ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Machine Learning For Computer Vision

ANOMALY DETECTION FOR LINE CLEARANCE IN INDUSTRIAL SYSTEMS

CANDIDATE
Lorenzo Galfano

SUPERVISOR
Prof. Samuele Salti

CO-SUPERVISOR
Lucente Stefano

Academic year 2023-2024

Session 4th

Contents

List of Figures

List of Tables

Abstract

With the rapid advancement of technology, industries are increasingly seeking solutions that operate efficiently, autonomously, and reliably. Line clearance, a critical procedure designed to ensure work areas are free of residual products or contaminants, is no exception, as it demands automation to enhance productivity and reduce human error. This project explores the application of anomaly detection techniques for automating line clearance across five distinct views of a machine, leveraging two neural network architectures to evaluate their feasibility and performance in operational scenarios.

The study employs advanced techniques such as masking, image registration, and data augmentation to enhance the robustness and precision of the anomaly detection models. Comparative analyses focus on assessing detection accuracy across all views and the overall system reliability in industrial workflows. This research aims to establish a robust framework for deploying automated anomaly detection systems, contributing to safer, more efficient manufacturing processes. Through a comprehensive evaluation, the study provides insights into the applicability and comparative strengths of the two networks for automating line clearance and highlights their potential for broader industrial adoption.

1 Introduction

Line clearance is an emerging procedure that is rapidly gaining acceptance in the industrial sector. It is designed to ensure that equipment and work areas are devoid of residual products, documents, and materials between production batches. This process is divided into three distinct phases:

- **Clearing**: The physical removal of all materials from the previous batch, including unused parts, labels, and packaging not required for the subsequent process.
- **Cleaning**: Disinfecting and drying all surfaces and equipment used in the previous process.
- **Checking**: A thorough inspection of the production line, conducted by the operator, before resuming production.

This pipeline is crucial, if not essential, for processes within the Life Sciences sector, where pharmaceutical products must be produced in a sterile environment free from external contamination. Nevertheless, machines are not infallible and may occasionally fail to correctly process a batch, potentially resulting in residual product remaining within the equipment. Such residual material can pose a risk of contamination to subsequent batches and needs to be cleared.

In this case study, two potential risk zones within the machine where the product could become lodged were identified and evaluated. Additionally, five possible camera placements were selected to capture images for anomaly detection algorithm implementation.

As industrial processes become increasingly automated, the checking phase is expected to be performed by an algorithm, specifically an anomaly detection system. This approach is anticipated to reduce the likelihood of errors that may occur with manual inspection and offer greater speed and accuracy.

Surface anomaly detection is a particularly challenging task due to the fact that anomalies typically occupy only a small fraction of the image pixels and often closely resemble the normal data distribution found in the training set. Given the rarity of images containing anomalies and the time-consuming nature of labeling them, datasets used in this domain are often highly imbalanced, necessitating the use of anomaly-free images for training. This situation typically leads to an unsupervised learning approach.

Existing anomaly detection algorithms can be broadly categorized into three main types:

- **Reconstruction-based methods**: These methods detect anomalies by comparing the reconstructed image to the original image. The underlying assumption is that the model, trained solely on normal data, will struggle to reconstruct anomalous features. However, modern convolutional neural network (CNN) architectures, being highly adaptable, may inadvertently learn to reconstruct anomalies as well, thereby reducing the efficacy of this approach. Diffusion models, which also fall under the category of reconstruction-based methods, have shown promise in overcoming the limitations typically faced by other approaches. These models have demonstrated improved performance, offering better anomaly localization and detection results compared to traditional reconstruction methods.

- **Embedding-based methods**: These approaches utilize pre-trained networks to extract and compress image features into a compact latent space, aiming to separate the feature clusters of normal data from those of anomalies. Nonetheless, this method may suffer from several limitations, such as under-representation of anomaly features or a failure to adequately distinguish between normal and anomalous data due to insufficient anomaly representation in the feature space.

- **Synthesis-based methods**: In this approach, anomalies are synthetically generated from normal samples to introduce anomaly-specific information into the detection model, thereby enhancing its performance. An example of this would be the use of Perlin noise to simulate anomalies.

This experiment involves two different architectures, the first one is DRÆM [18], which follows a synthesis-based approach. Anomalies are synthetically generated using the Describable Textures Dataset (DTD) [6] and Perlin noise, as detailed in Section 3. The second architecture is DDAD (Denoising Diffusion Anomaly Detection) [13], a reconstruction-based method that leverages denoising diffusion probabilistic models (DDPM) for anomaly detection by learning the normal data distribution and reconstructing clean outputs from corrupted images. A unique aspect of this model is its integration of a feature extractor such as wide resnets[10] [17], which helps enhance performance by capturing detailed features. To ensure the pretrained feature extractor adapts to the specific anomaly detection task, an unsupervised domain adaptation technique is applied, effectively shifting the domain to better address the problem at hand.

Unlike other commonly used datasets, each view in our custom dataset presents unique challenges, such as limited camera placement options to avoid obstructing the machinery, cameras occasionally being out of focus, moving machine parts, and difficult-to-learn regions. To address some of these challenges, masking is extensively applied during the network's evaluation. Consequently, performance metrics like AUROC, AUPRO, and AP are evaluated both with and without masking to determine whether masking provides any tangible benefit in improving the detection results.

2 Related Works

The tasks of anomaly detection and anomaly classification have gained significant attention due to their critical applications in domains like industrial monitoring, healthcare, and cybersecurity. Anomaly detection typically involves identifying data points that deviate significantly from the norm, often without prior knowledge of anomaly types, making it inherently unsupervised or semi-supervised. In contrast, anomaly classification assumes labeled datasets, enabling more granular categorization of detected anomalies. Both tasks are challenging due to the rarity of anomalous samples and their high variability, which necessitates the development of robust, generalizable models. Over time, various approaches have emerged to tackle these challenges, ranging from reconstruction-based methods to embedding-based and synthetic anomaly generation techniques.

Reconstruction-based methods are widely used in anomaly detection, leveraging the assumption that models can accurately reconstruct normal data but struggle with anomalous regions. Early approaches often utilized autoencoders and generative adversarial networks (GANs)[9]. However, these methods faced challenges in distinguishing between normal and abnormal samples, as neural networks tend to generalize well, leading to good reconstructions of both normal and anomalous data. To address this, more advanced techniques such as Variational Autoencoders (VAEs)[11] and Latent Space Autoregression[1] have been proposed, improving the detection of outliers by focusing on variations in latent space. Methods like the one proposed by Bergmann et al.[4] introduced structural similarity measures in combination with autoencoders, offering improved results in defect segmentation tasks by emphasizing pixel-wise differences.

Embedding-based methods focus on feature extraction and anomaly detection in high-dimensional spaces. These methods rely on pre-trained networks that map data into a feature space, where anomalies are detected based

on distance metrics like K-nearest neighbors (KNN). Techniques such as SPADE[16] and PaDiM[7] extract features from normal samples and measure the anomaly score of test samples by evaluating their distance from known normal instances.

Synthetic-based methods focus on generating synthetic anomalies or augmenting datasets to improve the model's robustness. One notable technique is CutPaste[12], which uses self-supervised learning to train models by cutting out sections of an image and pasting them onto different locations to simulate anomalies. The combination of synthetic anomaly generation with robust embedding strategies has been crucial in addressing the scarcity of anomalous training data, particularly in industrial applications where real anomalies may be rare.

3 Method

3.1 Data Labelling

While anomaly detection typically does not require data labelling for the majority of the process when using unsupervised methods, labelling becomes valuable during the testing phase to generate evaluation metrics. Following data acquisition, the first step was labelling the anomaly images. This task was performed using an open-source web application called ”VGG Image Annotator” (VIA) [8]. The results were saved in a .csv file, which was subsequently used to create functional 2D masks, later passed as input for evaluation.

3.2 DRÆM

DRÆM is composed of two main components: a reconstructive sub-network and a discriminative sub-network. The reconstructive sub-network focuses on generating a reconstruction of the input image that excludes anomalies. For images containing anomalies, the reconstruction quality tends to degrade specifically in anomalous regions, making them easier to detect. The discriminative sub-network, meanwhile, takes as input the concatenation of the original and reconstructed images and produces refined anomaly segmentation maps. Typically, training is conducted on anomaly-free images, as real anomalous examples are often limited. To address this, DRÆM synthetically generates anomalous images just before they are passed to the discriminative sub-network, allowing for effective training without real anomalous data.

Reconstructive Sub-Network

The Reconstructive Sub-Network employs a typical AutoEncoder structure, aiming to recreate the original, anomaly-free version of the input image. By comparing the reconstructed image to the input, anomalies can be identified based on areas where the reconstruction diverges from the original. For this

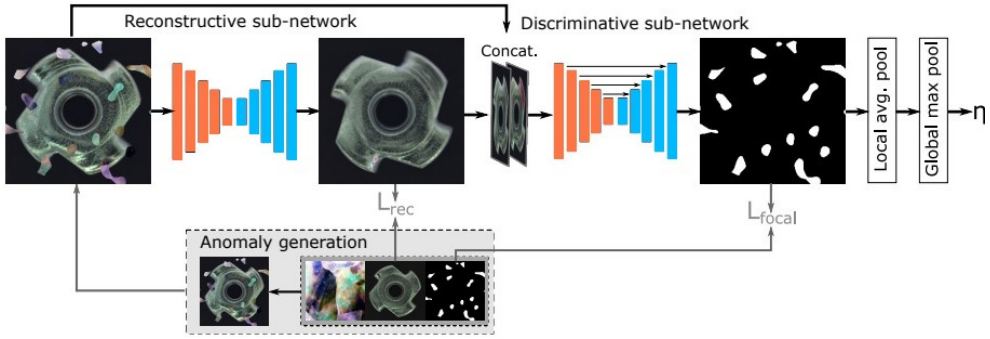


Figure 1: DRÆM architecture

type of task, SSIM (Structural Similarity Index Measure) loss is commonly used, as it assesses image similarity in a way that captures structural information.

SSIM is grounded in the concept that pixels exhibit strong inter-dependencies, particularly among those in close spatial proximity. These dependencies are significant as they contain structural details about the objects within the visual scene. SSIM calculates similarity between two images, I and I_r, through three key comparison metrics:

- $luminance(l) : l(I, I_r) = \frac{2\mu_I\mu_{I_r}+c1}{\mu_I^2+\mu_{I_r}^2+c1}$
- $contrast(c) : c(I, I_r) = \frac{2\sigma_I\sigma_{I_r}+c2}{\sigma_I^2+\sigma_{I_r}^2+c2}$
- $structure(s) : s(I, I_r) = \frac{\sigma_{II_r}+c3}{\sigma_I\sigma_{I_r}+c3}$

where μ_I and μ_{I_r} are the pixel sample mean of I and I_r respectively, σ_I^2 and $\sigma_{I_r}^2$ are the variances of I and I_r, σ_{II_r} is the covariance of I and I_r. $c_1 = (k_1L)^2$ and $c_2 = (k_2L)^2$ are two variables to stabilize the division with weak denominator, where L is the dynamic range of the pixel-wise values ($2^{bitsperpixel} - 1$ and $k_1 = 0.01$ and $k2 = 0.03$, also $c_3 = c_2/2$).

Finally SSIM can be computed as the weighted combination of these comparative measures:

$$SSIM(I, I_r) = l(I, I_r)^{\alpha} \times c(I, I_r)^{\beta} \times s(I, I_r)^{\gamma} = \frac{(2\mu_I \mu_{I_r} + C_1) + (2\sigma_{II_r} + C_2)}{(\mu_I^2 + \mu_{I_r}^2 + C_1)(\sigma_I^2 + \sigma_{I_r}^2 + C_2)} \tag{1}$$

This formula represents the global SSIM, however, for our purposes, we utilize a localized version of SSIM that calculates these components within a window centered on each pixel (i, j) and will be written as $SSIM(I, I_r)_{(i,j)}$.

The SSIM can then be transformed into a loss function as:

$$L_{SSIM}(I, I_r) = \frac{1}{N_p} \sum_{i=1}^{H} \sum_{j=1}^{W} (1 - SSIM(I, I_r)_{(i,j)}) \tag{2}$$

where H and W are the height and width of image I, respectively. N_p is equal to the number of pixels in I. I_r is the reconstructed image output by the network.

The network also incorporates the L2 norm to capture pixel-wise differences, supplementing the SSIM loss. SSIM can sometimes overlook minor pixel-level anomalies because of its broader focus on local structure, the L2 norm, however, directly measures differences at each pixel, providing precise, localized feedback on pixel-level deviations. Thus, the combination of L2 norm with SSIM provides a balance: L2 norm detects detailed pixel-wise deviations, while SSIM verifies these differences in the context of their structural neighborhoods, reducing the likelihood of false positives due to isolated pixel variations.

The final loss can be written as:

$$L_{rec}(I, I_r) = \lambda L_{SSIM}(I, I_r) + l_2(I, I_r) \tag{3}$$

where λ is a loss balancing hyper-parameter.

Discriminative Sub-Network

The Discriminative Sub-Network receives as input the concatenated channels of the original image I and the reconstructed image I_r generated by the Reconstructive Sub-Network. This network is designed to generate a precise anomaly map, M_o, which highlights regions in the input image where discrepancies with the reconstruction indicate potential anomalies. The output M_o is an anomaly score map, maintaining the same dimensions as the input image I. For this pixel-wise anomaly scoring task, a Cross-Entropy loss could serve as a basic loss function. However, using Cross-Entropy alone may be insufficient, as the generated synthetic anomalies can vary greatly in difficulty, and the goal is to make the network consistently responsive to challenging examples. To address this, Focal Loss is applied instead, which down-weights well-classified instances and focuses learning on harder examples. This adjustment helps improve the model's performance on difficult anomaly detection cases.

The Focal Loss can be expressed as:

$$L_{seg} = -\alpha_i \sum_{i=1}^{i=n} (1 - p_i)^{\gamma} * ln(p_i) \tag{4}$$

Making the final loss of the entire network as:

$$L(I, I_r, M_a, M) = L_{rec}(I, I_r) + L_{seg}(M_a, M) \tag{5}$$

Perlin Noise Generation

Autoencoders tend to over-generalize anomalies, while discriminative approaches often over-fit to synthetic anomalies, limiting their ability to generalize to real-world data. DRÆM addresses this by generating just-out-of-distribution appearances, allowing the network to learn an effective distance function by capturing deviations between normal and anomalous patterns.

To achieve this, DRÆM first generates a noise pattern using Perlin noise [2], which is converted into a binary anomaly map M_a by applying a random

threshold. Then, an anomaly texture image A, sourced from the DTD dataset, which contains textures unrelated to the input distribution, undergoes several random augmentations like posterization, sharpening, or changes in brightness and color to create variation.

This anomaly texture A is blended with the original image I, guided by the anomaly map M_a. This introduces anomalies that are unusual but not overly distant from normal data, helping the network sharpen its ability to distinguish anomalies. The process creates the augmented image I_a using the following equation:

$$I_a = \overline{M_a} \odot I + (1 - \beta)(M_a \odot I) + \beta(M_a \odot A) \tag{6}$$

where $\overline{M_a}$ is the inverse of M_a, $\odot$ represents element-wise multiplication, and β is a blending opacity parameter, uniformly sampled from $\beta \in [0.1, 1.0]$. The random blending and augmentation allows for the creation of diverse anomalous images from a single texture source, the process can be seen in Figure 2.

This method allows for the creation of varied training data, consisting of the original image I, the augmented image I_a containing simulated anomalies, and the corresponding anomaly map M_a, helping the model better handle real-world anomaly detection scenarios.

Cutout

Preliminary experiments with other networks revealed difficulties in detecting anomalies like leaflets and sometimes blisters, which are primarily white in color. Since both product types are found most of the times in a rectangular shape, the cutout method was tested to see if it improved anomaly detection for these cases. The process was straightforward: a white rectangle was randomly added to the image at a feasible size (not too large or too small) and position.

During training, the images could follow one of three paths:

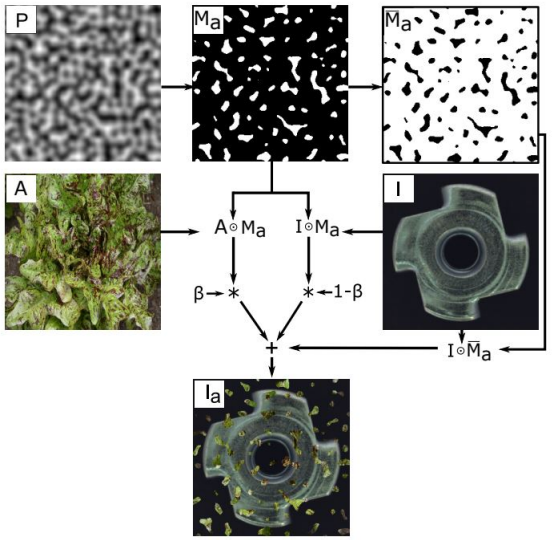


Figure 2: Anomaly generation process. First Perlin noise is applied to create a suitable mask M_a, then an image is sampled from the DTD dataset and applied to the original image according to M_a

- The image was left unaltered.
- Perlin noise was applied to the image.
- The cutout method was used.

Image level anomaly score

To calculate the image-level anomaly score from the output of the discriminative sub-network, which produces a pixel-level anomaly detection map M_o, a simple yet effective approach can be used. First, the anomaly map M_o is smoothed using a mean filter convolution, which helps aggregate anomaly responses across neighboring pixels. This step ensures that local variations are reduced, making the anomaly more evident.

The image-level anomaly score is then computed as the maximum value of the smoothed anomaly score map. Mathematically, it can be expressed as:

$$\eta = max(M_o * f_{s_f \times s_f}), \tag{7}$$

where $f_{s_f x s_f}$ is a mean filter of size $s_f \times s_f$ and * is the convolution operator.

3.3 DDAD

Diffusion Models

Diffusion models are generative models that were originally inspired by non-equilibrium thermodynamics,aiming to learn a distribution $p_\theta(x)$ that closely resembles the data distribution $q(x)$. The model operates in two key phases: the forward process and the reverse process.

In the forward process, Gaussian noise is incrementally added to an input image (or data sample) over a series of steps, ultimately converting it into pure noise. Mathematically, this can be expressed as:

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{8}$$

where x_t represents the noisy version of the data at step t, and is a time-dependent noise variance, while $\beta_t \in (0, 1)$ representing the variance schedule that determines how noise is added at each timestep. The reverse process aims to reverse this noise by learning to gradually denoise the image, step by step, until the original data distribution is recovered. This is modeled as a Markov chain, with each step predicting the denoised data from a noisy sample. The goal is to approximate the posterior distribution:

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{9}$$

where μ_θ and Σ_θ are the parameters learned by the model to estimate the mean and covariance, and θ represents the model's parameters. During training, diffusion models minimize a variational bound on the negative log-likelihood of the data, this function can be written as:

$$\nabla_{x_t} log_{p_\theta}(x_t) = -\frac{1}{\sqrt{1-\alpha_t}}\epsilon_\theta^{(t)}(x_t) \tag{10}$$

Here, α_t is defined as the cumulative product of a noise schedule β_t, where:

$$\alpha_t = \prod_{s=1}^{t}(1 - \beta_s), \tag{11}$$

The term $\epsilon_\theta^{(t)}(x_t)$ represents the noise predicted by the model at timestep t, which approximates the true noise ϵ added to the data during the forward diffusion process. This is used to compute the gradient of the log-probability $\nabla_{x_t} \log p_\theta(x_t)$, facilitating the denoising process.

Essentially, they learn to map noisy samples back to their original form by estimating the noise at each step. Once trained, the model can sample new data by reversing the diffusion process, starting from random noise and progressively denoising it to generate realistic samples.

Conditioned Denoising Process for Reconstruction

As discussed in Section 3.3, reconstructing a target image y involves applying the reverse diffusion process, wherein a perturbed image x_t is progressively denoised. A critical component of this reconstruction lies in conditioning the score function on the target image, resulting in a posterior score function $\nabla_{x_t} log p_\theta(x_t|y)$. Nevertheless, directly computing this posterior score is challenging due to the differing signal-to-noise ratios between x_t and y. A key assumption simplify this: if the reconstructed image x_0 closely approximates y, adding noise to y at the same level as x_t will yield a noisy version y_t that aligns with x_t. This approach facilitates guiding x_t towards y_t at each step of the denoising process. The noisy image y_t is derived by adding noise, predicted by the diffusion model as $\epsilon_\theta^{(t)}(x_t)$, to y. Consequently, the conditioning on y is updated to y_t, leading to the posterior score function $\nabla_{x_t} log p_\theta(x_t|y_t)$, which guides the denoising process. By applying Bayes' rule, the posterior score can be expressed as:

$$\nabla_{x_t} \log p_\theta(x_t \mid y_t) = \nabla_{x_t} \log p_\theta(x_t) + \nabla_{x_t} \log p_\theta(y_t \mid x_t). \tag{12}$$

A primary challenge lies in computing the conditional score $\nabla_{x_t} log_{p_\theta}(y_t|x_t)$, as it is often intractable. However, once y_t is obtained, this likelihood can be estimated. Intuitively, the conditional score serves as a correction factor to account for the divergence between x_t and y_t at each denoising step. The divergence is quantified as $x_t - y_t$, and the updated noise term $\hat{\epsilon}$ is given by:

$$\hat{\epsilon} = \epsilon_\theta^{(t)}(x_t) - w\sqrt{1-\alpha_t}(y_t - x_t), \tag{13}$$

where w is a weighting parameter that controls the influence of the conditioning. Using the updated noise term $\hat{\epsilon}$, the new prediction $\hat{f}_\theta^{(t)}(x_t)$ is computed as previously described. The denoised image x_{t-1} is then obtained via the following step:

$$x_{t-1} = \sqrt{\alpha_{t-1}}\hat{f}\theta^{(t)}(x_t) + \sqrt{1-\alpha t - 1 - \sigma_t^2}\epsilon_\theta^{(t)} + \sigma_t\epsilon_t, \tag{14}$$

Reconstruction for Anomaly Detection

The denoising process aims to reconstruct a cleaner version of an image by removing anomalies. Similar to the DRAEM architecture 3.2, the model is trained exclusively on nominal data, with anomalies appearing in low-probability regions of the distribution $p_\theta(x)$. During the denoising trajectory, early steps capture coarse, abstract features of the image, while later steps refine the finer details. Since anomalies typically manifest in these finer details, the denoising process can begin earlier in the trajectory $(T' < T)$, where the signal-to-noise ratio remains sufficient to distinguish nominal patterns from anomalous ones.

Anomaly Scoring

To compute the anomaly score for the network, two complementary approaches are employed: a pixel-wise comparison and a feature distance-based comparison, both performed between the input and its reconstruction. The use of both methods addresses the limitations of relying solely on pixel-wise comparisons,

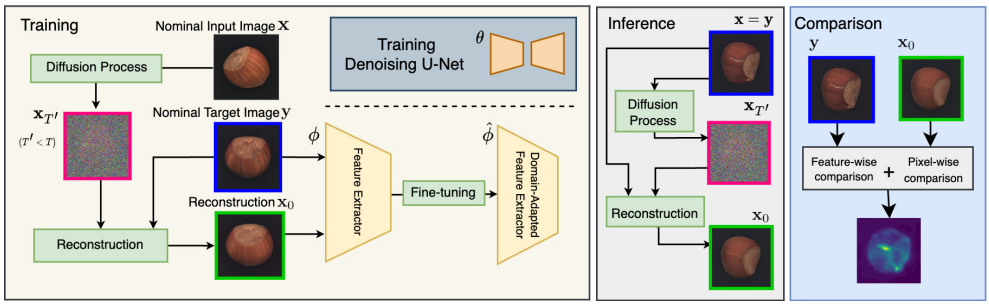


Figure 3: DDAD architecture

which may fail to capture subtle differences such as variations in edges, structures, or colors. By integrating these approaches, the detection capabilities are significantly enhanced.

The pixel-wise metric D_p is computed using the L_1 norm in pixel space. For the feature distance D_f, a method inspired by PatchCore [14] and PaDiM [7] is used. Adaptive average pooling is applied to smooth each feature map spatially, aggregating features within a specified patch into a single representation while retaining the input dimensions. Cosine similarity is then employed to calculate the feature distance.

$$D_f(x_0, y) = \sum_{j \in J}(1 - cos(\phi_j(x_0), \phi_j(y))), \tag{15}$$

In this context,x_0 is the input image, y it's reconstruction, ϕ represents a pre-trained feature extractor, and $j \in J$ indicates the set of layers considered for feature extraction. To maintain the generality of the features used, only layers $j \in 2, 3$ are selected. The pixel-wise distance D_p is normalized so that it shares the same upper bound as the feature distance D_f. The final anomaly score is then derived by combining both the pixel and feature distances:

$$D_{anomaly} = (v\frac{max(D_f)}{max(D_p)})D_p + D_f \tag{16}$$

where v controls the importance of the pixel-wise distance.

Domain Adaptation

Since the pretrained feature extractor was trained on ImageNet, it often struggles to capture domain-specific nuances relevant to anomaly detection, particularly since we use our own custom dataset. To address this, domain adaptation is employed. The approach involves selecting a random image x from the training dataset, adding noise to generate a perturbed image x_t, and choosing a target image y also from the training dataset. The goal is to use a trained denoising model θ to reconstruct an approximation of y by denoising x_t to obtain x_0. Given the expectation that $x_0 \sim y$, their features, denoted as $\phi_j(x_0)$ and $\phi_j(y)$, should be closely aligned. Fine-tuning is done by minimizing the feature distance, using a loss function $L_{Similarity}$ based on cosine similarity, transforming the pretrained model into a domain-adapted network $\hat{\phi}$.

To maintain the generality of the original model, a distillation loss is introduced using a frozen feature extractor $\overline{\phi}$, representing the state of the network ϕ before domain adaptation. The domain adaptation loss is therefore expressed as:

$$\begin{aligned} &\mathrm{L}_{DA} = L_{Similarity}(x_0, y) + \lambda_{DL} L_{DL}(x_0, y) \\ &= \textstyle\sum_{j \in J}(1 - cos(\phi_j(x_0), \phi_j(y))) \\ &+ \lambda_{DL} \textstyle\sum_{j \in J}(1 - cos(\phi_j(y), \overline{\phi_j}(y))) \\ &+ \lambda_{DL} \textstyle\sum_{j \in J}(1 - cos(\phi_j(x_0), \overline{\phi_j}(x_0))) \end{aligned}$$

where λ_{DL} is the significance of distillation loss L_{DL}.

3.4 Masking

Empirical Masking

In this project, images often contained highly detailed sections or were occasionally blurred due to the constraints of the camera's position. This variability caused the networks to identify numerous false positives. To mitigate this issue, heavy masking was extensively applied across all images. However,

no predefined masking strategy was used; the masking was entirely empirical. Each network was initially evaluated without masking, which allowed for a thorough error analysis to identify key areas where masking might be beneficial. Examples of masking across all views are shown in Figure 4.

The term ”heavy” masking is used because, in some cases, it covered portions of the anomaly. Although this might not be ideal in many scenarios, the goal here is not perfect segmentation. Instead, the objective is to highlight areas that could help the user locate the anomaly. Therefore, even if a section of the anomaly is masked, as long as the overall anomaly remains detectable, the network’s performance is considered satisfactory.

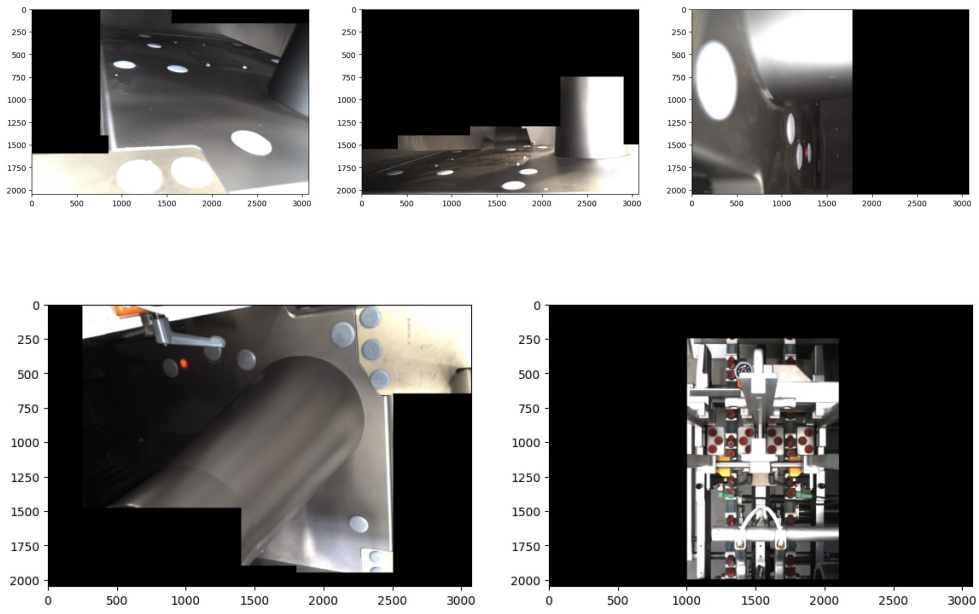


Figure 4: Visualization of each view with heavy masking applied

Image Registration

A portion of the dataset consisted of images with slight shifts in the camera angle or position. This was done to simulate a real-world scenario, where an operator might accidentally nudge the camera, causing a change in its viewpoint. A typical solution to handle this issue is Image Registration, which

involves aligning different data sets to a common coordinate system. The procedure used here was simple: a secondary dataset containing shifted images was created, then an image from the training set, captured with a consistent camera angle, was chosen as the reference image.

The image registration process was carried out as follows:

- **Convert Images to Grayscale**: Both the reference and the image to be aligned were first converted to grayscale for processing.
- **Feature Detection**: ORB (Oriented FAST and Rotated BRIEF)[15] from the OpenCV[5] library was used to identify and extract key points and their descriptors in both images. These key points represent distinctive patterns in the images that can be matched.
- **Feature Matching**: The key points between the reference and the image to be aligned were compared using BFMatcher (Brute Force Matcher). BFMatcher.match() was used to retrieve the best match, while BFMatcher.knnMatch() allowed the retrieval of the top K matches for further analysis.
- **Filtering Matches**: The top matches were selected, and noisy or irrelevant matches were filtered out to ensure accuracy.
- **Homography Computation**: A homography transform was calculated based on the filtered matches. This transform describes how the image needs to be warped to align with the reference.
- **Transformation Application**: The computed homography was applied to the original unaligned image (and the ground truth if present) to produce the aligned output.

This process allowed the dataset to maintain consistency despite slight shifts in the camera's position, during Section 5 the method will be explored in more depth to see if it actually gave any improvements, an example of image registration can be seen in Figure 6.

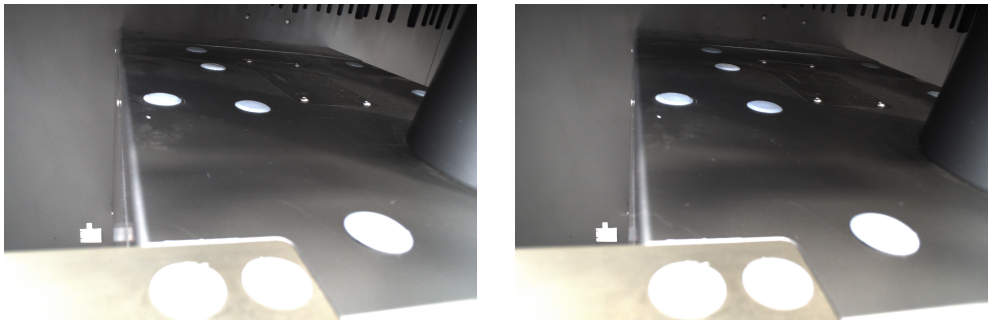


Figure 5: The image on the left shows the A view without any shifts, while the one on the right demonstrates the same view after applying a shift.

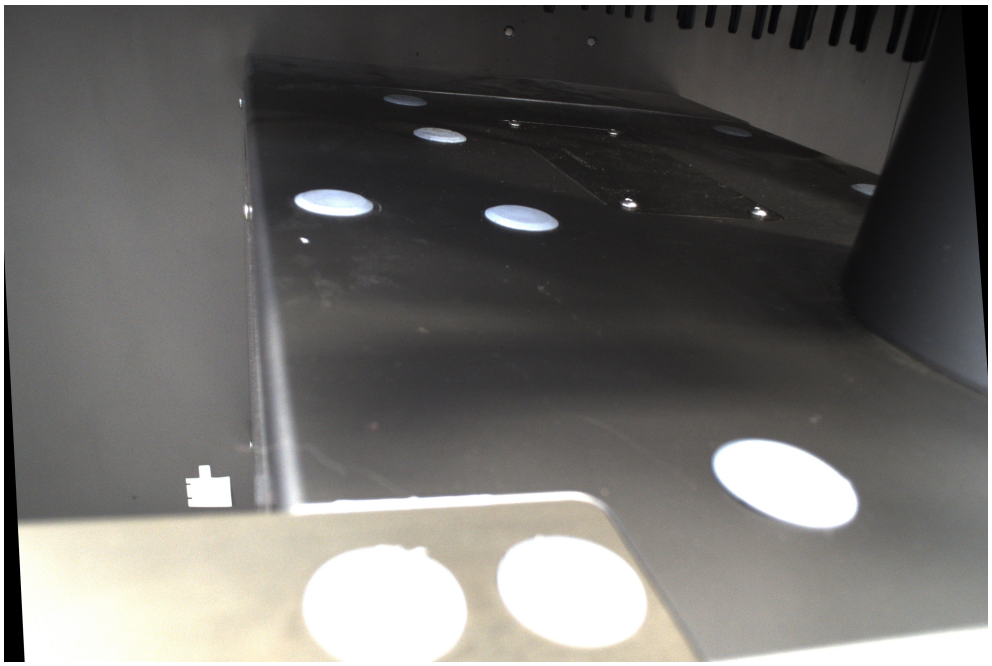


Figure 6: Image registration applied on the A view. It can clearly be seen how the image was shifted by looking at the bottom left and up right corners.

Timing of masking

The timing of when masking is applied is critical in this experiment, as it can significantly influence the results. There are two main stages where masking can be introduced:

- **Pre-Anomaly Map Computation**: In this approach, masking is applied before any anomaly map is generated. Both the input image and its reconstruction are masked prior to calculating the anomaly maps. This means that the masked areas do not contribute to the computation, potentially leading to less noise in the anomaly detection process.
- **Post-Anomaly Map Computation**: Here, masking is applied after the anomaly map has been computed. In this case, the designated masked regions are set to 0 in the anomaly map, effectively silencing any potential anomaly signals from those areas. This method ensures that masked regions do not trigger false anomalies during the final evaluation.

Impact of masking on DRAEM

DRAEM applies a softmax operation to enhance anomaly regions while reducing values in normal areas. Masking before or after the softmax affects the result differently. When masking is applied before softmax, the masked regions do not contribute to the softmax calculation. This means that if the highest values were in the masked areas, the output distribution might shift. Conversely, masking after softmax simply nullifies the output values in the masked regions without altering the computed distribution.

Impact of masking on DDAD

The impact of masking is quite evident in DDAD. When masking is applied before computing the anomaly maps, it affects both the pixel-wise and feature-wise distance calculations, with portions of the image intentionally obscured. Additionally, the pixel-wise distance undergoes normalization before the final mask is produced. This preprocessing can lead to varying contributions when

the final anomaly map is constructed, potentially altering the sensitivity to certain types of anomalies.

3.5 Threshold selection

Setting the threshold is arguably the most crucial aspect of the entire anomaly detection pipeline, as it can make the difference between an accurate anomaly segmenter and a scenario where no anomalies are detected at all. Inspired by the methodology used in the MVTEC paper [3], three distinct thresholding methods were tested:

- Maximum Threshold.
- p-Quantile Threshold.
- k-Sigma Threshold.

All thresholds were initially determined using a validation dataset and later applied during the evaluation phase.

Maximum Threshold

As the name implies, this method selects the highest anomaly score found in the validation set as the threshold. In theory, images without anomalies should not exhibit high scores; hence, if the method functions correctly, no anomaly should be detected in anomaly-free images. However, this method is sensitive to outliers, as a single pixel with a high anomaly value can set an unusually high threshold. While this approach can minimize false positives, it may increase the false negative rate, potentially missing some anomalies. Segmentation quality may suffer slightly under this method, but it's generally acceptable given that the primary focus is on detection over precise localization.

p-Quantile Threshold

This thresholding method aims to mitigate the influence of outliers by considering the entire distribution of anomaly scores in the validation set. It allows for a certain percentage of validation pixels to be classified as anomaly-free. The threshold is determined by selecting a p-quantile, where p usually falls within the range from 95% to 99%. This ensures that only a small percentage of high anomaly scores (potentially anomalies) exceed the threshold, making the estimation more robust.

k-Sigma Threshold

In the final method, the mean μ and standard deviation σ of the validation set's anomaly scores are calculated. The threshold is then set as $t = \mu + k\sigma$ where k is typically 3, assuming a Gaussian distribution. A k-value of 3 captures about 99% of the distribution's area. However, anomaly scores rarely follow a perfect Gaussian distribution, so the actual false positive rate may vary significantly from this expectation. This approach balances sensitivity and robustness but may still be prone to some misclassifications if the distribution deviates from Gaussian assumptions.

4 Evaluation metrics

When evaluating the performance of anomaly detection and segmentation methods, a range of metrics is used to capture different aspects of accuracy and robustness. Key metrics include the Area Under the Receiver Operating Characteristic Curve (AUROC), the Per-Region Overlap (PRO), Average Precision (AP), and the Intersection over Union (IoU). Each metric offers unique advantages that make it suitable for different aspects of anomaly evaluation.

4.1 Area Under the Receiver Operating Characteristic Curve

AUROC is a common metric for binary classification tasks, measuring the ability of a model to differentiate between normal and anomalous data. It calculates the area under the ROC curve, which plots the true positive rate against the false positive rate at various threshold settings. The primary advantage of AUROC is its threshold-independence, providing a single performance score that encapsulates how well the model separates normal from anomalous data across all possible thresholds. In this experiment, AUROC will be computed at two levels:

- **Image-level AUROC:** This measures the model's ability to classify entire images as either normal or anomalous, evaluating its global performance in distinguishing between the two categories.
- **Pixel-level AUROC:** This evaluates the model's performance in distinguishing normal and anomalous regions at a finer granularity, assessing how accurately it identifies individual anomalous pixels within the images.

This dual-level evaluation provides a comprehensive understanding of the model's performance, both in terms of overall classification and precise anomaly localization.

4.2 Per-Region Overlap

PRO, or Per-Region Overlap, specifically measures the overlap between predicted and true anomalous regions. This metric accounts for the importance of localizing anomalies accurately rather than just detecting them, which is particularly crucial in segmentation tasks. It rewards predictions that closely match the size and shape of true anomalies, making it a valuable measure when accurate localization is essential.

4.3 Average Precision

Average Precision (AP) evaluates the precision-recall trade-off, summarizing the precision at various recall levels. This metric is particularly sensitive to class imbalances, often seen in anomaly detection tasks where anomalies are rare compared to normal instances. AP is useful because it emphasizes the importance of high precision (low false positives) at all levels of recall, making it a strong indicator of a model's robustness.

4.4 Intersection over Union

IoU, or Intersection over Union, quantifies the overlap between predicted and ground truth segmentation masks by computing the ratio of the intersection to the union of these areas. This metric is widely used in segmentation tasks because it directly measures the accuracy of spatial predictions. Higher IoU values indicate better alignment between predicted and true anomaly regions, making it a crucial measure for assessing how precisely the anomalies are localized.

5 Experiments and Results

5.1 Dataset

Anomaly detection datasets like MVTecAD [3] and VisA [19] typically focus on objects in the foreground, occupying most of the image. This setup allows for assumptions about the potential location of defects, even if they are small. In contrast, our dataset presents a more complex scenario, as it involves images captured inside a machine, often cluttered with irrelevant details such as moving parts, wires, and dents, which complicates the anomaly detection task. The dataset was carefully curated, consisting of 193 images from five different viewpoints. Of the 114 anomaly-free images, only 76 were selected for training. This subset was further divided into training and validation sets, with 80% used for training and 20% set aside for validation to later assist with threshold computation. The remaining 38 anomaly-free images featured slight shifts in camera angle and small variations such as moving wires and mechanical parts. These were used during evaluation to assess the network's robustness to minor viewpoint change, which is an important consideration given that, in real-world scenarios, cameras are rarely perfectly stationary and may be slightly displaced by operators. The dataset also included 79 anomalous images, which were used to evaluate the network's ability to detect defects, including those with slight viewpoint shifts combined with anomalies. Examples of both normal and anomalous images for each viewpoint can be seen in Figure 7. Each image in the dataset has a high resolution of 2048x3072, which introduces several challenges if not treated properly. Firstly, training becomes impractical due to the excessive time and memory requirements needed to process such large images. Secondly, while evaluation speed may not be critical for our task, the time difference can still be significant from a practical standpoint. Processing a high-resolution image can take much longer, with times ranging from 20 seconds or more, compared to the more efficient 1-5 seconds that a lower resolution could achieve. This disparity can be crucial for end

users, as faster evaluation times are generally preferred in real-world applications.

5.2 Batch size

In this study, the batch size was set to 1 due to several constraints. First, each view contained a limited number of images, typically fewer than 16, which restricted the feasible batch size as training was conducted independently for each view. Furthermore, attempts to increase the batch size resulted in memory overflows, halting the training process. These interruptions were partly due to the hardware limitations inherent to this project, as well as the larger dimensions at which images were processed. In contrast, standard datasets, such as MVTecAD, feature images at lower resolutions, and existing models, such as DDAD, are trained at 256x256 (while DRAEM does not specify an input size). In our project, however, images were trained and evaluated at a resolution of 512x512 to preserve more fine-grained detail, which substantially increased memory requirements.

5.3 DRAEM results

The network was trained for 3000 epochs, following the original paper's setup. The learning rate was initialized at 10^{-4} and reduced by a factor of 0.1 after 1200 and 2100 epochs. Additionally, Cutout augmentation was applied during training.

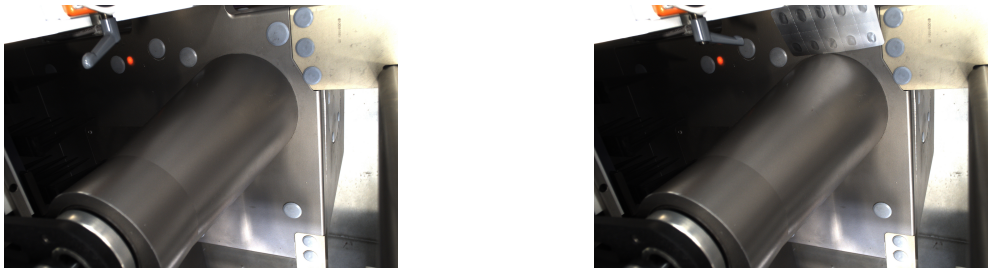
Metrics overview

From epoch 2000 onwards, models were saved at intervals of 250 epochs, resulting in five different saved models. Each saved model was evaluated across eight scenarios, combining the factors of post-masking (PO) vs. pre-masking (PRE), masking (M) vs. no masking (NM), and image registration (R) vs. no

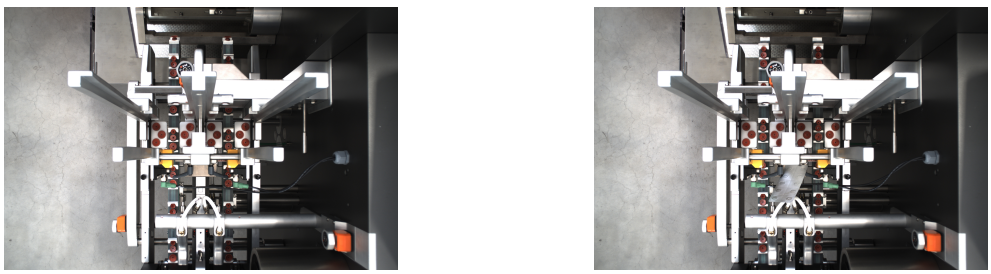
A view

A2 view

B view



B2 view



C view

Figure 7: Anomaly-free and anomalous images from each view of the machine

image registration (NR). The results of these evaluations are presented in Tables 1, 2, 3, 4, and 5. Only the performance metrics of the best-performing model are included in the final results.

The tables highlight the unique advantages of different techniques for each view. For evaluation, each metric provides a distinct perspective on the model's performance. However, accuracy stands out as the most critical metric, as it directly reflects the system's ability to identify anomalies without false positives or false negatives essential for end users.

For View A, the optimal configuration appears to be PO+M+NR, though a similar setup not incorporating masking achieves comparable results with less accuracy. This demonstrates that masking facilitates more effective anomaly detection in this view.

View A2, on the other hand, exhibits robust performance across all masking-based configurations. This resilience likely stems from the presence of conveyor belts in the images, which can lead to false positives if masking is not applied. By integrating masking, the model effectively identifies anomalies without interference from the conveyor belt.

View B presents significant challenges due to blurry images and small anomalies. In this case, the combination of masking and image registration proves instrumental in enhancing performance. Conversely, View B2 shows great potential. Masking is less intrusive here, and the images provide a clear view of anomaly locations. Each configuration offers distinct strengths, making this view particularly adaptable and effective.

Lastly, View C is the most complex to manage. It requires extensive masking and benefits substantially from image registration, both of which are crucial for improving accuracy in anomaly detection.

Results visualization

A key aspect of the evaluation involves visualizing the detected anomalies for the user. While an image may be labeled as anomalous, the localization of

	A view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(68.0,93.9)	(60.0,99.5)	(86.7,94.0)	(61.3,**99.6**)	(68.0,93.9)	(60.0,99.5)	(**91.6**,97.5)	(69.1,98.7)
PRO	91.7	91.3	**92.5**	92.4	91.7	91.3	92.1	87.8
IoU	52.4	54.9	52.6	**55.2**	53.9	54.9	53.6	16.7
AP	75.2	78.0	75.0	**80.3**	75.2	78.0	76.0	60.6
ACC	70.0	70.0	**75.0**	70.0	**75.0**	70.0	**75.0**	**75.0**

Table 1: DRAEM Performance metrics under different configurations for A view, the AUROC is divided into image level AUROC and pixel level AUROC

	A2 view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	**(100,99.7)**	(100,98.8)	**(100,99.7)**	(100,98.8)	**(100,99.7)**	(100,98.8)	**(100,99.7)**	(100,98.8)
PRO	**97.3**	86.7	**97.3**	86.7	**97.3**	86.7	**97.3**	86.7
IoU	**58.2**	54.7	**58.2**	54.7	57.4	54.7	57.4	54.7
AP	**81.3**	76.4	**81.3**	76.4	**81.3**	76.4	**81.3**	76.4
ACC	**100.0**	80.0	**100.0**	80.0	**100.0**	80.0	**100.0**	80.0

Table 2: DRAEM Performance metrics under different configurations for A2 view

	B view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(**73.3**,97.9)	(61.3,99.1)	(38.7,97.8)	(38.7,**99.2**)	(**73.7**,97.9)	(61.3,99.1)	(38.7,99.2)	(38.7,**99.2**)
PRO	**92.2**	85.3	90.2	85.7	**92.2**	85.3	90.2	85.7
IoU	**45.2**	42.7	41.3	42.6	42.8	42.7	41.1	42.6
AP	**73.3**	70.7	70.7	72.2	**73.3**	70.7	70.7	72.2
ACC	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**

Table 3: DRAEM Performance metrics under different configurations for B view

	B2 view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(83.3,95.8)	(75.0,**99.8**)	(**85.0**,95.8)	(73.3,**99.8**)	(83.3,95.8)	(75.0,**99.8**)	(**85.0**,95.8)	(73.3,**99.8**)
PRO	96.0	**98.3**	95.9	98.1	96.0	**98.3**	95.9	98.1
IoU	51.1	**53.5**	50.8	51.2	51.1	**53.5**	50.8	51.2
AP	**88.2**	86.8	87.5	83.8	**88.2**	86.8	87.5	83.8
ACC	**78.9**	**78.9**	**78.9**	**78.9**	**78.9**	**78.9**	**78.9**	**78.9**

Table 4: DRAEM Performance metrics under different configurations for B2 view

	C view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(91.6,97.5)	(69.1,**98.7**)	(**92.0**,97.3)	(70.3,98.4)	(91.6,97.5)	(69.1,98.7)	(**92.0**,97.3)	(70.3,98.4)
PRO	**92.1**	87.8	90.4	85.4	**92.1**	87.8	90.4	85.4
IoU	**53.6**	16.7	48.5	25.7	**53.6**	16.7	48.5	25.7
AP	**76.0**	60.6	73.4	63.3	**76.0**	60.4	73.4	63.3
ACC	85.4	60.4	81.2	62.5	**87.5**	60.4	85.4	62.5

Table 5: DRAEM Performance metrics under different configurations for C view

the anomaly could still be inaccurate. In Figure 8, a correct localization is demonstrated, where the predicted anomaly mask closely matches the ground truth. In contrast, Figure 9 shows a case where the localization is correct,

but the anomaly score is too low, causing the image to be classified as non-anomalous. Finally, Figure 10 illustrates a poor localization example, where the image is classified as anomalous, but the predicted anomaly map bears no resemblance to the ground truth.

These examples highlight two important concerns. First, while the network may accurately localize an anomaly, the image may be discarded due to a low anomaly score. Second, an image labeled as anomalous may have the anomaly predicted in entirely incorrect locations. These issues underscore the challenges of using a threshold as a reliable indicator for anomaly classification. Ultimately, these results point to the need for strategies to improve the model's robustness.

5.4 DDAD results

Metrics overview

DDAD followed a similar methodology to DRAEM in terms of model saving and testing procedures. The results are summarized in Tables 6, 7, 8, 9, and 10.

For the A view, the model performs significantly better with masking and without image registration, achieving an accuracy of 95.0%. This improvement is largely attributed to the masking process, which effectively addressed false positives caused by anomalies detected in anomaly-free images, particularly in challenging areas for the model to learn.

Similarly, the A2 view benefits from masking, which mitigates false positives arising from features like the conveyor belt that previously caused issues.

In contrast, the B view shows consistent accuracy across all implementations, indicating that the architecture struggles to reliably detect certain anomalies, regardless of the applied techniques. However, some methods demonstrate superior localization of detected anomalies, as evidenced by the metrics.

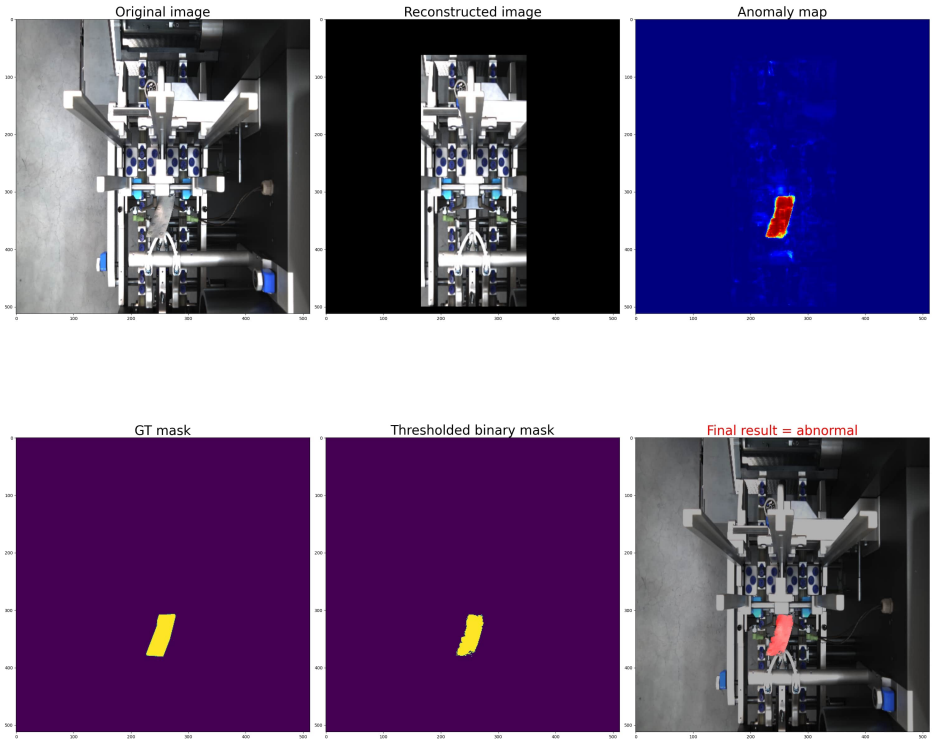


Figure 8: DRAEM example of correctly localized anomaly of the C view, from top left to bottom right, the original image, the reconstructed image with masking applied, the anomaly map generated, the ground truth, the predicted anomaly map and finally the anomaly displayed on the image

The B2 view yields strong results for both masking and non-masking approaches when image registration is not applied. This is consistent with observations in DRAEM, as the B2 view provides clear images where anomalies are relatively easy to detect. Errors primarily stem from machine components in areas unlikely to trap products. To better evaluate the model's robustness, anomalies were intentionally placed in these regions for testing.

Finally, the C view highlights the model's inability to learn effectively from this perspective as seen in Figure 11. The best performance occurs when all images are labeled as anomalies, suggesting a lack of meaningful feature extraction. Masking and image registration offer slight improvements, but

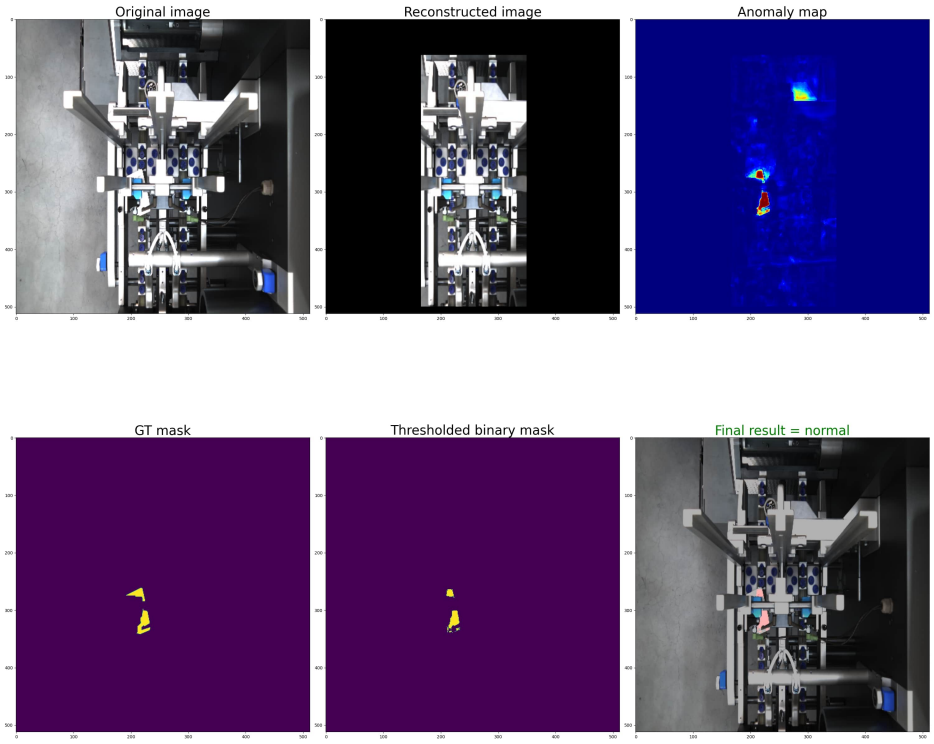


Figure 9: DRAEM example of correctly localized anomaly of the C view, however the image anomaly score is too low and the final image is treated as a normal image

the overall results remain suboptimal and require further refinement. Using a lower threshold might solve some problems but could also lead to many false negatives.

Results visualization

DDAD demonstrates strong performance in identifying anomalies. Unlike DRAEM, which tends to produce highly detailed anomaly maps closely resembling the ground truth, DDAD often encapsulates anomalies in a blob-like feature. From a practical perspective, this behavior can be advantageous, as it visually aids users in quickly locating anomalies within the image. A clear

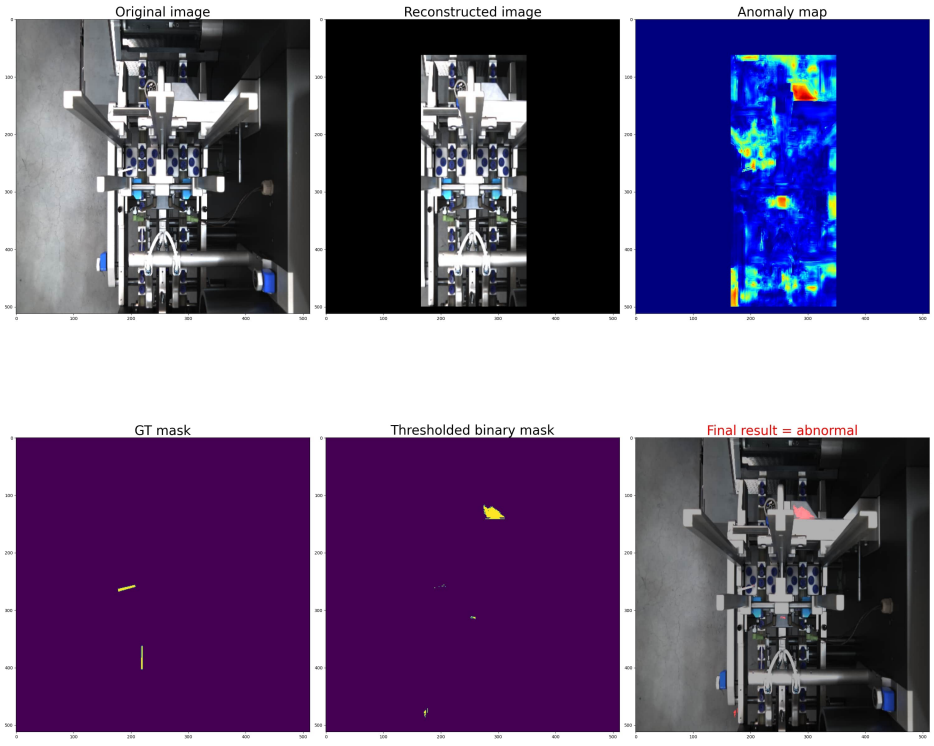


Figure 10: DRAEM example of uncorrectly localized anomaly of the C view

example of this can be observed in Figure 12.

However, DDAD is not without its limitations, as illustrated in Figures 13 and 14. The first example presents a challenging scenario due to the camera setup and the inherent blurriness of the image, causing portions of the image to be incorrectly flagged as anomalies, resulting in the entire image being classified as anomalous. In the second example, within the same view, some regions of the image are mistakenly identified as anomalies. On the positive side, the actual anomaly is still correctly localized, showcasing DDAD's potential to retain critical information despite occasional errors.

These results once again emphasize the critical importance of selecting an appropriate threshold and employing more refined masking techniques to minimize the occurrence of false anomalies.

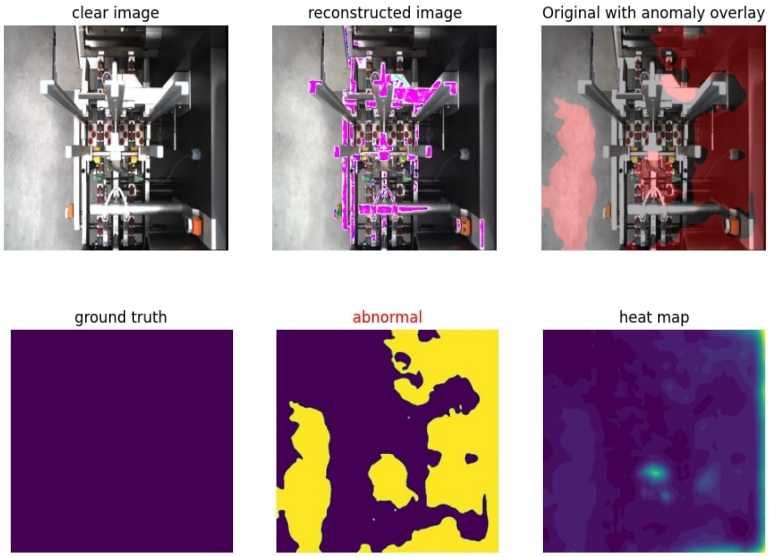


Figure 11: C view image after being processed with DDAD, the model is unable to extract meaningful features and treats most of the image as an anomaly

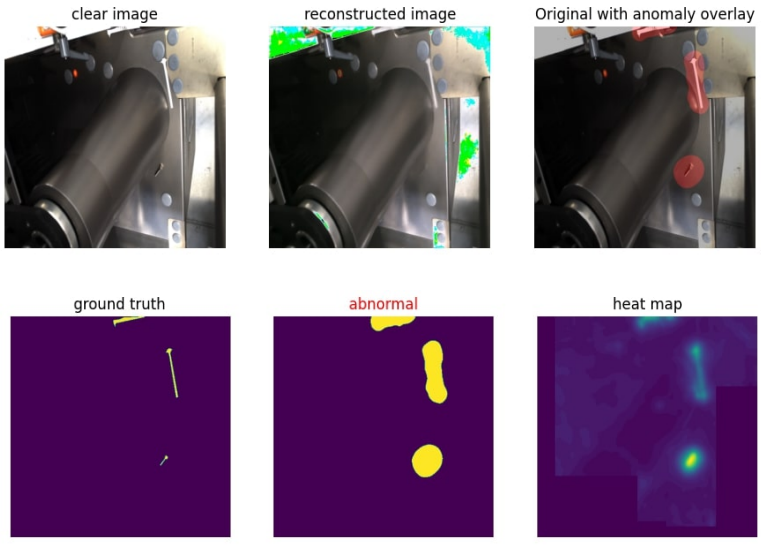


Figure 12: DDAD example of correctly localized anomaly in the B2 view

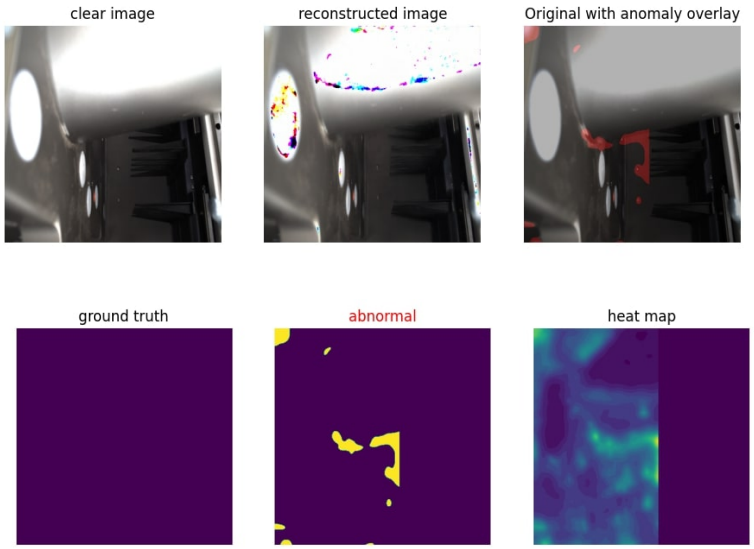


Figure 13: DDAD example of a bad detection In the B view

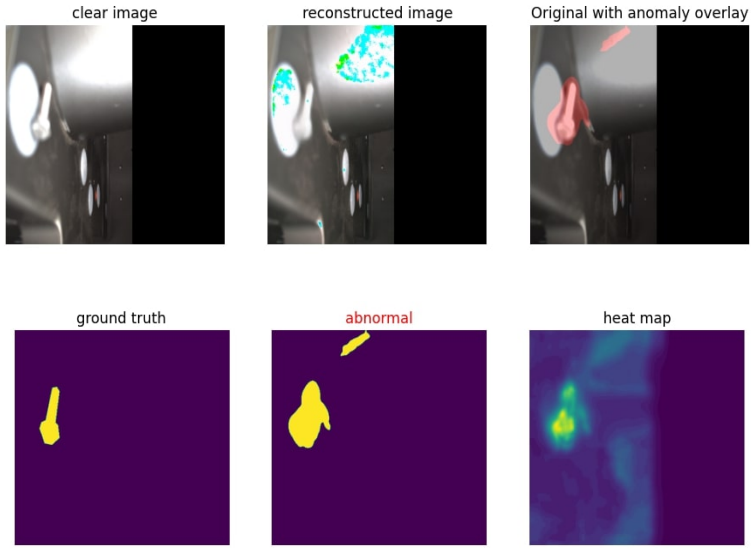


Figure 14: DDAD example of a bad localization In the B view

	A view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(53.3,93.4)	(44.0,98.0)	(92.0,93.5)	(80.0,**98.6**)	(74.7,94.1)	(44.0,98.0)	(**93.3**,94.1)	(80.0,**98.6**)
PRO	95.9	94.7	96.7	**96.9**	96.1	94.7	96.5	**96.9**
IoU	39.0	30.9	**43.8**	39.9	17.1	30.9	19.3	39.6
AP	55.5	46.1	**59.5**	58.3	56.9	46.1	59.4	58.3
ACC	75.0	75.0	**95.0**	75.0	75.0	75.0	90.0	75.0

Table 6: DDAD Performance metrics under different configurations for A view

	A2 view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	**(100,99.7)**	(75.0,99.5)	**(100,99.7)**	(75.0,99.5)	**(100,99.7)**	(75.0,99.5)	**(100,99.7)**	(75.0,99.5)
PRO	**97.7**	97.0	**97.7**	97.0	97.4	97.0	97.4	97.0
IoU	**31.1**	14.0	**31.1**	14.6	17.3	14.0	17.3	14.0
AP	77.5	72.4	77.5	72.4	**78.4**	72.4	**78.4**	72.4
ACC	90.0	80.0	90.0	80.0	**100.0**	80.0	**100.0**	80.0

Table 7: DDAD Performance metrics under different configurations for A2 view

	B view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(45.3,96.4)	(49.3,93.9)	(80.0,95.8)	(57.3,93.9)	(66.7,**97.5**)	(49.3,93.9)	(**85.3**,97.1)	(57.3,93.9)
PRO	**90.8**	76.8	87.7	81.9	89.7	76.8	88.6	81.9
IoU	31.6	26.4	32.2	**32.4**	27.2	25.1	27.2	**32.4**
AP	**54.4**	42.3	52.7	47.0	51.8	42.3	50.9	47.0
ACC	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**	**75.0**

Table 8: DDAD Performance metrics under different configurations for B view

	B2 view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(60.0,95.5)	(55.0,**99.6**)	(78.3,95.3)	(78.3,99.4)	(80.0,99.4)	(55.0,**99.6**)	(**85.0**,99.1)	(78.3,99.4)
PRO	95.3	**97.9**	94.4	97.0	97.0	**97.9**	96.2	97.0
IoU	44.5	38.9	32.5	28.1	**47.4**	35.9	35.4	28.5
AP	76.5	75.3	71.8	74.5	**78.7**	75.3	74.6	74.5
ACC	78.9	78.9	**89.5**	**89.5**	78.9	78.9	**89.5**	**89.5**

Table 9: DDAD Performance metrics under different configurations for B2 view

	C view							
	PO+M+R	**PO+NM+R**	**PO+M+NR**	**PO+NM+NR**	**PRE+M+R**	**PRE+NM+R**	**PRE+M+NR**	**PRE+NM+NR**
AUROC	(55.4,97.2)	(34.1,99.0)	(51.0,93.2)	(51.0,93.4)	**(56.6,99.7)**	(34.1,99.0)	(37.9,94.4)	(51.0,93.4)
PRO	93.9	92.0	68.0	58.7	**96.8**	92.0	67.5	58.7
IoU	**23.4**	6.5	1.4	0.7	16.1	6.6	1.3	0.7
AP	**63.4**	37.0	31.4	32.0	62.4	37.0	25.7	32.0
ACC	**54.2**	**54.2**	**54.2**	**54.2**	**54.2**	**54.2**	**54.2**	**54.2**

Table 10: DDAD Performance metrics under different configurations for C view

5.5 Threshold results

Throughout the document, the importance of threshold selection has been emphasized as a pivotal aspect, if not the most critical component, of the anomaly

detection pipeline. To clarify this point, a brief overview of the three thresholding methods tested is provided below.

Starting with max-threshold, this method is the ”safe” option. It generates a significantly lower number of false positives compared to the other methods, making it ideal for particularly challenging scenarios. However, this comes at a cost: the lower false positive rate often translates to a lower true positive rate, meaning the model may miss true anomalies. This trade-off can be suboptimal for tasks where detecting all anomalies is crucial.

On the other hand, p-quantile and k-sigma thresholds behave similarly in most cases, often leading to comparable threshold values. Of the two, k-sigma tends to yield slightly more accurate results than p-quantile. These thresholds are well-suited for the majority of the views tested. However, their higher accuracy comes with a downside: an increase in false positives and a greater likelihood of poor anomaly localization. Despite these drawbacks, this behavior is often preferred, as the higher likelihood of detecting an anomaly outweighs the inconvenience of additional false positives. In the worst-case scenario, the user may need to inspect the machine before resuming operation, a precaution that is generally acceptable in industrial settings.

An example with the same image tested with the different thresholds can be seen in Figures 15, 16 and 17.

5.6 Neural Networks comparison

Graphs overview

For each configuration tested, the corresponding graphs were also computed, including img_AUROC, pixel_AUROC, pixel_PRO, the precision-recall curve, and the IoU curve. These graphs provide a visual summary of the metrics, offering an additional layer of analysis.

To derive meaningful insights, two types of comparisons were conducted.

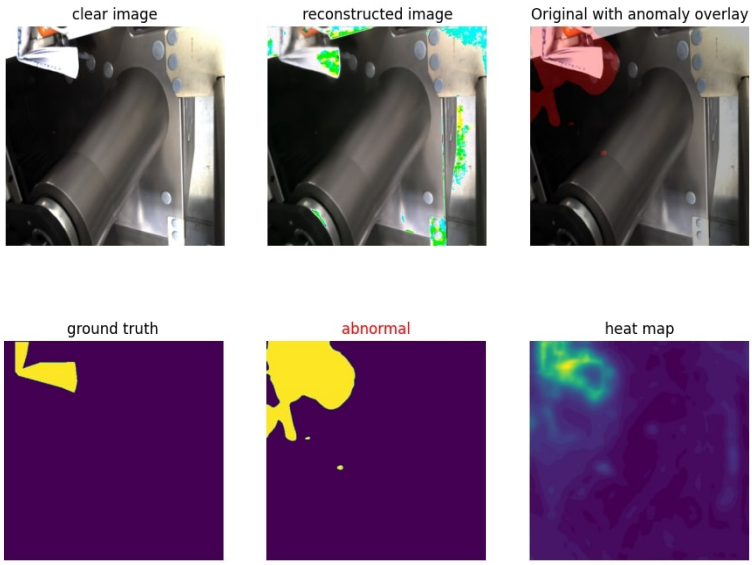


Figure 15: Anomaly detection using k-sigma threshold

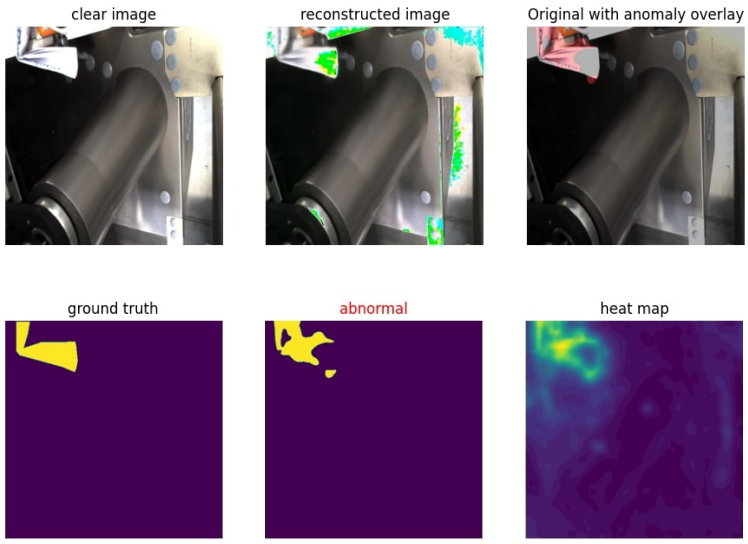


Figure 16: Anomaly detection using max threshold

The first compared performance across different views for each saved checkpoint, allowing for an assessment of how well the model generalized across views. The second comparison focused on the same view, analyzing performance across different model checkpoints. This was crucial for evaluating whether training for additional epochs improved performance or yielded diminishing returns.

Starting with DRAEM, Figure 18 illustrates the view comparison for the model's checkpoint at epoch 3000, utilizing post-masking without image registration. All curves were generated using the optimal threshold, defined as the one yielding the highest accuracy. In cases where multiple thresholds achieved the same accuracy, the max-threshold was prioritized, followed by the k-sigma threshold and then the p-quantile threshold. This prioritization was chosen because the max-threshold is more ”conservative”, ensuring a more stable and reliable localization of anomalies.

The results reveal that the B view exhibits the greatest inconsistency, which aligns with the findings presented in Section 5.3. Conversely, the DDAD results for the same checkpoint are displayed in Figure 21, showing significant performance discrepancies in the C and B views compared to the others. These observations are consistent with the analysis in Section 5.4, further highlighting that these two views present the most significant challenges across both models.

For the checkpoint comparison, the C view was selected as it posed the greatest challenge among all the views. Figure 20 illustrates the performance of View C using the DRAEM model. The results demonstrate a general trend of improved performance with longer training durations, as models trained for more epochs often achieve better results compared to earlier checkpoints. This trend is consistent across all views, with models trained for higher epochs often outperforming earlier checkpoints or, in some cases, matching their performance. However, occasional drops in performance were observed, suggesting that the benefits of extended training may plateau or vary depending

on the view and specific configurations.

In contrast, DDAD results, shown in Figure 21, indicate that the model struggles to extract meaningful signals from View C. Performance across epochs remains largely stagnant, particularly in terms of the image AUROC score, which fails to show significant improvement. These findings align with previously discussed results, reinforcing the view's status as a challenging scenario for the model.

5.7 Visualization comparison

An additional evaluation was conducted by comparing the visualization results of the networks, as shown in Figures 22, 23, 24, 25, 26, and 27. Although these figures represent only a subset of the dataset, similar trends were observed across other examples. As previously noted, the DDAD model tends to envelop anomalies in blob-like shapes, prioritizing coverage over precision, while the DRAEM model strives for precise segmentation, closely mimicking the contours of the anomaly.

The performance of the models varies significantly depending on the view. For instance, in View C, the DRAEM model demonstrates superior strength and robustness, effectively identifying anomalies with high accuracy. Conversely, for View A2, the DDAD model provides more reasonable results, particularly in challenging scenarios. However, both models exhibit noticeable shortcomings when masking is absent, highlighting the critical role of masking techniques in improving anomaly localization.

In conclusion, the comparative evaluation underscores the importance of model selection and view-specific adaptation in anomaly detection tasks. While DRAEM and DDAD each have distinct strengths, their performance is highly context-dependent, suggesting that a hybrid approach or model ensemble could potentially leverage the strengths of both architectures.

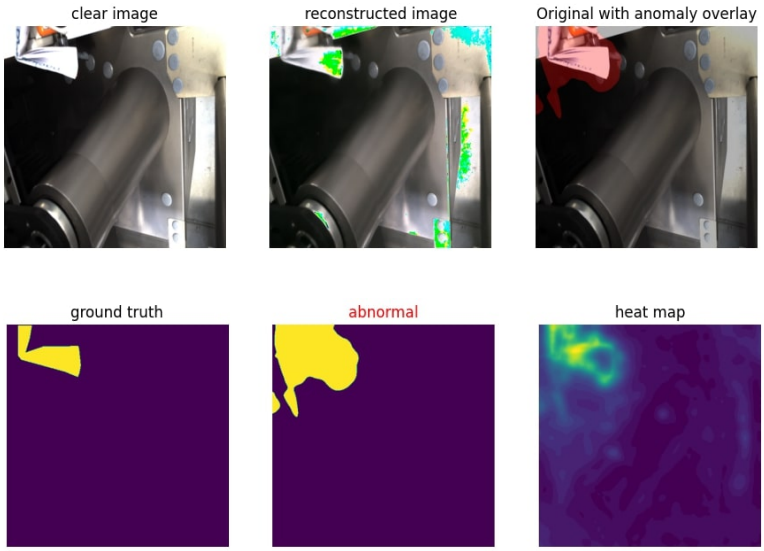


Figure 17: Anomaly detection using p-quantile threshold

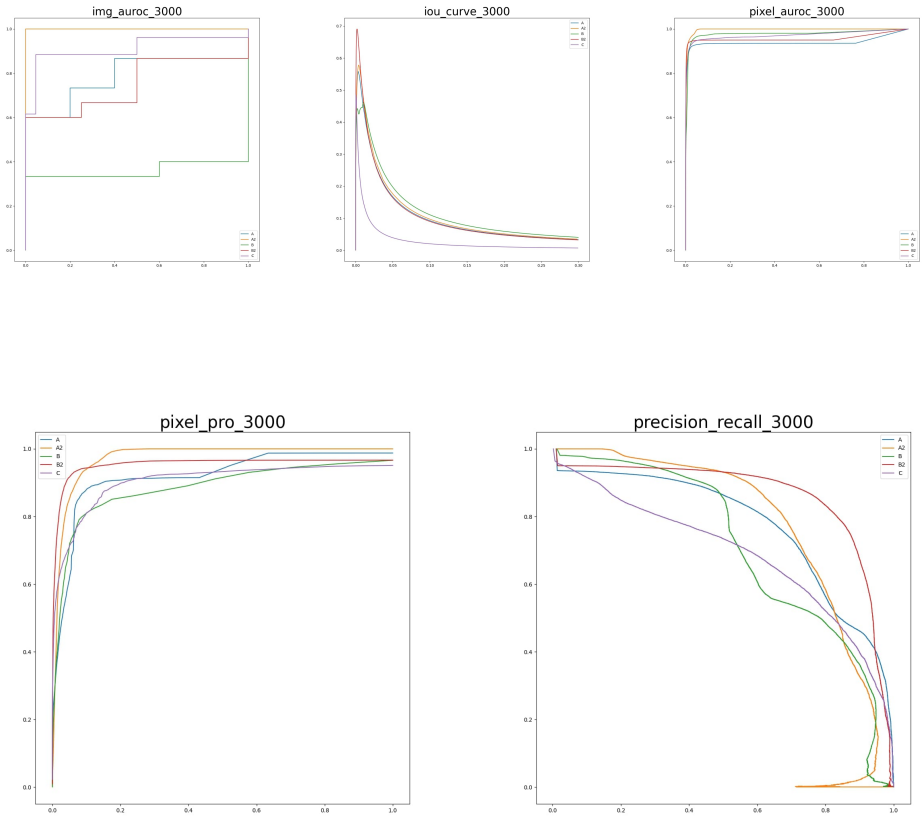


Figure 18: Metrics visualization of DRAEM model utilising post masking and no image registration, the model's checkpoint is the epoch 3000

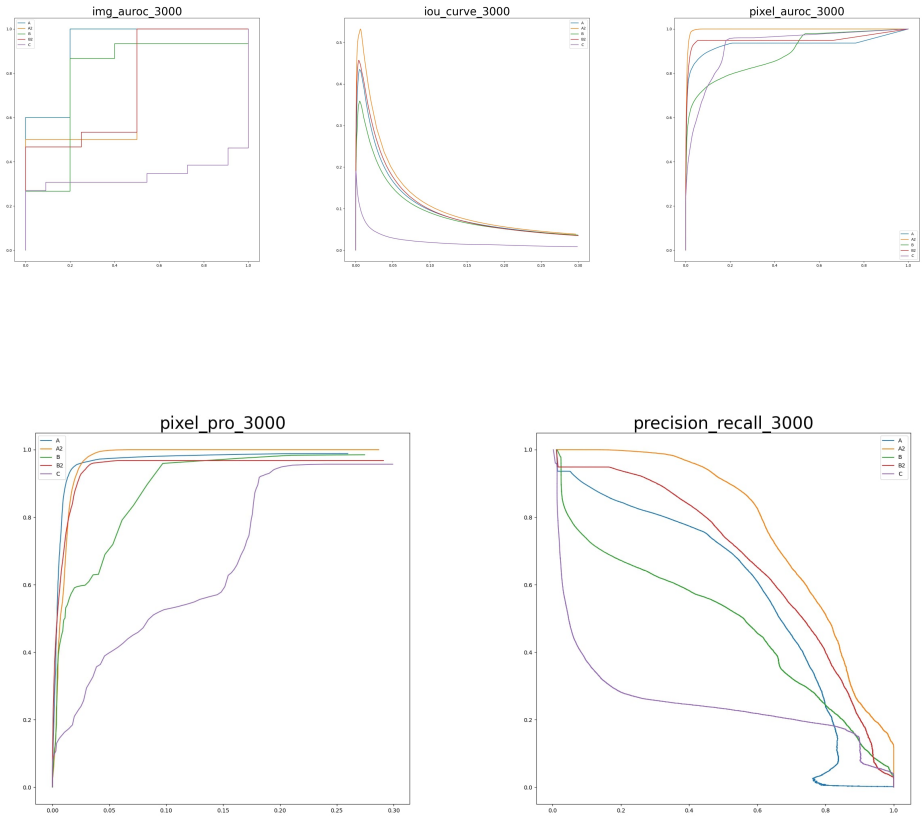


Figure 19: Metrics visualization of DDAD model utilising post masking and no image registration, the model's checkpoint is the epoch 3000

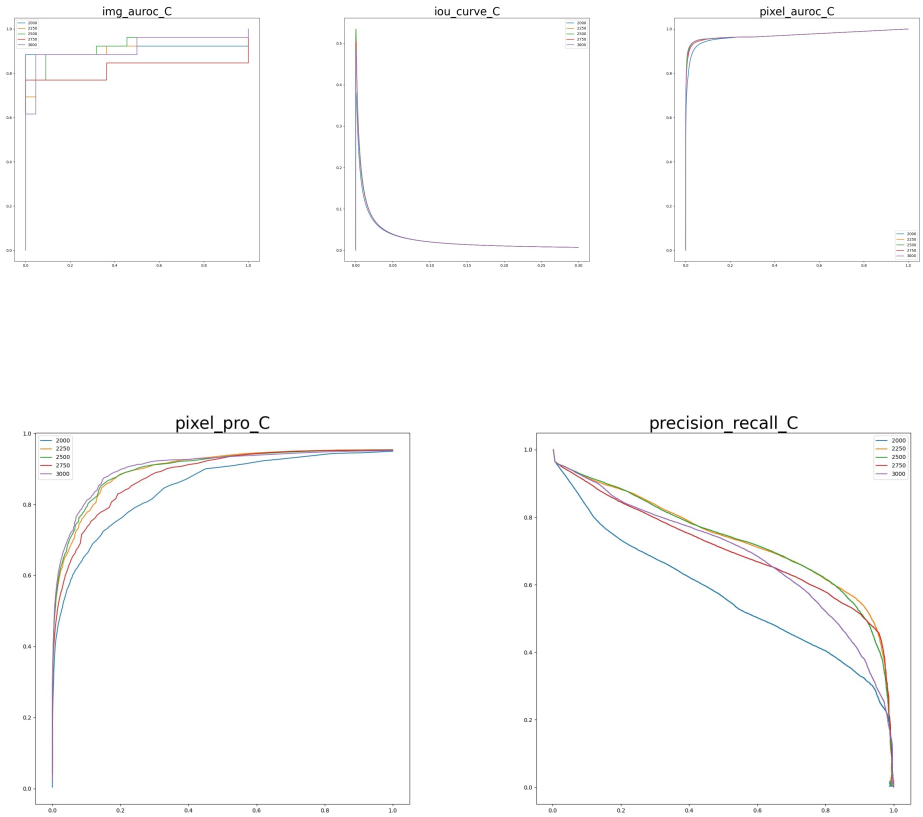


Figure 20: Metrics visualization of DRAEM model on the C view, best model configuration was taken

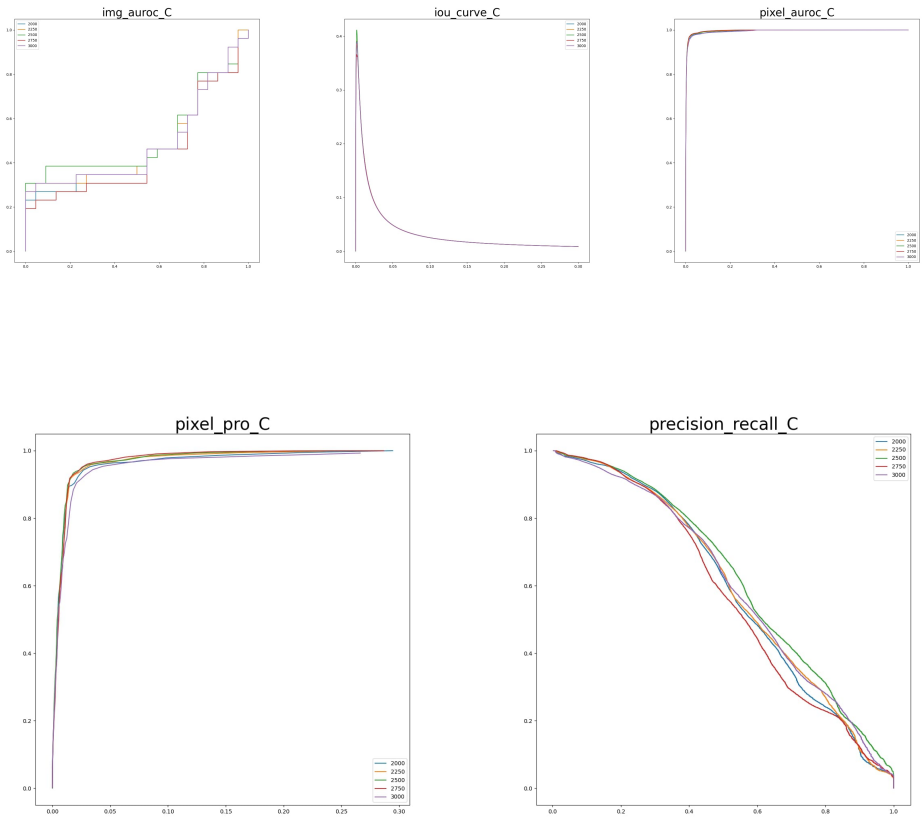


Figure 21: Metrics visualization for all epochs of DDAD C view

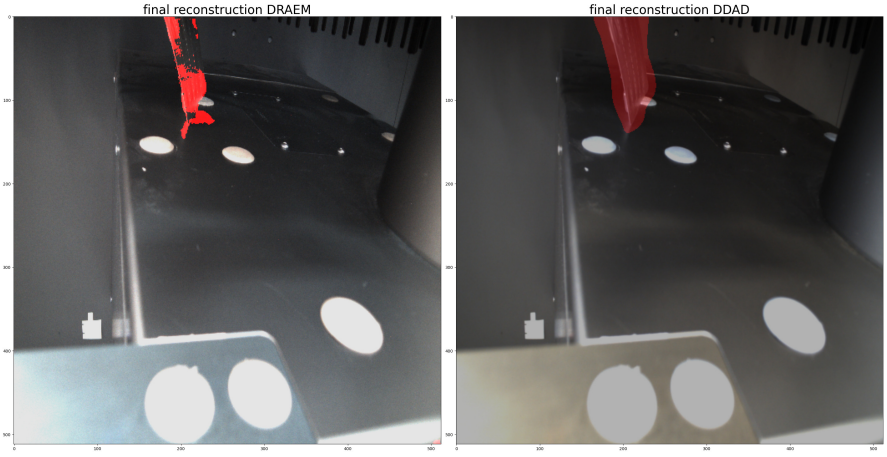


Figure 22: Comparison between DRAEM and DDAD in the A view.

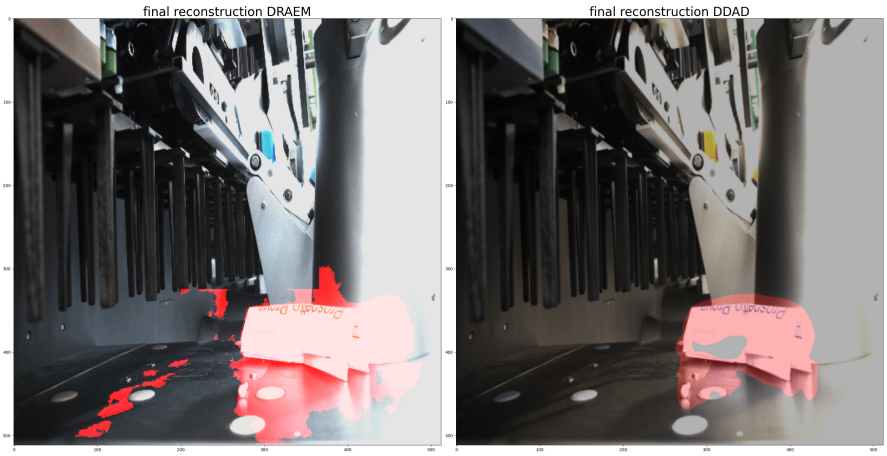


Figure 23: Comparison between DRAEM and DDAD in the A2 view using masking.

Figure 24: Comparison between DRAEM and DDAD in the A2 view, no masking is applied and it can be seen that both networks struggle to detect only the correct anomaly, hinting at the fact that masking is needed in this context.

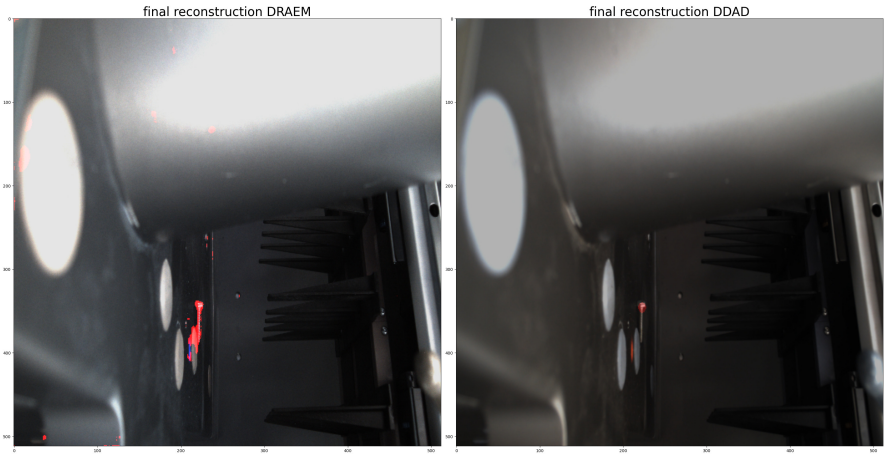


Figure 25: Comparison between DRAEM and DDAD in the B view.

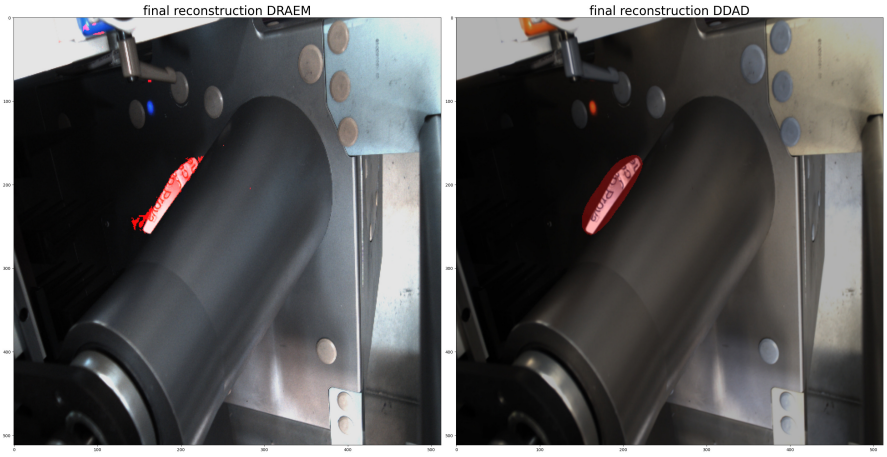


Figure 26: Comparison between DRAEM and DDAD in the B2 view.

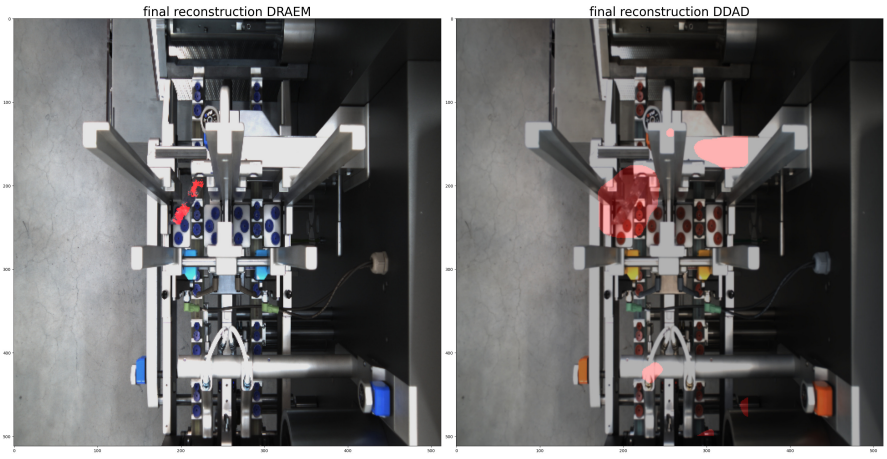


Figure 27: Comparison between DRAEM and DDAD in the C view.

6 Conclusion

This work presented an in-depth evaluation of two leading anomaly detection networks, DRAEM and DDAD, applied to a challenging real-world industrial scenario. Each network demonstrated unique strengths and weaknesses, highlighting their suitability for different aspects of anomaly detection tasks.

DRAEM proved to be robust across various views, showcasing consistent performance. However, it struggled with precise localization, often producing more false positives compared to DDAD. On the other hand, DDAD excelled in anomaly detection and localization in most views, yet its performance faltered significantly on the C view, indicating a lack of generalization to certain complex scenarios.

Several techniques were employed to enhance the performance of both networks, including masking and image registration. These techniques demonstrated their utility, with specific combinations being more effective for particular views. Additionally, the choice of threshold played a critical role in achieving optimal results. Three thresholding methods were thoroughly tested, each proving more suitable for specific conditions. This highlights the importance of tailoring hyperparameters and processing techniques to the nature of the data and operational context.

While the networks showed promising results, this study also uncovered some limitations. A significant concern is their dependence on training data and settings that may not fully align with real-world applications. This overreliance could hinder their adaptability and robustness in dynamic environments.

6.1 Future Works

In industrial applications, solutions often evolve significantly to meet real-world constraints and operational needs, and this project is no exception. This

work was conducted several months ago and has provided a strong foundation for improvements, but developments in the machine setup and use-case requirements have necessitated a reevaluation and refinement of the approach.

One of the most significant changes has been to the dataset itself. Of the five original views, only two or three have been deemed meaningful for the final application, with one still under consideration. These adjustments stem from the fact that the machine under test was not fully constructed during the initial evaluation. Subsequent modifications to the machine, such as new components added to the C view, have led to significant changes in how anomalies need to be detected.

The C view, in particular, now features a new point of view and must contend with a protective covering added to the blister area. While this enhancement improves safety and usability, it introduces potential challenges for anomaly detection due to reflections, occlusions, or distortions caused by the cover. For the other views, a new version of the A view has been captured, featuring an improved angle that better highlights the areas of interest. Additionally, a revised version of the B2 view is under consideration, aiming to improve clarity and relevance.

To address the challenges posed by these changes, the use of specialized illumination techniques is being explored. Improved lighting could help enhance visibility, especially for areas beneath the conveyor belt, making it easier to detect anomalies even in less accessible or darker regions.

These updates represent a critical step in adapting the anomaly detection pipeline to evolving requirements. By refining the dataset and introducing targeted improvements to the imaging setup, the system is becoming more tailored to real-world industrial constraints. This iterative process underscores the importance of flexibility and ongoing evaluation in developing effective solutions for complex applications.

Future efforts could also focus on developing a new neural network or refining existing architectures to better handle diverse scenarios, ensuring they

can tackle challenges that current models struggle to resolve.

Bibliography

[1] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara. Latent space autoregression for novelty detection, 2019. arXiv: 1807.01653 [cs.CV]. URL: https://arxiv.org/abs/1807.01653.

[2] *SIGGRAPH Comput. Graph.*, 19(3), 1985. B. A. Barsky and J. C. Beatty, editors. ISSN: 0097-8930.

[3] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592, 2019. DOI: 10.1109/CVPR.2019.00982.

[4] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019. DOI: 10.5220/0007364503720380. URL: http://dx.doi.org/10.5220/0007364503720380.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[7] T. Defard, A. Setkov, A. Loesch, and R. Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization, 2020. arXiv: 2011.08785 [cs.CV]. URL: https://arxiv.org/abs/2011.08785.

[8] A. Dutta and A. Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, Nice, France. ACM, 2019. ISBN: 978-1-4503-6889-6/19/10. DOI: 10.1145/3343031.3350535. URL: https://doi.org/10.1145/3343031.3350535.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[12] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister. Cutpaste: self-supervised learning for anomaly detection and localization, 2021. arXiv: 2104.04015 [cs.CV]. URL: https://arxiv.org/abs/2104.04015.

[13] A. Mousakhan, T. Brox, and J. Tayyub. Anomaly detection with conditioned denoising diffusion models. *arXiv preprint arXiv:2305.15956*, 2023.

[14] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler. Towards total recall in industrial anomaly detection, 2022. arXiv: 2106.08265 [cs.CV]. URL: https://arxiv.org/abs/2106.08265.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. DOI: 10.1109/ICCV.2011.6126544.

[16] J. Yoon, K. Sohn, C.-L. Li, S. O. Arik, and T. Pfister. Spade: semi-supervised anomaly detection under distribution mismatch, 2022. arXiv: 2212.00173 [cs.LG]. URL: https://arxiv.org/abs/2212.00173.

[17] S. Zagoruyko and N. Komodakis. Wide residual networks, 2017. arXiv: 1605.07146 [cs.CV]. URL: https://arxiv.org/abs/1605.07146.

[18] V. Zavrtanik, M. Kristan, and D. Skocaj. Draem - a discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8330–8339, October 2021.

[19] Y. Zou, J. Jeong, L. Pemula, D. Zhang, and O. Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation, 2022. arXiv: 2207.14315 [cs.CV]. URL: https://arxiv.org/abs/2207.14315.

Acknowledgements

It has been another two years since the last time I found myself writing a thesis, and I must say these two years have been filled with new experiences, both positive and negative. These experiences, in one way or another, have shaped me into the person I am today. So, first and foremost, I want to give myself a pat on the back for staying resilient during this period and pushing forward despite the various challenges I encountered along the way.

Now, moving on to the real protagonists of this section, I would like to express my heartfelt gratitude to Professor Samuele Salti for his guidance in selecting and writing this thesis and for his valuable advice on how to navigate the project effectively. I also want to thank my colleagues at work, who helped me better understand a world I was entirely unfamiliar with, providing insights that allowed me to see how my project connects to the industrial landscape. I want to thank my family for always being supportive, even though I know full well that most of them have absolutely no idea what half of the exams I took were about. Despite this, they always trusted that I knew what I was talking about (a serious mistake on their part).

I would like to thank my lifelong friends, starting with Francesco, whom I've known for 13 years (and I'm only 24!), and who has always supported me and continues to do so every single day. Adrian, who I met at the beginning of high school, and has been someone with whom I've shared everything, without too many filters, for just as long. Then there's Davide, whom I think I met in the third year of high school (though my memory might fail me), and with whom I've shared countless experiences. Knowing these people for so long

and so deeply, it's amazing to see how each of us has managed to maintain our individuality while changing so much over the years. And it's equally fascinating to see how each of them, in their own unique way, fits perfectly into my identity.

I am grateful to my friend Angelo, whom I met during my first year of high school as well, and who is currently a coursemate of mine. He supported me across countless challenges during this journey, providing both motivation and encouragement when it was most needed.

I would like to thank the friends I met during the bachelor's degree, Mirko, Vittorio, and Andrea, without whom the journey would have been far more boring. What always surprises me is how all three of them were roommates. Out of all the people I could have met, I became so close to three people who rented the same house! Not to mention that the aforementioned house was an excellent base for when I needed to stay overnight in Bologna. Don't judge me, but as a commuter, you have to do what you can to survive!

However, over the course of two years, you also meet new people! I want to thank Pietro, who was a delightful discovery and whom I met during the second semester of my first year. His encyclopedic knowledge of history amazes me every single time. By the way, the way I've phrased it might make it sound like he's a coursemate of mine, he's not. I actually met him through Francesco, that guy I mentioned earlier (you remember him, right?).

Finally, I'd like to thank Carlotta, who only entered my life a few months ago but has managed to add a touch of color to it in a way that only a few people ever have.

With that said, I think I've gone on long enough. I've only mentioned a few people, those who, in my view, played the most significant roles over these past two years. If I had to name every single person who has helped me, even in the smallest way, I'd probably need to write another thesis.

It's been a wonderful journey, and like any good commuter, I think it will end with one last train home.