



Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Artificial Intelligence

Beatrix Tugyi

ANOMALY DETECTION FOR UNPACKING MACHINE USING DEEP LEARNING STRATEGIES

SUPERVISORS

Bálint Gyires-Tóth, Phd

Dániel Unyi

Tamás Fischl, Phd

Ellák Somfai, Dsc

BUDAPEST, 2024

Contents

Abstract	5
Kivonat	6
1 Introduction.....	7
2 Theoretical background	10
2.1 Anomaly detection.....	10
2.2 Artificial Intelligence and Deep Learning	12
2.2.1 Dimension reduction techniques.....	16
2.2.2 Clustering methods	18
2.3 Deep learning methods for visual anomaly detection.....	19
3 Proposed methods.....	21
3.1 Integration with an existing system	21
3.2 Dataset.....	22
3.2.1 Data pre-processing	22
3.2.2 Dataset categorizations.....	24
3.2.3 Train-test split.....	25
3.2.4 Data augmentations.....	26
3.2.5 Prepared dataset	26
3.3 Anomaly detection with autoencoders.....	28
3.4 Anomaly detection with clustering.....	31
3.4.1 Feature extractor	32
3.4.2 Scaling methods.....	33
3.4.3 Dimension reduction.....	33
3.4.4 Clustering	34
3.4.5 Filtering	34
3.4.6 Prediction.....	34
3.5 Anomaly detection with Segment Any Anomaly+	35
3.6 Evaluation metrics	37
3.6.1 Subjective metrics.....	37
3.6.2 Objective metrics	39
4 Implementations	40
4.1 PyTorch.....	40

4.2 Scikit -learn	40
4.3 Albumentations.....	40
4.4 Weights & Biases	41
4.5 Label Studio	41
4.6 Docker.....	41
4.7 GPU	41
5 Results.....	42
5.1 Autoencoder approach	42
5.2 Anomaly detection with clustering.....	47
5.3 Anomaly detection with Segment-Any-Anomaly+.....	52
6 Discussion	57
7 Human-Centred Considerations.....	59
7.1 Ethics	59
7.2 Safety	60
7.3 Legal	60
7.4 Security	60
8 Future work.....	61
9 Summary.....	62
References.....	63

STUDENT DECLARATION

I, Beatrix Tugyi, the undersigned, hereby declare that the present MSc thesis work has been prepared by myself and without any unauthorized help or assistance. Only the specified sources (references, tools, etc.) were used. All parts taken from other sources word by word, or after rephrasing but with identical meaning, were unambiguously identified with explicit reference to the sources utilized.

I authorize the Faculty of Electrical Engineering and Informatics of the Budapest University of Technology and Economics to publish the principal data of the thesis work (author's name, title, abstracts in English and in a second language, year of preparation, supervisor's name, etc.) in a searchable, public, electronic and online database and to publish the full text of the thesis work on the internal network of the university (this may include access by authenticated outside users). I declare that the submitted hardcopy of the thesis work and its electronic version are identical.

Full text of thesis works classified upon the decision of the Dean will be published after a period of three years.

Budapest, 6 December 2024

.....
Beatrix Tugyi

Abstract

Industrial automation is expanding and evolving rapidly, with a huge technological boost from Artificial Intelligence (AI), especially Deep Learning (DL). The accelerating development of AI technologies has revolutionized information processing and decision-making processes, which are the main pillars of automation methods. Industrial automation encompasses a wide range of different methods that can be applied to a manufacturing process to execute production steps without human intervention. When these methods are applied in real life, it is crucial that such systems detect in time if they are facing an input that is not supposed to be handled because it is faulty, damaged, wrong, or unknown. One of the most widespread methods to solve this problem is image-based anomaly detection, through which these anomalous cases can be identified.

This research is focused on an industrial company's factory automation process. It aims to integrate a deep learning-based anomaly detection control step into this workflow. This leads to one of the specialities and challenges of the topic, which is that the implementation has to be adapted to the demands of a real-life scenario. The necessary images for training the models were provided by the company, thereby fully adapting the input to a real factory environment. An extensive literature review was conducted on the most common and high-performing anomaly detection techniques, exploring the capabilities of each method. Based on this, three directions were identified, for which different methods have been studied and implemented, such as solutions based on autoencoders or clustering. Both qualitative and quantitative methods were used to evaluate the performance of these methods and their suitability for the given use case. Their accuracy and reliability were further improved through various optimizations and improvements. Besides the construction of well-performing models, the analysis of the results with adequate accuracy was an emphasized part of the research. General conclusions were drawn on the necessary considerations to be taken into account in real-world applications. I have identified the limits of the methods' abilities and the possible preconditions for their successful use. Furthermore, industrial automation raises several concerns that fall within the scope of human-centred artificial intelligence, including legal, ethical, and safety aspects, which will be further discussed.

Kivonat

Az ipari automatizálás terjedésére és fejlesztésére óriási hatással van a mesterséges intelligencia (MI), azon belül is a mély tanulás. Ezen technológiák jelenleg is tartó, robbanásszerű fejlődése forradalmasította az információfeldolgozási és döntéshozatali folyamatokat, amelyek alappilléreit alkotják az automatizációs eljárásoknak. Az ipari automatizáció számos különböző módszert foglal magába, amelyek lehetővé teszik, hogy a gyártási folyamat során az egyes lépések emberi beavatkozás nélkül végrehajthatóak legyenek. Ezen módszerek valós életben való felhasználása során kiemelten fontos, hogy időben észleljük, ha a rendszer olyan bemenettel áll szemben, amit nem szabad kezelni, mert hibás, sérült, rossz vagy esetleg még nem ismert. Ennek megvalósítására a képeken alapuló anomália detekció egy elterjedt módszer, melynek segítségével azonosítani lehet ezeket az eseteket.

Kutatásom egy nagyipari vállalat gyári automatizációs folyamatához kapcsolódik. Célja egy mély tanuláson alapuló anomália detekciós ellenőrzési lépés integrálása ebbe az eljárásba. Ebből adódik a téma egyik sajátossága és egyben kihívása, hogy a megvalósítás során a valós életben való alkalmazás igényeihez kell igazodni. A modellek tanításához szükséges képeket a vállalat biztosította, teljes mértékben adaptálva ezzel a bemenetet egy valós gyári környezethez. Az anomália detekciós technikákról széleskörű irodalomkutatás készült, mely során az egyes módszerek sajátosságai elemezve lettek. Ez alapján három irányvonal került azonosításra, melyekhez különböző módszerek lettek tanulmányozva és implementálva, mint például az autoenkóderen vagy a klaszterezésen alapuló megoldások. Kvalitatív és kvantitatív módszerekkel is kiértékelésre került a módszerek teljesítménye, valamint, hogy az adott felhasználási célra mennyire felelnek meg. Különböző optimalizálások és továbbfejlesztések által tovább növelve lett a pontosságuk és a megbízhatóságuk. A jól teljesítő modellek létrehozása mellett hangsúlyos része volt a kutatásnak az eredmények megfelelő pontossággal történő kielemezése is. Általános érvényű következtetéseket vontam le arról, hogy milyen elveket kell figyelembe venni a valós felhasználás során. Meghatároztam, hogy mik a módszerek képességeinek határai, valamint milyen esetleges előfeltételei vannak a sikeres használatuknak. Mindemellett az ipari automatizáció több olyan kérdést is felvet, amely az emberközpontú MI kérdései alá tartoznak, ideértve felhasználásának jogi, etikai és biztonsági kérdéseit, melyek további elemzésre kerülnek.

1 Introduction

This chapter presents the motivation and purpose of the research, along with the use case in which it will be deployed, contributing to the realization of an industrial automation project.

There is an increasing focus on workflow automation nowadays, including the domain of various industrial processes [1]. The seamless operation of machinery stands as a pivotal factor for efficiency, precision, and product quality. Moreover, by automatization, employees can be freed from procedures that are exhausting, tedious, dangerous, or difficult to carry out. These methods represent a major innovation and could provide a huge benefit to a company and its employees. Most industrial automation realizations require some kind of computer vision tool to map the working area and identify what kinds of components are available. These are necessary for the system to get information about its surroundings and to be able to determine, what should be the next step. Since these are not closed systems in which we can explicitly predict what will be in front of the cameras, it is important to have some sort of mechanism that is prepared for unusual cases as well. Especially, when it is used in a hazardous environment. For example, when the machine is transporting or cutting things, it is essential that it can identify if a human is in the way, or if a hand or some clothes are hanging inside the work area, which could lead to accidents. In such situations, these machines must not continue to operate as normal, but give some kind of signal, as the resolution of the action requires human intervention. A powerful solution for identifying these cases is image-based anomaly detection. With the rise of deep learning, image processing techniques improved tremendously, involving anomaly detection. This research aim is to investigate different deep learning-based anomaly detection methods, that can be used in the given industrial environment.

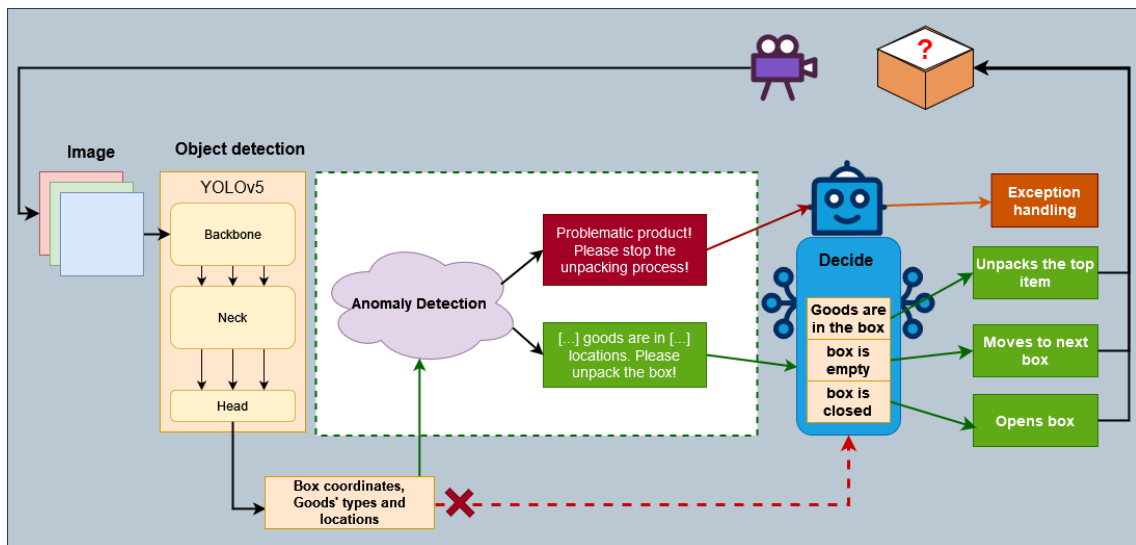
The Unpacking Machine is the name of a machine in the concept phase, which is a current industrial initiative aiming to unpack the received boxes on a conveyor belt, separate their contents, and sort the leftover packaging materials selectively. It has advanced capabilities, it can move, lift, and sort different items, and has a cutting blade to cut the boxes open at their localized coordinates. With object detection techniques, it can observe what is in front of its camera, locate each product, and then decide what to

do next. Figure 1.1. shows a case study about a rudimentary version of the Unpacking Machine, while it sorts goods based on its camera view.



1.1. Figure - Images of a rudimentary version of the Unpacking Machine in work. It sorts the products based on its camera view, which is illustrated on the upper left side of the images.

In the given use case, the AI model should recognize goods (e.g., PCBs (Printed Circuit Boards), Reels), stacked boxes, plastic packages, and paper packages, due to recycling. The object detection technique is suitable for this purpose, but unfortunately not enough. In a real environment, the machine also has to be prepared for the edge cases when some unexpected input or anomaly is encountered. Object detection is unable to identify these because these are previously unknown incidents, and not enough data is available of them. The aim of the proposed anomaly detection algorithms is to be incorporated into this system to detect these cases. The full automation process is shown in Figure 1.2, with the novel anomaly detection-based verification step highlighted with a white background.



1.2. Figure - The automatization workflow, updated with the novel anomaly detection step, which is highlighted with a white background. It replaces the step shown with a red dashed arrow.

As the illustration shows, the automation process runs in a loop and only stops when an anomaly is encountered, and an exception-handling process is needed.

Establishing a suitable dataset is critical for training a deep learning algorithm, which requires many images. For this research, the company provided images from one of their factories. After carefully planned pre-processing steps, they were used to train and test the proposed methods. The investigated techniques can be used as part of any other automation project if a sufficient amount of images is available of the environment. This use case is only one example of the numerous possibilities where these solutions may be appropriate.

The literature on anomaly detection with deep learning methods is quite broad. The main principles of these methods are presented in the second chapter. The basics of artificial intelligence, deep learning, and the main components that were used for anomaly detection are also introduced.

In the third chapter, the used dataset and the various pre-processing steps are presented. Then the research is pursued in three main directions. First comes an approach leveraging the popular, well-established method of autoencoders. Next, another method is examined, that seeks to identify outliers through clustering the latent space of an AI-based object detection model. Finally, comes a modern approach to detect anomalies using zero-shot detection, called Segment Any Anomaly+. The different characteristics of these methods are described along with the engineering choices made to design them, and the modifications made to customize them. The used evaluation methods are also detailed here.

The fourth chapter provides implementation details, together with a description of the used libraries. The fifth chapter presents the results of the evaluation of all the models, highlighting their advantages, disadvantages and limitations. In the sixth chapter, the three methods are compared, and it is determined which one of them would be the most suitable to use for the specific application. Finally, in chapter seven, the ethical, legal, and safety-critical issues of the proposed methods are discussed.

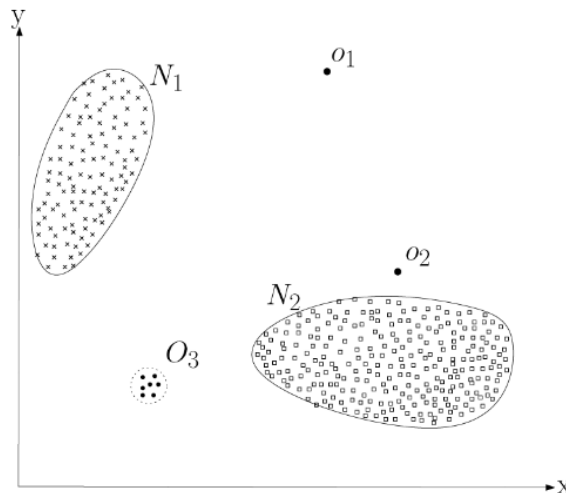
2 Theoretical background

In this chapter, the theoretical background of anomaly detection, artificial intelligence, and the fundamentals of the applied deep learning methods are presented.

2.1 Anomaly detection

Anomaly detection has established itself as an important method, with many applications and presence in almost all scientific domains. Its significance arises from its capability to discover critical incidents that could cause serious damage with time. It can be used for fraud detection, identifying cybersecurity threats or terrorist activity, forecasting possible machine failures, diagnosing medical disorders, sanitizing data, or verifying the proper functioning of workflows. The concept of anomalies and their identification is not new, mathematicians have been interested in the statistical approach to this problem for quite a long time. As early as the second half of the 19th century, there are records of this problem in a paper by the famous mathematician Edgeworth [2].

Anomalies are patterns in data that do not conform to a well-defined notion of normal behaviour [3]. A simple example of them in two dimensions is visualized in Figure 2.1.



2.1. Figure - An example of anomalies in two dimensions. The data has two normal regions N_1 and N_2 , points that are sufficiently far from these are considered as anomalies (O_1, O_2, O_3). Source: [3].

Anomalies can arise in various ways in our life. They can be the consequence of deliberate attacks, they can result from unintentional mistakes, or they can just happen by the laws of nature. The concept of anomaly also appears in many other areas, with the more general

meaning of unexpected, unusual patterns that are different from normal. They are also known as outliers, discordant observations, or exceptions. Originally, anomalies' characteristics are easy to confuse with novel ones, as they have also never occurred before. The difference between the two is that the new patterns occur repeatedly over time and will become incorporated into the existing distribution.

The discipline that studies these anomalies is called anomaly detection, with the aim of separating outlier data points from the regular data. In anomaly detection, the goal is the classification of normal and abnormal data, yet it cannot be considered as a simpler binary classification problem. Unlike a binary classification, here the distribution of data is unbalanced, and anomalies are mainly previously not seen data, so they are not available for training and cannot be labelled. In addition, anomalies can vary in appearance, shape and colour and they do not have stable statistical characteristics [4], that would define them. Anomaly detection can be used on a huge variety of data types, including binary files, texts, videos and images. In this research, the focus is on image-based anomaly detection methods. There are many traditional, so-called “pre-deep learning” anomaly detection methods, such as Principal Component Analysis [5], one-class Support Vector Machine [6], Local Binary Patterns [7], and Gaussian Mixture Models [8]. However, deep learning-based methods are more robust on complex, high-dimensional data, like images, due to their ability to automatically learn and extract hierarchical features, whereas traditional methods struggle due to the reliance on handcrafted features [9]. In this research, only deep learning-based solutions are investigated since the image dataset is quite complex so the traditional methods would not be feasible.

A straightforward approach would be to define a region that is representative of normal behaviour and to declare all other observations as anomalies. However, there are confounding factors that make it difficult in practice [10], such as:

- Labelled anomalous data is often not available for training and testing.
- Normal noisy data can be similar to anomalies, so they are hard to differentiate.
- Anomalies can occur in a wide variety of forms; they cannot be defined, and it is not possible to know their main features.
- To detect an anomaly, a deep learning-based neural network must learn the distribution of every possible form of the normal data very accurately, for which they usually require very huge datasets.

- It is difficult to define the region of the normal data, that encompasses every possible normal behaviour as there is no clear boundary between the normal and anomalous samples. If we define an exact threshold for anomalies, it is very common to have normal data detected as an anomaly and vice versa.
- The exact definitions of anomaly vary from application to application. For example, in the medical field, a small deviation from the norm (e.g., a fluctuation in body temperature) should be considered an anomaly, while a similar deviation in the world of the stock market (e.g., a fluctuation in the value of a share) should be considered normal. Thus, reusing a technique developed in one area to another is not straightforward.

These challenges confirm that anomaly detection is not easy to address. Most existing techniques only solve a specific formulation of the problem, which is influenced by factors such as the nature of the available data, the type of anomalies to be detected, and the application domain.

2.2 Artificial Intelligence and Deep Learning

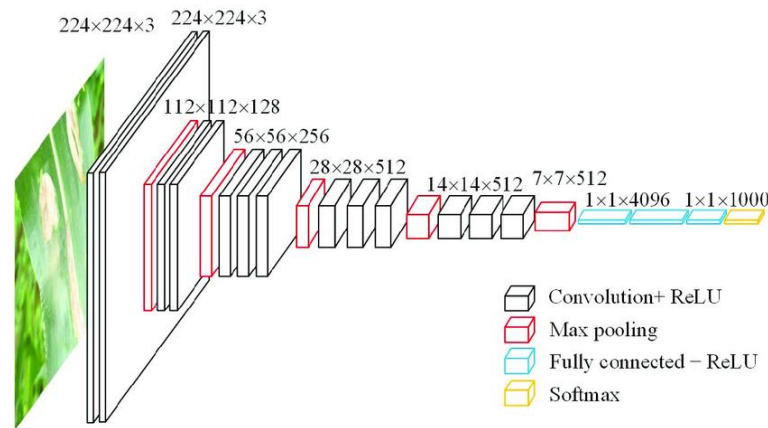
Artificial Intelligence (AI) covers a broad area, including Machine Learning (ML), a part of which is Deep Learning (DL). Deep learning uses complex structures called deep neural networks, which have many layers to process information. They usually work with a large amount of data and excel in tasks such as image recognition, natural language processing, and data generation.

The very first neural network model became known as the Perceptron in 1957, invented by Frank Rosenblatt [11]. It was inspired by earlier theoretical works, which laid the foundation for computation methods modelled after biological neural processes. In those earlier years, many ideas and concepts had been proposed about what AI could achieve, but most of them could not be realized because of technological limitations, such as a lack of computational power, and theoretical challenges. One major limitation of single-layer Perceptrons was their inability to solve non-linearly separable problems, like the XOR problem. These have led to the neglect of this field, called AI Winter. In the following years, AI research experienced cycles of optimism and setbacks, where a discovery or research result only kept the interest alive in the subject for a shorter amount of time. The development of backpropagation in the 1980s revived interest in multi-layered neural networks, enabling them to learn more complex patterns. However, progress remained constrained by limited computational power and available data. In

2012, the real breakthrough came with the emergence of deep learning methods, especially convolutional neural networks for image recognition [12]. This has brought undeniable successes, that fundamentally changed the public view of machine learning methods, triggering a revolution in the sector. To this day AI technologies have proved their worth, conquering most fields of science. We can expect more reliable, more accurate, and faster solutions, thanks to the continuous development of models, software tools, hardware, and the increasing amount of data.

Artificial Intelligence solutions can be divided into three main categories, depending on how the training is performed: supervised, unsupervised, and semi-supervised learning. **Supervised learning** is the most frequently used, where models are trained on data with predefined labels, which serve as the target class or value. They aim to map the relationships between data points and labels during training so that afterwards, the model can predict the target for new, unlabelled data. While it is straightforward and effective, its main drawback is the cost of labelling large datasets. In **unsupervised learning** the model learns from a dataset without any predefined labels, with the aim of discovering patterns, structures, or hidden relationships. The model identifies these on its own, which can be valuable for exploratory data analysis or understanding complex datasets. However, due to the lack of clear ground truth answers, it can be more challenging to evaluate their performance, and they may require more sophisticated techniques to ensure meaningful results. **Semi-supervised learning** is a hybrid approach, that leverages both labelled and unlabelled data, as it combines the advantages of supervised and unsupervised learning. Usually, the model is initially trained on the unlabelled dataset and then further fine-tuned using the labelled data, which helps improve its generalization abilities. Hence the performance can significantly improve, while the dependency on fully labelled datasets is reduced, offering a more practical solution in real-world applications. In recent years the emergence of foundation models has introduced a new trend alongside these methods, known as **prompt-based learning**. This approach leverages pre-trained foundation models, which are large-scale models that can subsequently be adopted to solve multiple different tasks e.g., GPT-4 [13], Segment Anything (SAM) [36]. They finetune these large, existing models for specific tasks using prompt engineering, rather than creating new ones from scratch. This method can also be referred to as zero-, one-, or few-shot learning, which aims to reduce the need for large training sets, using only a few or only one data sample for prediction.

Convolutional Neural Networks (CNNs) operate on data, whose representation has a grid-like structure in space or time, like images, videos, and audio files. Their most popular application is image processing, which includes image recognition, object detection, image segmentation, and image generation. They are much more effective on these than traditional computer vision methods or the fully connected neural network approaches. This is because CNNs take advantage of the translation invariance of images. Its convolutional layers contain a kernel, which is moved across the entire image in a sliding window approach, extracting the key information. The underlying concept is that the pixels that belong together have a high probability of being spatially close to each other. The trainable weights correspond only to the size and number of kernels, so it has far fewer parameters. It is essential that the same weights are used in all parts of an image so that if a feature is considered important by the network in one part of the image, it should be equally important elsewhere. Another relevant layer is the pooling layer, which reduces the spatial resolution to extract global features. As an example, the architecture of a popular CNN model can be seen in Figure 2.2.



2.2. Figure -The architecture of a commonly used CNN, called VGG16. Source:[28].

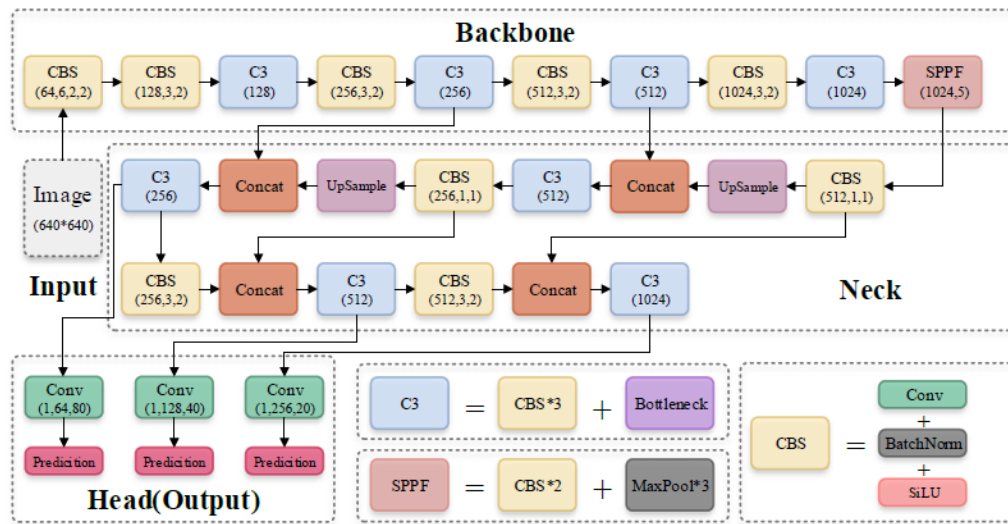
CNN's structure has two main parts, which are easily locatable in Figure 2.2. The first, consists of many convolution layers separated by activation functions, named ReLU (Rectified Linear Unit) and pooling layers, to identify the most important features. ReLU is a commonly used non-linear activation function that leaves the positive parts of its argument unchanged and replaces the negative parts with 0. The second part consists of fully connected layers, which serialize the results. At the end of the model, there is an activation function to determine the predicted output. For multi-class classification tasks, it is usually a Softmax function, which transforms a vector of K integers into a probability distribution with K possible outcomes. It determines the probability of being represented

in the image for each possible object. In the few years that followed the success of the first CNN network [12], many novel and versatile architectures were created, achieving better and better results. One popular benchmark is the ImageNet competition[14], which aims to find an algorithm that could correctly classify diverse content images into 1000 categories. In this work, several different CNNs were used, such as VGG16, ResNET18, ResNET50, and DenseNET [15], as part of more complex deep learning-based architectures. VGG16 (Visual Geometry Group) is a simple, yet deep CNN known for its consistent architecture of 16 layers, using small 3x3 filters to achieve depth and representational capacity, but it has high computational costs due to its 140 million parameters. ResNet (Residual Network) introduces residual connections to combat the vanishing gradient problem, enabling the training of very deep networks efficiently while maintaining lower computational complexity compared to earlier models like VGG. DenseNet enhances information flow by connecting each layer to all subsequent layers via feature concatenation, reducing redundancy but increasing computational demands due to its complex layer interconnectivity.

Transformer networks were introduced in 2017 [16], and represented a breakthrough in the field of Artificial Intelligence, especially in natural language processing. Their main strength is the so-called attention mechanism, which allows the network to efficiently focus on relevant parts of the input data. They also use positional encoding, which helps to maintain and manage the sequence of input data. Transformers differ significantly from previous models such as recurrent neural networks and CNNs. They are built up of layers that can operate in parallel, including encoder and decoder modules. This parallelization allows for significant speed-up and simplicity. Its modular design enables the easy integration of new functions and components, allowing for further development and specialization of the models. In recent years, transformer networks have rapidly taken the lead in AI, because of their versatility and their continuous development has kept them at the forefront in many fields. These models form the basis of several state-of-the-art foundational models.

Object detection aims to determine the exact location of objects in the images, via bounding box coordinates. Two popular implementations are YOLO (You Only Look Once [17]) and SSD (Single Shot MultiBox Detector [18]). The choice between them depends on the specific use case. In real-time scenarios, usually YOLO is used because its detection time is faster, and it has a simpler architecture. In this approach, all objects

in the image are detected and classified at once, as hinted by the name of the architecture: “You Only Look Once”. This way it significantly reduces the required time to perform the task. It works by dividing the image into several grid elements and assigning several predefined bounding boxes for each, which are used to make estimates of what objects are located there. The network consists of three main parts, named Backbone, Neck, and Head. The Backbone is a CNN, which aims to learn and extract key features from the input image. The Neck is responsible for aggregating and enhancing these features across different scales, it connects the Backbone and Head through three different resolution levels, allowing the model to detect objects of various sizes. The Head makes the actual predictions. Figure 2.3 shows the architecture of the YOLOv5 model, that was used for this research.



2.3. Figure - The architecture of the YOLOv5 object detection model. Source: [19]

The YOLO model uses a complex loss function, which consists of several components: the bounding box localization error, the classification error, and their confidence error and it penalizes detections of non-existing objects. The required ratio of these parts in the loss function depends on the specific application, so usually extra attention should be given to tuning the parameters that control the weights of the different losses.

2.2.1 Dimension reduction techniques

Dimension reduction aims to reduce the number of features or variables in a dataset while preserving as much important information as possible. In high-dimensional spaces, points tend to become uniformly distant from each other, making it difficult for

distance-based algorithms to find meaningful structures. This causes the "curse of dimensionality" [20], which describes challenges, such as increased data sparsity, reduced effectiveness of distance metrics, and computational complexity. By reducing the dimensionality, these methods simplify the data, making it easier to visualize, interpret, and analyse, while also improving the efficiency and performance of machine learning models.

Principal Component Analysis (PCA) [5] is a standard statistical method that aims to reduce the dimension of the data, while preserving the most meaningful variables that are likely to contain the most relevant information, to re-express the dataset. PCA assumes that the task is linear and tries to solve it as a linear problem. This way the whole problem is simplified to finding a suitable change of basis. This linear approach allows for a simpler, yet informative representation of the data structures, that often underlie it. It is particularly useful in cases where the dataset is redundant, contains a large amount of noise, or has simply too many dimensions, that need to be reduced to make it manageable and visualizable. The principal components that form the new basis of the data are orthogonal to each other. This orthogonality is achieved by choosing the eigenvectors, which are perpendicular to each other on a symmetric matrix. PCA is closely related to Singular Value Decomposition (SVD), in fact, SVD is a more general method of understanding change of basis. It is a matrix decomposition method, which decomposes a matrix into two orthogonal (U , V) and one diagonal (Σ) matrices. It can be used to directly compute the eigenvalues in PCA.

Uniform Manifold Approximation and Projection (UMAP) is a nonlinear dimensionality reduction technique [21], which is particularly effective for visualizing high-dimensional data. It models data as a geometric manifold and seeks to preserve both the global structure and local relationships between points when projecting them into a lower dimension. It achieves this by building a weighted graph representation of the data's nearest neighbours in high-dimensional space and optimizing its layout in the lower-dimensional space. It is computationally efficient, scalable to large datasets and often yields more meaningful embeddings than other methods like t-SNE (t-distributed Stochastic Neighbour Embedding) or PCA.

Dimensionality reduction is one popular task of **autoencoders**, but their utility extends beyond this, encompassing a variety of forms such as sparse autoencoders, denoising autoencoders, and more [22]. The primary objective is centred around learning

meaningful representations of the data. An autoencoder with a single layer along with a linear activation function is nearly equivalent to PCA. However, the potential power of autoencoders is much larger, as it enables both linear and nonlinear transformations [23]. Its purpose is to learn the identity function: during the training process, it compresses the data into a small latent space and then reconstructs the original input. This way, it can discover a more efficient and compact representation of the data. It has two main parts, called Encoder and Decoder.

2.2.2 Clustering methods

The purpose of clustering is to divide data into categories so that similar data are grouped into one and different data into another cluster. It aims to find natural patterns or structures within a dataset. It helps uncover hidden patterns, simplify data complexity, and can provide insights for decision-making by revealing how data points relate to one another. There are three main approaches, Partition-Based Clustering, Hierarchical Clustering, and Density-Based Clustering. In this research, K-means was used for categorizing the dataset by product types and DBSCAN and HDBSCAN for anomaly detection purposes.

One of the most popular partition-based clustering algorithms is called **K-Means** [24]. It aims to divide M points in N dimensions into K clusters so that inside each cluster the square of the sum of the distances between the data points is minimized. The number of clusters is defined by the K parameter. The K-means algorithm assigns each data to the cluster, which centroid is the closest to it and tries to find the best centroids by alternating between two steps. First, it assigns data points to the clusters, based on fixed centroids, then it chooses new centroids based on the current assignment of data points to clusters.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a powerful density-based clustering algorithm [25]. Unlike traditional clustering methods, it does not require the number of clusters to be defined and can identify clusters of arbitrary shapes while effectively handling outliers. It clusters the data by identifying dense regions of points, expanding clusters from core points within a specified distance (ϵ) while marking sparsely populated points as outliers. This behaviour can be tuned with parameters, for example, the maximum distance between two samples, to be considered in the same cluster, and the number of samples in a neighbourhood for a point to be considered as a core point can be specified. By focusing on density rather than distance,

DBSCAN is particularly effective for datasets with irregularly shaped clusters, making it useful for applications like anomaly detection. Furthermore, the algorithm inherently identifies outliers as part of its clustering process, without requiring a separate method. **Hierarchical DBSCAN (HDBSCAN)** [26] extends DBSCAN by converting it into a hierarchical clustering algorithm, through the construction of a dendrogram, which represents the arrangement of clusters formed at different data density levels. In this hierarchy, each point is connected to its nearest neighbour of higher density, creating a tree of points that gradually merges into denser clusters as one moves up the hierarchy. Then it uses a technique called Condensed Tree to extract a flat clustering by selecting the most stable clusters from this dendrogram across different density levels. It allows variable-density clusters without needing to manually set the ϵ parameters, making it a more flexible and robust version of DBSCAN. It is particularly effective for imbalanced datasets with varying density, noise, or irregular shapes, as it can identify clusters of different densities.

2.3 Deep learning methods for visual anomaly detection

Visual anomaly detection can be divided into two different categories: **Image-level** and **Pixel-level** anomaly detection [4]. The first focuses on separating the images, which are considered as anomalies from the normal ones. The second approach is more sophisticated, it locates the abnormal regions inside the images, identifying which pixels can be considered anomalous. Another categorization is defined based on the used deep learning technique: **supervised**, **semi-supervised**, and **unsupervised** deep anomaly detection [23]. **Supervised** anomaly detection uses labels for both normal and anomalous data. It involves training a multi-class classifier on the labeled data, to distinguish anomalies. Their performance is usually suboptimal, because of the class imbalance and the high in-class variability, and they cannot be used for unknown or new types of anomalies. Thus, these methods are usually not applicable, because of the lack of availability of labelled training samples. On the other hand, **Semi-supervised** methods, which leverage existing labels of the normal, positive inputs are widely adopted. They can be realized using a deep learning-based vision model, which is trained on labelled data, without anomalies. It should learn a discriminative boundary around the normal instances, and the ones, which do not belong to the majority are considered as outliers. Then an unsupervised technique, such as clustering, can be applied to distinguish between normal and anomalous images. It assumes, that data points with the same label, are close

to each other both in input space and learned feature space. The use of labelled data can cause performance improvement over unsupervised techniques. The disadvantage of this method is that the hierarchical features extracted within hidden layers may not be representative enough to distinguish every kind of anomalous instance. **Unsupervised** methods are only based on the intrinsic properties of the data instances to detect outliers. They operate under the assumption that normal data instances are much more common than abnormal ones, so with sufficient training samples these models should produce lower reconstruction errors for them. If this assumption is incorrect, it can lead to a high rate of false positive predictions. It is cost-efficient because labelling is not required. On the other hand, it can be challenging to learn commonalities within data in a high-dimensional space. Thus, they are also sensitive to noise and need much tuning to achieve accurate results. The output of vision-based anomaly detection models is usually a single value, called anomaly score. It quantifies the level of outlierness for each data. The data instances may be ranked according to these scores, and a domain-specific threshold must be selected, which determines the value above which an individual sample is considered an anomaly.

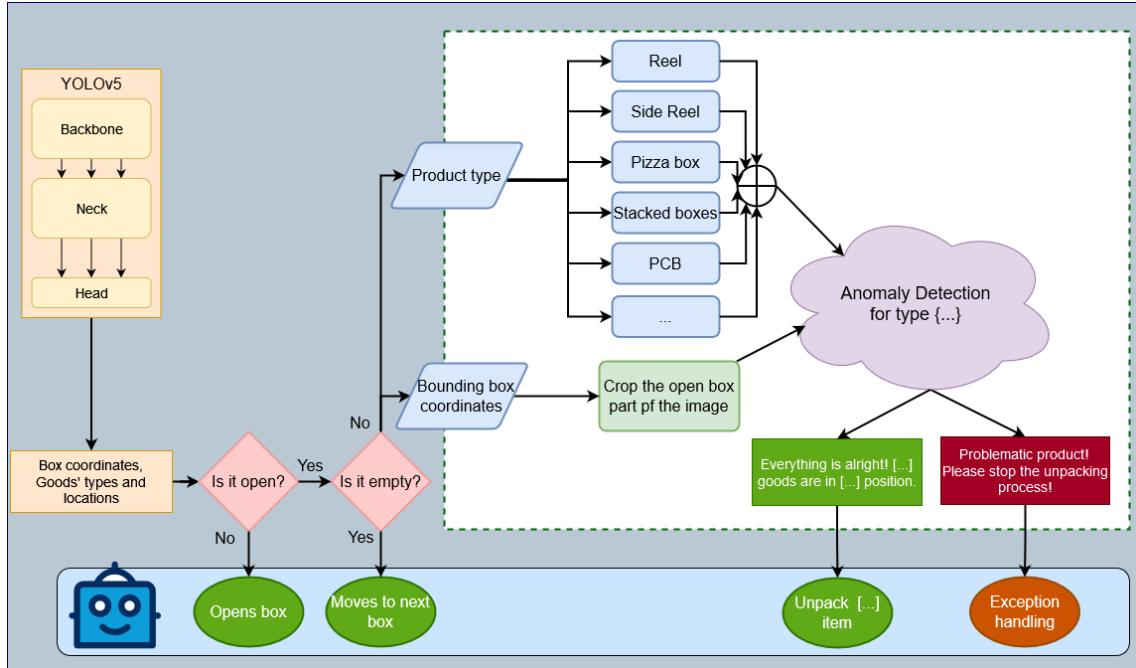
3 Proposed methods

In this chapter, the main parts of the workflow, where the novel anomaly detection method will be integrated are presented, along with the steps and considerations of data preparation, and the details of the different methods.

3.1 Integration with an existing system

The proposed technique is going to be integrated into an existing industrial automatization system. The first building block of this is the Unpacking Machine, which has various remarkable capabilities and is equipped with several arms, allowing it to unwrap, cut, and lift items. It is crucial, for this machine to know what is in front of it before it can perform such a precise operation. To address this challenge, a team has been working on building and optimizing an object detection model that will provide its image recognition capabilities. This will probably be a YOLO model, but any other method would be suitable, which can determine with high accuracy where a particular product is located, via rotated bounding box coordinates. In a simulated environment where all the prepared boxes contain exactly what and how we expect it, this might be enough to complete the task. But in a real-life industrial scenario, the risk also must be assumed that the received products may include defective, damaged, or differently packaged items (i.e., not allowed package configuration of the supplier), that should be handled somehow differently. Relying only on the YOLO, these items would be categorized into one of the predefined classes and the machine would try to handle them in the same way as any other item from that class. This could lead to mishandled, damaged, or ruined products and even cause the whole system to shut down. To prevent such errors, the proposed anomaly detection step will be integrated into the workflow after the YOLO's detection, but before the machine starts the movement. This will provide a safety enhancement to prevent accidents by detecting faulty products. Given this, it can be assumed, that the output information of the YOLO (the types and locations of the products) is available and can be used as prior knowledge for anomaly detection. The target objects for unpacking are always inside a box, so the anomaly detection step should also focus exclusively on the inside of the boxes. This is achieved by cropping the parts of the original image containing the open boxes and using only those as input for anomaly detection. The anomaly

detection is only needed if the box is open and not empty. These additional steps before the anomaly detection in the workflow are visualized in Figure 3.1.



3.1. Figure - The detailed steps before and after the anomaly detection in the final workflow.

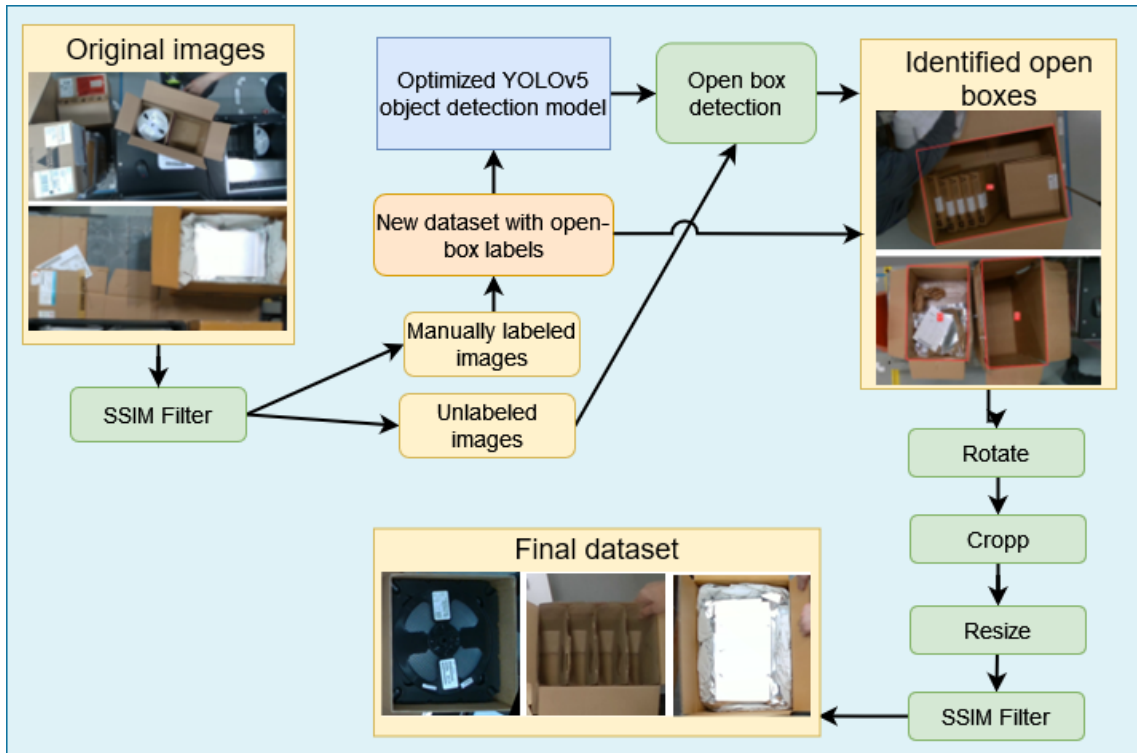
3.2 Dataset

The methods are designed to be suitable for use in any manufacturing process. However, to develop a deep learning-based solution, images of a specific location are needed. For this purpose, an image dataset was collected in one of the company's factories. This contains a series of images about the process of manual box placement, opening, and unpacking. The two main products that can be found in the boxes are reels and PCBs. In addition to these, there are various kinds of paper or plastic packaging materials, bubble wraps, strips, and labels on the boxes. These high-resolution pictures show a large surface area. In the final process, an object detection model will define the location of the open boxes, and they will be cropped. Therefore, this dataset should also contain only the parts of open boxes.

3.2.1 Data pre-processing

The images were captured at fixed, dense intervals, without taking into account movements on the surface. Therefore, many identical images were retrieved, that needed to be eliminated. Structural Similarity Index Measure (SSIM [27]) was used to identify the images that were too similar to each other. The next step was to locate the open boxes,

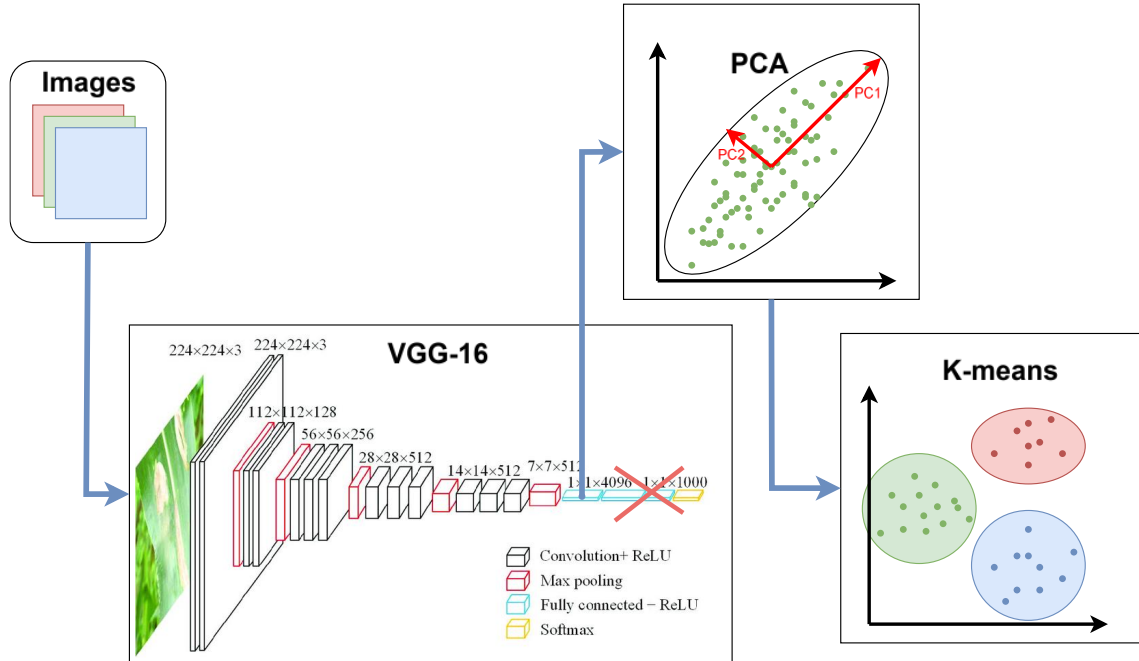
but the object detection model that would be integrated into the workflow was not available at the time of this research. Therefore, another model is needed, to predict these coordinates. A subset of the data was manually labelled before, with multiple different types of labels including 7000 open box labels. A YOLOv5 model was trained on these with transfer learning, to locate the open boxes' rotated bounding box coordinates. Since now false positive detections are of more concern than possible missed ones, the confidence limit was set higher, than the default. In addition, low-confidence results are mostly found in low-quality, blurred images, so omitting them also improves the quality of the dataset. After these coordinates were obtained, the original images were rotated to make their borders parallel to the sides of the boxes. If the bounding box coordinates were out of the image, those parts were replaced with black pixels. Since neural networks generally expect fixed-sized inputs, the images were padded with black pixels, to make them as high as they are wide. Then each box on every image was cropped. An SSIM filter was used once more since it is still possible that there are identical boxes if the differences were only in the trimmed part. Separate functions were implemented for each of these steps, which were connected to fully automate the data preparation steps. The detailed steps of the data pre-processing pipeline are visualized in Figure 3.2.



3.2. Figure – The steps of the data pre-processing pipeline.

3.2.2 Dataset categorizations

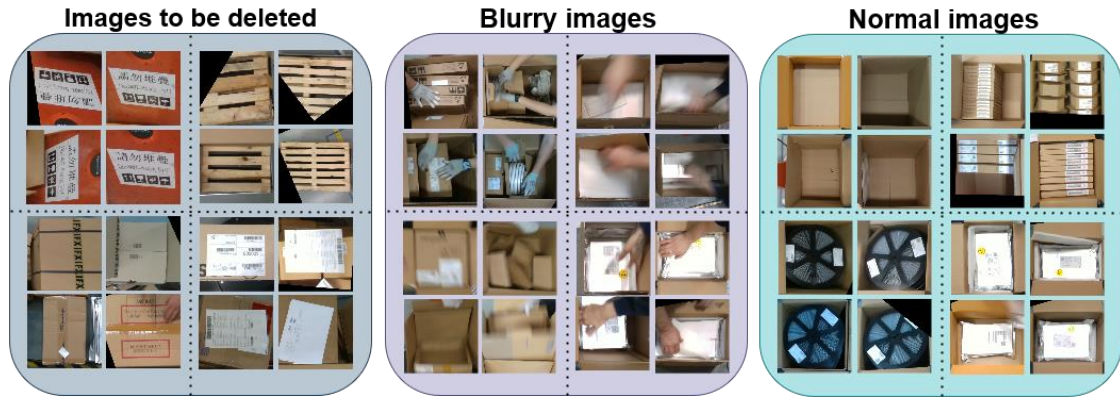
The proportion of different types of products was uneven and the characteristics of the resulting images were also very different, which makes anomaly detection more difficult. Moreover, examining different types of products together makes it impossible to find those cases which count as anomalies for some products but not for others. For example, a tearing or creasing is considered normal on a paper package, but anomalous on a box. So, it was necessary to group the images from the dataset according to the type of products. For this categorization, the images were the input of a pre-trained VGG16 network, without its classification head for feature extraction (new dimension: 4096). Next PCA was applied, for additional dimension reduction to 600 components. Finally, these features were clustered using K-means, with 100 clusters. These categorization steps are visualized in Figure 3.3.



3.3. Figure - The steps of the automatized part of the data categorization. VGG16 image source: [28].

This way the algorithm managed to efficiently separate images with different features. The images that the detector incorrectly detected as open boxes were also separated, these were deleted. Also, many images that happened to be blurry, or had people's hands or heads hanging in them, were also put into separate clusters. Figure 3.4 shows examples of images in each group. This process helped the categorization, but manual methods were unfortunately still needed. If such a simple clustering algorithm

could sort out the anomalous cases from the others, the whole task would be accomplished. Instead, it put normal and anomalous images of the same item into one cluster, so they had to be sorted manually.



3.4. Figure - Examples of the images from different clusters from the K-means results, and their categorization.

Whether an image should be considered an anomaly is an objective, difficult, and domain-specific decision. After careful consideration, only the ones that seemed completely normal were kept as training images. The ones containing any irregularities were sorted into different folders based on the type of deviation. For evaluation reasons, two more datasets were created, which should always be considered anomalous, with high confidence. First, images from the ImageNet dataset were chosen, that do not belong to the dataset and contain animals and objects. In the other, the images were selected from the original dataset, but they deliberately did not include the target products. The applied DL-based methods were sensitive to the cleanliness of the data, so any remaining blurry, noisy, out-of-focus images were manually eliminated.

3.2.3 Train-test split

There can be significant similarities between the images, as they were captured sequentially. For example, in the case of unpacking, the images that come after each other show one less product, but the rest of the image is the same, as shown in Figure 3.5.



3.5. Figure – An example of the similarity in an image sequence.

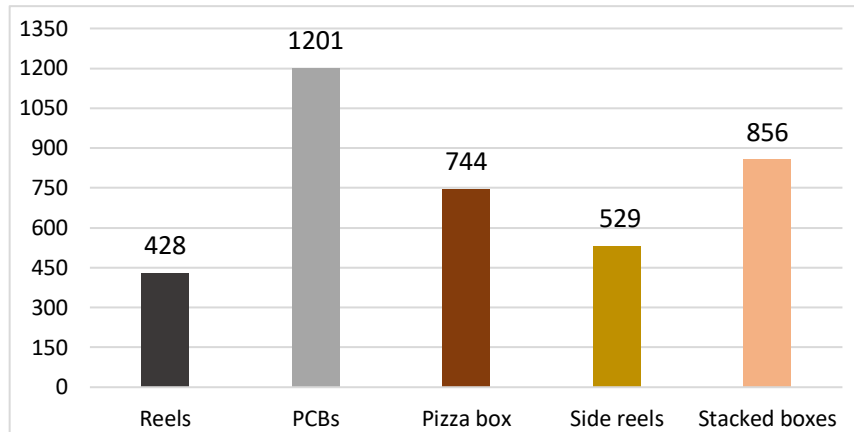
If some of these images were put in the training set and others in the testing set, it could cause significant data leakage, which could distort the results. To avoid this, the split of the test and train data was not random. The images taken on the same day were aggregated, ensuring that they were assigned to the same set. I trained the models with cross-validation, since unfortunately there were not so many days, and the distribution of products can be highly varied from day to day. For cross-validation, the dataset needed to be divided into more parts, and each of them was used as a testing set while the DL model was trained with the others. For this, the days with different numbers of images had to be divided into separate folds, in such a way that the number of images in them was the same as far as possible. This task corresponds to the Partition problem, which is NP-hard. It was solved with a greedy approximate algorithm, where the days were sorted in descending order based on the number of contained images and the next days' images were put in the smallest fold.

3.2.4 Data augmentations

Augmentation is a frequently used technique to make the models more robust and improve their performance. It is the process of performing different transformations on the data in each epoch. This way the network does not see the same image twice, just a slightly altered version of it. This reduces the chance of overfitting, increases the generalization ability, and makes the network less sensitive to noise. The amount and type of the possible transformations depends on the specific application, it is important to only perform transformations that can occur in real situations. The used augmentation techniques made the model insensitive to rotation with 90 degrees, and small changes in colour and contrast.

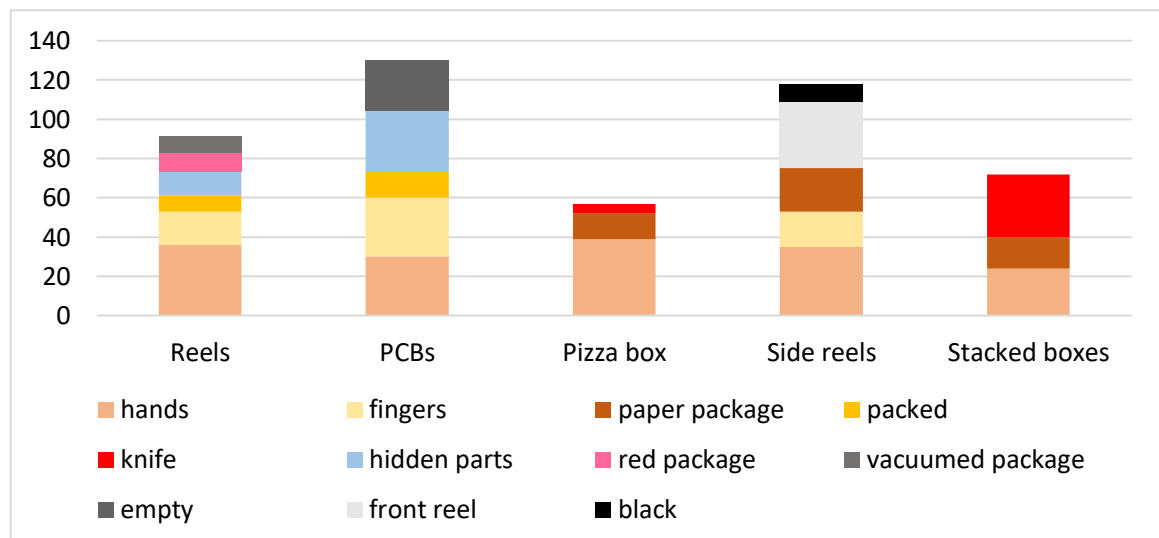
3.2.5 Prepared dataset

A separate dataset was prepared for each of the products, that were included in the images. The exact sizes of the different datasets (without any anomalies) are illustrated in Figure 3.6. This separation of the product types was advantageous because each has different properties and types of anomalies associated with them, so it was possible to observe which ones are more easily found.



3.6. Figure - Distribution of the datasets without any anomaly.

It is important to note that the anomalies were identified according to the outliers, that were present in the dataset. In the final production process, there should be anomalies, other than these which are not yet encountered. Also, the current images were taken during manual packing, so for example, in-hanging hands is currently an anomaly, but it may not be an issue in the final system as people would not be around the machine. The distribution of the anomaly types in the five created datasets is visualized in Figure 3.7. and their characteristics are detailed below.



3.7. Figure - The distribution of the different anomaly types across the different datasets

The datasets:

- **Reels:** Contains a top view of reels. The types of anomalies are hands (hands or arms of the workers), fingers (only the fingers of the workers), hidden parts (A paper or plastic package material is on the top of it), red packaging, vacuum packaging, and other packaging.

- **Side Reels:** Contains several different types of wrapped and unwrapped reels from a side view. The groups of anomalies are hands, fingers, paper package (the box contains unnecessary paper package material on the top), black (the reels are black), and front reel (one reel is pulled out and its front side can be seen on top of the others).
- **Pizza boxes:** Contains the top view of a thin closed box, which looks like a pizza box. The groups of the anomaly images are hands, fingers, paper packages, and knives (a small red box cutter knife is found on top of the boxes).
- **Stacked boxes:** Contains side-by-side, thin closed boxes from a side view. The anomaly groups are hands, fingers, paper packages and knives.
- **PCBs:** This dataset contains the top view of packed PCBs with different types of packaging materials under and next to them. The groups of anomaly images are hands, fingers, hidden parts, missing PCB (only the package, the PCB has been removed), packed (unnecessary packaging material wrapped around it), and knives.

One of the implemented methods, called Segment Any Anomaly+ is a segmentation approach, which requires segmentation masks. Fortunately, they are only needed for the evaluation of anomalous cases, as this method does not need training data. An example of the created anomaly masks can be seen in Figure 3.8.



3.8. Figure - Examples of the segmentation masks for the anomalous cases.

3.3 Anomaly detection with autoencoders

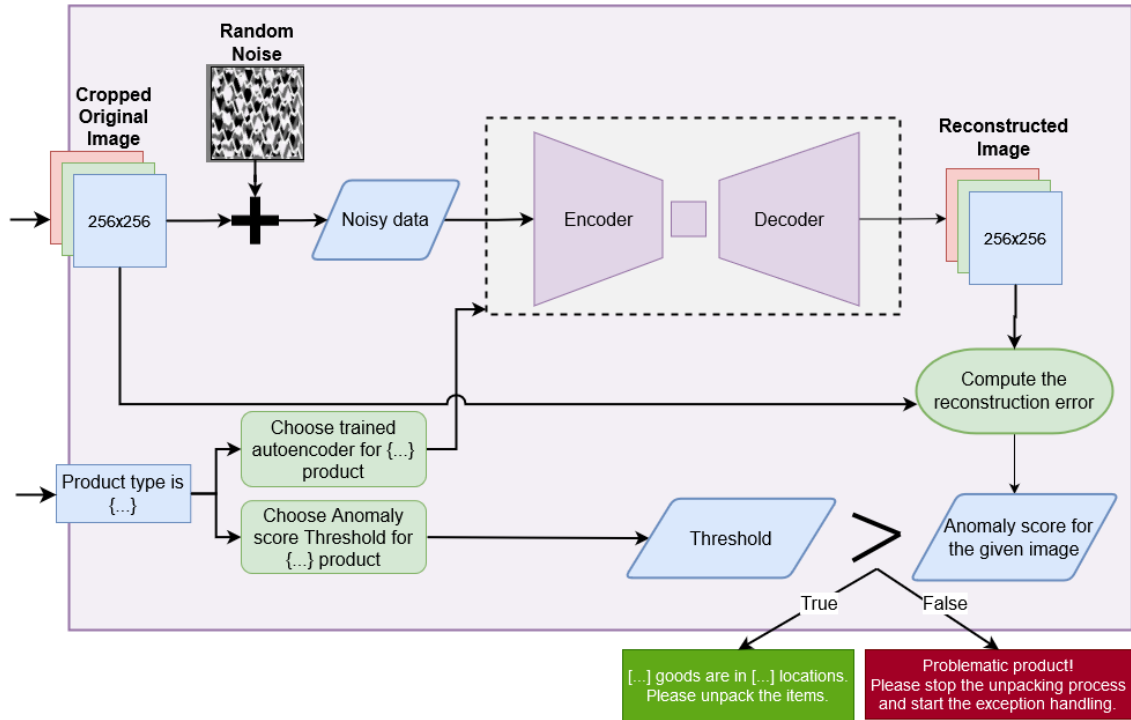
A deep autoencoder approach is a popular choice for unsupervised image anomaly detection problems [23]. The motivation for using it to detect anomalies is that when we train an autoencoder on normal data instances, it captures their most representative properties as it learns to reconstruct them. This way they should produce a larger reconstruction error for the anomalous instances, as they struggle to accurately reconstruct them. They work best when plenty of data is available and there is a significant difference between the normal and anomalous ones. Their performance significantly depends on the construction of a large, clean and representative dataset.

An early yet influential application of autoencoders for anomaly detection on tabular data [29], showed that the reconstruction error could effectively distinguish between normal and abnormal data. Building on this foundational work, several variants of autoencoders further advanced the field with varying architectures, parameters, and optimization techniques across different domains. In the field of image-based anomaly detection studies with the combination of autoencoders and CNNs achieved remarkable results (e.g. [30] on medical images). Beyond their general application, specific implementations in different industrial settings demonstrate their versatility and effectiveness. There is no single, universally accepted architecture, different approaches can work better for each dataset, anomaly characteristic and application domain. The relevance of this research is to find the best solution with the best parameters for the given use case. For example, they use generative approaches with Variational autoencoder in [30] and GAN-based adversarial autoencoder approach in [31]. However, in the given case generative solutions might be too time-consuming. Some methods suggest increasing the difficulties of image reconstruction by adding some noise or transformations to the input image and then training it to reconstruct the original input (e.g. [32]). Unlike simple augmentation, where the goal is to create a more diverse dataset, as the model learns to reconstruct the augmented image. This method can effectively increase the reconstruction difference between normal and abnormal images and is known as a de-noising autoencoder [22]. This research followed this approach, and different kinds of transformations were performed on the images, each with a different probability. The used functions were Gauss Noise, Gaussian Blur, Elastic Transform, Sharpening, Motion Blur, Coarse Dropout, Random Gravel, Pixel Dropout, Optical, and Grid Distortion. Some examples of their results can be seen in Figure 3.9.



3.9. Figure - Examples of the noises added to the input images.

The encoder was implemented as a commonly used CNN (e.g., ResNet50, ResNet18, DenseNet), and the decoder was its transpose. It means that each layer performed the inverse operation as the original CNN. It is a common practice to design autoencoders to be symmetric, as here, but this is not strictly necessary. The steps of the defined de-noising autoencoder-based method are illustrated in Figure 3.10. This can be integrated into the complete process, in place of the purple-coloured anomaly detection block in Figure 3.1., as it has the same inputs and outputs.



3.10. Figure – The flow diagram of the complete autoencoder-based anomaly detection workflow.

The loss function aims to compute the reconstruction error for every pixel of the original image, which the autoencoder tries to minimise. For this purpose, MSE or SSIM-based functions were used. A similar metric was used to evaluate the performance. For this, the SSIM-based techniques proved to be better, as the MSE or MAE-based methods were very sensitive to noise in the image, which was not a problem while training, but could degrade the evaluation. The reconstructed images of the first batch along with their heatmaps computed with the different metrics, were saved to visually inspect the process. An example of this can be seen in the results in Chapter 5, Figure 5.5.

Several different versions of the SSIM function were created to refine the final decision. The different methods were compared and the best-performing one was assigned to each product type. **MS-SSIM** calculated the Multi-scale SSIM score [3.6.1].

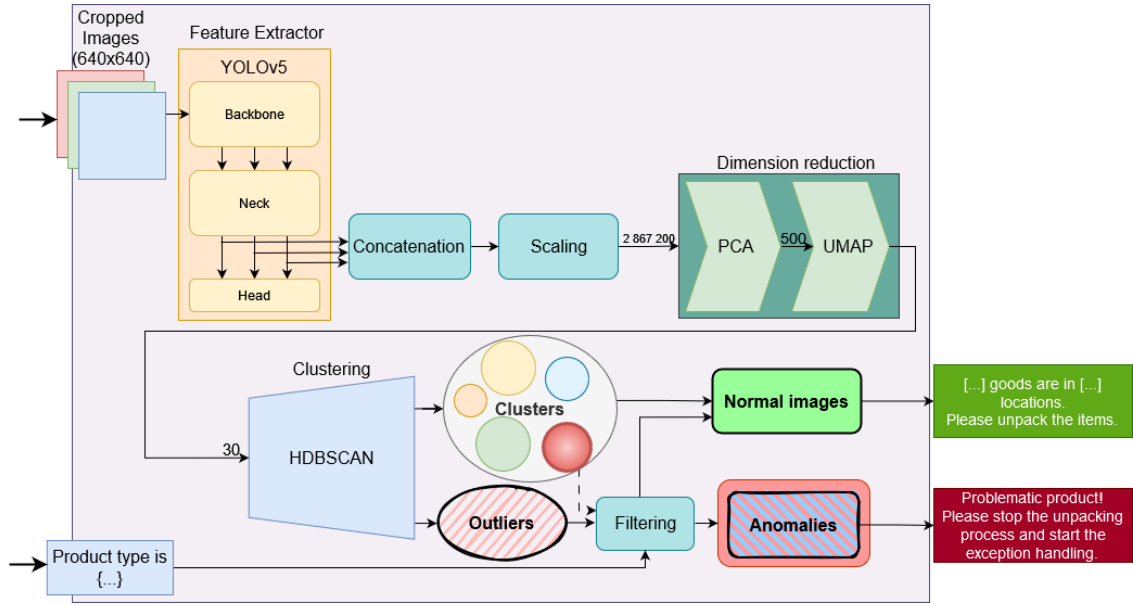
Patched SSIM was created to help the model detecting smaller anomalies. The idea is to calculate the error on only a smaller part of the image, doing this over the whole image several times in a sliding window approach, and then taking the value of the largest error as the result. **Thresholded SSIM** was intended to make the model less sensitive to noise. First, a median Gauss filter was used on the original and reconstructed image to eliminate the isolated high-intensity pixels. Then the differences that scored under 0.1 were zeroed out because they are probably not anomalies and the ones that were above 0.9 were set to 1 because they probably are. As the images are coloured, there was another option on how to aggregate the result of the different channels, for example, they could be summed or averaged.

All parameters can be specified in a YAML file, and each run's result is saved in a separate folder. Cross-validation and augmentation were used for training and a special method was implemented for the train-test split [3.2.3.]. The different types of anomalies were evaluated with the models from each fold, and their results were averaged. The validation images were different in each fold, these values were aggregated, this way a prediction was obtained for each image. Based on the validation images a threshold on the anomaly scores was calculated, from which the images were considered as an anomaly.

3.4 Anomaly detection with clustering

The use of deep hybrid models is a common technique for semi-supervised anomaly detection [23]. They use deep neural networks as feature extractors, and the learned latent features are inputs of traditional anomaly detection algorithms like clustering to detect outliers. The dimensionality of the input is reduced within the layers of the neural network which ensures scalability for high dimensional data. For the AI model usually, CNN or autoencoder-based models are used, as in [33] and [34].

For the given use case a novel architecture is proposed, which uses the YOLOv5 object detection model, which is part of the original workflow. Its steps are visualized in Figure 3.11. This can also be integrated into the complete process, in place of the purple-coloured anomaly detection block in 3.1.



3.11. Figure - The whole architecture of the proposed clustering-based anomaly detection method.

3.4.1 Feature extractor

The object detection model, which is part of the original workflow process can be used as a feature extractor. It is worth choosing this, as it doesn't cost any extra time. Also, the successful object detection demonstrates that the model successfully discovers the main characteristics of each product. Since this model was not yet available, another YOLOv5 model was trained to detect the products and was used in this research. The YOLO can learn location and rotation invariant features instead of pixel intensities in the original images, which information can be much better used by clustering. This model was truncated after an earlier layer, to extract a higher dimensional, complex representation. There were more possible places for this truncation:

- **Before the Neck** (Dim.: $20 \times 20 \times 1024 = 409\,600$): The backbone of the network is a CCN, called CSPDarknet53. It should capture only high-level feature maps, which has not gone through any object-detection-specific layer. These features could be useful for general clustering, although important information may be lost, that is specifically provided by object detection, such as the location and number of products.
- **Before the Head** (Dim.: $20 \times 20 \times 1024 = 409\,600$): At this location the features are more processed. It may provide more object-detection-specific patterns, but could lose important features, which are significant for anomaly detection, but not for object detection.

- **All outputs before the Neck or the Head:** (Dim.: $20 \times 20 \times 1024 + 512 \times 40 \times 40 + 256 \times 80 \times 80 = 2\,867\,200$): Here not only the output with the highest scaling would be considered, but all three outputs of the neck or head, aggregating their results. This could be effective because it captures multi-scale information from different resolutions. The downside is that the aggregation can cause larger dimensionality and can contain more noise and redundant information. For aggregation, the easiest approach is to concatenate the vectors, but it significantly increases the size. The addition of the vectors is an alternative option, which has the drawback of losing information.
- **All outputs in the middle of the Neck** (Dimension: $(20 \times 20 \times 512) + (556 \times 40 \times 40) + (256 \times 40 \times 40) = 1\,024\,000$): Inside the layers of the neck, there is a part, where it also can be truncated at each resolution. This is the thinnest part of the network, so it contains more compact information but still aggregates information from all scales. It could balance the result of dimension complexity and scale aggregation.

3.4.2 Scaling methods

The features extracted from the network should be scaled, to ensure that each feature contributes equally, especially in distance-based algorithms. Experiments were made with MinMax, Standard, and Robust scaling. It is important to choose a solution, that does not suppress any outlying features, as these may indicate anomalies. The best choice seems to be Robust scaling, because it maintains a good balance by scaling the data based on the majority of the data points, without letting the anomalies influence the scaling too much.

3.4.3 Dimension reduction

It is crucial to reduce the dimensions of the representation, because clustering usually only works well with small dimensional data, due to the “Curse of dimensionality” [20]. The most common linear dimension reduction approach is PCA, and its only parameter is the number of the final dimensions. For high dimensional data, like images, the non-linear algorithms often perform better. From these UMAP was used, because it is fast and tends to produce the best results. It has many configurable parameters, with which the balance between global and local structure in the data, the metrics of the distance between the data points, and how tightly the points are allowed to pack together can be controlled. When it combines the different local data points, it uses a union, but for anomaly detection, taking the intersection may be better, because it could

ensure that outliers remain disconnected. However, it could also break up the resulting simplicial sets into disconnected components. For a better solution, there is a parameter that controls the interpolation between the two. Using UMAP before a density-based algorithm can be effective in practice, but it is also controversial because UMAP does not completely preserve density and can create false tears in clusters [35].

In this particular problem, neither PCA nor UMAP worked well enough. The clustering algorithms that were applied after them, failed to cluster the features properly, suggesting that the process lost essential information. However, a hybrid solution, that used both approaches provided an efficient and successful solution. First PCA reduced dimensionality while preserving the global linear structure, removing noise, and simplifying the data, then UMAP further reduced dimensions while capturing both local and global nonlinear relationships. They created a low-dimensional, denoised, and well-structured representation.

3.4.4 Clustering

For anomaly detection, the best approach is to use density-based clustering. Unfortunately, DBSCAN failed to correctly identify the clusters, because it was overly sensitive to the parameters and could not handle varying sizes of distortions. HDBSCAN performed much better, due to its capability to search clusters across different density levels. It was less sensitive to parameters, more configurable, and gave more stable results.

3.4.5 Filtering

As a last step the images, which the clustering algorithm considered as anomaly must be examined. Detectable false positive cases should be filtered out. Moreover, if the clustering is performed, with the known anomalous images included, they might be identified as a different cluster, instead of anomalies. These clusters must be checked and marked because if the algorithm predicts a new point in this cluster, it is also an anomaly.

3.4.6 Prediction

It would be too slow to re-cluster all points with this complex architecture, for every new data point. Especially, because this time will be much more with a bigger dataset. Therefore, at prediction, the existing clusters do not change. The processing steps are only performed for the new data, and it is decided, which pre-defined cluster it would

have been assigned to. These new images do not change the clusters, even if the result would have been different if they had been in the training set.

3.5 Anomaly detection with Segment Any Anomaly+

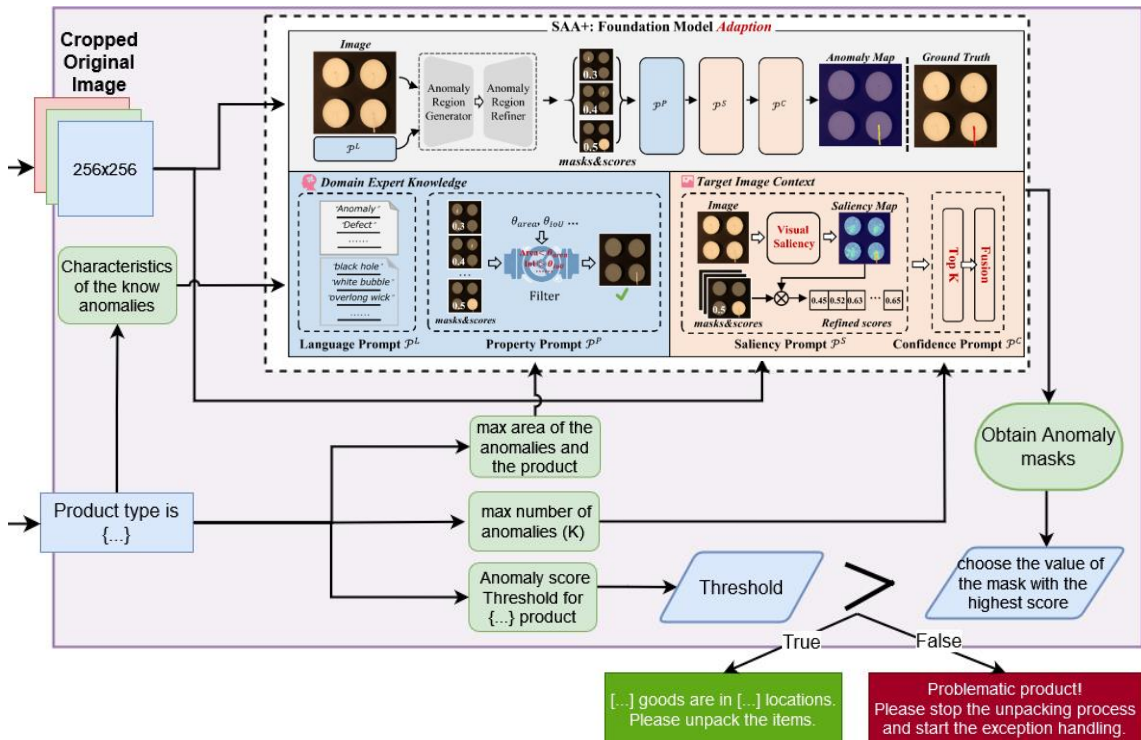
The biggest limitation of previous anomaly detection methods is that they require a large dataset and cannot be used for example at the start of a new industrial process, until enough data has been collected. In a real-world scenario, there could be many different products, so it is not cost-effective to collect a large training set for each. Thus, afterwards, their performance is still strongly influenced by the quality of the dataset. However, in zero-shot techniques, like the Segment Any Anomaly+ (SAA+) [36] model, this problem is avoided as there is no need for training data. SAA+ is a recent and novel model for anomaly segmentation in images. This method achieved significant results in the zero-shot anomaly detection field and outperformed previous models on several benchmark datasets. However, there are no publicly available records yet of its performance and usability on more complex, real-life datasets, like the one used in this research.

It works by leveraging pre-trained foundation models to identify the anomalies, without requiring explicit training, by retrieving prior knowledge stored in these models via prompting. The authors first constructed the Segment Any Anomaly (SAA) model, by cascading a prompt-guided object detection model, named Grounding DINO [37], and a segmentation foundation model, named Segment Anything [38]. These serve as Anomaly Region Generator to identify the possible anomalous parts of the images and Anomaly Region Refiner, to create the segmentation masks, respectively. For these naive language prompts were utilized, like “defect” or “anomaly”. This solution tended to cause many false detections because the word “anomaly” could mean very different things in different contexts, making it difficult for linguistic models to interpret the word, the authors called this language ambiguity.

The SAA+ was an improved version of this, in which they integrated domain expert knowledge and target image context. The domain expert knowledge describes the characteristics of the relevant anomaly types for the given product. The target image context includes creating a **visual saliency map** from the input image, which helps the model focus on key areas, that may show anomalies. By highlighting these important regions, the model can better detect subtle or hidden anomalies. The **maximum number**

of anomalies (K) can also be defined, and with **confidence ranking-related prompts**, the top K results are kept. Besides language prompts, which detail the characteristics of the most possible anomalies, **property prompts** are used to address foundation models' lack of awareness of specific properties, such as “count” and “area”. They are constructed in the form of rules, instead of language and the detections that do not meet these criteria, can be filtered out. (e.g. the maximum size of the anomalies can be specified, and the bigger ones, can be easily omitted.)

The appropriate language and property prompts were constructed for every dataset. The main features of the anomalies were defined, as well as the maximum area and maximum number of anomalies. The outputs of the SAA+ were the obtained segmentation masks. The value of the mask with the highest confidence was assigned to the whole image as the anomaly score. Based on the available normal images a threshold could be defined, above which the images are considered as anomalies. The architecture of the SAA+-based approach is illustrated in Figure 3.12. This can also be integrated into the complete process, in place of the purple-coloured anomaly detection block in 3.1.



3.12. Figure – The Segment Any Anomaly + (highlighted with dashed contour, source: [36]) and the proposed necessary steps for its integration into the industrial workflow.

3.6 Evaluation metrics

There are several different approaches to evaluating anomaly detection, from both subjective and objective perspectives. In this chapter the most significant ones are detailed. The evaluation of the methods was conducted based on these metrics, which results are presented in Chapter 5.

3.6.1 Subjective metrics

Many different metrics can be used to obtain the anomaly scores, that measure the outlierness of each image in the dataset:

- **Mean Average Error (MAE)** and **Mean Squared Error (MSE)** are the most used cost functions for regression problems. They work with a simple principle, are fast to compute and are well-suited to a wide range of problems. **MAE** computes the mean of the absolute value of the difference between the expected (Y_i) and the predicted (\hat{Y}_i) result.

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i|$$

- **MSE** computes the mean of the square of the difference between the expected (Y_i) and the predicted (\hat{Y}_i).result.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

- **Structural Similarity Index Measurement (SSIM)** [27] is an image quality assessment metric used to determine the ratio of similarity between two images. It is not intuitive to measure the quality distortion of an image, because it is difficult to mathematically define what are the most significant changes for a human eye. Traditional methods, such as MSE and MAE are simple to calculate and have clear physical meaning, but they do not always reflect structural or visually noticeable differences between images. SSIM, on the other hand, assesses the quality of images by considering the specificities of human visual perception and based on the assumption that it is most sensitive to structural information change. First, it compares the average brightness of the images:

$$l(x, y) = \frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}, \text{ where } \mu_x \text{ and } \mu_y \text{ are the average brightness of images x and y.}$$

Then measures contrast based on the standard deviation of the images:

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \text{ where } \sigma_x \text{ and } \sigma_y \text{ are the standard deviation of images } x \text{ and } y.$$

The structure is measured by the covariance between the normalized pixels of each image:

$$s(x, y) = \frac{\sigma_{x,y} + C_3}{\sigma_x\sigma_y + C_3}, \text{ where } \sigma_{x,y} \text{ is the covariance between the two images.}$$

C_1 , C_2 and C_3 are small additional values for computational stability. The final SSIM score is calculated by combining these three components:

$SSIM(x, y) = l(x, y)^\alpha \times c(x, y)^\beta \times s(x, y)^\gamma$, where α, β and γ correspond to the weight of each component, they are usually set to 1.

In the end, SSIM derives an index, where 1 indicates full similarity and 0 indicates full dissimilarity. With this approach, SSIM offers a more accurate assessment of image quality.

- **Multi-scale SSIM (MS-SSIM)** extends SSIM by evaluating across multiple scales or resolutions [39]. This is done by applying SSIM at different image scales, capturing both local and global structural information. It allows the algorithm to account for image details at different levels of granularity. It captures more complex distortions, which makes it more robust for images with varying sizes of deviations.

After an appropriate anomaly score is obtained for all images, the metrics to measure the performance of anomaly detection are in practice the same as the metrics used for binary classifiers, since the goal is to separate normal and anomalous images. Thus, the most used are precision, recall, and F1-measure. A confusion matrix can be created to visualize the distribution of true positive (TP), false positive (FP), true negative (TN) and false negative (FN) detections. Precision measures the accuracy of true predictions among the positive predictions. Recall measures the ability of the model to identify all relevant instances (anomalies). The F1 score is the harmonic mean of precision and recall, providing a balance between the two when the class distribution is uneven.

$$Precision = \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN}, \quad F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall}$$

- For the **evaluation of the segmentation masks** in SAA+, several metrics were used at both instance (*i_prefix*) and pixel (*p_prefix*) levels [36]. Instance level measures the proportion of correctly predicted anomalous instances, while pixel level measures the proportion of correctly classified pixels. The **i_ROC** and **p_ROC** (Receiver Operating

Characteristic) metrics measure the model's ability to detect anomalies, while **i_AP** and **p_AP** measure the Average Precision scores and **i_F1** and **p_F1** provide the F1-scores.

3.6.2 Objective metrics

- By visual inspection it can be seen whether the images with high anomaly scores are anomalous, or whether they have any salient features that cause the network to consider them as anomalies. To facilitate this, the **heatmap** of the differences between the original and target image can be generated. This method reveals which pixels of the image caused the model difficulty to recover, i.e., detected as possible anomalies.

4 Implementations

In this chapter details about the used hardware, software and libraries are provided. The implementation of the different methods is available in a public GitHub repository¹.

4.1 PyTorch

In this research, Python was used as the main language, as it is the most popular language for deep learning nowadays. PyTorch and Keras are two commonly used libraries, that are specifically designed to effectively solve deep learning problems. From these, PyTorch was used. This library [40] has the advantage of being more flexible and customizable. It offers dynamic computation graphs, enabling users to build and modify models on the fly, making experimentation and debugging easier. PyTorch provides a comprehensive set of tensor operations with GPU acceleration support, simplifying tensor manipulation. It provides a variety of pre-defined modules, models, and layers.

4.2 Scikit -learn

Scikit-learn is a Python library, which contains simple and efficient tools for predictive data analysis [41]. It offers pre-defined, optimized solutions for a wide range of machine learning algorithms. It is open-source, accessible, and reusable to everybody. It's easy to use and import. The library is based on Numpy, SciPy, and Matplotlib. It was used for simpler machine learning operations, like clustering and dimension reduction.

4.3 Albumentations

Albumentations is a Python-based computer vision library that aims to boost the performance of deep learning-based vision methods [42]. It's a popular, open-source library, that is widely used in industry and deep learning research. It provides techniques for fast and flexible image augmentations by efficiently implementing a rich variety of image transform operations that are optimized for performance. It supports any computer vision tasks and works well with data from different domains: photos, medical images, satellite images, manufacturing, and industrial applications. It can be integrated into

¹ <https://github.com/TBeatrix/Anomaly-Detection-For-Industrial-Automatization.git>

various deep-learning frameworks such as PyTorch and Keras. This was used for augmentation techniques and for adding different kinds of noises to the images.

4.4 Weights & Biases

Weights & Biases (WandB) is an outstanding tool for tracking, managing, version controlling and visualizing machine learning training experiments. It is simple and intuitive to use, easy to integrate into an existing environment and offers a wide range of extra features. It logs the results, creates easy-to-view charts, and stores all the parameters and results in a spreadsheet. The different trainings can be easily grouped or ordered according to any criteria.

4.5 Label Studio

Label Studio is an open-source tool for annotating data like text, images, and audio. It's customizable for various tasks, helping create labelled datasets for many machine learning projects. The labels and segmentation masks were obtained with this tool.

4.6 Docker

Docker is an open-source platform that allows developers to automate the deployment, scaling, and management of applications inside lightweight, portable containers that include everything needed to run the software, such as code, libraries, and system dependencies. The experiments were run within a Docker container on a server.

4.7 GPU

Since the training of an anomaly detection model is very time- and computation-consuming, the Graphics Processing Unit (GPU) played an important role in the successful completion of the training. Two 40GB NVIDIA A100 GPUs were used. The autoencoder-based method was trained the longest, as it used cross-validation. The training of five folds took about 20 GPU hours.

5 Results

In this chapter, the performance of the three different methods to detect anomalies is presented. Each of them has different advantages and weaknesses, which are discussed in detail. The results have been numerically evaluated and are also visualized with diagrams.

5.1 Autoencoder approach

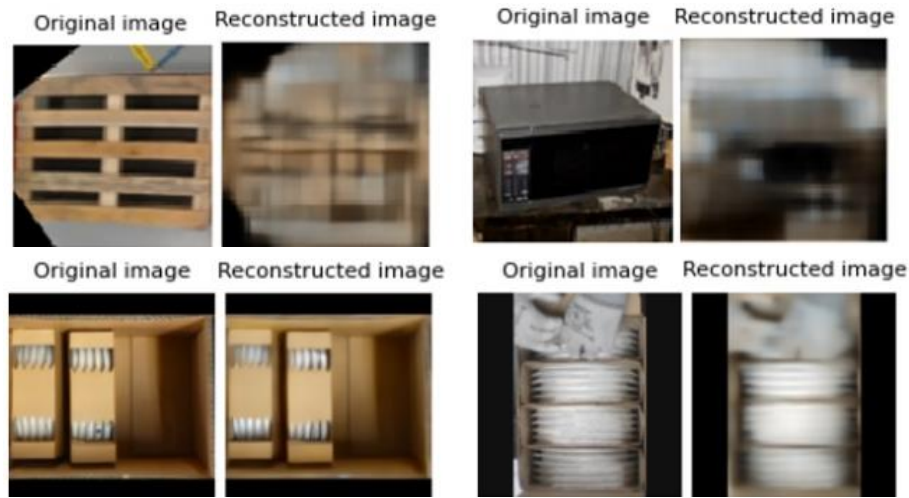
Different autoencoder models have been trained for each product type. The exact hyperparameters and transformations are detailed in Appendix A. All models were trained on clean, normal images with randomly initialized weights. The vanilla autoencoder compresses the images into a low-dimensional space and restores them with minimal errors, which is the original aim of autoencoders. However, in anomaly detection, the aim is not the perfect restoration of the whole image, but to learn and restore the characteristics of a particular product, and only that. This model failed to do that, as it even restored objects that represent something completely different, as can be seen in Figure 5.1. Therefore, this method is not suitable for anomaly detection since the reconstruction error does not increase significantly for anomalous cases.



5.1. Figure - The Vanilla autoencoder method well-reconstructed images of the product (first), even if it contained anomalies (second), as well as images completely unrelated to the product (third, fourth).

The purpose of the de-noising autoencoder was to make the model's task more difficult, forcing it to focus more on the given products. It became an additional task to

remove the noise mixed with the images and to fill in any missing details. This way it learned the main characteristics of the items and if something looked different than usual, it tried to recreate the original product in its place, as seen in Figure 5.2.



5.2. Figure – The de-noising autoencoder well-reconstructed images of the product (first), but not its anomalous parts (second), and nor the ones, completely unrelated to the product (third, fourth).

In this case, as intended, the reconstruction of the anomalous parts appears to be much worse. Thus, it is appropriate for anomaly detection. The drawback was that the restoration of normal images could also become blurrier. To find the best-performing model, several optimization was performed with different types of noises and transformations. The final ones are detailed in Appendix A. It can be concluded that for simpler-looking products (e.g. pizza boxes), less noise led to the best results, while for more complex configurations (e.g. side reels), more noise was needed.

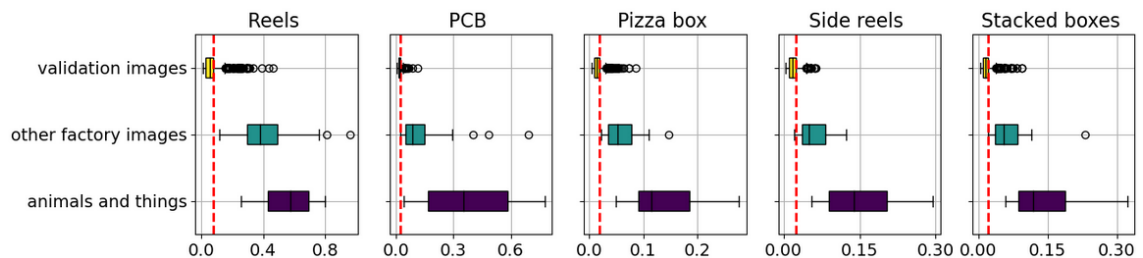
The model's evaluation was complicated by the fact that insufficient data were available from each product for a good performance. This drawback was especially clear when setting a threshold, above which the images are classified as anomalous. Ideally, this would completely separate the two types, but since the notion of anomaly can also be questionable, there should inevitably be some overlap between the two. A separate test set was not created, as there were already few images available, and reducing them would have hurt the quality of the training, thus currently the exact numerical evaluation is less meaningful. The threshold was set at the value, where 75% of the validation set is correctly classified. This would be weak in practice, as it is assumed that every fourth detection would be a false positive. However, this is a reasonable expectation, due to the small dataset, which is not sufficiently representative for all data types. By examining these false positive detections, they mainly contain images that are poorly represented or

have some special (but not anomalous) characteristics, like too many labels, a different colour, dissimilar shape, or simply were captured from an unusual angle, such as the ones in Figure 5.3.

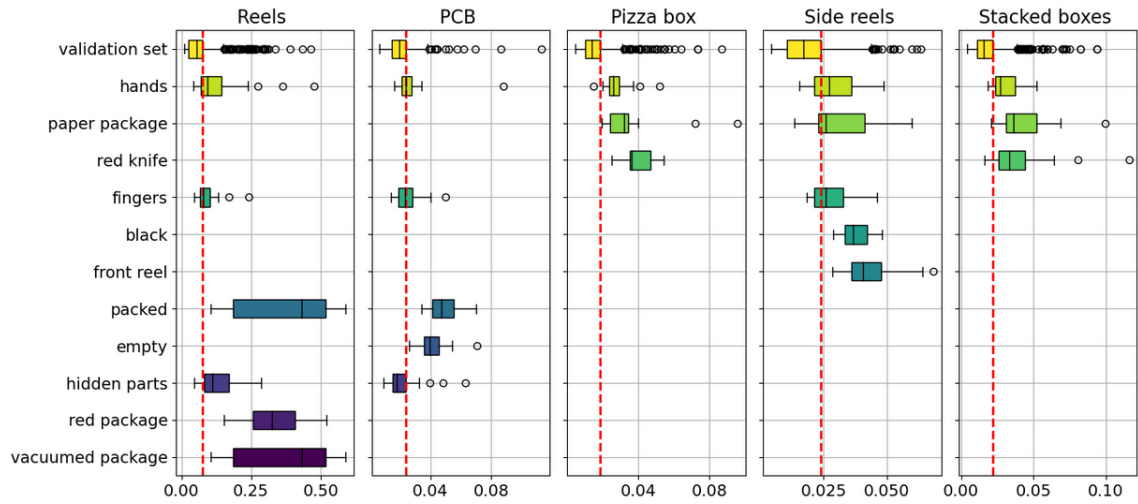


5.3. Figure - Examples for false positive detections, with unique characteristics. (1. is from stacked boxes, 2. is from side reels, 3. and 4. are from pizza boxes and 5. and 6 are from reels.)

This does not mean that the method was incorrect at these detections, since product types, that occur too few times are correctly considered as anomalies by the algorithm. Even if they are not domain-specific anomalies. This problem could be reduced and even a much higher threshold could be established as soon as more images are collected. Despite this, the results demonstrate that the method is capable of finding anomalies and will be worth using. The results are visualized in Figures 5.4 and 5.5, separately for each product and for each anomaly type, that was observed for them. The top yellow bar in each diagram belongs to the validation set of the normal data. The threshold is visualized with a dashed red line. Figure 5.4 shows the anomaly groups, that contain completely different images from the task, these were almost perfectly separated. Figure 5.5 shows all the other anomaly types.



5.4. Figure - Boxplots of the performance of the de-noising autoencoders on the different datasets (columns) with the anomaly types, that contained different images than the given product (rows). The threshold is visualized with a dashed red line.



5.5. Figure - Boxplots of the performance of the de-noising autoencoders on the different datasets (columns) with different anomaly types (rows). The threshold is visualized with a dashed red line.

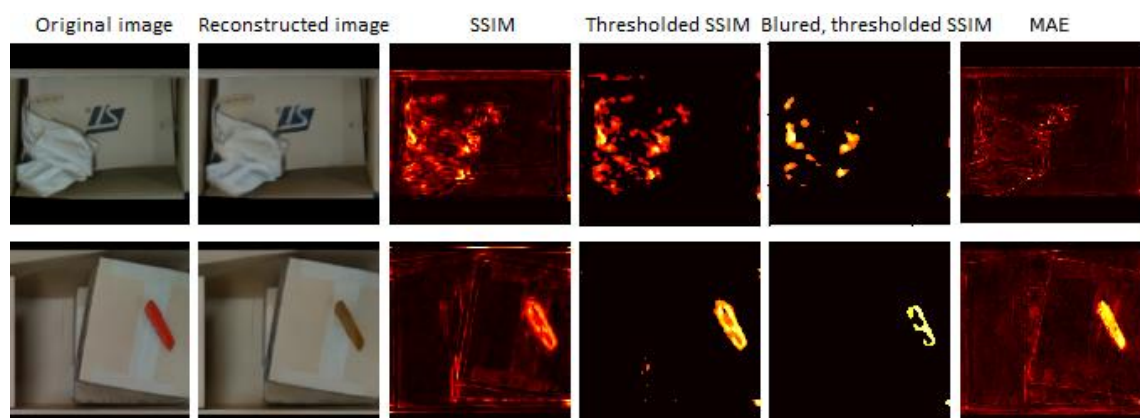
The exact evaluation of the accuracy of the detected anomalies by the model, for different threshold values (75, 80, 90, 95) is shown in Table 1.

1. Table - The accuracies of the detected anomalies for each product and anomaly type in percentages, at different thresholds, which shows the accuracy of the correctly identified images in the validation set: 75 (blue), 80 (green), 90 (yellow), and 95 (orange).

	Reels (%)				PCBs (%)				Side reel (%)				Pizza box (%)				Stacked box (%)			
validation images	75	80	90	95	75	80	90	95	75	80	90	95	75	80	90	95	75	80	90	95
animals and things	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
other factory images	100	100	96	85	96	96	92	88	100	96	88	69	96	100	96	96	96	96	85	69
hands	67	64	31	14	50	33	13	3	97	56	40	23	60	97	74	41	83	71	42	25
paper package	-	-	-	-	-	-	-	-	68	50	36	32	100	92	77	62	94	88	75	44
red knife	-	-	-	-	-	-	-	-	-	-	-	-	100	100	100	80	91	84	56	31
fingers	59	47	18	6	47	40	17	7	56	50	28	17								
black	-	-	-	-	-	-	-	-	100	100	89	44								
front reel	-	-	-	-	-	-	-	-	100	100	94	69								
packed	100	100	88	62	100	100	100	92												
empty box	-	-	-	-	100	100	88	65												
hidden parts	73	73	45	9	26	19	13	10												
red packed	100	100	100	80																
vacuum packed	100	100	88	62																

As can be seen, this method worked best for the Pizza and Stacked boxes. This is because their features are more simply captured by the autoencoder, as these items have much simpler characteristics, without a lot of edges or fine details. Among the different

types of anomalies, the larger ones (red and vacuumed package, packed, front reel) scored higher, due to the fact that larger anomalies imply larger deviations in reconstruction. However, it is not true that the smaller the anomaly is, the harder it is to detect. For example, knives were small, but because of their striking red colour, they had high anomaly scores. Hands are more detectable than fingers, but they became harder to spot when the worker wore a white glove that blended into the background. In general, the autoencoder managed to detect better those anomalies, which had bigger differences from the normal data, in size or colour. Based on the differences between the original and reconstructed images, heatmaps can be created, to obtain information about the location of the anomalies. Figure 5.6 shows examples of these, based on different metrics.



5.6. Figure – Heatmaps, based on different metrics. They well reflect the differences between them.

Preconditions:

- To effectively use this approach a huge amount of data is needed.
- The images should be diverse, high-quality, and have a clear resolution.
- The anomaly types should be clearly distinguishable from the normal.
- The construction of the dataset requires a lot of pre-processing steps.

Advantages:

- Do not need labels.
- Fast prediction.
- It is not considered an anomaly if some parts do not always appear in the same place and the same number.
- Even small deviations can be detected.
- The heatmaps produce pixel-based information about the location of the anomalies, which explains the decisions.

Limitations:

- Every product requires a separate model and a sufficiently large dataset.
- Can not be used right away for new products.
- It has difficulty detecting anomalies with similar characteristics to the normal.
- Does not consider an anomaly when two separately already known things appear together or parts that differ from the norm only in size or number.

5.2 Anomaly detection with clustering

The proposed architecture for clustering involved a lot of hyperparameters, which had to be carefully adjusted, as it was very sensitive to them. In the final models, the YOLO was truncated before the Neck, only at the biggest resolution, so no concatenated method was needed. Both PCA and UMAP were used for dimension reduction, the scaling type was MinMax or Robust scale and HDBSCAN was the clustering algorithm. The exact parameters of the best models are detailed in Appendix B.

This method was not able to detect small anomalies such as hands or knives. This is because there are normal images that share too many similarities with the anomalous ones and clustering tends to classify based on the greater similarities (e.g. the style of the reel) and neglects the minor differences. Examples of this are shown in Figure 5.7. This was also complicated by the fact, that the used dataset contained complex images, which have varying appearances even within the same product. In other words, in the case of small anomalies, the variance within the normal data was larger and more significant for clustering, than the variance between anomalous and normal images. This is due to the way clustering works. This does not cause a problem for autoencoders, where the model distinguishes based on the differences, rather than similarities.

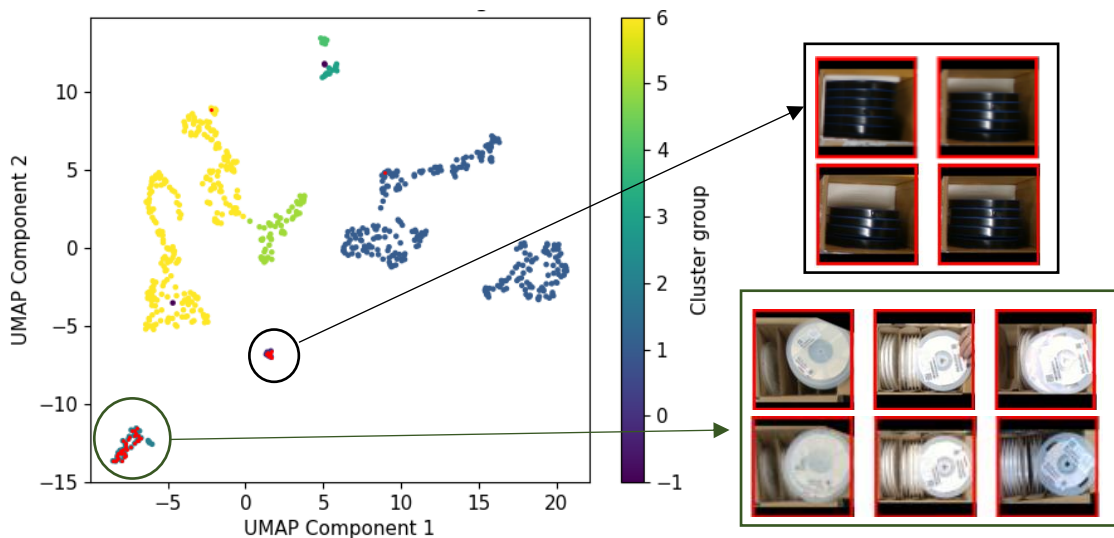


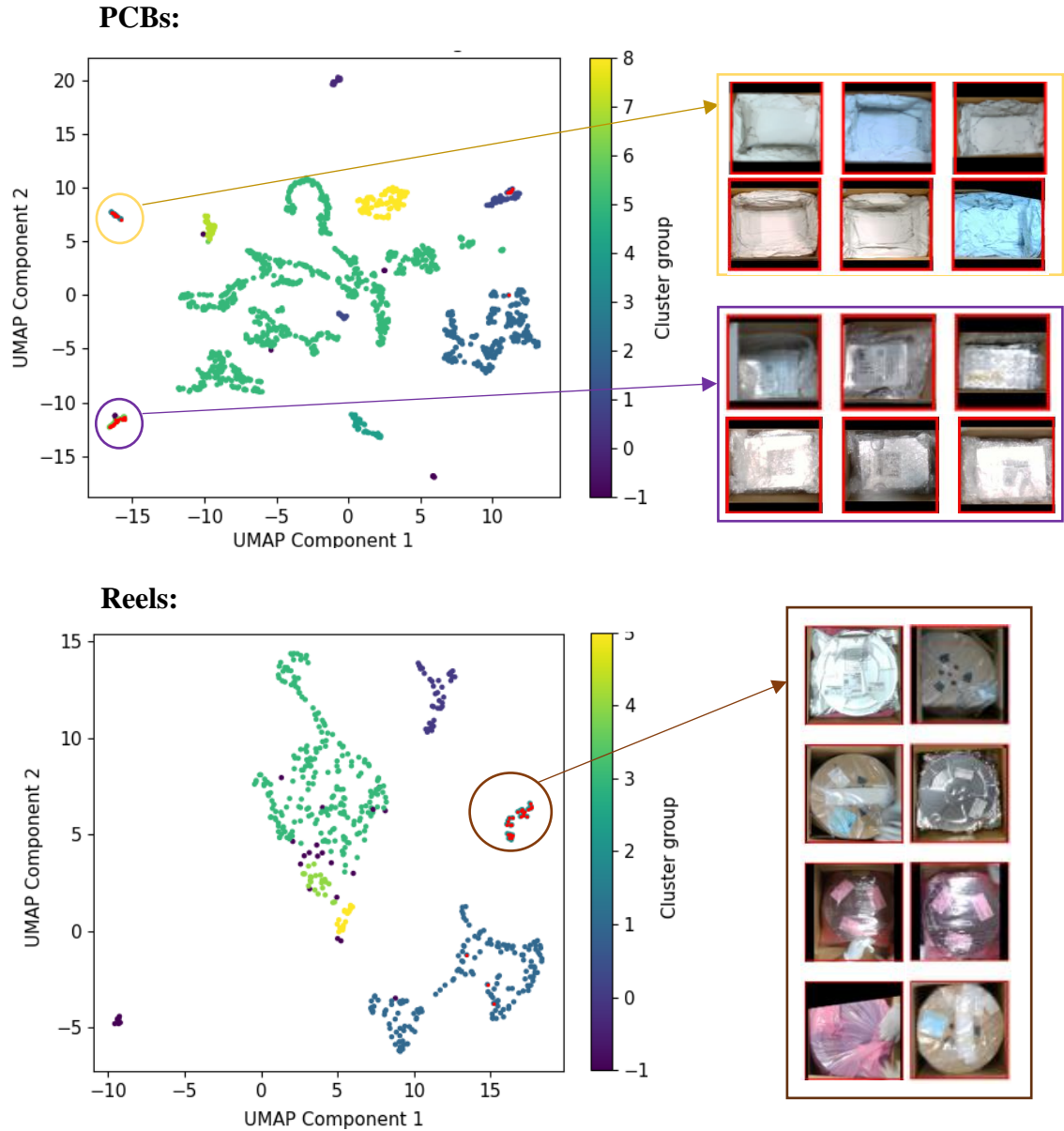
5.7. Figure – Anomalous images of reels, and normal images, which were highly similar to them.

However, the model successfully detected images with larger anomalies. The datasets that contained bigger anomalies were Reels (the red, vacuumed, and other packages), Side reels (front and black reels), and PCBs (empty and packed), so only these were used with this method. Currently, the most essential aspect of the use case is to identify these bigger outliers. Minor ones cause fewer problems, as most of them will not even occur later,

when the workflow is automated, such as hands or knives. But there might be new types of smaller anomalies, in new situations. For these, one of the other solutions should be used. The best performance was achieved when both anomalous and normal images were used for clustering. This way the anomalies made it easier for the clustering algorithm to recognise what should be separated and it was also easier to determine which parameters needed to be tuned. Moreover, the obtained results were more stable and robust. This way there were anomalous images not detected as anomalies by the algorithm but assigned to a separate cluster when enough instances of an anomaly type occurred in the dataset to form its own cluster. This could only happen with known anomalies. After the training of these methods, these specific clusters can be identified and marked. When a new data is assigned to these clusters, they should be considered as anomalies. Thus, information about the anomaly type of this new data is also obtained (based on the cluster, it was assigned to), which is important for the explanation of the detections. New kinds of anomalies should be selected as anomalous by the algorithm, as they do not fit into any predefined cluster. The method was evaluated based on the original clustering partitioning on the full dataset. No separate test dataset and prediction were used, as there were only a few instances of the larger types of anomalies, and they were highly similar to each other. Hence, it was not possible to construct a proper and suitable test dataset, not even for cross-validation. However, if the clustering analysed this way separates the anomalies properly, similar behaviour can be expected during prediction. The clustering was performed in more dimensions, but for visualization purposes, a 2-dimensional representation of the data was created with UMAP, which is visualized in Figure 5.8. The points were coloured based on the assigned cluster, and anomalies were coloured red.

Side Reels:























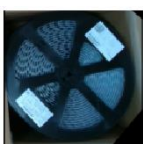























5.8. Figure - 2D UMAP visualizations of the clustering result for the Side reels, PCBs and Reels datasets. Each colour corresponds to a different cluster. The anomalous points are coloured red and some samples from them are visualized.

It can be seen from the results, that the bigger anomalies are indeed assigned to separate clusters. Moreover, examining the different clusters showed that, the method was capable of well separate the different types within a product. This way this method can also be used to detect special types of the product, which are not anomalies, if we want to distinguish them for some reason. To demonstrate this, Table 2 shows some of the elements of the clusters created in the Reel dataset. This clearly shows that the different types have been distinguished into separate clusters. Their colour indicates their location on the plot. This also shows that the 4th, 5th and 6th clusters, which contain similar

elements, are also close to each other in the visualisation. And the 1st, 2nd and 3rd clusters, which are more dissimilar, appeared further apart.

2. Table - The clustering results of the Reels dataset, with visualized instances of each cluster.

Cluster -1 corresponds to the outliers detected by the model.

1.						
2.						
3.						
4.						
5.						
6.						
-1						

Since all the bigger anomaly types in the dataset formed their own cluster, the anomalies that are currently detected by the HDBSCAN clustering algorithm are all false positive detections in each dataset, but their numbers are small. These are the images, that the algorithm could not assign to any of the other clusters for some reason (e.g. they are underrepresented in the dataset). Examples of these can be seen in Table 2 at the -1 cluster. Nevertheless, these detections cannot be ignored, because new anomaly types will be included here. If perhaps in a later model, despite proper tuning of the parameters, the number of these false positive elements grows out of control, it may be worth introducing

extra filtering on these items to find out which ones are actually anomalous, for example using the autoencoder-based solution.

Table 3 shows the numerical results of the clustering as the recall, precision and F1 score of the detected anomalies, and the numbers of different positive and negative detections. In this method, each image is an anomaly or not, depending on which cluster it is assigned to. No confidence values or anomaly values are associated with the images.

3. Table - The Recall of the detected anomalies, and the number of True Positive (TP), False Negative (FN), False Positive (FP), and False Negative (FN) detections. The second element of the addition at the FPs corresponds to the number of anomalies detected by the clustering.

	TP	TN	FP	FN	Recall of anomalies	Precision of anomalies	F1 score of the anomalies
Reels	24	491	1 + 27	2	92,3%	46%	61,4%
Side reels	46	593	6 + 3	2	95,8%	83%	91,5%
PCBs	37	1282	5 + 13	7	92,5%	67%	77,7%

Anomalies were found with a high degree of accuracy for all data sets. The future increase of the dataset will also have a positive impact on this approach, since more types of normal data will occur, thereby improving the performance of clustering and false positive detections will be reduced by increasing the number of currently underrepresented types.

Preconditions:

- To use this approach a large amount of good resolutioner, clean data is needed.
- The data should be of only one product, and it should not contain too diverse images.
- The anomaly types should be large and distinguishable from the normal ones.

Advantages:

- It can be easily integrated into workflows, with a trained object detection model.
- It can detect those cases, in which the object detection model is assigned to a given product but has some major deviation that suggests they might need to be handled differently.
- It finds large anomalies fast and with high accuracy.

- The cases when two separately already known things appear together, and items, that only differ from the normal in size or number could be detected as anomalies.

Limitations:

- Do not work well for smaller anomalies or new products.
- Require a re-clustering and a new parameter tuning, after many new images are obtained.
- Every object type requires a separate model and a sufficiently large dataset.

5.3 Anomaly detection with Segment-Any-Anomaly+

To run Segment Any Anomaly+, the implementation was used from the original article [43]. For each product, a separate model was used, with its own prompt. As this method does not require training, its successful use entirely depends on the construction of the prompts. These should summarise the most significant features of the already known or expected domain-specific anomalies for every product. Hence, its limitation is that it mainly focuses on the known anomalies. However, with the continuous monitoring of the workflow, the list of possible anomalies can be expanded, so the system can be prepared for the most possible errors. The characteristics of the normal data can only be described with one term. This may be enough for general objects from benchmark datasets (e. g. tile, wood, screw, hazelnut), but not for complex or special products, which the large language models, may not be capable of properly recognising. For example, after the word "reel" it may not recognise the exact object in this dataset and the more abstract categories (e.g. PCB in a tinfoil package, stacked boxes) were even harder to describe with a single term. Consequently, the model occasionally gave false positive detections for the normal products, that it did not recognise. This could be resolved by upgrading the model to include one or two normal images as input. Currently, the authors of the SAA+ model are working on an enhancement to achieve this, using few-shot techniques. By using this technique, the model could get implicit information about the characteristics of the normal product.

The detected anomaly masks were highly dependent on the defined size of the anomalies, if too large value was determined, it did not detect smaller anomalies well. Therefore, the anomalies in the datasets were separated based on their size into two groups, and a different model was used on them. One aimed to find the smaller anomalies and the other for the bigger ones. Their evaluation is also done separately because they

achieved different results on the normal data, which influences the value of the threshold. In general, this method was less successful for larger anomalies, because when the defined size could be larger than the size of the whole product, the model gave too much false positive detection as it considered the entire product as an anomaly (because it did not recognise it well). The defined prompt for the small reels was: (the rest are detailed in Appendix C):

```
Manual prompts:'reels_small': [
['package material on top. ', 'reel disc'], ['hands.', 'reel disc'],
['narrow rubber band around it.', 'reel disc'], ['fingers. ', 'reel disc'],
['paper package on top.', 'reel disc'], ['gloves. ', 'reel disc'],
['skin. arm.', 'reel disc']],
Property_prompts = {'reels': 'the image of reels have 1 dissimilar
reel_disc, with a maximum of 3 anomalies. The anomaly would not exceed 0.3
object area.'}
```

In many cases, the model was successful in detecting the desired anomalies and the detected anomaly masks were almost perfect. Examples of correct detections are visualized in Figure 5.9. The strength of the anomaly score is indicated by the colour, with red being the strongest, then yellow, green, and blue as the weakest.



5.9. Figure - Examples of the obtained segmentation masks from SAA+.

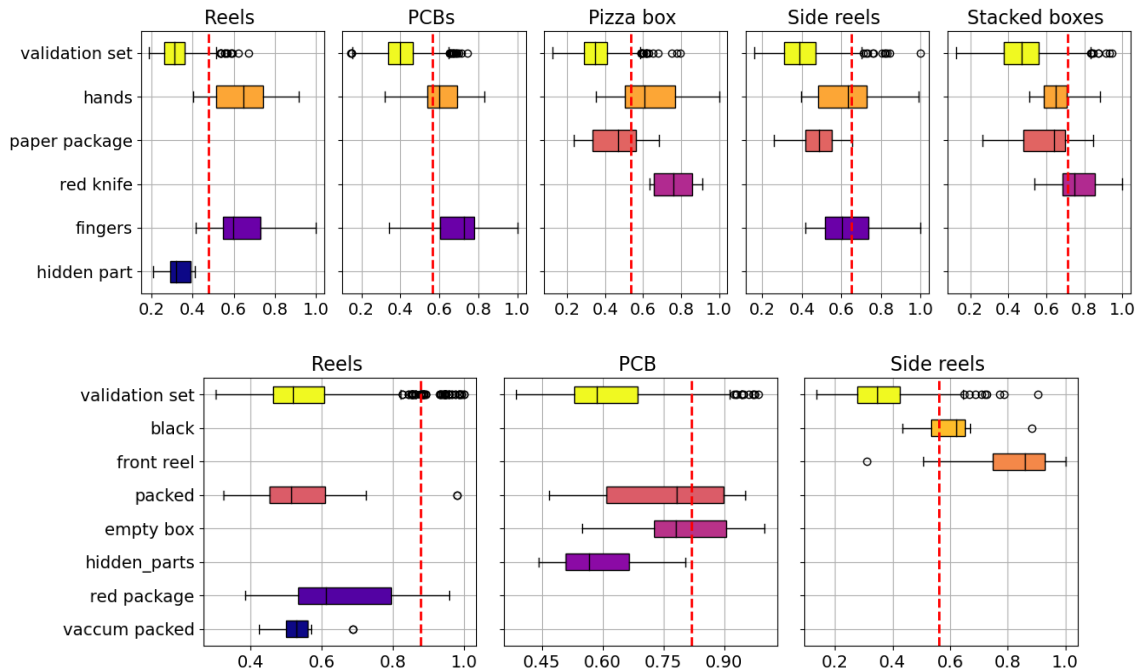
The evaluation results of the created segmentation masks, based on the evaluation methods implemented in SAA+, can be seen in Table 4. The accuracy of these results is not too high, as the results are greatly affected by the fact that the ground truth masks were obtained by hand and are not accurate at a pixel level. Furthermore, these metrics also take into account anomaly detections that have low confidence, which are later filtered out, when the final anomalous images are obtained. Thus, the real scores would

be even higher, but these metrics were perfect to measure the performance of different tests and to select the best prompt for each product.

4. Table - - Numerical results of the SAA+ model on the five different datasets.

	I_ROC	P_ROC	I_AP	P_AP	I_F1	P_F1
Reels (small)	90.43	72.60	77.78	37.70	71.76	46.69
Side reels (small)	88.33	71.44	61.93	26.91	56.54	80.0
Stacked boxes	81.12	74.02	41.48	7.07	46.61	16.17
Pizza boxes	88.18	79.63	61.01	28.99	58.59	38.83
PCBs (small)	91.48	85.16	65.03	39.78	62.96	45.41

The anomaly scores of the whole images were the value of the highest-scored anomaly mask. The results of these scores are presented in Figure 5.10., in the same ways as the autoencoder's. The threshold is higher here because this method does not require a training dataset, so later increasing it, will not yield better results. Therefore, a threshold (95%) was required, that is already acceptable for real-world use. The thresholds were defined based on the percentage of true positive results on the available images, without anomalies (called validation set).



5.10. Figure - Boxplots of the performance of the SAA+ on the five datasets (columns) with different anomaly types (rows). On the top for the smaller anomalies. The anomaly threshold (red dashed line) is set to correctly classify 95% of normal data.

The exact evaluation of the accuracy of the detected anomalies by the model, for different threshold values (75, 80, 90, 95) is shown in Table 5.

5. Table – The accuracies of the detected anomalies for each product and anomaly type in percentages, at different thresholds 75 (blue), 80 (green), 90 (yellow) and 95 (orange), based on the correctly identified images in the validation set.

	Reels (%)				PCBs (%)				Side reel (%)				Pizza box (%)				Stacked box (%)			
validation images	75	80	90	95	75	80	90	95	75	80	90	95	75	80	90	95	75	80	90	95
hands	100	100	94	92	83	83	80	56	83	66	60	49	95	90	82	69	88	75	62	25
paper package	-	-	-	-	-	-	-	-	59	36	23	5	62	61	38	38	62	62	50	19
red knife	-	-	-	-	-	-	-	-	-	-	-	-	100	100	100	100	94	94	84	66
fingers	100	100	94	82	90	90	83	77	83	78	56	44								
black	-	-	-	-	-	-	-	-	100	89	89	67								
front reel	-	-	-	-	-	-	-	-	97	97	97	91								
packed	33	22	11	11	62	54	54	47												
empty box	-	-	-	-	88	85	50	38												
hidden parts	36	36	0	0	23	17	7	0												
red packed	50	40	30	20																
vacuum packed	12	12	0	0																

It can be seen from Table 5, that this model achieved very diverse results on different products and anomaly types. Overall, it can be concluded that it gave exceptionally high results in cases where both the anomaly type and the product type could be well defined using prompts (e.g. fingers and hands for reels, front reels for side reels, or red knife for the pizza boxes). In cases where the model did not understand correctly the characteristics of the anomaly or product in question, very poor results were obtained. (e.g. vacuum package for reels, paper package for pizza box, and stacked box). For the remaining groups, the results were intermediate, suggesting that there were several variants within the anomaly types, some of which were correctly detected, but others were not. All in all, the model is useful for detecting some pre-defined anomalies, but it is not sufficient on its own to find all anomalies.

Preconditions:

- The anomaly types must be well describable and explainable.

Advantages:

- Do not need training images. Easy to apply for new products.

- Domain knowledge can be incorporated. The attributes of the anomalies can be precisely defined, making them more easily detectable.
- For some known anomaly types, the results are better than the other methods.
- The pixel-based segmentation masks of the anomalies can also be obtained.

Limitations:

- Function worse for unexpected anomalies, that were not added as domain knowledge.
- Sensitive for the parameters, i.e., the size of the anomaly.
- Cannot detect the absence of something.
- The model has trouble figuring out what the original products look like, in the case of complex items, which may not be known by the foundation models.
- The model cannot be told exactly what the normal data looks like, its characteristics are only determined by the prompt. (Using few-shot learning would be a solution for this, however, this is not yet implemented in SAA+.)
- It is slower than the previous approaches, as many foundation models are used.

6 Discussion

All three approaches performed well for some anomaly cases. Based on the previous results, in Table 5 the most appropriate method is defined for each anomaly type. These are not final decisions, because the results of the different solutions are not yet fully comparable, as the autoencoder and clustering results will improve later, due to the collection of a larger dataset. However, general conclusions and trends can already be drawn from the current results and the table has been filled based on these. The SAA+ solution should be used where current results show high accuracy. The clustering-based solution has provided a good solution wherever it was used. For those types where SAA+ did not provide a good solution and clustering cannot be used, the autoencoder-based solution should be used.

5. Table – The most suitable methods for the different kinds of anomalies, in the different datasets.

	Reels	PCB	Side reels	Pizza box	Stacked box
hands	SAA+	SAA+	SAA+ / autoencoder	SAA+	autoencoder
paper package	-	-	autoencoder	autoencoder	autoencoder
red knife	-	-	-	SAA+ / autoencoder	SAA+ / autoencoder
fingers	SAA+	SAA+	autoencoder		
black	-	-	clustering		
front reel	-	-	clustering / SAA+		
packed	autoencoder / clustering	autoencoder / clustering			
empty box	-	clustering			
hidden parts	autoencoder	autoencoder			
red packed	autoencoder / clustering				
vacuum packed	clustering				

All three methods worked based on quite different approaches and therefore, had different advantages and limitations, which were detailed after each one. Among them, if the best solution must be chosen, the autoencoder solution has the highest potential, as it has been able to detect the most types of anomalies. Also, by increasing the dataset, it can be made much more accurate than the published results. The clustering approach is appropriate as long as there is no need to find smaller, more subtle anomalies. For new

industrial processes and products without a proper dataset only the Segment Any Anomaly+ model can be used.

However, the use case is not limited to the selection of only one method. Since each model works in different ways and hence excels in different aspects, it could be valuable to combine their strengths to make more reliable, robust, and accurate predictions for each anomaly type and reduce variance and bias. This can also increase the detection time, so it should be used with caution. For their combination, there are many possible solutions, such as ensemble models, stacking, and hybrid modelling. A simple ensemble-based approach could be to perform predictions with all methods for all the images, and one should be only considered anomalous if one of the methods has a high anomaly score or at least two of them predict a medium score. As more complex approaches, meta-models or simpler ML-based models could be optimized to select the best decision, based on the three models. The first step towards this will be to collect a much bigger dataset for better autoencoder and clustering results.

The research of finding suitable anomaly detection methods for the given use case has concluded successfully, with methods, which can be integrated into the industrial process.

7 Human-Centred Considerations

Nowadays, when it comes to artificial intelligence-based applications, there is a growing focus on human-centric approaches, especially in real-world settings, with direct human interaction. When it comes to using these solutions in a factory environment, they provide plenty of additional challenges, beyond the development of the appropriate AI models. These issues are addressed in this chapter, along four major dimensions: ethical, safety, security, and legal aspects. These considerations not only help in the operation of AI-based systems but also ensure a smooth, socially responsible transition for their use that is aligned with social values.

7.1 Ethics

In every application, the AI-based decisions should be transparent, to build trust among workers and enable them to better understand the system. The autoencoder and SAA+ methods give information about the exact pixels that indicate anomalies, thereby explaining the decisions. The clustering approach also provides some level of explainability, as we can analyse the resulting clusters.

The automatization of a workplace can have a large social impact, as the workplace will be transformed, and the employees will need retraining programs as different types of human support will be needed. They need to be provided with adequate information about the changes and it must be ensured, that they are not negatively affected by the transition. The AI-based algorithms should be free from bias, which can occur due to mishandled training data or flawed design. Bias in anomaly detection could lead to unequal attention to different parts of the production process. The method could become discriminative against specific brands or products, which could give a misleading impression of them. For example, if fewer products were received from a given supplier, when the dataset was collected, as a result, the model might give false positive detections more frequently for their items in production. This can mistakenly trigger negative feelings towards the supplier or cause possible conflicts, which can implicitly affect the employees working there.

7.2 Safety

When an anomaly detection system is deployed in a real environment, it should be reliable and minimize inaccurate detections as they could lead to accidents. In case of an anomaly, there should be straightforward rules for a safe intervention and possible system shutdowns. To ensure maximum safety, the Unpacking Machine will operate in a closed, human-inaccessible environment, as sharp, narrow, or moving parts may make its interior unsuitable for human work. The problematic products will be transferred to a safe external location where the employees can handle them, or if it is necessary, the machine can be accessed, but only after a complete shutdown.

7.3 Legal

AI-based applications must comply with the industry-specific regulations, the GDPR for the protection of private images (especially those, that include potentially recognisable parts of people, which are also carefully excluded from this thesis), and the AI Act [44], to align with the human-centred considerations. The used AI solutions in this research were open source, which was further improved and combined with the novel dataset.

7.4 Security

From the perspective of security, it is a strong protection, that the solutions use only the company's data from a trusted source and do not use any online datasets or pre-trained weights for the trainings. Except the SAA+-based solution, which is based on foundation models. A potential vulnerability could be, if suppliers are packaging products in some special way, for example, to deliberately increase the number of incorrect anomaly detections, thereby slowing down the unpacking process. However, by using the combination of the three proposed models, and making a joint decision, this could be effectively mitigated, because it is much more difficult to confuse more different models with one adversarial input. These potential cases can also be identified by the regular analysis of false positive detections.

8 Future work

In future works, an approach could be used, which focuses on the individual products instead of the whole open boxes. The removal of the box and the unnecessary adjacent items can help the model to focus more on the specific items. The methods had trouble with the small white labels on the products, as their location and appearance varied, so they could be rightly considered as an abnormality. As a workaround, with the help of object detection, these labels could also be identified, and their areas could be masked out during evaluation to not affect the results. This could decrease the number of false positive detections. Furthermore, as discussed before, it would be beneficial to explore the possibilities of using the three models together.

9 Summary

Industrial automation is becoming a more and more popular method nowadays, driven by the continuous development of various AI solutions. For their appropriate use in a real environment, these complex systems must be prepared for any potential problem. Anomaly detection can provide a solution for identifying these unexpected cases so that they can be diagnosed and resolved in real-time. Consequently, this research explored deep learning-based anomaly detection techniques to be integrated into the automation processes of a company's existing workflow. It was identified between which steps of the process it should be incorporated, and what should be its preconditions, inputs, and outputs. Three different approaches were proposed, all with comprehensive evaluation. The limitation of the autoencoder and clustering-based approaches was that they require a very large dataset. The de-noising autoencoder-based, unsupervised method provided the most appropriate solution, which could detect a very large variety of anomalies. The second was a semi-supervised, clustering-based solution, which was able to detect the major anomalies with high probability but struggled to find the smaller ones. The third approach leveraged a novel model, called Segment Any Anomaly+. This model does not require a training set, so it can be used for completely new processes immediately. It was not able to detect all anomalies, instead, it mainly focused on known anomalies, that can be described well in an appropriate prompt. All approaches were capable of providing great results, but each stood out in different aspects with different advantages and limitations. It would be beneficial to use them together, as they could complement each other to provide a more robust and reliable solution. These developed solutions are under patent registration, and they will make a major contribution to the company's automatization system.

References

- [1] Mohd Aiman Kamarul Bahrin , Mohd Fauzi Othman, Nor Hayati Nor Azli, Muhamad Farihin Talib, „Industry 4.0: a review on industrial automation and robotic” (2016), Jurnal Teknologi (Sciences & Engineering), 78(6-13), DOI: <https://doi.org/10.11113/jt.v78.9285> (Accessed: Oct 2024)
- [2] EDGEWORTH, F. Y. 1887. On discordant observations. Philosoph. Magazin. 23, 5, 364–375. (Accessed: Apr 2024)
- [3] Chandola, Varun, Arindam Banerjee, Vipin Kumar, “Anomaly detection: A survey.”, Doi: 10.1145/1541880.1541882 (Accessed: Apr 2024)
- [4] Chong Zhou and Randy C. Paffenroth. 2017. „Anomaly Detection with Robust Deep Autoencoders”, <https://doi.org/10.1145/3097983.3098052>, (Accessed: Apr. 2024.)
- [5] Jonathon Shlens, „A Tutorial on Principal Component Analysis”, arXiv:1404.1100, (Accessed: Apr. 2024.)
- [6] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, John Platt (1999). “Support Vector Method for Novelty Detection”. NIPS. 12. 582-588. (Accessed Okt 2024)
- [7] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution grey-scale and rotation invariant texture classification with local binary patterns," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987, (2002), doi: 10.1109/TPAMI.2002.1017623., (Accessed: Okt 2024)
- [8] Te-won Lee, Michael Lewicki „The Generalized Gaussian Mixture Model Using ICA. Int. workshop on ICA”. (2000), https://www.researchgate.net/publication/2620375_The_Generalized_Gaussian_Mixture_Model_Using_ICA, (Accessed: Okt 2024).
- [9] Dan Hendrycks, Mantas Mazeika, Thomas Dietterich, „Deep Anomaly Detection with Outlier Exposure“, 2019, <https://openreview.net/forum?id=HyxCxhRcY7>, (Accessed: Okt. 2024)
- [10] Ahad Alloqmani, Yoosef B., Asif Khan, Fawaz Alsolami, “Deep Learning based Anomaly Detection in Images: Insights, Challenges and Recommendations.” International Journal of Advanced Computer Science and Applications. 12. 10.14569/IJACSA.2021.0120428. (Accessed: Apr 2024)
- [11] Rosenblatt, Frank, “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” (1958) Psychological Review, 65(6), 386–408. <https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf>, (Accessed: Nov 2024)

- [12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (June 2017), 84–90. <https://doi.org/10.1145/3065386> (Accessed: Nov 2024)
- [13] OpenAI, Josh Achiam et al., “GPT-4 Technical Report” , [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) (Accessed Okt 2024)
- [14] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009).“ *ImageNet: A large-scale hierarchical image database.*” In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 248-255). IEEE. (Accessed: Apr 2024)
- [15] Laith Alzubaidi , Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie and Laith Farhan „Review of deep learning: concepts, CNN architectures, challenges, applications, future directions”. *J Big Data* 8, 53 (2021)., (Accessed: Apr 2024)
- [16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, „Attention Is All You Need”, Curran Associates, Inc., (2017) https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf, (Accessed: Apr. 2024.)
- [17] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, „You Only Look Once: Unified, Real-Time Object Detection”, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91. (Accessed.: Apr. 2024.)
- [18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, “SSD: Single Shot MultiBox Detector”, *Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science()*, vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2 (Accessed: Apr. 2024.)
- [19] Liu Haiying, Fengqian Sun, Jason Gu, and Lixia Deng. 2022. "SF-YOLOv5: A Lightweight Small Object Detection Algorithm Based on Improved Feature Fusion Mode" *Sensors* 22, no. 15: 5817. <https://doi.org/10.3390/s22155817>, (Accessed: Apr. 2024.)
- [20] Dehua Peng, Zhipeng Gui, Huayi Wu, „Interpreting the Curse of Dimensionality from Distance Concentration and Manifold Effect” (2023), [arXiv:2401.00422](https://arxiv.org/abs/2401.00422) (Accessed: Okt 2024)
- [21] Leland McInnes, John Healy, James Melville, „UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”, (2018) *Journal of Open Source Software*, 3(29), 861, <https://doi.org/10.21105/joss.00861>, (Accessed: Okt 2024)

- [22] Lilian Weng, „From Autoencoder to Beta-VAE”, Lil’Log, <https://lilianweng.github.io/posts/2018-08-12-vae/>, (Accessed: Apr. 2024.)
- [23] Jie Yang, Ruijie Xu, Zhiquan Qi, Yong Shi, “Visual Anomaly Detection for Images: A Survey”, arXiv:2109.13157, (Accessed: Apr. 2024.)
- [24] Hartigan, J. A., and M. A. Wong. “Algorithm AS 136: A K-Means Clustering Algorithm.” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, 1979, pp, <https://doi.org/10.2307/2346830>. (Accessed: Apr. 2024.)
- [25] Ester, M., Kriegl, H.-P., Sander, J., & Xu, X. (1996). „*A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*”. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)* (pp. 226–231), <https://cdn.aaai.org/KDD/1996/KDD96-037.pdf> (Accessed: Oct 2024)
- [26] Ricardo Campello, Davoud Moulavi, Joerg Sander. “Density-Based Clustering Based on Hierarchical Density Estimates”. . (2013) 7819. 160-172. doi: 10.1007/978-3-642-37456-2_14., (Accessed: Apr. 2024.)
- [27] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861. (Accessed: Apr. 2024.)
- [28] Xiangpeng Fan, Zhibin Guan (2023). “VGNet: A Lightweight Intelligent Learning Method for Corn Diseases Recognition”. *Agriculture*. 13. 1606. 10.3390/agriculture13081606. (Accessed: Apr. 2024.)
- [29] Mayu Sakurada and Takehisa Yairi, “Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction.” (2014), <https://doi.org/10.1145/2689746.2689747>, (Accessed: Apr 2024)
- [30] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, Nassir Navab, Deep Autoencoding Models for Unsupervised Anomaly Segmentation in Brain MR Images,(2018) [arXiv:1804.04488v1](https://arxiv.org/abs/1804.04488v1) (Accessed: Apr 2024)
- [31] Xiaoran Chen, Ender Konukoglu, “Unsupervised Detection of Lesions in Brain MRI using constrained adversarial auto-encoders” (2018), *Medical Imaging with Deep Learning*, <https://openreview.net/forum?id=H1nGLZ2oG>, (Accessed: Apr 2024)
- [32] Polykarpos Thomadakis, Angelos Angelopoulos, Gagik Gavalian, Nikos Chrisochoides, “De-noising drift chambers in CLAS12 using convolutional auto encoders”, *Computer Physics Communications Volume 271*, 2022, 108201, ISSN 0010-4655, <https://doi.org/10.1016/j.cpc.2021.108201>., (Accessed: Apr 2024)
- [33] Pramuditha Perera, Vishal M. Patel. “*Learning Deep Features for One-Class Classification*”, (2019), *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5450-5463, doi: 10.1109/TIP.2019.2917862. (Accessed: Okt. 2024)

- [34] Chen, Jinghui & Sathe, Saket & Aggarwal, Charu & Turaga, Surya Deepak. "Outlier Detection with Autoencoder Ensembles." (2017) 10.1137/1.9781611974973.11., (Accessed: Okt. 2024.)
- [35] Umap-learn, UMAP: "Uniform Manifold Approximation and Projection for Dimension Reduction", <https://umap-learn.readthedocs.io/en/latest/clustering.html>, (Accessed: Okt. 2024.)
- [36] Yunkang Cao, Xiaohao Xu, Chen Sun, Yuqi Cheng, Zongwei Du, Liang Gao, Weiming Shen, "Segment Any Anomaly without Training via Hybrid Prompt Regularization" 2023, arXiv:2305.10724 (Accessed: Apr. 2024.)
- [37] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang "Grounding dino: Marrying dino with grounded pre-training for open-set object detection", (2023), <https://openreview.net/forum?id=DS5qRs0tQz>, (Accessed: Apr. 2024.)
- [38] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, Ross Girshick, "Segment anything". Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages: 4015-4026 arXiv:2304.02643, (Accessed: Apr. 2024.)
- [39] Z. Wang, E. P. Simoncelli and A. C. Bovik, "Multiscale structural similarity for image quality assessment," *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Pacific Grove, CA, USA, 2003, pp. 1398-1402 Vol.2, doi: 10.1109/ACSSC.2003.1292216., . (Accessed: Apr. 2024.)
- [40] Pytorch, <https://pytorch.org/>, (Accessed: Apr. 2024.)
- [41] Scit-learn, <https://scikit-learn.org/stable/>, (Accessed: Apr. 2024.)
- [42] Buslaev A, Iglovikov VI, Khvedchenya E, Parinov A, Druzhinin M, Kalinin AA. „Albumentations: Fast and Flexible Image Augmentations.” *Information*. 2020; 11(2):125. <https://doi.org/10.3390/info11020125>, (Accessed: Apr. 2024.)
- [43] Github, "Segment Any Anomaly Official Implementation ", <https://github.com/caoyunkang/Segment-Any-Anomaly> (Accessed: Okt. 2024.)
- [44] European Parliament, EU AI Act: first regulation on artificial intelligence, <https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence>, (Accessed: Okt. 2024.)

Appendix

Appendix A:

- Hyperparameters of the autoencoder methods (for all products):

Hyperparameters	
learning rate	0.001
batch size	32
num of epochs	500
num of folds	5
Encoder model	ResNET50
criterion	SSIM
optimizer	Adam
image size	256x256
Eval method	MS-SSIM

- The added noises and transformations:

PCBs and Reels:

- CoarseDropout(p=0.7, num_holes_range=(2, 6), hole_height_range=(30, 30), hole_width_range=(30, 30))
- OneOf([GaussNoise(p=0.5, var_limit=(10.0, 100.0), per_channel=False, mean=0.0, noise_scale_factor=1.0), GlassBlur(p=0.3, sigma=0.0005, max_delta=1, iterations=1, mode='fast'), RandomGravel(p=0.3, gravel_roi=(0.0, 0.0, 1.0, 1.0), number_of_patches=4),] p=0.7),
- PixelDropout(p=0.5, dropout_prob=0.01, per_channel=False, drop_value=None, mask_drop_value=None)
- OneOf([MotionBlur(p=0.2, blur_limit=(3, 3), allow_shifted=True), Sharpen(p=0.5, alpha=(0.2, 0.5), lightness=(0.5, 1.0)), Blur(p=0.2, blur_limit=(3, 3)),], p=0.4)
- OneOf([OpticalDistortion(p=0.3, distort_limit=(-0.05, 0.05), shift_limit=(-0.05, 0.05), interpolation=1), GridDistortion(p=0.1, num_steps=5, distort_limit=(-0.3, 0.3), interpolation=1], p=0.3)

Pizza boxes:

- CoarseDropout(p=0.7, num_holes_range=(2, 6), hole_height_range=(30, 30), hole_width_range=(30, 30))
- OneOf([GaussNoise(p=0.5, var_limit=(10.0, 100.0), per_channel=False, mean=0.0, noise_scale_factor=1.0), GlassBlur(p=0.3, sigma=0.0005, max_delta=1, iterations=1, mode='fast'), RandomGravel(p=0.3, gravel_roi=(0.0, 0.0, 1.0, 1.0), number_of_patches=4),], p=0.7)
- PixelDropout(p=0.5, dropout_prob=0.01, per_channel=False, drop_value=None, mask_drop_value=None)

Side Reels and Stacked boxes:

- CoarseDropout(p=0.7, num_holes_range=(2, 6), hole_height_range=(30, 30), hole_width_range=(30, 30))
- OneOf([GaussNoise(p=0.5, var_limit=(10.0, 100.0), per_channel=False, mean=0.0, noise_scale_factor=1.0), GlassBlur(p=0.3, sigma=0.0005, max_delta=1, iterations=1, mode='fast'), RandomGravel(p=0.3, gravel_roi=(0.0, 0.0, 1.0, 1.0), number_of_patches=4),], p=0.7)
- PixelDropout(p=0.5, dropout_prob=0.01, per_channel=False, drop_value=None, mask_drop_value=None)

Appendix B: Parameters of the clustering method

	Reels	Side Reels	PCBs
YOLO layers	before the Neck	before the Neck	before the Neck
Dim red type	PCA & UMAP	PCA & UMAP	PCA & UMAP
umap params:			
- metrics	correlation	correlation	correlation
- set_opt_mix_ratio	1.0	0.9	0.5
- n_components	30	30	30
- n_neighbors	50	50	30
- min_dist	0.2	0.1	0.0
clustering_method	HDBSCAN	HDBSCAN	HDBSCAN
HDBSCAN params:			
- min_cluster_size	10	8	10
- cluster_selection_epsilon	0.1	0.1	0.08
- min_samples	3	3	5
- alpha	1.0	1.0	1.0
- metrics	l2	l2	l2
- cluster_selection_method	eom	eom	eom
scaling_type	MinMax	MinMax	robust
image_size	640	640	640

Appendix C: Prompts for Segment Any Anomaly+

```
manual_prompts = {
  'stacked': [
    ['Crumpled paper package on top. ', 'stacked boxes'],
    ['package material on top.', 'stacked boxes'],
    ['paper sheet on top.', 'stacked boxes'], ['arm.', 'stacked boxes'],
    ['red knife.', 'box cutter.', 'stacked boxes'], ['hands.', 'stacked boxes'],
    ['anything other than boxes next to each other.', 'stacked boxes'],
    ['fingers. ', 'stacked boxes'], ['opened box.', 'stacked boxes']
  ],
  'side_reels_small': [
    ['paper package on top. plastic package on top.', 'stacked reels'],
    ['red knife', 'red box cutter.', 'stacked reels'],
    ['black reels.', 'stacked reels'],
    ['circular object. a reel disc lays on top.', 'stacked reels'],
    ['hands. ', 'stacked reels'], ['fingers.', 'stacked reels'],
    ['glove. ', 'stacked reels'], ['skin. arm.', 'stacked reels'],
  ],
}
```

```

'side_reels_big': [
    ['paper package on top.', 'stacked reels'], ['black reels.', 'stacked reels'],
    ['circular.', 'stacked reels'], ['a reel disc lays on top.', 'stacked reels'],
    ['plastic package on top.', 'stacked reels'],
],

'reels_small': [
    ['package material on top.', 'reel disc'],
    ['paper package on top.', 'reel disc'], ['skin. arm.', 'reel disc'],
    ['narrow rubber band around it.', 'reel disc'], ['hands.', 'reel disc'],
    ['fingers.', 'reel disc'], ['gloves.', 'reel disc'],
],

'reels_big': [
    ['paper package on top.', 'reel disc'], ['red package.', 'reel disc'],
    ['not circular.', 'reel disc'], ['vacuumed packed.', 'reel disc'],
    ['in a metalized foil package.', 'reel disc'],
],

'pizza_boxes': [
    ['plastic package on top.', 'pizza box shaped box'],
    ['paper sheet on top.', 'pizza box shaped box'],
    ['red knife.', 'pizza box shaped box'], ['hands.', 'pizza box shaped box'],
    ['fingers.', 'pizza box shaped box'], ['gloves.', 'pizza box shaped box'],
    ['skin. arm.', 'pizza box shaped box'],
    ['paper package next to.', 'pizza box shaped box'],
],

'PCBs_small': [
    ['fingers.', 'metalized foil bag'], ['hands.', 'metalized foil bag'],
    ['gloves.', 'metalized foil bag'], ['skin. arm.', 'metalized foil bag'],
],

'PCBs_big': [
    ['polystyrene material on top', 'item in a metalized foil bag'],
    ['paper package on top.', 'item in a metalized foil bag'],
    ['plastic package on top.', 'item in a metalized foil bag'],
    ['empty box. the foil bag is missing', 'item in a metalized foil bag'],
],
}

property_prompts = {
    'stacked': 'the image of boxes have 1 dissimilar stacked_boxes, with a maximum of 3 anomaly. The anomaly would not exceed 0.6 object area. ',
    'side_reels_small': 'the image of reels have 1 dissimilar stacked_reels, with a maximum of 2 anomaly. The anomaly would not exceed 0.35 object area. ',
    'side_reels_big': 'the image of reels have 1 dissimilar stacked_reels, with a maximum of 2 anomaly. The anomaly would not exceed 0.8 object area. ',
    'reels_big': 'the image of reels have 1 dissimilar reel, with a maximum of 2 anomaly. The anomaly would not exceed 1.0 object area. ',
    'pizza_boxes': 'the image of box have 1 dissimilar box, with a maximum of 2 anomaly. The anomaly would not exceed 0.5 object area. ',
    'PCBs_small': 'the image of PCB have 1 dissimilar metalized_foil_bag, with a maximum of 3 anomaly. The anomaly would not exceed 0.4 object area. ',
    'PCBs_big': 'the image of PCB have 1 dissimilar item_in_a_metalized_foil_bag, with a maximum of 2 anomaly. The anomaly would not exceed 1.0 object area. ',
    'empty_box': 'the image of box have 1 dissimilar box, with a maximum of 3 anomaly. The anomaly would not exceed 0.4 object area. ',
}

```

Self-assessment for Human-Centred Artificial Intelligence (AI) Master's

All relevant HCAI aspects are detailed in Chapter 7, according to the ethical, safety, security and legal considerations.

Activity	Y/N	Documents to be provided as appendix
<i>Does this activity involve the development, deployment and/or use of Artificial Intelligence-based systems?</i>	Y	Yes, the research involves the development and deployment of Artificial Intelligence-based systems. It focuses on deep learning-based image anomaly detection methods, that can be integrated into an existing industrial automatization process.
<i>Could the AI based system/technique potentially stigmatise or discriminate against people (e.g. based on sex, race, ethnic or social origin, age, genetic features, disability, sexual orientation, language, religion or belief, membership to a political group, or membership to a national minority)?</i>	Y	In general, such cases cannot happen, as the research is entirely focused on industrial use and the classification decision is only made for products, not people. However, as detailed in Chapter 7, it can have a small explicit impact on the employees of a given supplier, if the model is discriminatory towards a supplier's products, and thus the supplier is perceived negatively, so it is important to use the same number of images from all suppliers.
<i>Does the AI system/technique interact, replace or influence human decision-making processes (e.g. issues affecting human life, health, well-being or human rights, or economic, social or political decisions)?</i>	Y	Yes, the industrial process in which the anomaly detection will be integrated, is related to industrial employees. This requires, everyone to be well-informed about the exact operation of the machine and the rules for its use. The methods can also give explanations for their decisions. In addition, to ensure maximum protection of employees, the Unpacking Machine will work in an area isolated from people and will only be accessible after the machine has been completely shut down. This AI-based solution will not replace or influence human decision-making processes.
<i>Does the AI system/technique have the potential to lead to negative social (e.g. on democracy, media, labour market, freedoms, educational choices, mass surveillance) and/or environmental impacts either through intended applications or plausible alternative uses?</i>	Y	Yes, because the AI-based model will be part of an automatization process in a company's factory. Currently, employees are doing manually the tasks that the AI-based solution will replace. This will have a high social impact. For the transition of the workplace, employees must be provided with appropriate information about the changes and they need retraining programs. Overall, it will have a positive impact by allowing workers to do a different, easier and less monotonous job. But the changeover must be done in a way that they are well informed and do not suffer negative consequences.
<i>Does this activity involve the use of AI in a weapon system?</i>	N	No, the research is entirely focused on industrial automatization.
If YES	<i>Is it possible to establish which specific function/functions are automated/autonomous in the weapon system?</i>	
	<i>If the weapon system has AI-enabled functions, could these functions render the weapon system indiscriminate?</i>	
	<i>Does the design include the possibility of an autonomous mode for selfprotection? If yes, can the system reliably distinguish between targets (threats) and non-targets?</i>	
<i>Does the AI to be developed/used in the project raise any other ethical issues not covered by the questions above (e.g., subliminal, covert or deceptive AI, AI that is used to stimulate addictive behaviours, lifelike humanoid robots, etc.)?</i>	N	No, all relevant aspects had been covered.