However, it may overemphasize large anomaly regions. [61]

Intersection over Union

Intersection over Union (IoU), also known as the Jaccard Index, measures segmentation accuracy; see equation (22).

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (22)$$

IoU is a strict metric that penalizes even very small misalignment. It is commonly used in object detection and segmentation, but it can be too harsh in unsupervised anomaly detection due to subtle defects or class imbalance. [61]

5 EXPERIMENTAL PART

In this section, the proposed solution is presented in detail. It begins with an introduction to the pre-filter component, which serves as the main focus of this work. The motivation for selecting this specific part is also explained. Next, the experimental setup is outlined, followed by a discussion on dataset collection and its specific characteristics. Finally, the development of two reconstruction-based models is described. They are trained and evaluated at both the pixel level and the image level.

5.1 Pre-filter Component

The part chosen for the experimental section of this thesis is a plastic pre-filter. This part serves as the first stage of air filtration in a car. It filters the incoming air to remove coarse particles such as gravel, stones, or larger insects. Then it further channels the air into the main air filter.

The manufacturing process begins with injection molding. The pre-filter consists of two main parts, which are molded in an *Injection Molding Machine* (IMM) from polypropylene blended with talc (a mineral filler composed of magnesium silicate). The next step involves equipping both parts with the rubber components necessary for sealing and compatibility with the rest of the filtration system. Finally, the two parts are welded together using vibration welding. After this, the part is inspected for functional and cosmetic defects before being packed. The final completed part is shown in Fig. 15.



Fig. 15: Complete pre-filter part

The main functional elements of the pre-filter are the grid, which filters the air, and the internal ribs, which channel the air. The pre-filter also features an upper section that is very smooth and visually appealing (see Fig. 16). This section is clearly visible in the engine bay and must therefore comply with the customer's strict visual quality standards. Several stages of the manufacturing process present a risk of creating visual defects. During injection molding, the part may not be fully formed, resulting in missing essential geometry. Molding can also leave burn marks on the smooth upper surface.

During handling and assembly, the upper section is highly prone to scratching. A critical moment occurs when placing or removing the part from the welding fixture.



Fig. 16: Pre-filter upper part

There are several quality checks throughout the process. Some are automated, such as weld quality control using conventional edge detection and alignment techniques. However, the final visual quality checks are still performed manually by operators. At the end of the welding process, before packing, operators are required to thoroughly inspect the upper surface for scratches or other abnormalities. Given the visually intensive nature of this task, combined with human fatigue during long shifts and strict customer quality requirements, it is evident that future efforts should focus on automating quality control for this smooth upper section.

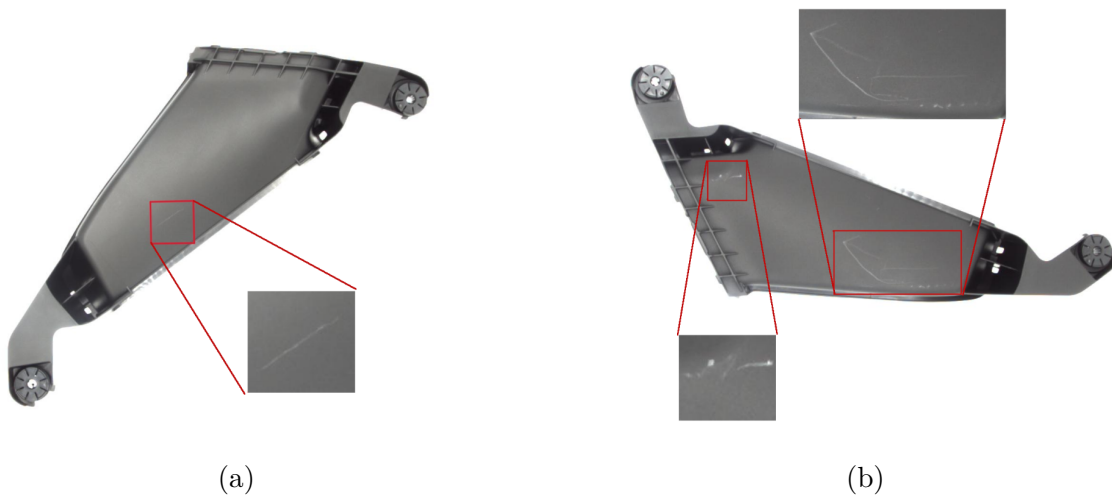


Fig. 17: (a) Subtle anomaly on the upper side, (b) Larger anomalies on the upper side

Defects on the upper surface of the pre-filter can vary widely in shape and size. The most common defects are scratches, but their size and severity can differ significantly – from small to larger (see Fig. 17). Upper part defects do not occur frequently because operators are well trained and handle this part with great care. However, this also means that there is a limited number of defective parts available to train supervised machine

learning models. The wide variability in scratch size and severity, combined with the scarcity of faulty parts, makes this component an excellent candidate for unsupervised image anomaly detection.

5.2 Experimental Setup

To accurately capture abnormalities, such as subtle scratches, the following setup was used for data acquisition. To eliminate ambient lighting effects and ensure a homogeneous and well-lit environment, a professional photography lightbox was selected. The camera was chosen based on its compatibility with a Python control library and its adjustable focal length. It is connected to the standard laptop with the necessary libraries. The exact specifics of the camera and the lighting are described further. The experimental setup used for data acquisition and later for inference testing is shown in Fig. 18.

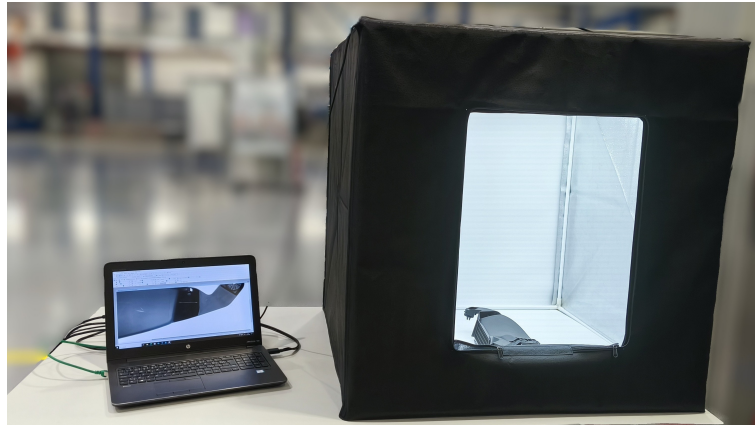


Fig. 18: Experimental setup for image acquisition

Lightbox Environment

This lightbox was selected for its excellent ability to eliminate potentially disruptive lighting biases typical for a factory environment and for its reflective inner surfaces, which help evenly distribute light. Two LED light sources are mounted inside the box. The inner sides of the box are reflective, allowing light to be dispersed in all directions. The back and bottom panels allow for interchangeable backgrounds of various colors. For our purposes, a white background was used to provide a high contrast with the black pre-filter part.

The lightbox features two camera entry points, one on the front side and one on the top. The top entry was used in our case, as the upper surface of the pre-filter is clearly visible from above when placed inside the box. The pre-filter can be freely positioned at the bottom without any fixture, as it remains stable in this orientation. The upper surface

then faces the camera directly. The camera itself was mounted using a desk mount (see Fig. 19). Key specifications of the lightbox environment are listed below:

- **Dimensions of the box:** $80 \times 80 \times 80$ cm
- **Lighting specifications:**
 - Power consumption: 24 W
 - Luminous intensity: max. 8000 lx
 - Color temperature: 5500 K (± 200 K)
 - Number of LED diodes: 2×42
 - White light – wavelength 400–700 nm

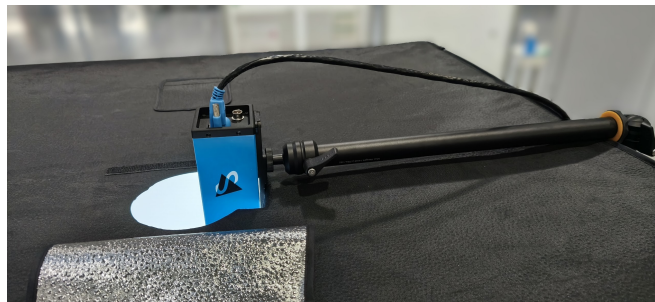


Fig. 19: Camera mounted at the top of the lightbox

Lighting Configuration

Although the setup was already partially presented, the exact configuration of the light-camera setup is necessary to discuss to better understand its functional benefits and limitations. This system employs two parallel rectangular LED strips positioned on opposite sides of a centrally-mounted camera (refer to Fig. 20). The LED strips came integrated with the lightbox, making their use a practical choice rather than a custom design. This bidirectional setup directs light perpendicular to the inspection surface, diverging from traditional circular lighting arrangements.

Scratches aligned perpendicular to the LED strips (vertical in this setup) benefit greatly from the directional lighting. Strong shadows form due to the disruption of the light path caused by the edges or depth of the scratch. This creates contrast that the camera can effectively capture. However, scratches running parallel to the LED strips (horizontal orientation) align with the light direction, resulting in minimal shadow formation and reduced visibility.

The diffusion introduced by the reflective enclosure, which is covered with a droplet-like texture, mitigates the drawback of reduced visibility for scratches that are parallel. The droplets disperse light into space, which softens the strictly directional nature of the LEDs.

Operating at 20 W, the lighting system demonstrates sufficient intensity for a reliable surface inspection. The combination of bidirectional lighting and diffusion offers a practical and hybrid solution. Using just integrated components, this setup provides reliable illumination in a variety of defect orientations without requiring a specialized lighting solution.

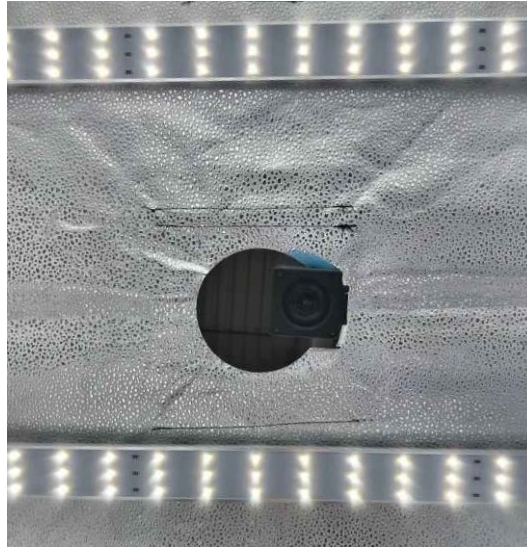


Fig. 20: Two rectangular LED strips parallel on opposite sides of camera

Camera

The requirements for the camera included a variable focal length, a color sensor, and compatibility with Python (via dedicated libraries).

Although the task itself was clearly defined, the specific details regarding the focal length required further consideration. It was important to strike a balance between capturing the entire part in the frame and focusing on the central area of the pre-filter. Capturing the whole part ensures visibility of its overall structure, but subtle anomalies may become less noticeable. In contrast, zooming in on the central area allows anomalies to occupy more pixels, making them easier to detect and analyze. However, this approach comes at the cost of losing some information about the edges of the pre-filter, as they are no longer visible in the zoomed-in view. As a result, a camera with a variable focal length was needed.

Another requirement of the industry partner was that the camera should be a color model rather than monochrome, to allow for future potential applications that involve color differentiation. This decision could be debated as a monochrome camera would have been superior for this task. Monochrome cameras are preferred in machine vision applications due to their higher light sensitivity and superior signal-to-noise ratio [24].

After evaluating the available options on the market, the DFK Z12GX236 color zoom camera from The Imaging Source was selected. This decision was based on several key factors. The camera includes a well-documented Python library [62], which simplifies installation and integration into the project. It also features a variable focal length, enabling zoom adjustments. Additionally, the camera is equipped with a color sensor, fulfilling the industrial partner's requirement for potential future applications involving color differentiation. The key parameters of the selected camera are listed below:

- **Sensitivity:** 0.05 lx
- **Dynamic range:** 8-bit
- **Resolution:** 1920×1200 pixels
- **Sensor type:** CMOS Exmor
- **Sensor specification:** Sony IMX236LQ
- **Shutter:** Rolling
- **Format:** 1/2.8 inch
- **Pixel size:** H: $2.8 \mu\text{m}$, V: $2.8 \mu\text{m}$
- **Lens:** Integrated
- **Focal length:** 4.8 mm (wide) to 57.6 mm (tele)
- **F-Stop:** 2.2 (wide) to 2.3 (tele)
- **Minimal object distance:** 3 cm (wide) to 70 cm (tele)

Geometric Imaging Calculations

For sufficient data acquisition and to ensure reliable visibility of abnormalities, it is necessary to calculate the geometrical properties of the camera and the setup itself. The pre-filter measures approximately 44 cm in length, 17 cm in width, and 10 cm in height. For a better understanding of the setup's geometry, refer to Fig. 21.

First, note that although the camera sensor is 1/2.8 inch, corresponding to a diagonal size of approximately 6.4 mm, the effective area of the sensor is reduced. This is due to cropping to 1024×1024 pixels from the original 1920×1200 resolution. This cropping is applied to obtain square images, which are preferred as input to neural networks. It simplifies data processing and ensures that the resulting images retain sufficient detail. When considering a pixel size of $2.8 \mu\text{m}$, the resulting effective size of the camera sensor is 2.87 mm (length and width).

There are a few essential parameters that must be calculated. The first important parameter is the angle at which the camera sees the scene at a fixed focal length. The *Angle of View* (AoV) can be calculated with the following formula (23):

$$\text{AoV} = 2 \cdot \arctan \left(\frac{d}{2f} \right) \quad (23)$$

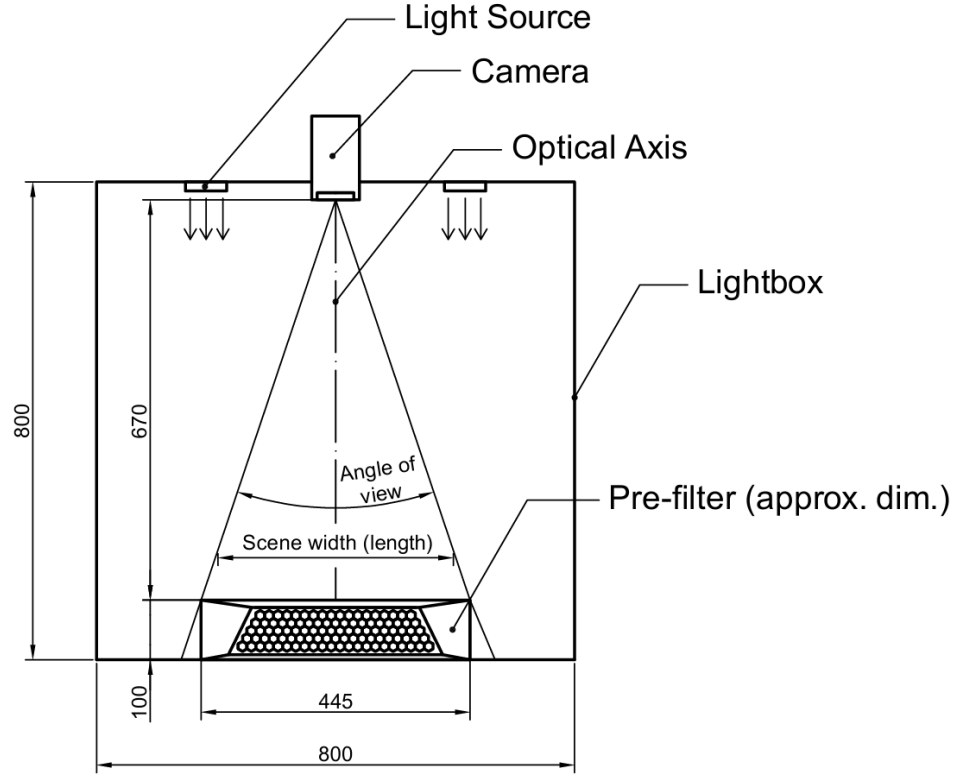


Fig. 21: Geometric illustration of camera's setup

where d is the sensor size and f is the focal length. Another crucial parameter is the width (or length) of the scene, which determines how much of the object is visible at a given distance. Knowing this is essential to ensure proper coverage of the target area [24]. The width W of the scene is calculated in the following formula (24):

$$W = 2 \cdot h \cdot \tan\left(\frac{\text{AoV}}{2}\right) \quad (24)$$

where h is the distance to the scene and AoV is the angle of view calculated previously (in radians). Later we can also calculate the diagonal scene width W_d using the Pythagorean theorem. Finally, we can calculate the spatial resolution; see equation (25), which indicates how much detail can be resolved in the image (i.e., the number of millimeters per pixel).

$$R = \frac{W}{N} \quad (25)$$

where R is spatial resolution in mm/px, W is scene width and N is a number of pixels in an image.

Although the previous geometric parameters depend solely on the imaging setup, we must now account for sensor-specific limitations that impact resolution. The effective pixel size remains $2.8\mu\text{m}$ – cropping the sensor to 1024×1024 pixels does not alter

the physical dimensions of the individual pixels. However, the Bayer color filter array introduces a critical trade-off. While enabling color imaging, it reduces resolution by distributing red, green, and blue filters in a 1:2:1 pattern. This configuration means that the green channel operates at 512×512 pixels (half the original resolution), while the red and blue channels operate at 256×256 pixels, further limiting the color-specific detail [24].

The Nyquist-Shannon theorem defines the theoretical resolution limit of the sensor. For a monochrome sensor, this limit is determined by equation (26), that calculates R_{mono} – the maximum resolvable spatial frequency in line pairs per millimeter (lp/mm).

$$R_{mono} = \frac{1}{2 \times \text{pixel size}} \quad (26)$$

For color sensors, interpolation between Bayer-filtered pixels reduces the maximum resolvable spatial frequency by approximately 30%:

$$R_{color} \approx 0.7 \times R_{mono} \quad (27)$$

To translate this sensor-limited resolution to the object plane, we calculate the effective resolvable feature size (R_{eff}), defined as the minimum distance between two distinguishable points on the object – equation (28). This metric combines the sensor’s Nyquist limit, focal length f , and working distance h .

$$R_{eff} = \frac{h}{R_{color} \cdot f} \quad (28)$$

To accommodate varying inspection requirements and computational constraints, five focal lengths were analyzed. The shortest focal length (4.8 mm) captures the entire pre-filter only when rotated diagonally due to its diagonal scene width W_d exceeding the longest dimension of the part. Longer focal lengths prioritize resolution over coverage by cropping peripheral mounting regions of the pre-filter, focusing on the central area, where defects are most critical. This approach provides enough spatial resolution to detect subtle anomalies, even after resizing the images for neural network processing. However, shorter focal lengths do not allow the entire part to be captured in a single image. Tab. 1 summarizes the geometric and resolution parameters for each focal length.

The spatial resolution R quantifies the amount of physical space each pixel covers, ranging from 0.39 mm/px to 0.13 mm/px. The effective resolvable feature size R_{eff} reflects the smallest detectable detail, spanning from 1.12 mm/lp to 0.36 mm/lp. Although R suggests single-pixel detection of submillimeter features, R_{eff} accounts for optical and sensor limitations. For example, a 0.39 mm defect would occupy one pixel but would require 1.12 mm to be distinguishable as a distinct feature.

In practice, anomalies such as scratches typically measure several millimeters, which means that they are going to be distinguishable across all focal lengths. Longer

Tab. 1: Results of geometrical calculations for each focal length chosen

f [mm]	AoV [°]	W [mm]	W_d [mm]	R [mm/px]	R_{eff} [mm/lp]
4.8	33.26	400.21	565.99	0.39	1.12
5.5	29.22	349.28	493.95	0.34	0.97
9.2	17.71	208.81	295.30	0.20	0.58
11.7	13.97	164.19	232.20	0.16	0.46
14.7	11.14	130.68	184.81	0.13	0.36

focal lengths (while cropping peripheral mounting regions) retain more detail during resizing, while shorter ones risk information loss but allow for full-part imaging with diagonal placement.

5.3 Data Acquisition

The objective of this study is to develop a model based on unsupervised learning, using only normal images without anomalies for training. However, anomalous images are required for performance evaluation.

Image Gathering Workflow

Data acquisition of defects-free images was carried out within the factory environment, near the production line, where pre-filters are assembled from molded components. Each pre-filter was taken from the production line, subjected to a visual inspection and placed in the lightbox for imaging. A Python script was executed seven times, with the part oriented differently for each run, and the resulting images were saved to the training folder. Once the imaging sequence was completed, the pre-filter was returned to the production line for further processing.

The acquisition of anomalous images followed a process similar to that of defect-free data. Parts with visible abnormalities were identified primarily by operators during routine visual inspections on the production line. When such parts were detected, they were either handed over to the author or, in the author's absence, set aside by the operator. The imaging process involved placing the defective part in the lightbox and running the script seven times, with different orientations for each run. The script was executed more times than during normal data acquisition to compensate for the scarcity of anomalous parts. The resulting images were saved to the evaluation folder under the label *Not OK* (NOK). Once the imaging sequence was completed, the defective part was retained for future inference testing. For a better understanding of the data acquisition procedure, refer to Fig. 22, which clearly shows the process.

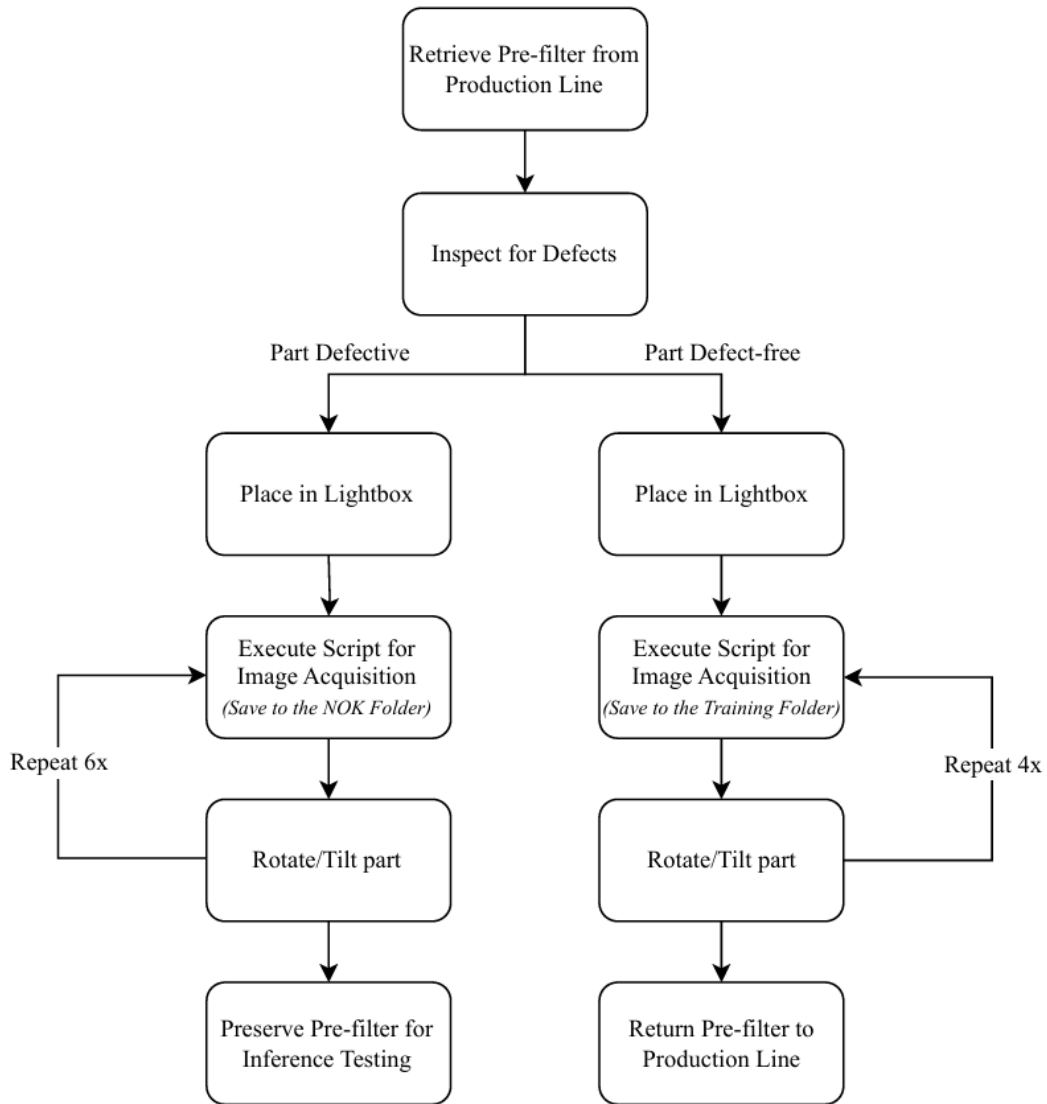


Fig. 22: Data acquisition workflow

Image Acquisition Script and Camera Settings

The image acquisition script was developed in Python using the Imaging Source library [62]. The scripts for capturing images of non-defective parts and NOK parts were nearly identical, differing only in the file path where the images were saved. Additionally, the number of script executions varied between OK and NOK parts, as discussed earlier. Before configuring the camera settings, the script sets the sensor crop to a resolution of 1024×1024 pixels to standardize the image dimensions and focus on the region of interest. The camera is switched to manual focus, and the focus value is adjusted to its optimal setting. The focus remained fixed throughout the imaging process. Besides that, several automatic parameters that could affect the consistency and reliability of image acquisition, such as *Exposure Auto-reference*, were deactivated. The script then configured the camera with the following settings:

- **Exposure time:** 1/10 s
- **Aperture:** f/4.1 (close to the lens's widest opening)
- **Gain:** 0

These parameters were carefully selected to accommodate the characteristics of the part being imaged. Since the part is black and subtle anomalies require significant illumination to become visible, the settings were chosen to produce images that show the black part clearly.

The exposure time was set to be relatively long because the acquisition speed was not a limiting factor and there was no movement that could cause motion blur. The aperture was set at f/4.1 to allow sufficient light into the camera while maintaining a balance with depth of field. The depth of field is not really a factor given the part's near-planar geometry. The gain parameter was maintained at zero to avoid introducing unnecessary sensor noise. The combination of a long exposure and a wide aperture provided sufficient image brightness without requiring signal amplification.

The script then proceeded to capture five images during each execution. For each image, it adjusted the focal length according to the predefined values listed in Tab. 1. Each captured image was assigned a unique timestamp and named according to its corresponding focal length: for simplicity, using filenames ranging from *zoom_0* to *zoom_5*, representing the shortest to longest focal lengths, respectively.

Total Images Gathered

The script produced five images during each execution. Each part was imaged seven times or five times, depending on whether it was defective or not. The precise number of parts subjected to imaging and the total number of images captured are summarized in Tab. 2.

Tab. 2: Summary of gathered images

	Parts imagined	Images per zoom	Images total
OK	450	2250	11250
NOK	35	245	1225

Although the parts were visually similar, 450 non-defective parts were subjected to imaging to account for slight visual variations that are still considered normal. This approach ensures a diverse dataset that represents acceptable deviations within normal parts. By capturing this variability, the unsupervised learning model will gain a better understanding of normal part variability.

5.4 Data Preprocessing

The acquired images provided high contrast and clear anomaly visibility. However, smaller anomalies or those with low visual salience remained challenging to see with the naked eye. To address this limitation, an enhancement algorithm was implemented to improve edge contrast and sharpen anomaly boundaries. This algorithm was applied to the entire test set, while its use in the training set was selective in order to mitigate the risk of overfitting.

The training dataset was further augmented to increase the number of training examples and enhance data variability. Finally, for all NOK examples in the test dataset, binary masks were created to enable the evaluation of the pixel-level accuracy for each proposed model.

Enhancement Algorithm

The script was implemented in Python using the OpenCV and NumPy libraries. Its primary objective was to enhance local contrast and sharpen the boundaries of defects. The script processes each image in three main steps:

1. **Contrast Enhancement with CLAHE:** The image is first converted to the LAB color space, and then the CLAHE is applied. This method increases local contrast, making faint or low-salience anomalies more noticeable without excessively amplifying noise.
2. **Gamma Correction:** After contrast enhancement, the gamma correction is applied with a gamma value less than 1 (specifically, $\gamma = 0.8$). This adjustment brightens darker regions of the image, further improving the visibility of subtle features.
3. **Unsharp Masking:** Finally, an unsharp mask filter is used to sharpen the image. This is achieved by subtracting a blurred version of the image from the original and then combining the result with the original image. The process emphasizes edges and fine details, making anomaly boundaries more distinct.

The script iterates over all images in the test set directory, applies the described enhancement pipeline, and saves the processed images to the output directory. Comparison of a defective part before and after enhancement can be seen in Fig. 23.

Augmentation

To improve the robustness and generalization of the model, a comprehensive data augmentation strategy was implemented. Each original image was used to generate three additional augmented variants, effectively increasing the size of the training dataset by a factor of four. Since the original images are already very bright, the augmentation

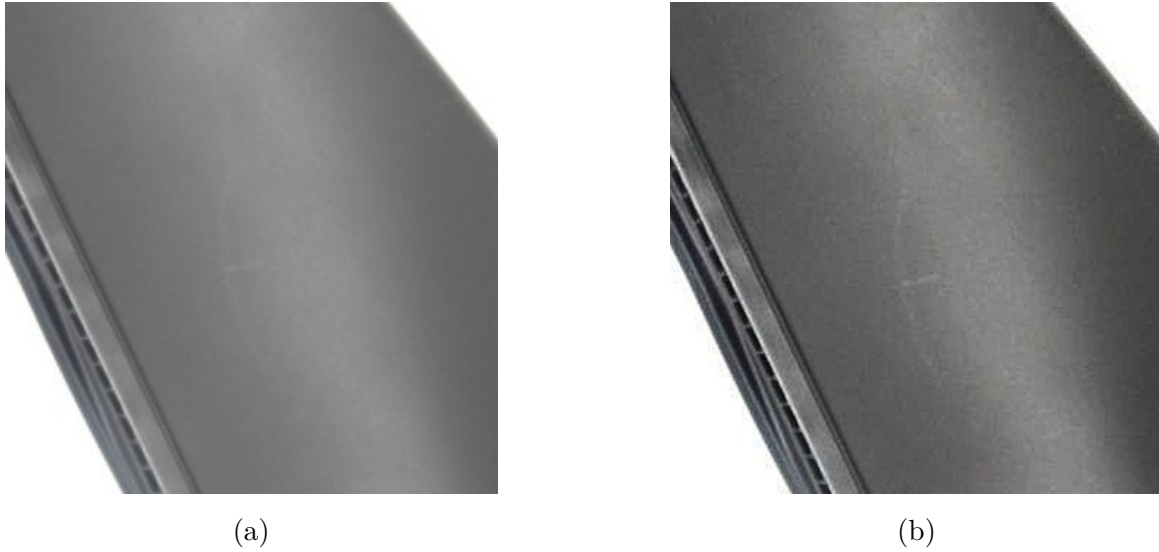


Fig. 23: Zoomed-in images of barely visible scratch: (a) Original image, (b) Enhanced image

parameters were chosen carefully to avoid overexposure. The following photometric and geometric adjustments were applied to the images with specified probabilities and ranges.

- **Gamma correction** (30% probability), to simulate varying lighting conditions and camera response in range of $[0.7, 1.25]$.
- **Brightness adjustment** (30% probability) to account for differences in illumination – sampled from $[0.8, 1.05]$, the upper value was kept low to avoid overexposure.
- **Contrast adjustment** (30% probability), drawn from $[0.8, 1.2]$.
- **Gaussian noise addition** (three intensity levels, 30% probability), to simulate sensor imperfections and electronic noise. The levels are: $[-2, 3]$, $[-5, 6]$, $[-10, 11]$.
- **Gaussian blur** (20% probability), to simulate defocus or motion blur. The size of the blur kernel is randomly chosen as 3×3 , 4×4 or 5×5 .
- **Horizontal and vertical flips** (40% probability each), to increase the invariance to the orientation of the part.
- **Rotation** (15% probability), rotates the part for 90° , 180° or 270° – it avoids the introduction of empty pixels or interpolation artifacts.

To ensure that every augmented image is truly unique, a minimal noise augmentation was applied whenever no other transformation was selected. Furthermore, for 15% of the original training data, a previously mentioned test enhancement was applied to ensure robust training.

Pixel-level Anomaly Annotation

Pixel-level annotation was performed using the Computer Vision Annotation Tool (CVAT) [63], an open-source web-based tool for creating segmentation masks and other types of annotation. The anomalous regions in the NOK images were manually labeled using the brush tool and exported as binary masks, where normal pixels are assigned a value of zero and anomalous pixels a value of one. Visually, these masks are represented as black and white images: black indicates normal regions, while white corresponds to anomalies (see Fig. 24).



Fig. 24: (a) Original image of anomaly, (b) Ground truth binary mask

5.5 Development of Models

The following section is a key component of this thesis. It describes the development of unsupervised anomaly detectors for the collected data.

For model development, the Python programming language was used in combination with the PyTorch framework. PyTorch was chosen over TensorFlow because of its more Pythonic syntax, strong community support, and modern library ecosystem. It also offers easy integration with *Compute Unified Device Architecture* (CUDA) for efficient *Graphics Processing Unit* (GPU) acceleration. The development environments included both Jupyter Notebook and Visual Studio Code. All key code developed for this thesis is publicly available in a GitHub repository.¹ The code also includes architectures that were not used in the final experiments.

All experiments and training were conducted on a workstation equipped with an NVIDIA GTX 1080 Ti (11 GB VRAM) and 32 GB RAM. Although the GPU supports CUDA acceleration, memory limitations occasionally arose, particularly when using the

¹ <https://github.com/Joey7500/Image-Anomaly-Detection-Models>

second model, which operates on high-dimensional tensors that can rapidly exhaust the available GPU memory.

5.5.1 Methodology

We develop two models. Both are based on reconstruction. This type of anomaly detection architecture was chosen primarily because of the author's familiarity with their design and because they are programmed intuitively. The way in which they discover anomalies is also quite intuitive and easy to understand.

The first is a **Baseline Autoencoder with Attention**. This model is developed from scratch by the author. It is presented more briefly, as it mainly serves to demonstrate the limited capabilities of plain image autoencoders for this task.

The second model is based on **DFR** paper [57]. Rather than reconstructing raw pixels, it relies on reconstructing intermediate features extracted by pretrained backbone. The autoencoder architecture was taken from this article, but the feature extraction and aggregation were redesigned. This approach is therefore rather redesigning than developing from scratch.

Datasets

Each model is tested on two datasets. The first dataset (Dataset-A) contains pre-filter images from *zoom_0* and *zoom_1*. In these images, the entire part is visible when placed diagonally or only slightly cropped when aligned parallel to the camera sensor edges. This setup allows the models to inspect the whole part in a single image, although the anomalies appear smaller. The second dataset (Dataset-B) includes images from *zoom_3* to *zoom_5*, where the part is cropped more significantly, especially in *zoom_5*. However, these images still capture the most important shiny surface in the center of pre-filter.

For computational efficiency, given that the datasets contain tens of thousands of images, the input resolution was reduced from 1024×1024 to 512×512 pixels. As explained in Section 5.2, there is a considerable margin for resizing.

Augmentation

Both models were trained on the same augmented datasets: Dataset-A with 16,000 images and Dataset-B with 24,000 images

Testing

Testing on image as well on pixel-level was conducted under the same conditions on the same testing datasets. Image level testing was performed on 438 images per class for Dataset-A and 580 images per class for Dataset-B. Pixel-level evaluation was performed on 480 ground truth NOK segmentation masks for Dataset-A, and 350 for Dataset-B.

5.5.2 Baseline Autoencoder with Attention

This proposed model is based on the principle that autoencoders typically struggle to accurately reconstruct anomalous regions that were not present during training. As suggested earlier, this model is developed from scratch by the author.

Model Architecture

The proposed model is a basic convolutional autoencoder, characterized by its symmetric encoder-decoder structure and a single skip connection. For a better understanding, refer to Fig. 25. The numbers in the figure refer to in-channels, out-channels, width and height (e.g. 3, 32, 256, 256).

The model consists of three main components: an encoder, a bottleneck, and a decoder. The encoder is made up of four convolutional stages. Each of the layers reduces the spatial resolution of the input image using 2D convolutional layers with a stride of 2. For example, a layer defined as `Conv2d(3, 32, 4, 2, 1)` takes an input with 3 channels (RGB), outputs 32 channels, and uses a kernel size of 4×4 , stride 2 (for downsampling) and padding 1 (to preserve alignment). Those parameters result in halving the spatial resolution at each step, producing feature maps with spatial sizes: $512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32$. Simultaneously, the number of channels increases across the encoder stages – from 3 to 32, 64, 128, and finally 256. In this way, the network learns richer and more abstract representations.

The bottleneck performs 1×1 convolution to reprioritize certain semantic features before reconstruction begins. This also further lowers the computational load. The decoder mirrors the encoder using `ConvTranspose2d` layers to upsample the feature maps back to the original size. It concludes with *Sigmoid* activation to map the output to the range $[0, 1]$.

To improve the capability of the model, two advanced techniques are integrated. First, a mid-level skip connection is implemented between encoder layer 3 and decoder layer 1. This connection allows the decoder to recover fine-grained spatial details by merging lower-level features with higher-level semantics. Second, each decoder stage includes a CBAM, which applies both channel and spatial attention mechanisms to adaptively refine feature maps and highlight informative regions.

Training

The proposed autoencoder was trained using the AdamW optimizer with a learning rate of 1×10^{-3} and a weight decay factor of 1×10^{-5} to promote generalization and reduce the risk of overfitting. Training was carried out for 25 epochs with a batch size of 16.

Each training iteration consisted of a forward pass, where the model reconstructed the input image and produced a latent representation. The loss function was a composite loss of the MSE and the SSIM, expressed in formula (29). The composite loss was chosen due to its potential superior results over standard approaches [38, 39].

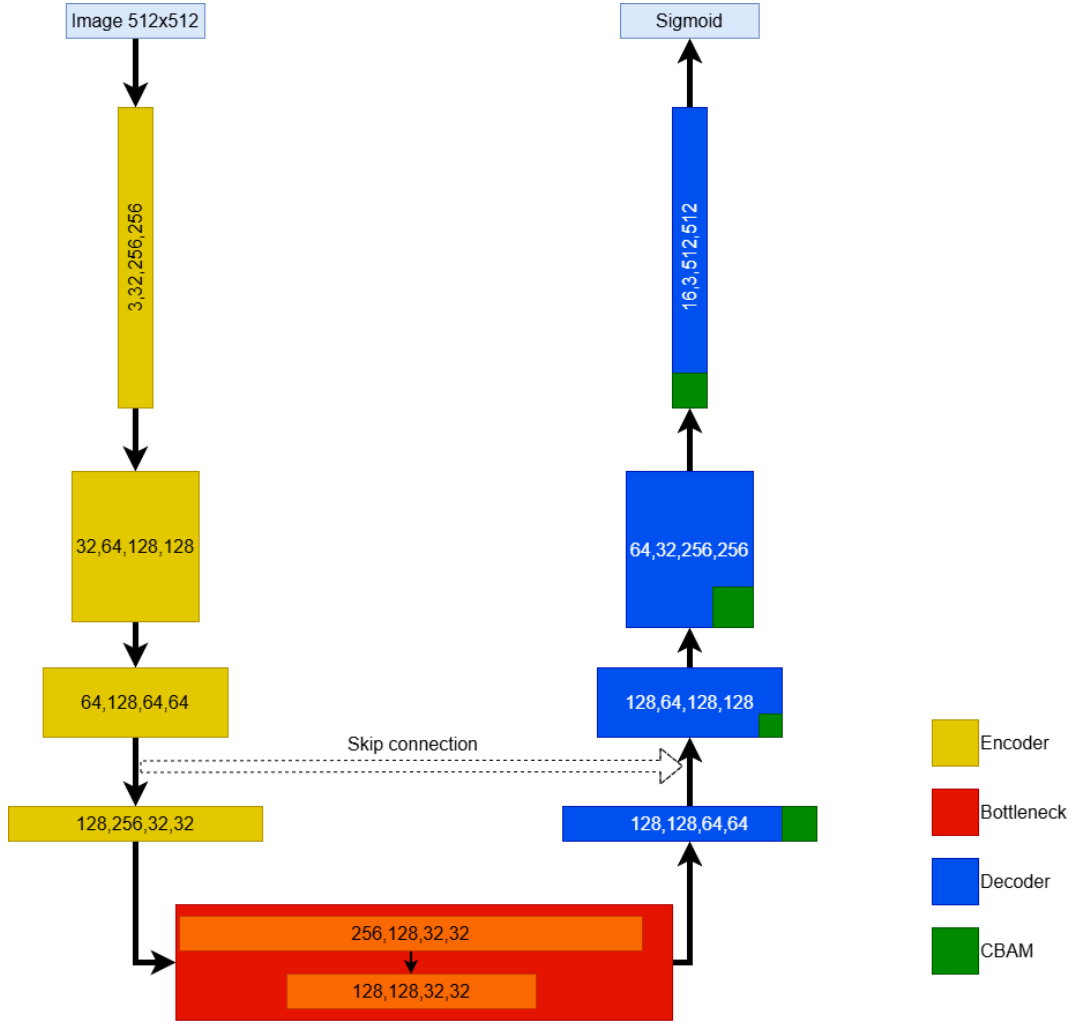


Fig. 25: Architecture of the baseline autoencoder with attention

$$\mathcal{L}_{\text{total}} = 0.15 \mathcal{L}_{\text{MSE}} + 0.85 \mathcal{L}_{\text{SSIM}} \quad (29)$$

The model parameters were updated through backpropagation and the training loss was monitored across epochs. Upon completion, the model weights were saved for later evaluation.

Testing

The model was evaluated first at the image level, distinguishing between normal and NOK images based on their true labels. The decision was made using a decision function that computed the mean of the k highest pixel-wise reconstruction errors. The final value was chosen to be $k = 1500$. Depending on the threshold, this mean value determined whether the image was classified as anomalous. Subsequently, the evaluation proceeded at the pixel level, assessing the model's ability to localize anomalous regions within images. A detailed analysis of the obtained results is presented in Section 5.6.

5.5.3 Deep Feature Reconstruction Autoencoder

In the following sections, the second model is described in detail. Although numerous configurations with varying hyperparameters were explored during development, the discussion below focuses solely on the setup that yielded the best performance. Occasionally, alternative hyperparameter choices are mentioned for context.

Overall Principle of the DFR Method

The DFR approach is an unsupervised anomaly detection technique introduced by Yang *et al.* [57]. It operates by extracting multi-scale regional features from a pre-trained convolutional network and then reconstructing these features with a convolutional autoencoder.

The key idea is that by generating spatially-aware feature descriptors for each image subregion and training an autoencoder to reconstruct them, anomalous features (which differ from normal training data) will cause large reconstruction errors. As Yang *et al.* [57] explain, DFR “generates multiple spatial context-aware representations from pre-trained deep convolutional networks for every subregion of an image” and then uses a “deep yet efficient convolutional autoencoder” to detect anomalies via fast feature reconstruction. The author has decided to build the second model upon this approach, use some parts of the existing solution, and redesign others.

Feature Extraction from EfficientNet-B6

The original approach [57] used the VGG-19 network [44], which is old and very heavy parametrically. In our approach to feature extraction, we use EfficientNet-B6 as the backbone. EfficientNet-B6 is a pretrained network known for its good accuracy–complexity trade-off. It achieves around 84.0% top-1 accuracy on ImageNet [47]. Its compound scaling makes it effective on high-resolution inputs (its pretrained input size is 528×528 , and here we use 512×512).

We hook six layers of the network (corresponding to different stages of its inverted bottleneck blocks) to capture features at multiple scales. The early blocks of EfficientNet downsample the input gradually (for a 512×512 input, feature maps are 256×256 , while the deeper blocks produce coarser maps (e.g. 32×32 or smaller). By choosing one layer from each major block (layers indexed 1 through 6 in the PyTorch implementation), we obtain feature maps that represent local regions in different receptive fields. These multi-scale features collectively encode both fine-grained and contextual information relevant to 512×512 images.

It should be noted that an additional experiment was conducted in which features from all seven layers of the backbone were extracted. However, incorporating the seventh layer introduced a 576-channel feature map, which significantly increased the computational demands of the subsequent processing stages. Ultimately, the seventh layer did not bring any increase in performance, probably because the features were too abstract for the task anyway.

Aggregating the Features

Each selected feature map is first upsampled back to the input resolution (512×512) using bilinear interpolation. Then we aggregate local neighborhoods by applying a 3×3 average pool (stride 3) to each upsampled map. This effectively reduces each map to 170×170 (since $512/3 \approx 170$), resulting in compact multi-scale representation. If the six hooked layers produce $c_1, \dots, c_6$ channels, respectively, the aggregation pipeline produces six tensors of shape $[B, c_i, 170, 170]$ (batch size B). Those tensors are concatenated along the channel dimension to form a single tensor of shape $[B, \sum_i c_i, 170, 170]$. In our implementation, the sum of channels is 832. The concatenated tensor is thus $[B, 832, 170, 170]$. These aggregated feature maps serve as input to the subsequent autoencoder.

Preliminary experiments also explored the use of a 4×4 average pooling kernel with a stride of 4, yielding feature maps of size 128×128 . This configuration produced promising results under the same training and testing conditions. At the image level, it was eventually outperformed by the 3×3 average pooling setup with stride 3. However, it performed better on the pixel-level evaluation, more in section 5.6.

Autoencoder Architecture

Despite the architectural simplicity, a 1×1 convolutional autoencoder from the original paper [57] was used in our final solution. Its parametric structure is clearly visualized in Tab. 3. In practice, this architecture proved to be remarkably effective for the task. Its low memory footprint and fast inference were critical given the high-dimensional input tensor of shape $[B, 832, 170, 170]$.

During development, multiple more complex alternatives were tested, including increasing the size of the convolutional kernel and incorporating attention mechanisms such as CBAM. However, these modifications did not produce any improvements in anomaly detection performance. They achieved similar results, but at the cost of prolonged training and higher computational load. This may be due to several factors:

- **1×1 convolutions already capture channel-level correlations effectively.** For DFR, which reconstructs deep features rather than pixel RGB values, inter-channel relationships may be more important than spatial ones.
- **The EfficientNet backbone already embeds spatial context.** By the time the feature maps are extracted from six different layers and aggregated, the spatial structures have been encoded. Additional spatial attention could be redundant.
- **Overfitting or training instability.** Attention modules add redundant parameters, which could have possibly led to slight overfitting.

Increasing the kernel size in our autoencoder to 3×3 in an effort to capture a richer spatial context proved to be extremely memory intensive and, despite employing mixed precision training, resulted in an out-of-memory overflow.²

As a result, the final autoencoder architecture was copied the “simple yet efficient” design used in the DFR paper [57]. Nevertheless, the choice of latent dimension remains a crucial hyperparameter and should be selected carefully based on the desired percentage of data variance to keep.

Tab. 3: Architecture of the feature autoencoder used in final experiments

Input: $f_{1:6}(\mathbf{x})$ ($h_o \times w_o \times c_o$)
[Layer 1]: Conv($1 \times 1, c_o \rightarrow \frac{c_o+2c_l}{2} \rightarrow 2c_l$); BN; ReLU
[Layer 2]: Conv($1 \times 1, \frac{c_o+2c_l}{2} \rightarrow 2c_l$); BN; ReLU
[Layer 3]: Conv($1 \times 1, 2c_l \rightarrow c_l$)
[Layer 4]: Conv($1 \times 1, c_l \rightarrow 2c_l$); BN; ReLU
[Layer 5]: Conv($1 \times 1, 2c_l \rightarrow \frac{c_o+2c_l}{2}$); BN; ReLU
[Layer 6]: Conv($1 \times 1, m_1 \rightarrow c_o$)

Here, $f_{1:6}$ are concatenated EfficientNet layers, h_o and w_o are input spatial dimensions, c_o is the number of input/output channels, c_l is the latent dimension and BN is batch normalization.

Dimensionality Estimation via PCA

Before training the autoencoder, we flatten and apply PCA to the aggregated feature maps to determine a suitable latent dimension. The 170×170 spatial feature maps with 832 channels are flattened into vectors of size 832 and stacked across all spatial positions and samples, forming a matrix. To keep the computation reasonable, we used 100 randomly selected training images to construct this matrix. We then fit PCA and examine the explained cumulative variance. The number of principal components required to retain 90% of the total variance determines the latent dimensionality of the autoencoder. In our case, this value consistently fell in the range of 200-260, depending on whether Dataset-A or Dataset-B was used. For the final training, a latent dimensionality of 210 was used for Dataset-A, and 260 for Dataset-B.

Training Setup and Hyperparameters

We use MSE as the loss criterion and Adam optimizer with a learning rate of 10^{-3} . The model is trained for 150 epochs with a batch size of 4. There is no composite loss function as in the first model training, because the features are not RGB pixel values and are too abstract for metrics like SSIM.

² Mixed precision training reduces memory usage by using 16-bit floating point arithmetic (FP16) for certain operations, while keeping critical ones in 32-bit (FP32).

During training, we also computed a validation loss on the test set, which includes both normal and abnormal images. Although this may initially appear counterintuitive, since the test set is typically reserved for final evaluation, this step serves as a useful indication of performance early on. A consistently higher validation loss indicates that the model struggles to reconstruct anomalous images, which is a desirable outcome in anomaly detection.

This behavior suggests a low risk of overfitting, where the model would otherwise learn to reconstruct even anomalous regions well. Importantly, the validation phase is conducted under `torch.no_grad()`, ensuring the model parameters remain unchanged and the test set is only used for inference. However, we also should not change the training settings based on this validation loss, as we would therefore indirectly bias towards test set overfitting. The training and validation losses during the training process of a best model are presented in Fig. 26.

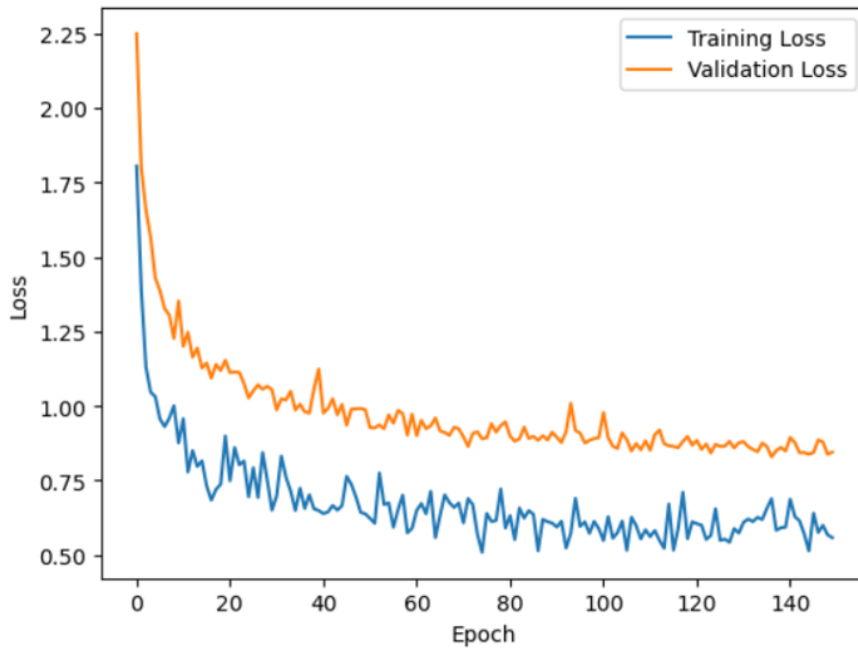


Fig. 26: The training and validation losses during the best model training

To assess convergence, we monitor the minimum training loss in the final few epochs and save the model weights for this configuration.

Testing

For each test image, we compute its feature map through the backbone and feed it through the autoencoder to obtain a reconstructed feature map. We then compute a residual map of squared errors averaged across the channel dimension to produce a single-channel anomaly heatmap of size 170×170 . To reduce edge artifacts (due to padding or border

effects in upsampling [57]), we cropped a 20 pixel border from each side of this heatmap before scoring. This heatmap is later upsampled to the original 512×512 resolution. It is visualized with the border artifacts, but the score itself is calculated without those values.

The decision function then flattens this cropped map, sorts the pixel values in descending order, and computes the mean of the top- k values. We used the mean of $k = 20$ as the anomaly score for the best performing model. This number is significantly lower than the one in the first model, where we took raw reconstructed pixel values. The results on image and pixel levels are discussed in the next section 5.6.

5.6 Final Results

The results are reported separately for the first and second model on image-level and pixel-level performance. For our application, image-level classification is the primary objective – determining whether a given pre-filter image contains an anomaly or not (OK/NOK binary classification). Pixel-level metrics, while secondary, offer valuable insight into whether the model truly detects localized anomalies or just generates false activations. The results section also includes qualitative examples such as reconstruction error heatmaps and thresholded binary masks, which are useful for interpreting the model’s pixel-level behavior.

5.6.1 Results of Baseline Autoencoder with Attention

The quantitative results on both the image and pixel level are summarized in Tab. 4. The threshold-specific metrics: accuracy, precision, recall, and the F_1 score are calculated to the threshold that resulted in the highest F_1 score. The confusion matrices for both datasets (with respect to the highest F_1 score), which are fundamental for the table calculations, are in Fig. 27.

Tab. 4: Performance of the Baseline Autoencoder with Attention for each dataset

Data	Accuracy	Precision	Recall	F_1	AUROC	Pixel AUROC
Dataset-A	0.59	0.58	0.61	0.60	0.61	0.85
Dataset-B	0.57	0.56	0.62	0.59	0.63	0.58

The results indicate that the model performed poorly on both datasets in terms of image-level classification. The image-level AUROC reached only 0.61 and 0.63 for Dataset-A and Dataset-B, respectively. These values suggest the model is only slightly better than a random classifier, with classification accuracies close to 60%.³ This also suggests that

³ Despite the unsatisfactory results, this model still performed better than Baseline Autoencoder, without attention and skip connection (its AUROC was lower by approximately 10% on both levels).

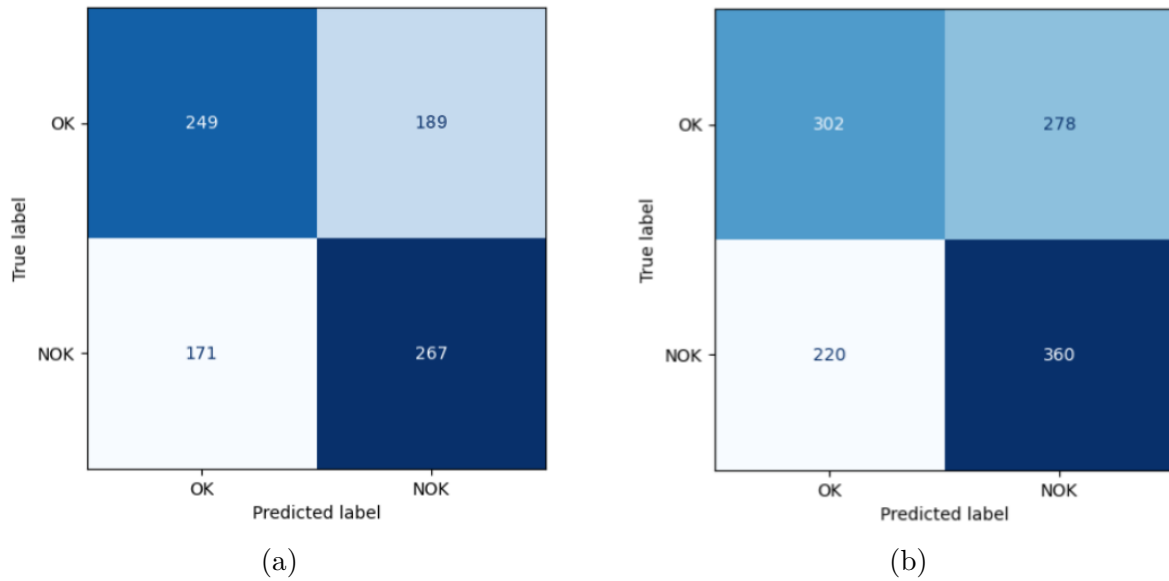


Fig. 27: (a) Confusion matrix for Dataset-A, (b) Confusion matrix for Dataset-B

on image level models will likely perform better on Dataset-B, since the anomalies are more visible.

What initially seemed surprising was the relatively high pixel-level AUROC on Dataset-A, reaching 0.85 despite the weak image-level performance. In contrast, for Dataset-B, the pixel-level AUROC dropped significantly to 0.58. This discrepancy can be explained by considering the structure of the datasets and the nature of the autoencoder's reconstruction. In Dataset-A, the parts are more zoomed out, so the images contain a larger proportion of background. The autoencoder effectively reconstructs this background while incorrectly marking most of the part anomalous; see Fig. 28. Since AUROC evaluates whether anomalous pixels tend to receive higher scores than normal ones, this behavior still yields a high score, even when the model fails to pinpoint the exact location of the anomaly. In Dataset-B, the parts are more zoomed in, and the images include much less background. The model no longer benefits from the reconstruction of large simple areas and struggles to separate anomalies from normal details within the part, leading to a lower and more realistic pixel-level AUROC.

Some indications of pixel-level awareness were still observed, particularly in cases with more visible anomalies (see Fig. 29). In the figure, highly visible anomaly is labeled as anomalous and neighboring pixels are labeled as normal. There is still a high contrast of the background and the part present – a lot of normal pixels on the part are labeled as anomalous.

In conclusion, the first model showed poor performance, which can be attributed to the limitations of its loss function and the complexity of the task. Although a hybrid loss combining MSE and SSIM was used, it did not sufficiently prevent the model from



Fig. 28: Failure of the pixel-level classification despite great pixel-AUROC on Dataset-A image: Original image (left), Ground truth mask (middle), Predicted binary mask (right)



Fig. 29: Indication of some sort of pixel level awareness: Original image (left), Ground truth mask (middle), Predicted binary mask (right)

focusing on easy-to-reconstruct regions such as the background. This led to the model ignoring most of the anomalies in more complex parts of the image – the pre-filter.

To address this issue, future improvements could design a loss function that penalizes overconfident reconstructions in uniform regions such as the background. More aggressive data augmentation could also be beneficial, for example, by adding even stronger Gaussian noise to eliminate easy uniform areas. Overall, due to its limited effectiveness, the model is not applicable for the intended anomaly detection task.

5.6.2 Results of the Deep Feature Reconstruction Autoencoder

For DFR models, we chose to present the results of the two best performing models. Those models perform similarly, but each highlights different benefits of the DFR approach. Although both models utilize feature maps extracted from layers 1 through 6 of EfficientNet-B6, they differ in their feature aggregation. The final concatenated feature map of the first model has size of 128×128 , while that of the second model 170×170 . All quantitative results are summarized in Tab. 5. The threshold-specific metrics (accuracy, precision, recall, and F_1) are calculated using the threshold that maximizes the F_1 score. The models are referred to as Model-128 and Model-170, based on their respective fea-

ture map resolutions. For efficiency, each configuration is named using the naming format $\langle \text{feature map size} \rangle - \langle \text{dataset} \rangle$, e.g. *128-A* denotes Model-128 evaluated on Dataset-A. The corresponding confusion matrices of all test configurations are in Fig. 30.

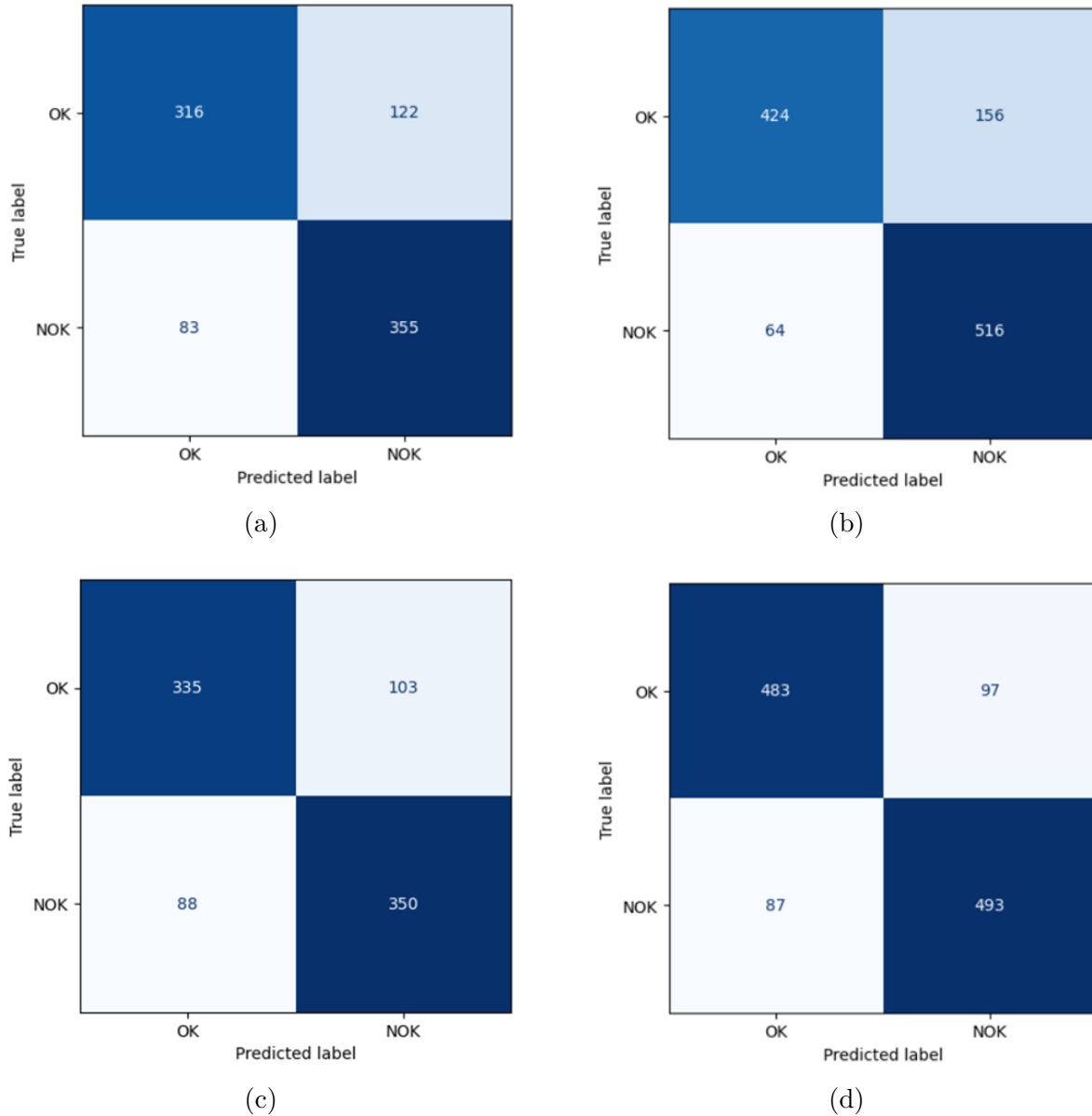


Fig. 30: Confusion matrices of all testing configurations: (a) 128-A, (b) 128-B, (c) 170-A, (d) 170-B

The evaluation demonstrates that both DFR models performed reliably across both datasets, with particularly strong results at the image level. Model-170 achieved higher classification accuracy and AUROC, especially on Dataset-B. On the other hand, Model-128 achieved the higher pixel-level score AUROC across both datasets.

At the image level, Model-128 achieved an accuracy of 77% and an AUROC of 0.86 on Dataset-A. Its performance further improved on Dataset-B, reaching 81% accuracy and

Tab. 5: Performance of the DFR Autoencoder models across both datasets

Config.	Accuracy	Precision	Recall	F ₁	AUROC	Pixel AUROC
128-A	0.77	0.74	0.81	0.78	0.86	0.89
128-B	0.81	0.77	0.89	0.82	0.90	0.82
170-A	0.77	0.78	0.80	0.79	0.86	0.88
170-B	0.84	0.84	0.85	0.84	0.92	0.79

an AUROC of 0.90. This confirms the hypothesis that the model benefits from zoomed-in views of the parts as the anomalies are more visible. Model-170 achieved the same accuracy and AUROC as Model-128 on Dataset-A but outperformed it on Dataset-B, reaching an accuracy of 84% and AUROC of 0.92 (see Fig. 31).

At the pixel level, all configurations also demonstrated decent performance, with AUROC scores consistently above 0.79. A slight increase in performance of both models can be observed on Dataset-A, likely due to the larger presence of a uniform white background. Nevertheless, the pixel AUROC score difference between datasets is less pronounced than in the baseline autoencoder model – indicating, the artificial pixel-level AUROC increase is insignificant. Specifically, Model-128 achieved pixel-level AUROC scores of 0.89 (Dataset-A) and 0.82 (Dataset-B), while Model-170 scored 0.88 (Dataset-A) and 0.79 (Dataset-B). Model-128 therefore outperformed Model-170 at a pixel level.

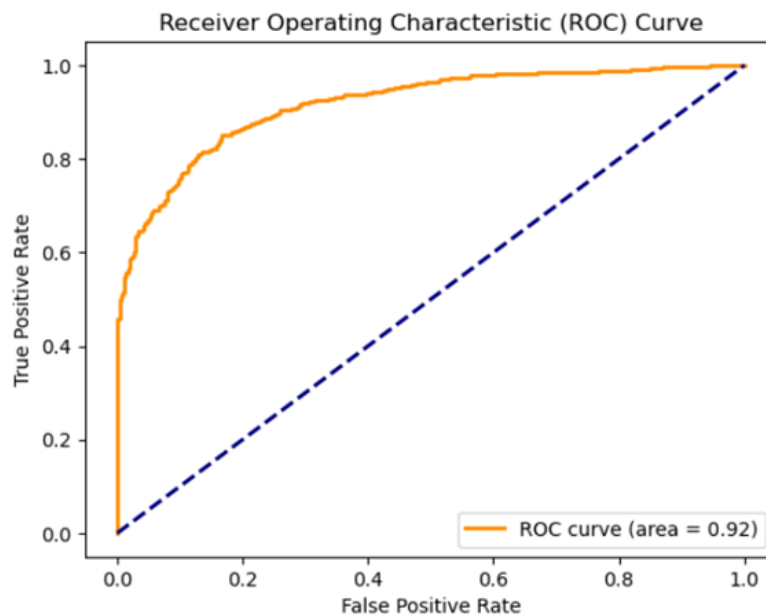


Fig. 31: ROC curve of 170-B model

At first glance, it may seem counterintuitive that the 128×128 model achieves better pixel-level scores than the higher-resolution 170×170 model, since the higher resolution should preserve more spatial detail. However, this can be explained. The author suggests that the more aggressive spatial pooling in the 128×128 feature maps smooths out fine textures and noise. It probably helps the model to focus more effectively on the most relevant regions. In contrast, the 170×170 resolution preserves fine-grained detail useful for detecting subtle anomalies, but this precision may also capture irrelevant image noise, leading to worse localization. The predicted binary masks and heatmaps that underline the Model-128 performance can be seen in Fig. 32, for Dataset-A, and in Fig. 33, for Dataset-B.

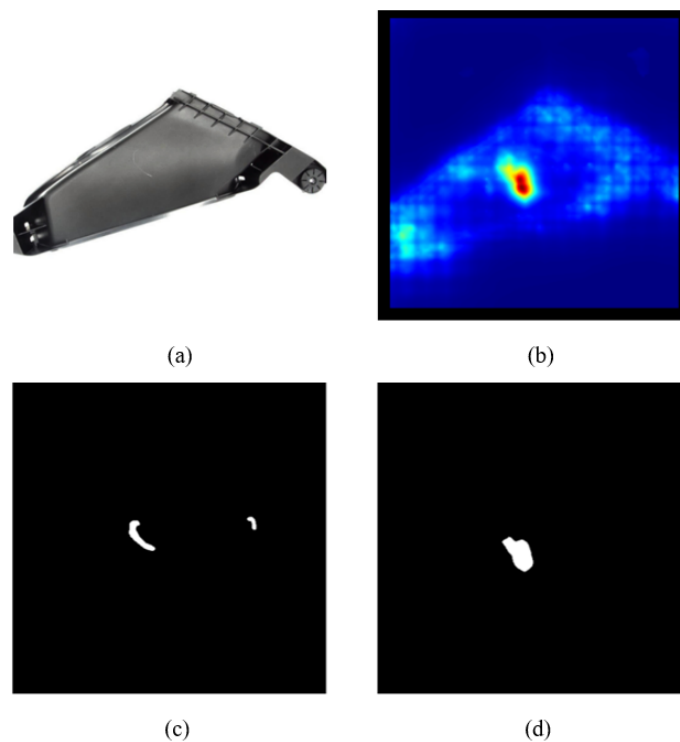


Fig. 32: 128-A performance: (a) Original image, (b) Anomaly heatmap, (c) Ground truth mask, (d) Predicted mask

In summary, Model-128 smooths out the noise and irrelevant texture variation, which helps it with pixel-level performance. On the other hand, it may overlook subtle anomalies that Model-170 picks up and includes them in its decision on the image classification level.

Overall, the DFR models performed with satisfactory results and can be implemented for inference. The final implementation of the 170-B configuration can be seen in the Fig. 34 and Fig. 35. The significant drawback of those models is that they in fact crop the classified results by approximately 10% from each side (because of padding artifacts). This will result in missed anomalies on the borders of the image.

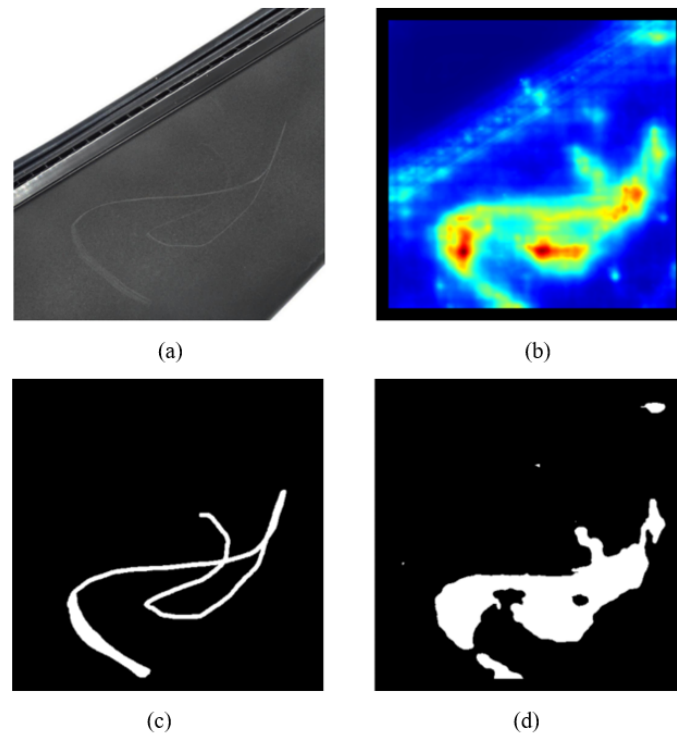


Fig. 33: 128-B performance: (a) Original image, (b) Anomaly heatmap, (c) Ground truth mask, (d) Predicted mask

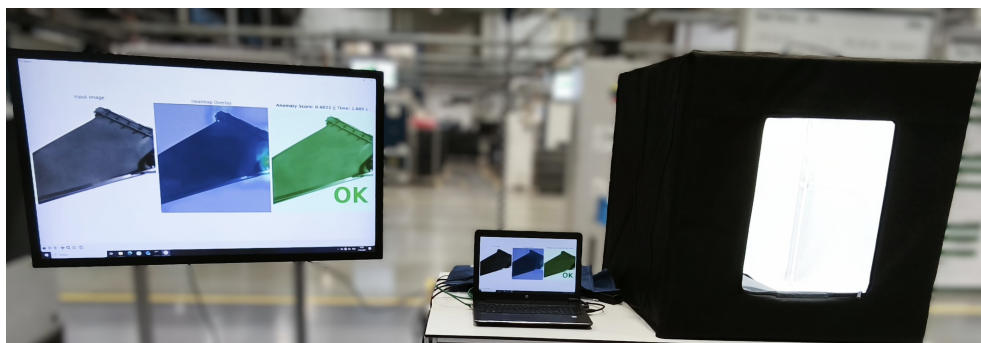


Fig. 34: Final implementation setup; example of a part classified as OK

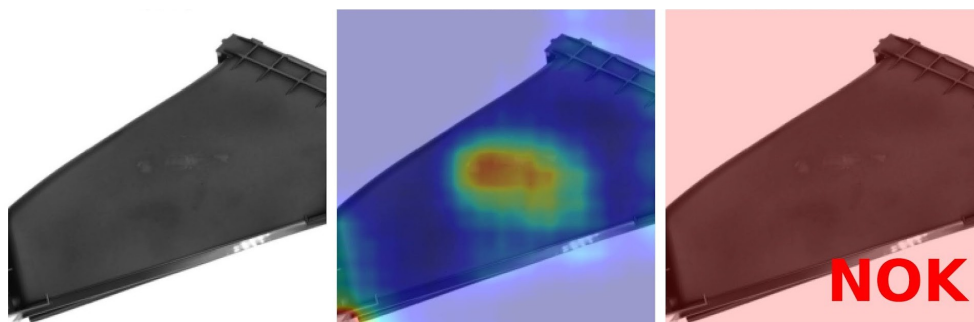


Fig. 35: Visual output of the final implementation; part classified as NOK

6 CONCLUSION

The primary objective was to develop an unsupervised deep learning model for detecting visual anomalies on pre-filter components, which are prone to surface defects such as scratches. To achieve this goal, first a dataset was collected directly from the production line. Each part was photographed under various zoom levels and orientations. This process yielded approximately 11,000 images of defect-free parts and 1,200 images of defective parts.

Two unsupervised models were developed using the PyTorch framework and trained on two datasets: Dataset-A (showing zoomed-out parts) and Dataset-B (showing more zoomed-in views). The first model was a conventional autoencoder with attention modules, developed from scratch by the author. It failed to achieve usable results, likely due to an insufficient loss function and the complexity of the pre-filter dataset. Looking back, incorporating a background penalization mechanism could have improved performance.

The second model, based on DFR, significantly outperformed the baseline autoencoder. Instead of reconstructing raw pixel values, it reconstructed intermediate features extracted from a custom EfficientNet-B6 backbone. The autoencoder architecture was adapted from the original DFR paper, but the backbone and feature aggregation strategy were redesigned. This approach provided strong results at both the image and pixel levels, with the best configuration (Model-170-B) achieving an image-level AUROC of 0.92. Pixel-level segmentation was limited by the lower spatial resolution of the feature map compared to the input image size (170×170 compared to 512×512). That inherently limited pixel-level AUROC, which ultimately remained below 0.90 for all tests. Ultimately, binary OK/NOK classification remains the primary requirement in industrial settings, and coarse anomaly localization is generally sufficient for this purpose. That makes Model-170-B sufficient for deployment.

In retrospect, the overall complexity of the task may have increased unnecessarily. The use of multiple zoom levels and varied orientations increased the diversity of the dataset, but also made the learning task more difficult, particularly for the baseline model. A more practical and robust solution might involve placing the pre-filter directly into a fixture, rotating it only slightly to either side. This would standardize the part's appearance while still capturing enough variability for generalization.

In summary, the main goals of the thesis were fully achieved. Although the implemented system is not entirely production ready, the solution demonstrates strong potential for unsupervised anomaly detection under real-world conditions. Furthermore, the proposed approach is not limited to the lightbox setup, it could be adapted to operate directly within the production line. Although lighting conditions would be less uniform and controlled in such a scenario.

BIBLIOGRAPHY

- [1] HAVLE, O. Strojové vidění II: Úlohy, nástroje a algoritmy. *Automa*, 2008, vol. 27, no. 2, p. 54–56. Available at: https://automa.cz/Aton/FileRepository/pdf_articles/36676.pdf.
- [2] *Presence Inspection (Quantity, Missing Parts)*. Available at: <https://www.keyence.com/ss/products/vision/visionbasics/use/inspection01/>.
- [3] ZHANG, R.; WANG, D.; GUO, J. and YUAN, Y. Inspecting spring clamp dimensions with machine vision. *Journal of Physics: Conference Series*, 2019, vol. 1303, no. 1, p. 012040. Available at: <https://doi.org/10.1088/1742-6596/1303/1/012040>.
- [4] *Introduction to Machine Vision*. 2024. Available at: <https://www.cognex.com/resources/white-papers-articles/whitepaperandarticlemain?event=?event=f6c6ef16-20ec-4564-bc74-7c42a9a4900a>.
- [5] *Barcode Readers*. Available at: <https://www.cognex.com/products/barcode-readers>.
- [6] *Data Matrix Codes*. Available at: <https://www.cognex.com/resources/symbologies/2-d-matrix-codes/data-matrix-codes>.
- [7] JOSHI, S. System for Optical Character Recognition Using Intelligent Template Matching. *International Journal of Engineering Research & Technology (IJERT)*, July 2014, vol. 3, no. 7. ISSN: 2278-0181.
- [8] KAE, A. and LEARNED MILLER, E. G. Learning on the Fly: Font-Free Approaches to Difficult OCR Problems. *Selected Works of Erik G Learned-Miller*, 2009. Available at: http://works.bepress.com/erik_learned_miller/59/.
- [9] PODOBNY, Z. and CONTRIBUTORS, T. *Tesseract OCR 4.0.0 Released* <https://github.com/tesseract-ocr/tesseract/releases/tag/4.0.0>. October 2018. Official release announcement and release notes.
- [10] KARATZAS, D.; BIGORDA, L. G. i; NICOLAOU, A.; GHOSH, S. K.; BAGDANOV, A. D. et al. ICDAR 2015 competition on Robust Reading. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015, p. 1156–1160. Available at: <https://api.semanticscholar.org/CorpusID:13322740>.
- [11] MOREH, F.; HASAN, Y.; HUSSAIN, B. Z.; AMMAR, M. and TOMFORDE, S. Deep Learning for Micro-Scale Crack Detection on Imbalanced Datasets Using Key Point Localization. *Journal of Display Technology*, 2024, no. 11, p. 1–12.

- [12] HUANG, Y. and XIANG, Z. RPDNet: Automatic Fabric Defect Detection Based on a Convolutional Neural Network and Repeated Pattern Analysis. *Sensors*, 2022, vol. 22, no. 16. ISSN 1424-8220. Available at: <https://www.mdpi.com/1424-8220/22/16/6226>.
- [13] MURAOKA, T.; UCHIMURA, S. and IKEDA, H. Evaluating Visual Fatigue to Keep Eyes Healthy Within Circadian Change During Continuous Display Operations. *Journal of Display Technology*, 2016, vol. 12, no. 11, p. 1464–1471. ISSN 1558-9323.
- [14] SEE, J. E. Visual inspection reliability for precision manufactured parts. *Human Factors*. SAGE, september 2015, vol. 57, no. 8. ISSN 0018-7208. Available at: <https://www.osti.gov/biblio/1236203>.
- [15] HAVLE, O. Strojové vidění IV: Osvětlovače. *Automa*, 2008, vol. 27, no. 4, p. 47–49. Available at: https://automa.cz/Aton/FileRepository/pdf_articles/36988.pdf.
- [16] REN, Z.; FANG, F.; YAN, N. and WU, Y. State of the Art in Defect Detection Based on Machine Vision. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 2022, vol. 9, no. 2, p. 661–691. ISSN 2198-0810. Available at: <https://doi.org/10.1007/s40684-021-00343-6>.
- [17] KEYENCE Vision Sensor IV series. Available at: <https://www.keyence.eu/products/vision/vision-sensor/iv/>.
- [18] COGNEX Fixed-Mount Barcode Readers. Available at: <https://www.cognex.com/products/barcode-readers/fixed-mount-barcode-readers>.
- [19] KEYENCE Vision Systems. Available at: <https://www.keyence.eu/products/vision/vision-sys/vs/>.
- [20] Intuitive Vision System CV-X series. Available at: <https://www.keyence.eu/products/vision/vision-sys/cv-x100/>.
- [21] COGNEX VisionPro Deep Learning. Available at: <https://www.cognex.com/products/machine-vision/vision-software/visionpro-deep-learning>.
- [22] ZIKMUND, D. *Expanding AI Quality Inspection Options*. Available at: <https://www.pekatvision.com/expanding-ai-quality-inspection-options/>.
- [23] CHAKI, J. and DEY, N. *A Beginner's Guide to Image Pre-processing Techniques*. Taylor Francis Group, june 2018. ISBN 9781138339316.
- [24] SZELISKI, R. *Computer Vision Algorithms and Applications*. 2nd ed. Springer Cham, 2022. ISBN 978-3-030-34374-3.

- [25] STEGER, C. Machine Vision Algorithms. In: *Handbook of Machine Vision*. 2006, p. 511–692. ISBN 9783527610136. Available at: <https://doi.org/10.1002/9783527610136.ch8>.
- [26] JOSHI, M. and VYAS, A. Comparison of Canny edge detector with Sobel and Prewitt edge detector using different image formats. *International Journal of Engineering Research & Technology (IJERT)*, 2014, vol. 2, no. 03. Available at: <https://www.ijert.org/comparison-of-canny-edge-detector-with-sobel-and-prewitt-edge-detector-using-different-image-formats>. ETRASCT – 2014.
- [27] DEEP. *Introduction to SIFT (Scale Invariant Feature Transform)*. Available at: <https://medium.com/@deepanshut041/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40>.
- [28] HASSANIN, A.-A.; EL SAMIE, F. A. and BANBY, G. E. A real-time approach for automatic defect detection from PCBs based on SURF features and morphological operations. *Multimedia Tools and Applications*, 2019/12/01, vol. 78, p. 1–21.
- [29] KAMOONA, A. M.; GOSTAR, A. K.; BAB HADIASHAR, A. and HOSEINNEZHAD, R. Point Pattern Feature-Based Anomaly Detection for Manufacturing Defects, in the Random Finite Set Framework. *IEEE Access*. Institute of Electrical and Electronics Engineers (IEEE), 2021, vol. 9, p. 158672–158681. ISSN 2169-3536. Available at: <http://dx.doi.org/10.1109/ACCESS.2021.3130261>.
- [30] SINGH, P.; RATTAN, M.; GREWAL, N. S. and AGGARWAL, G. Distributed Feature Matching for Robust Object Localization in Robotic Manipulation. *IEEE Access*, 2024, vol. 12, p. 161679–161687. ISSN 2169-3536.
- [31] WASZKEWITZ, P. Machine Vision in Manufacturing. In: *Handbook of Machine Vision*. 2006, p. 693–776. ISBN 9783527610136. Available at: <https://doi.org/10.1002/9783527610136.ch9>.
- [32] MVTEC. *Blob Analysis*. Available at: <https://www.mvtec.com/technologies/blob-analysis>.
- [33] AGGARWAL, C. C. *Neural Networks and Deep Learning*. 2nd editionth ed. Cham: Springer International Publishing, 2023. ISBN 978-3-031-29641-3. Available at: <https://link.springer.com/book/10.1007/978-3-031-29642-0>.
- [34] GOODFELLOW, I.; BENGIO, Y. and COURVILLE, A. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.

- [35] RAMACHANDRAN, P.; ZOPH, B. and LE, Q. V. Searching for Activation Functions. *ArXiv preprint arXiv:1710.05941*, 2017. Available at: <https://arxiv.org/abs/1710.05941>.
- [36] RICHMOND, A. *Loss Functions in Machine Learning Explained*. 2024. Available at: <https://www.datacamp.com/tutorial/loss-function-in-machine-learning>.
- [37] LIU, J.; XIE, G.; WANG, J.; LI, S.; WANG, C. et al. Deep Industrial Image Anomaly Detection: A Survey. *Machine Intelligence Research*. Springer Science and Business Media LLC, january 2024, vol. 21, no. 1, p. 104–135. ISSN 2731-5398. Available at: <http://dx.doi.org/10.1007/s11633-023-1459-z>.
- [38] BERGMANN, P.; LÖWE, S.; FAUSER, M.; SATTLEGGGER, D. and STEGER, C. Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders. In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019. Available at: <http://dx.doi.org/10.5220/0007364503720380>.
- [39] SHRIVASTAVA, A.; RAMESHAN, R. and AGNIHOTRI, S. Latent Space Characterization of Autoencoder Variants, december 2024.
- [40] MUSSTAFA. *Optimizers in Deep Learning*. 2021. Available at: <https://musstafa0804.medium.com/optimizers-in-deep-learning-7bf81fed78a0>.
- [41] HU, J.; SHEN, L.; ALBANIE, S.; SUN, G. and WU, E. Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, vol. 42, no. 8, p. 2011–2023. ISSN 1939-3539.
- [42] WOO, S.; PARK, J.; LEE, J.-Y. and KWEON, I. S. CBAM: Convolutional Block Attention Module. *ArXiv preprint arXiv:1807.06521*, 2018. Available at: <https://arxiv.org/abs/1807.06521>.
- [43] RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015, vol. 115, no. 3, p. 211–252.
- [44] SIMONYAN, K. and ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv preprint arXiv:1409.1556*, 2015. Available at: <https://arxiv.org/abs/1409.1556>.
- [45] HE, K.; ZHANG, X.; REN, S. and SUN, J. Deep Residual Learning for Image Recognition. *ArXiv preprint arXiv:1512.03385*, 2015. Available at: <https://arxiv.org/abs/1512.03385>.

- [46] SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. et al. Going Deeper with Convolutions. *ArXiv preprint arXiv:1409.4842*, 2014. Available at: <https://arxiv.org/abs/1409.4842>.
- [47] TAN, M. and LE, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv preprint arXiv:1905.11946*, 2019, abs/1905.11946. Available at: <https://arxiv.org/abs/1905.11946>.
- [48] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L. et al. Attention Is All You Need. *ArXiv preprint arXiv:1706.03762*, 2023. Available at: <https://arxiv.org/abs/1706.03762>.
- [49] DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ArXiv preprint arXiv:2010.11929*, 2021. Available at: <https://arxiv.org/abs/2010.11929>.
- [50] SHORTEN, C. and KHOSHGOFTAAR, T. M. A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 2019, vol. 6, no. 1, p. 60. ISSN 2196-1115. Available at: <https://doi.org/10.1186/s40537-019-0197-0>.
- [51] KRIZHEVSKY, A.; SUTSKEVER, I. and HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems*. 2012, vol. 25, p. 1097–1105. Available at: https://papers.nips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
- [52] YUN, S.; HAN, D.; OH, S. J.; CHUN, S.; CHOE, J. et al. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. *ArXiv preprint arXiv:1905.04899*, 2019. Available at: <https://arxiv.org/abs/1905.04899>.
- [53] CUBUK, E. D.; ZOPH, B.; MANE, D.; VASUDEVAN, V. and LE, Q. V. AutoAugment: Learning Augmentation Strategies From Data. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019, p. 113–123. Available at: <https://doi.org/10.1109/CVPR.2019.00020>.
- [54] ANWAR, A. *Difference between AutoEncoder (AE) and Variational AutoEncoder (VAE)*. 2021. Available at: <https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/>.
- [55] KULLBACK, S. and LEIBLER, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics*. Institute of Mathematical Statistics, 1951, vol. 22, no. 1, p. 79–86. ISSN 00034851. Available at: <http://www.jstor.org/stable/2236703>.
- [56] BERGMANN, P.; BATZNER, K.; FAUSER, M.; SATTLEGER, D. and STEGER, C. The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for

- Unsupervised Anomaly Detection: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. *International Journal of Computer Vision*, 2021, vol. 129, no. 4, p. 1038–1059. ISSN 1573-1405. Available at: <https://doi.org/10.1007/s11263-020-01400-4>.
- [57] SHI, Y.; YANG, J. and QI, Z. Unsupervised Anomaly Segmentation via Deep Feature Reconstruction. *Neurocomputing*. Elsevier BV, february 2021, vol. 424, p. 9–22. ISSN 0925-2312. Available at: <https://doi.org/10.1016/j.neucom.2020.11.018>.
- [58] SCHLEGL, T.; SEEBÖCK, P.; WALDSTEIN, S. M.; LANGS, G. and SCHMIDT ER-FURTH, U. F-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, 2019, vol. 54, p. 30–44. ISSN 1361-8415. Available at: <https://www.sciencedirect.com/science/article/pii/S1361841518302640>.
- [59] BERGMANN, P.; FAUSER, M.; SATTLEGER, D. and STEGER, C. Uninformed Students: Student-Teacher Anomaly Detection With Discriminative Latent Embeddings. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, p. 4183–4192. Available at: <http://dx.doi.org/10.1109/CVPR42600.2020.00424>.
- [60] ROTH, K.; PEMULA, L.; ZEPEDA, J.; SCHÖLKOPF, B.; BROX, T. et al. Towards Total Recall in Industrial Anomaly Detection. *ArXiv preprint arXiv:2106.08265*, 2022. Available at: <https://arxiv.org/abs/2106.08265>.
- [61] TAO, X.; GONG, X.; ZHANG, X.; YAN, S. and ADAK, C. Deep Learning for Unsupervised Anomaly Localization in Industrial Images: A Survey. *IEEE Transactions on Instrumentation and Measurement*. Institute of Electrical and Electronics Engineers (IEEE), 2022, vol. 71, p. 1–21. ISSN 1557-9662. Available at: <http://dx.doi.org/10.1109/TIM.2022.3196436>.
- [62] *IC Imaging Control 4 Python Library*. Available at: <https://www.theimagingsource.com/en-us/documentation/ic4python/index.html>.
- [63] CORPORATION, C. *Computer Vision Annotation Tool (CVAT)*. CVAT.ai Corporation, 2023. Available at: <https://cvat.ai/>.

SYMBOLS AND ABBREVIATIONS

AE	Autoencoder
Adam	Adaptive Moment Estimation
AdaGrad	Adaptive Gradient Descent
AoV	Angle of View
AUROC	Area Under Receiver Operating Characteristic Curve
CBAM	Convolutional Block Attention Module
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DFR	Deep Feature Reconstruction
FP	False Positives
FN	False Negatives
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IoU	Intersection over Union
IMM	Injection Molding Machine
MAE	Mean Absolute Error
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NOK	Not OK
OCC	One-class Classification
OCR	Optical Character Recognition
OCV	Optical Character Verification
PCA	Principal Component Analysis

PCB	Printed Circuit Board
PLC	Programmable Logic Controller
ReLU	Rectified Linear Unit
RMSProp	Root Mean Square Propagation
ROC	Receiver Operating Characteristic
ROI	Region of Interest
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SSIM	Structural Similarity Index Measure
SURF	Speeded-Up Robust Features
SVDD	Support Vector Data Description
TP	True Positives
TN	True Negatives
VAE	Variational Autoencoder
ViT	Vision Transformer

LIST OF FIGURES

1	Schematic diagram of typical illumination modes: a) Bright field forward lighting, b) Dark field forward lighting, c) Coaxial forward lighting, d) Scattering forward lighting of dome structure, e) Back lighting [16]	17
2	Comparison of reflective surface illumination: ordinary bright field forward lighting (left) and a lighting with dome structure (right) [15]	18
3	Comparison of interpolation methods: (A) Original image, (B) Nearest-neighbor interpolation, (C) Bilinear interpolation, and (D) Bicubic interpolation. [23]	21
4	Hough transform: (a) Original image of capacitor, (b) Image after Canny edge detection, (c) Hough accumulator array, (d) Circle obtained with Hough transform and later thresholding. [25]	23
5	(a) Original image, (b) Image with extracted blobs and calculated center points. [32]	25
6	Deep learning-based solution for final assembly quality inspection with collaborative robot [22]	26
7	Perceptron: a neuron that computes a weighted sum of inputs, adds bias b , and applies an activation function f	28
8	Example of underfitting and overfitting in regression task	32
9	Architecture of a vision transformer [49]	36
10	Training setting of a GAN network [58]	41
11	Architecture of distribution-map based models [37]	43
12	Architecture of Student-teacher model [37]	44
13	Architecture of one-class classification models [37]	45
14	Architecture of memory bank based methods [37]	46
15	Complete pre-filter part	48
16	Pre-filter upper part	49
17	(a) Subtle anomaly on the upper side, (b) Larger anomalies on the upper side	49
18	Experimental setup for image acquisition	50
19	Camera mounted at the top of the lightbox	51
20	Two rectangular LED strips parallel on opposite sides of camera	52
21	Geometric illustration of camera's setup	54
22	Data acquisition workflow	57
23	Zoomed-in images of barely visible scratch: (a) Original image, (b) Enhanced image	60
24	(a) Original image of anomaly, (b) Ground truth binary mask	61
25	Architecture of the baseline autoencoder with attention	64

26	The training and validation losses during the best model training	68
27	(a) Confusion matrix for Dataset-A, (b) Confusion matrix for Dataset-B . . .	70
28	Failure of the pixel-level classification despite great pixel-AUROC on Dataset-A image: Original image (left), Ground truth mask (middle), Predicted binary mask (right)	71
29	Indication of some sort of pixel level awareness: Original image (left), Ground truth mask (middle), Predicted binary mask (right)	71
30	Confusion matrices of all testing configurations: (a) 128-A, (b) 128-B, (c) 170-A, (d) 170-B	72
31	ROC curve of 170-B model	73
32	128-A performance: (a) Original image, (b) Anomaly heatmap, (c) Ground truth mask, (d) Predicted mask	74
33	128-B performance: (a) Original image, (b) Anomaly heatmap, (c) Ground truth mask, (d) Predicted mask	75
34	Final implementation setup; example of a part classified as OK	75
35	Visual output of the final implementation; part classified as NOK	75

LIST OF TABLES

1	Results of geometrical calculations for each focal length chosen	56
2	Summary of gathered images	58
3	Architecture of the feature autoencoder used in final experiments	67
4	Performance of the Baseline Autoencoder with Attention for each dataset .	69
5	Performance of the DFR Autoencoder models across both datasets	73

LIST OF APPENDICES

A	Attachment	89
---	------------------	----

A Attachment

Contents of the attached .zip file

- **Augmentation.py** – file with data augmentation code
- **Baseline_AE_with_attention.py** – first model discussed in the thesis
- **Baseline_wo_attention.py** – autoencoder plain without advanced techniques
- **Feature_AE_final.py** – second model discussed in the thesis
- **Feature_AE_with_attention_and_skips.py** – improved second model (not used)
- **Feature_extraction_and_aggregation.py** – process of extracting features with EfficientNet-B6 backbone and aggregating them together
- **Final_implementation_of_Real-time_inference.py** – implementation of the best model to real time inference
- **PCA_Latent_dimension_estimation.py** – computation of latent space dimension via PCA
- **SSIM_loss_class.py** – SSIM loss function used for first model
- **Test_enhancement.py** – enhancement of testing images