Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line

Integration of deep generative Anomaly Detection algorithm in high-speed industrial line

Niccolò Ferrari*†‡ Nicola Zanarini† Michele Fraccaroli*
Alice Bizzarri* Evelina Lamma*

14/03/2024

Abstract

One of the main challenges of modern industrial automation is the exploitation of increasingly intelligent and efficient Quality Control solutions. The pharmaceutical industry requires high quality standards, for both economic reasons and health concerns. Human operators still perform inline checks on packages, but they face obvious obstacles, such as the risk of overlooking a defect, in addition to low efficiency, which limits productivity. Automatic algorithms, conversely, can execute inline high-speed controls due to their lower error rate, improved systematicity, and increased throughput. Classical approaches to visual inspection rely on blob-analysis algorithms with limited adaptability to varied settings or acceptable samples due to their huge number of parameters and inherent rigidity. To overcome aforementioned issues, machine learning algorithms are being increasingly integrated into industrial applications. The state-of-the-art of anomaly detection is achieved with Deep Learning infrastructures that are capable of generalizing complex and variable features distribution. Industry research has focused on applying machine learning algorithms to computer vision, but their applicability to real-case scenarios remains a challenge. Industrial production has numerous good samples available, however, defects are only a small fraction of the production, so obtaining a set as large as the good one is difficult; As a result, it is important to manage extremely unbalanced datasets, which often exclude supervised networks. This paper proposes the implementation of a Generative Adversarial Network-based architecture on a high-speed industrial line that produces pharmaceutical Blow Fill Seal vials. The architecture detects anomalies during production and is trained on over 2500000 images.

*Department of Engineering, University of Ferrara, Via Saragat 1, 44122 Ferrara, Italy
†Bonfiglioli Engineering, Via Amerigo Vespucci 20, 44124 Ferrara, Italy
‡niccolo.ferrari@unife.it, nferrari@bonfiglioliengineering.com

1 Keywords

Anomaly Detection; Industrial automation; Machine Vision; Generative Adversarial Network; Automated Quality Control; Big data

2 Acknowledgments

The authors would like to thank Bonfiglioli Engineering for providing a real-case dataset to test the software developed in this work. The first author is supported by an industrial PhD funded by Bonfiglioli Engineering, Ferrara, Italy. Alice Bizzarri is supported by a National PhD funded by Politecnico di Torino, Torino, Italy and Università di Ferrara, Ferrara, Italy.

3 Nomenclature

AE	AutoEncoder
VAE	Variational AutoEncoder
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Time Memory
GAN	Generative Adversarial Network
Generator	Generative subnet of the GAN
Discriminator	Adversarial subnet of the GAN
Discriminative net	U-Net subsequent to the GAN used for segmentation
CRAE	fully-Convolutional Residual AutoEncoder
DRAE	Dense-bottleneck Residual AutoEncoder
AUROC	Area Under the Receiver Operating Characteristic
ROI	Region Of Interest
SSIM	Structural Similarity Index Measure
BFS	Blow Fill Seal

4 Introduction

Industrial applications increasingly require machine learning solutions to overcome difficult tasks, such as inline visual anomaly detection at production time. Implementations must take in account several physical limitations related to the hardware and to the requested specifications, maximizing the trade-off between accuracy performance and realization cost. Therefore, feasibility plays a crucial role in implementation.

One of the primary applications is anomaly detection for machine vision, and one of the main emerging areas is the pharmaceutical industry, which requires extensive non-destructive controls on production capable of meeting high accuracy and high throughput, thus pharma companies commenced to invest a lot of effort into research and integration of the mentioned architectures within

production line automation. Usually requirements rely on finding a trade-off between the following constraints:

1. Accuracy rate (either in terms of missing rejects rate and false rejects rate)
2. Hardware
3. Costs

Accuracy has a double impact: the primary and most relevant is the patients health and their safety; the latter has a business impact. The other two items are highly correlated but do not overlap; while hardware does contribute to the overall cost, encompassing computing clusters, automation handling, and the size of the machinery itself, it affects the footprint of the machine on the line and increase its maintenance complexity, creating additional constraints.

Nowadays, the majority of pharmaceutical companies entrust their quality controls to manual operators. Unsystematic errors caused by oversights or attention deficits can lead to efficiency deviations. Cosmetic inspection and particle search are among the most critical operations performed by visual inspectors. They learn how to recognize production flaws. This task is very difficult to formalize by defining an analytical description of the problem, as the decision taken by the human operator relies on a perceptual understanding of the visualizing context, which can make it very complex to structure an algorithm based on classical blob-analysis operations.

Classical procedural algorithms are developed ad-hoc usually for a small set of examples, both compliant and anomalous, exploiting pattern-matching heuristics, topological and colors features extraction based on thresholds given during the designing phase, by utilizing a large number of parameters. This approach could give the illusion of being more adjustable by using the mentioned parameters as knobs. In truth, the implementation is highly connected to the product itself and even to the representation provided in the form of images, resulting in a rigid and unscalable infrastructure. Furthermore, to be analytically treatable, a problem of such complexity needs to be simplified, avoiding the formalization of perceptual aggregation of features in the given images, which would result in losing the global sense that is easily extrapolated from a human brain. Moreover, the oversimplification results in the noise becoming indistinguishable from real defects, such as the inability to distinguish bubbles generated by the liquid contained in a vial from stuck alien particles or other cosmetic defects on the product surface. Finally, it is nearly impossible to encode the noise variance of the entire production using this type of algorithm.

Deep learning for machine vision represents the state of the art to accomplish the task of spotting anomalies in real-time during production. Classification using supervised learning is a very mature solution and widely treated in literature. However, such solutions usually perform best in contexts where the dataset is balanced across classes [1].

In real cases, on production lines, conforming products are far superior in number as compared to anomalous products, for this reason, the training dataset

would be extremely unbalanced in favor of nominal examples. This makes it difficult to perform proper training of a supervised model. To overcome this issue we can use semi-supervised models, which learn feature distribution from a single label dataset, namely the conforming distribution. This approach relies on the inability of the model to reconstruct or distill well extracted anomalous features [2, 3]. This methodology is divided into two main categories: *reconstruction-based* methods, *embedding similarity-based* methods.

4.1 Reconstruction-based

This method is based on the intuition that a network trained solely on conforming images cannot reconstruct anomalous regions. Taking the above considerations as a starting point, it has been thoroughly investigated since it makes possible to learn a robust reconstruction subspace using only images without anomalies. Examples of these kind of models are: autoencoders (AE) [4, 5], variational autoencoders (VAE) [6, 7], generative adversarial networks (GAN) [8], and architectures based on the latter, such as GANomaly [9] (and its variant [10]) or DRÆM [11]. Thanks to the inability to rebuild anomalous regions, which were not contained within nominal images during training, the network fails to reproduce the out-of-distribution area. For this reason, it is possible to detect discrepancies between the two images by *thresholding*; for example, the absolute value of the difference between them or using other kind of similarity algorithms, such as Structural Similarity (SSIM) [12]. Abovementioned nets are usually quite heavy to be trained and inferred, given that at best, the network consists of two parts: the encoder and the decoder. On the other hand they manage to compress extracted features in a small and well manageable latent space.

4.2 Embedding similarity-based

This approach uses deep neural network, pre-trained on huge and sparse datasets, to extracts meaningful vectors of embeddings which aggregate input images features. The concept behind this method is that deep neural networks can synthesize features from images that are far from those used during training, thanks to the dataset's sparsity and the structure of the net, such as Residual Networks [13]. Determining regularity in the distribution can be achieved through various models, including PaDiM [14], PatchCore [15, 16], or normalizing-flow approaches, such as FastFlow [17] and more recent architectures such as PNI [18] and MSFlow [19], which hypothesize that a Multivariate Gaussian Distribution can approximate an industrial process [20]. In contrast, this approach tends to be less interpretable, as described by T. Defard et al. [14], because the anomaly score is the distance from the embeddings extracted from the input image and reference embeddings contained in the latent space. Furthermore, in industrial applications where computing clusters, specifically on-board machines, have limited resources, it is often difficult to handle the latent space as it grows with the training dataset.

4.3 Adopted solution

Taking the above considerations as a starting point, we managed to build an intelligent system for an inline real-time cosmetic surface analysis machine, which performs quality control on BFS plastic strip of vials, filled by liquid chemical. One of the main challenge was not only to get a good performance in terms of accuracy, but also to make the application feasible within project constraints.

The hardware infrastructure is very different between training server and inference computer. For the former:

- Intel® Xeon® Silver 4216 CPU @ 2.10 GHz as CPU
- 64GB of DDR4 Synchronous @ 3200 MHz as system memory
- Nvidia® A100 with 40GB of VRAM as GPU.

For the latter:

- Intel® Xeon® E-2278GE CPU @ 3.30 GHz as CPU
- 32GB of DDR4 Synchronous @ 3200 MHz as system memory
- Nvidia® A4500 with 20GB of VRAM as GPU.

The architecture of the application is based on a previous work [21], which has been readapted in order to fulfill the requirements and the limits imposed by the hardware and the requested test ratio; thus the model is defined by only one GAN that underlay a residual autoencoder (RAE), with a dense bottleneck (DRAE). In addition, the Perlin noise was randomly overlapped onto the input image, just like in DRAEM, not to create a synthetic defect, but to improve the network's denoising capability, while on the other to counteract the trend that has the network to lose the ability to generalize and instead copy the image.

The dataset is composed of 2815200 images obtained from 782 different strips of 5 vials, acquired 10 times each with 16 frames, plus 2 ranked images, per acquisition. Each image has been divided into 20 patches: 5 regions, one per vial, are subdivided into 4 sub-regions each.

In the context of this work, summarizing:

1. generative network is composed by a residual autoencoder (RAE) [22, 23, 24] with a fully-connected bottleneck, designed to work within a GAN architecture [9]
2. the images in the dataset have been pre-processed and pre-patched to divide the product's single vial into four logical regions
3. during training image augmentation have been performed on positive batch examples, and moreover, a Perlin noise have also been superimposed randomly during training phase, in order to improve augmentation and to enhance the capability of the network to remove out-of-distribution noise on images

4. the system's effectiveness has been assessed using a test dataset that is evenly divided between positive and negative cases. The evaluation process began with the raw patches. However, a second aggregation was conducted to assess the accuracy on the good vials and anomalous vials dataset in order to provide a realistic quantification of the machine performance

5. the inference is integrated within machine system, that performs inference on-line, using C++ TensorFlow APIs

5 Related Work

The implemented approach is mainly based on GRD-Net [21], as above mentioned, which is inspired by the already widely treated DRÆM [11]. This model is based on the *reconstruction-based* method which is widespread within the surface anomaly detection field, as they rebuild an input image exploiting the inability to reproduce the out-of-distribution regions contained in the input image [25, 26]. Usually autoencoders (AE) [4, 5, 27, 28], variational autoencoders (VAE) [6, 29] and GAN-based networks [9, 10, 30, 31], are used for image reconstruction. The representation of the image lies in the latent space, which compress the features extracted though the encoder [4]. Structural Similarity index (SSIM) [27] or the pixel-wise reconstruction error [32] can be used both as loss function during training and as anomaly score during inference phase. Also transformers can perform reconstruction and segmentation as in [33, 34], or a combination of convolutional and transformer net [35]. The interpretability of the network results using this approach is a positive aspect, and another positive aspect is that the latent space is limited and fixed in size, despite the computationally heavy network architecture during both training and inference. A good improvement seems to be the proposed contrastive learning technique [36],

Another important mention goes to the family of anomaly detection methods defined as *embedding similarity-based* approach. These techniques extract useful vectors describing an entire image for anomaly detection [37, 38] or an image patch for anomaly localization [39] using deep neural networks. Comparing this approach to the reconstructive one, it is significantly less computationally demanding during the inference phase, which is one of its advantages, since the neural architecture consists of a simple encoder , from which embeddings are extracted. On the other hands embedding similarity method suffers from a lack of interpretability, as it is not possible to pinpoint the precise feature of an anomalous image that led to its anomaly score. The score is calculated as a kind of distance between the embeddings inferred from the input sample and the representation of the complaint products' embeddings distribution. One of the pioneers of this method is SPADE [40], which conveys the normal reference as the central point of a hypersphere that contains the whole collection of normal embeddings. Numerous designs that were later developed based on

this methodology include PaDiM [14], PatchCore [15], and those that use the normalizing flow (NFLOW [41]) approach on extracted embeddings, such as PNI [18] or FastFlow [17]. As previously indicated, these architectures provide a strong throughput even though they require a very high memory consumption, since they use a memory bank as the storage of mined embeddings from the normal dataset. This approach becomes unmanageable in real hardware architectures for large, complicated, and highly variable images that necessitate huge datasets for the network to accurately learn salient features. In addition, another strengths of the reconstruction-based approach is the capability to spot logical defects better than the embedding-similarity one, as mentioned by K. Batzner et al. [42], which can be cited as a further example of an industrially effective strategy founded on the student-teacher paradigm.

6 Methods

To explain our approach it is essential to briefly introduce the GRD-Net [21] architecture.

6.1 GRD-Net

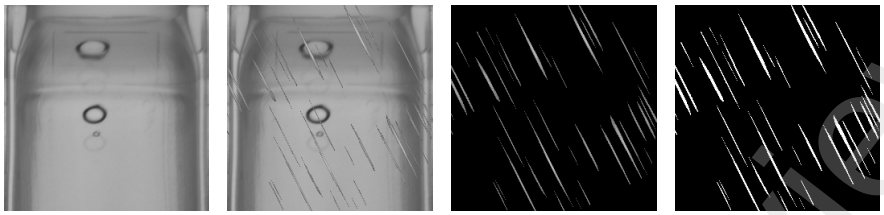
Taking inspiration from DRÆM [11] architecture, GRD-Net [21] consists of two primary networks. The first is a GAN that uses a fully-Convolutional Residual AutoEncoder (CRAE). The second network is a U-net with skip connections, which combines the original image X and the reconstructed image $\hat{X}$ from the autoencoder output. During the training phase, a perturbation function P_q superimposes Perlin noise onto the input image X with a probability of q, resulting in a perturbed image X_n and a map M. In this map, the perturbed pixels in X are represented by white pixels with a value of 1.0, and the unperturbed pixels are represented by black pixels with a value of 0.0. Throughout this process, the original image X remains unchanged and a new X_n is produced:

$$X_n, M = P_q(X). \tag{1}$$

In this design, the perturbation acts as an augmentation for the first network and compels the autoencoder to carry out a denoise task in addition to compressing significant features that are inferred from X. Additionally, it generates the map used by the second network to predict the anomalous region inside the image.

The superimposed noise forces the network to recreate the missing part of the image, further enhancing the aggregation of what are the essential features. Similar techniques have already been investigated by other researchers, one of the most notable being the work of Kaiming He et al., 'Masked Autoencoders Are Scalable Vision Learners' [43].

In the initial project, an attention map—a Region Of Interest (ROI)—was added during the training phase, for focusing the task inside the predetermined



(a) Original vial region image (X) (b) Original vial region image with Perlin noise X_n (c) The generated Perlin noise standalone N (d) The map of the noise M

Figure 1: The vial region in picture (a) is the original one from train set X, picture in (b) is the original with Perlin noise superimposed X_n. Last two images are (c) the noise N standalone and (d) the map M of the superimposed noise's region.

area, which varied from example to example, rather than throughout the whole image.

6.1.1 Residual Autoencoders

An autoencoder (AE) is a neural network architecture that has been trained to replicate the input image with the exception of any final artificial noise that is added. This structure is composed by two entities: an encoder (E), which maps the input features in a latent space z, and a decoder (D), that rebuild the input from z. The network must gather the most important features from the input and distinguish them from those that fall into the noise domain in order to complete the aforementioned task, which can be carried out by restricting z to be lower than X. Since they resolve issues with degradation during training, such as the vanishing gradient problem, residual networks have been extensively investigated in the literature [44]. They have been heavily utilized in supervised networks for classification tasks, but in recent times, they have also been used in unsupervised frameworks [21, 22, 23, 24].

6.1.2 Generative Adversarial Networks

GANs have been studied with the initial purpose of generating realistic synthetic data [8]. Two networks, the discriminator and the generator (autoencoder-like networks), compete with one another during training. The discriminator evaluates if the data is phony or real, while the generator attempts to generate it. Similar to GANomaly [9], the discriminator in this instance distinguishes between the rebuild and original images.

6.1.3 GRD-Net architecture

The first component of the GRD-Net architecture is the Generator (G), which is made up of an Encoder (G_E) that encodes the input features from X and generates a latent space z; a Decoder (G_D) that decodes the latent space and attempts to replicate the input, producing an image $\hat{X}$; and a second Encoder ($G_{\hat{E}}$) that accepts $\hat{X}$ as input and creates a latent space $\hat{z}$. The AutoEncoder (AE) is made up of G_E and G_D. The Discriminator (C), the second network in the GAN model, is a convolutional encoder with a dense layer on top that performs a binary classification with the goal of classifying the input as a real or a fake image.

The generator loss is composed by three components: the adversarial, the contextual and the encoder losses.

The adversarial loss $\mathcal{L}_{adv}$ is used to smooth the instability of the Discriminator. Let $\mathcal{F}_C$ be the function that produces the output of the last convolutional layer of C, so the loss is defined as:

$$\mathcal{L}_{enc} = \mathcal{L}_2(\mathcal{F}_C(X), \mathcal{F}_C(\hat{X})), \tag{2}$$

where $\mathcal{L}_2$ is the l2-norm loss.

The contextual loss, namely the reconstruction loss, adds contextual information to the final loss. It is the combination of two subloss:

$$\mathcal{L}_{con} = w_a \cdot \mathcal{L}_1(X, \hat{X}) + w_b \cdot \mathcal{L}_{\text{SSIM}}(X, \hat{X}), \tag{3}$$

where $\mathcal{L}_1$ is the l1-norm loss and $\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(X, \hat{X})$.

The encoder loss, as explained by Akcay et al. [9], minimize the distance of the latent space z derived from X and the latent space $\hat{z}$ inferred from $\hat{X}$. It is so defined:

$$\mathcal{L}_{enc} = \mathcal{L}_1(z, \hat{z}). \tag{4}$$

The resulting loss function will hence be the following:

$$\mathcal{L}_{gen} = w_1 \cdot \mathcal{L}_{enc} + w_2 \cdot \mathcal{L}_{con} + w_3 \cdot \mathcal{L}_{enc}, \tag{5}$$

where w_a, w_b, w_1, w_2 and w_3 are three hyper-parameters used to tune the effect of individual losses.

6.2 Application specific optimizations

With everything mentioned in section 6.1 as starting point, we can start looking into out actual application: we used a number of augmentation techniques, such adding perturbation alongside Perlin noise, also some other augmentation methods: a random rotation in the interval of $[-\frac{\pi}{8}, \frac{\pi}{8}]rad$; a random vertical flip. Horizontal flip and rotation larger that $\frac{\pi}{2}$ can be considered as an anomaly.

Among the advantages of these augmentation techniques, the network's generalization capability is increased; at the same time, the downside is also the introduction of a lot of entropy into the training, which can make this phase

difficult, to the point where the training becomes unstable or diverges. For this reason we introduced a *Noise Loss*, defined as:

$$\mathcal{L}_{nse} = w_4 \cdot \mathcal{L}_1(|\overset{*}{X} - \hat{X}|, N), \tag{6}$$

where $\overset{*}{X} = P_q(X)$, is the input after the perturbation and N is the Perlin noise generated on a black image. We empirically found that this enhance the reconstruction ability beneath the noise. Moreover l1-norm loss in the contextual loss was replaced with Huber loss, that enhances the former by removing the point of non-differentiability on the origin. So it becomes:

$$\mathcal{L}_{con} = w_a \cdot \mathcal{L}_{Huber}(X, \hat{X}) + w_b \cdot \mathcal{L}_{\text{SSIM}}(X, \hat{X}), \tag{7}$$

Using as a starting point DRÆM [11] and GANomaly [9] results, and subsequently using a branch-and-bound approach we found that the best setup for the described application was:

$$\begin{aligned} w_a &= 2.0 \\ w_b &= 1.0 \\ w_1 &= 1.0 \\ w_2 &= 50.0 \\ w_3 &= 1.0 \\ w_4 &= 3.0, \end{aligned} \tag{8}$$

despite the $\mathcal{L}_{\text{SSIM}}$ loss being the one that contributes the most to reconstruction of the input image, it has a function that tends to become unstable in complex images, with high entropy, so rising w_a this phenomenon was smoothed at the cost of more convergence time.

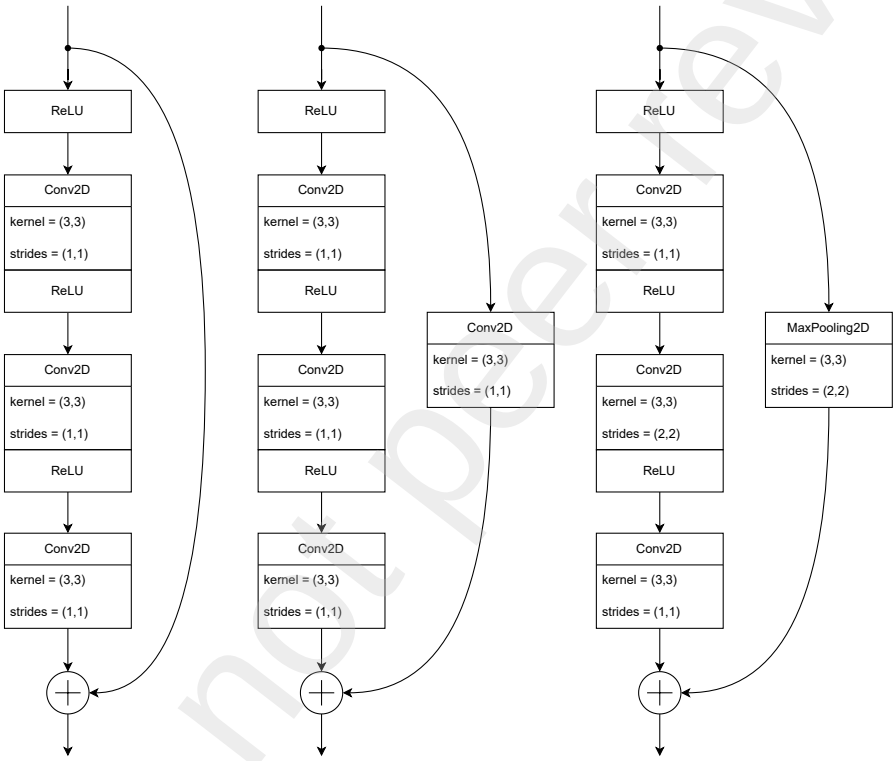
6.2.1 Network

The network used as generator for training is an encoder-decoder-encoder shaped net, where the encoder has a ResNet architecture (Figure 2), and the decoder (Figure 3) a reverse residual shape. As mentioned in section 6.1.1, residual network are widely used [44, 22, 23, 24], since they prevent several degradation problems during training, as the gradient vanishing. In the context of this work, residual structure was updated to the most recent ones, using as starting point the one designed in the previous work [21].

6.2.2 Classification and segmentation

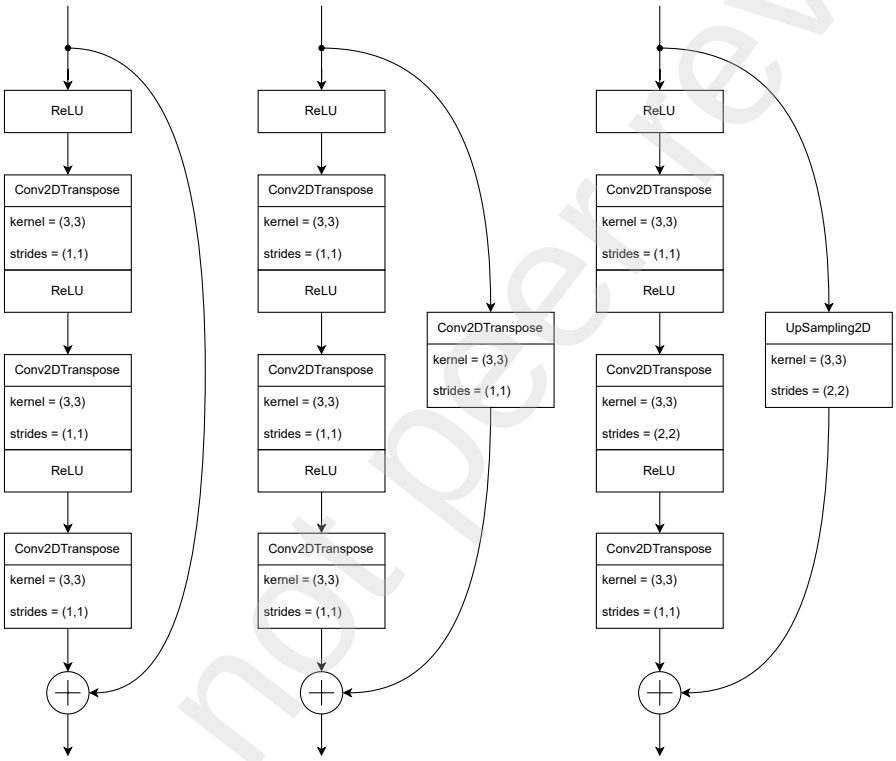
There are no segmentation requirements for this particular application; the anomalous patch and the relative heatmap are adequate to meet the project's objectives. Thus, only a generative network was used during training and inference phases. Furthermore, the division into patches makes it possible to identify the macroscopic region of the anomaly. So the anomaly score ϕ is:

$$\phi = 1 - \text{SSIM}(X, \hat{X}), \tag{9}$$



(a) Residual encoder

Figure 2: Three residual blocks in the encoder network. Only the last one halve the (H, W) size of the layer



(a) Residual decoder

Figure 3: Three residual blocks in the decoder network. Only the last one double the (H, W) size of the layer

and the heatmap is calculated as the absolute difference between input and output:

$$H = |X - \hat{X}|\Big|_0^1, \tag{10}$$

where $\Big|_0^1$ is the min-max normalization between 0 and 1:

$$H = a + b \cdot \frac{|X - \hat{X}| - \min(|X - \hat{X}|)}{\max(|X - \hat{X}|) - \min(|X - \hat{X}|)},\ a = 0 \wedge b = 1. \tag{11}$$

The optimal threshold ϕ_t that maximizes the accuracy on good examples and on anomalous real ones is determined on a subset of real production goods (both positive and negative). Consequently, if a patch is classed as a rejection during inference, it indicates that the entire product is a rejection. In this instance, 10 is used to create a heatmap for the subregion corresponding to the anomalous patch.

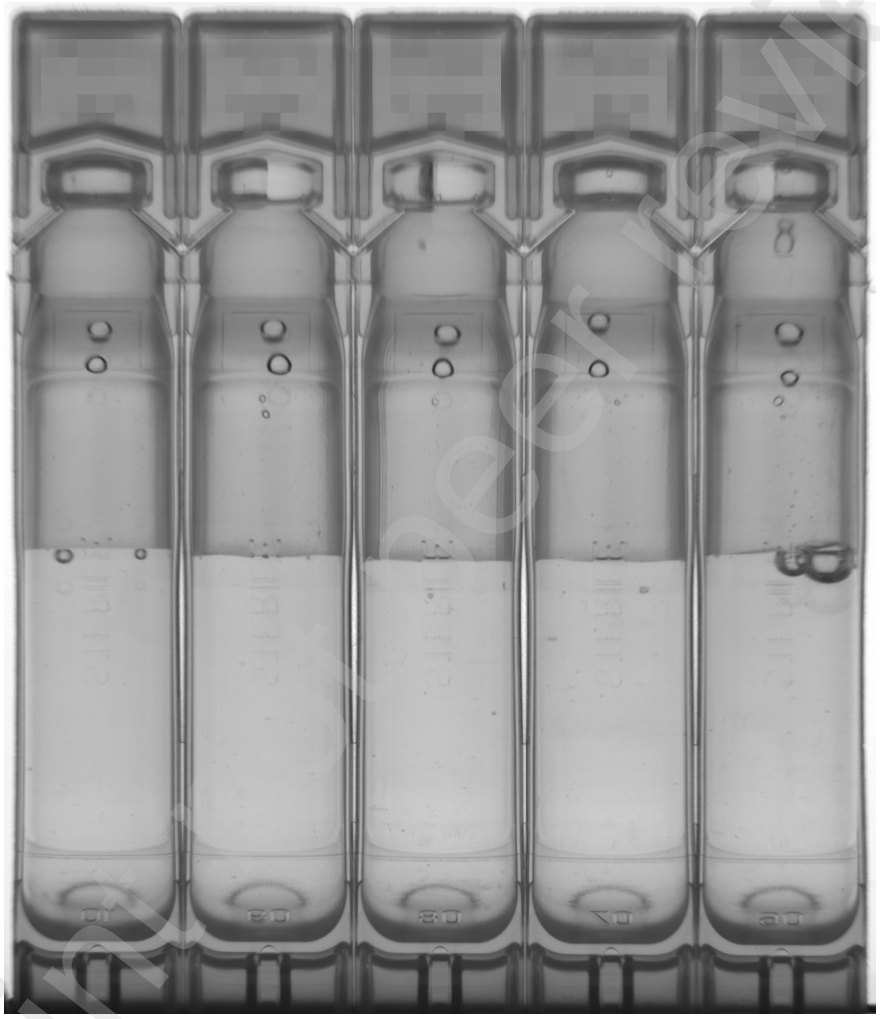
7 Experiments

Different experiments were conducted in order to optimize the result of the network and obtain a good performance, maintaining a low inference time, which can fit the required slot of 500 ms between two different acquisitions. However, in order to accurately describe the experiments the background and hardware of the machine that handles the products must be introduced.

Product The product, shown in the Picture 4[1], is a 5 vials BFS strip, filled by 10 ml of liquid excipient. It frequently causes bubbles to form in the liquid portion, particularly around the *meniscus*. Furthermore, because the *neck* is thin, some liquid may become caught inside it, and drops are typically found in the area above the liquid. As shown in the Picture 5, the product is divided into 5 vials, and each vial is further divided into 4 logic regions.. The red one (the upper), is the *flag* region, the blue one is the top *body* region, the green area is the liquid *body* region, finally the yellow area is the *bottom* region.

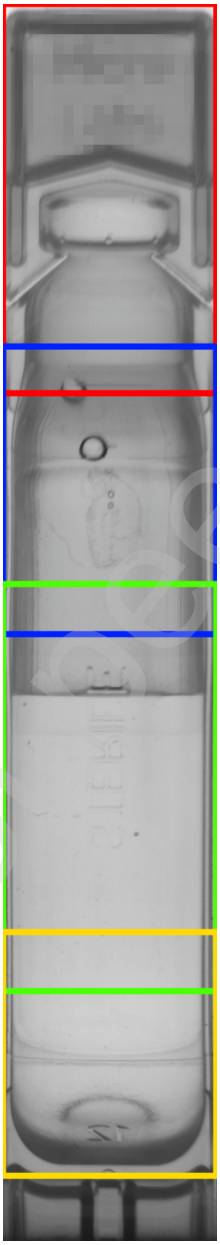
Train dataset acquisition The machine acquisition environment, where the acquisitions were carried out, was replicated at a laboratory scale; both setups used a telecentric lens, which mostly overcomes the distortion problem, found on side vials and on the bottom. Following a semicircular movement that was focused on the upper side of the vial strip, each product was acquired in this way 10 times. It required 16 frames for each acquisition in the end. At each set of frames are added two images obtained applying a *rank* filter to the entire set and selecting the lower and higher gray value from generated images by the filter

[1] Because of an NDA, the *flag* (the upper part) was blurred to conceal the company logo.



(a) The product

Figure 4: The product: a BFS strip composed by 5 vials[1].



(a) The vial regions

Figure 5: The vial regions[1]: the logic regions in which the vial is divided.

procedure. From MVtec definition of the filter[2]: Conceptually, the rank filter sorts all gray values within the mask in ascending order and then selects the gray value with rank *Rank*. The rank 1 corresponds to the smallest gray value and the rank A[3] corresponds to the largest gray value within the mask. Therefore, nothing that moved during handling is seen in the lighter image; conversely, all of the drops and bubbles that moved during handling are shown in the darker image. This technique is used to increase the variability of the positive images features. After patching every image we obtained a dataset of 2815200 images, which was then split into 90% training and 10% validation images. Each patch is a grayscale image of size $256 \times 256 \times 1$.

Test dataset acquisition A real-case set of defective strips was acquired in order to benchmark the algorithm's performance. Each acquisition is made of three frames, which are the first, the eighth, and the last in the series, respectively, to lessen the computational load on the system while it is operating. Consequently, the numerosity (N) of the batch is:

$$N = f \cdot v \cdot r = 3 \cdot 5 \cdot 4 = 60, \tag{12}$$

where f is the frame number, v is the vial number and finally, r is the region number.

Machine handling The machine is a rotating inline quality control machine that inspects the products at the end of the production line. Usually these machines are placed right after the filling device and before the packaging ones.

The product is gripped from the input conveyor belt and brought into the carousel and handled identically to how it was handled during the laboratory acquisitions. The handling and movement inside the carousel is the same as the one in the laboratory mockup, but that there is additional movement of the liquid inside due to how the product is handled before the carousel, but it has been observed on previous machines, that this further movement does not affect the performance of the algorithm, since it is much less and far away in time than the one produced on the inspection station. After handling the product, acquisitions take place, and the images begin to be processed[4].

Experiments Experiments were conducted over only 10 epochs due to the high numerosity of the training set, using the structure described in section 6 and more specifically in subsections 6.2, 6.2.1 and 6.2.2. The residual network takes inspiration from the ResNet V2 architecture [44]. In this section will be shown:

1. The hardware and software environment

[2] `https://www.mvtec.com/doc/halcon/2305/en/rank_image.html`

[3] In this case $A = 16$

[4] More specific details about machine hardware cannot be provided due to the NDA and company policies

2. The architecture of the used network
3. The training phase
4. The results in terms of accuracy over the positive and the negative examples in the test dataset.
5. The results in terms of time and throughput
6. Experiments on the generative network

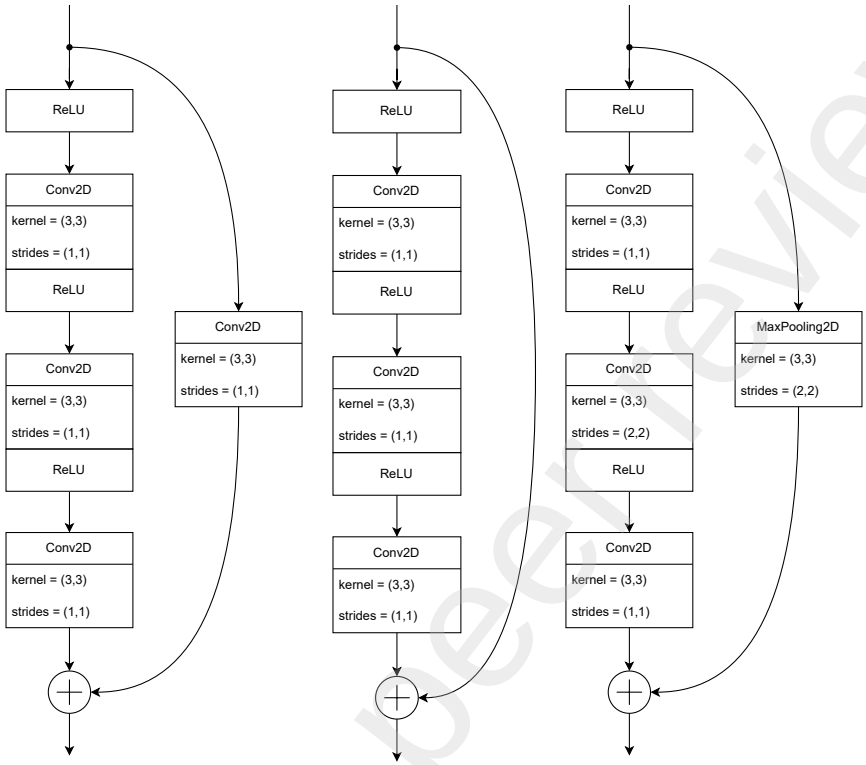
7.1 Hardware

The hardware, as already mentioned in subsection 4.3 is very different between the training server and the inference machine. The training process is carried out on a server that has been set up in this way:

- Hardware
  - Intel® Xeon® Silver 4216 CPU @ 2.10 GHz as CPU
  - 64GB of DDR4 Synchronous @ 3200 MHz as system memory
  - Nvidia® A100 with 40GB of VRAM as GPU.
  - 2 TB M.2 NVMe SSD
- Software
  - Ubuntu 22.04.3 LTS Server minimal-based o/s with 5.15.0-94-x86_64 kernel
  - Nvidia® driver Version: 535.104.12
  - CUDA® version: 12.2
  - TensorFlow 2.13.1 compiled from source

The inference machine is an industrial cluster installed on board and it is set up in this way:

- Hardware
  - Intel® Xeon® E-2278GE CPU @ 3.30 GHz as CPU
  - 32GB of DDR4 Synchronous @ 3200 MHz as system memory
  - Nvidia® A4500 with 20GB of VRAM as GPU.
  - 32 GB CFast flash for the operative system (read-only mounted)
  - 1 TB M.2 NVMe SSD for the data
- Software
  - Ubuntu 22.04.3 LTS Server minimal-based o/s with 5.15.0-94-x86_64 kernel
  - Nvidia® driver Version: 535.104.12
  - CUDA® version: 12.2
  - TensorFlow 2.13.1 compiled from source



(a) ResEnc Black A (b) ResEnc Block B (c) ResEnc Block C

Figure 6: Three residual blocks in the encoder (resEnc) network. 1 stage is composed by 3 convolutional blocks in the order: A, B, C

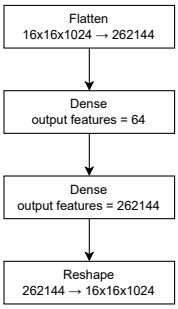
7.2 Network architecture

Encoder As mentioned in subsection 6.2.1, the network is residual based on ResNet version 2 architecture. Each stage is composed of 3 residual blocks:

1. the first (block A in the schema 6) concatenates the results of three consecutive convolutions with 3×3 kernel with the result of a single convolution with 1×1 kernel. In this case there are no reduction in the input $H_i \times W_i$ embedding size.

2. the second (block B in the schema 6) concatenates the results of three consecutive convolutions with 3×3 kernel with the input received from the block A. In this case there are no reduction in the input $H_i \times W_i$ embedding size.

3. the third (block C in the schema 6) concatenates the results of three consecutive convolutions with 3×3 kernel with the output of a downsampling



(a) Dense bottleneck

Figure 7: The bottleneck stage.

block, the middle convolution halves the sizes H_i and W_i of the input embedding. In this case H_i and W_i are both halved.

In total the network is composed by 4 stages, in order to obtain a final size of 16×16 and 1024 filters.

Bottleneck The bottleneck is fully-connected with a size of 64 features, as in figure 7.

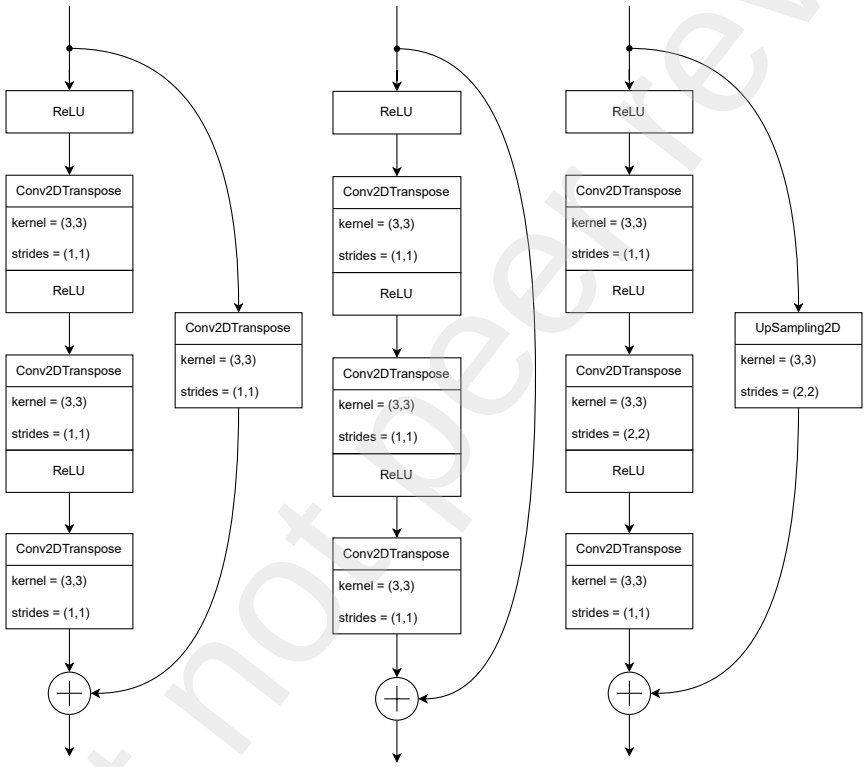
Decoder The decoder is the inverse architecture of the encoder, described above. Same as with the encoder, each stage is composed by 3 residual blocks:

1. the first (block A in the schema 6) concatenates the results of three consecutive convolutions transposed with 3×3 kernel with the result of a single convolution with 1×1 kernel. In this case there are no increasing in the input $H_i \times W_i$ embedding size.

2. the second (block B in the schema 6) concatenates the results of three consecutive convolutions transposed with 3×3 kernel with the input received from the block A. In this case there are no increasing in the input $H_i \times W_i$ embedding size.

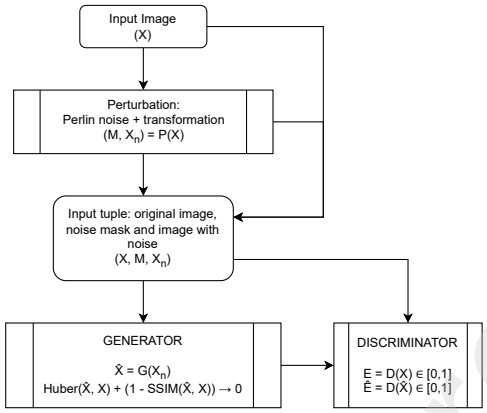
3. the third (block C in the schema 6) concatenates the results of three consecutive convolutions transposed with 3×3 kernel with the output of a upsampling block, the middle convolution double the sizes H_i and W_i of the input embedding. In this case H_i and W_i are both doubled.

In total the decoder network is composed by 4 stages, in order to obtain a final size of 256×256 and 1 channel, with a Sigmoid function as activation.



(a) ResDec Black A (b) ResDec Block B (c) ResDec Block C

Figure 8: Three residual blocks in the decoder (ResDec) network. 1 stage is composed by 3 convolutional transposed blocks in the order: A, B, C



(a) Train flowchart

Figure 9: The training flowchart.

7.3 Training phase

Training follows a GAN pattern, and is therefore composed by two steps: the training of the generator and the training of the discriminator. The starting learning rate is 1.5×10^{-4} with a cosine decay policy with restart every 2533680 steps at $\frac{1}{3}$ of last maximum learning rate and the optimizer used is for both GAN's networks the Adam algorithm. The batch size is 32 images and the probability to apply a perturbation to an input patch is $q = 0.75$. The training flowchart is illustrated in picture 9.

7.4 Results

Experiments were performed on test dataset, namely the knapp test kit provided by the client. The set contains 141 defects and 120 positive products, since the dataset is fairly balanced, accuracy is the chosen metric, but also true negative, true positive ratios, mean and max inference time are calculated. The results are presented on 3 stages:

1. Overall accuracy on the whole test dataset and inference time per patch
2. Accuracy after image aggregation per product and inference time per product
3. Accuracy, after image aggregation per product and run, based on acceptance policy of the client

Also some images are presented in Appendix A.

Acceptance policy To be validated, the machine, has to perform better or the same than the human operators on the knapp test kit provided by the

Overall results					
	R0	**R1**	**R2**	**R3**	**Vial**
Threshold	0.015589	0.038017	0.046568	0.029593	-
Accuracy	0.9919	0.9926	0.9957	0.9991	0.9870
True positive ratio	0.9966	0.9985	0.9986	0.9994	0.9942
True negative ratio	0.9093	0.9044	0.9779	0.9973	0.9581
Mean inference time (ms)	0.1689	0.1689	0.1689	0.1689	-

Table 1: Overall results using the equation 9 as anomaly score.

client. As mentioned, each product of the knapp kit is acquired 10 times, each acquisition is defined as a run. If a product is included in the regular set and gets a positive classification at least seven times out of ten, it is said to be correctly classified. Similarly, if an abnormal product receives a negative classification seven times out of ten, it is deemed properly classified. If not, a product will be considered misclassified.

Overall results Overall results are presented in table 1. A threshold is calculated for each region, since there is a significant correlation between every image that belongs to the same region, and hence a similar distribution between pixels. In this context the mean inference time is also calculated as $\mu_{t_f} = \frac{\mu_{t_b}}{f \cdot v \cdot r} = \frac{\mu_{t_b}}{60}$, where μ_{t_f} is the mean time per frame f, μ_{t_b} is the mean time per batch, v is the vials numerosity per strip and r is the number of region for each vial.

Per product aggregation The classification of a whole strip of vials is designed to minimize the GMP[5] impact of the error, thus if at least one region is labeled as rejection, the whole product is considered the same. From these considerations, the false positive ratio decreases to the detriment of the false rejection ratio which increase a lot; hence a new thresholds tuning is needed in order to satisfy the project requirements. To this end the final score function per vial becomes:

$$\theta_v = \max(\left\{1.0 - \text{SSIM}(X_i, \hat{X}_i)\right\}),\ i \geq 0\ \wedge\ i < f \cdot v \cdot r, \tag{13}$$

where i is the patch index.

Per run aggregation In order to be validated, the algorithm must overcome the manual inspectors' performance, conducted by the client on the same knapp test kit. Every product has to be acquired 10 times and in order to be considered well classified its label must be confirmed at least 7 times out of 10 (70%). Results are shown in table 3.

[5]Good Manufacturing Practice (GMP) describes the minimum standard that a medicines manufacturer must meet in their production processes.

Per strip results	
	Score
Threshold R0	0.016156
Threshold R1	0.038584
Threshold R2	0.046635
Threshold R3	0.029660
Accuracy	0.9593
True positive ratio	0.9694
True negative ratio	0.9467
Mean inference time	0.4873

Table 2: Per strip results using the equation 13 as anomaly score.

Per strip results	
	Score
Threshold R0	0.013222
Threshold R1	0.034650
Threshold R2	0.046201
Threshold R3	0.029226
Accuracy	0.9641
True positive ratio	0.9676
True negative ratio	0.9599

Table 3: Per run results using the equation 13 as anomaly score.

8 Conclusions and Future work

We developed an architecture that performs efficiently in an extremely demanding industrial environment, where performance has both business and GMP impact, which in turn affects people's safety and health. The proposed method manages a large dataset while adhering to hardware and time limits and striking a fair balance between costs and outstanding performance. In order to identify the primary features of the product and identify out-of-distribution anomalies, it completes the denoising process during training. With the use of this strategy, the network can be forced to perform better than a traditional autoencoder when it comes to compressing and eliminating extraneous features from the result. In fact, because autoencoder-based networks, like GANomaly, have learnt too smoothly to replicate the source image, they frequently have the issue of being able to repeat the anomaly region even in the reconstructed image. Therefore, selecting the noise to be superimposed on the input image becomes crucial since the more similar it is to the distribution of potential anomalies, the higher the yield will be in terms of detecting anomalies. Modern generative models were used as a first instance in the architecture's development, losses and base network design was optimized in order to overcome difficult challenges like the large size of the acquired images and the high variance between positive examples brought on by the movement of the liquid inside the vials. A heatmap for defective items is displayed on the HMI to give the operator a visual explanation of the outcome. The "hot" parts of the heatmap indicate the most anomalous locations in terms of pixels. This architecture was designed jointly by University of Ferrara and Bonfiglioli Engineering, while tests were conducted on the Bonfiglioli Engineering's servers.

While this research provides insightful information and specifics about how to integrate and implement an out-of-distribution anomaly detection via deep learning algorithm, several areas warrant further exploration. Nowadays, operators frequently need additional explanations in addition to an anomaly score and the associated categorization in order to assess the efficiency of the production line and the machine itself.

Several solution have been provided in the context of the embedding similarity-based approach, as in PaDiM [14], PatchCore [15] or EfficientAD [42], where the distance between the positives embedding collected during training, re-scaled to the original image size, constitute a visual representation of the anomalous regions, training a threshold on test dataset can be obtained a pixel-wise classifier. This is often accomplished by calculating a distance between the inferred embeddings of the input, and the knowledge base, which may be arranged in many ways, such as positive clusters or a stochastic process. Similarly, reconstruction-based networks provide a *heatmap* defined as the absolute value of the difference between the original and the rebuilt images, whereas the differences correspond to the *hottest* parts in the map.

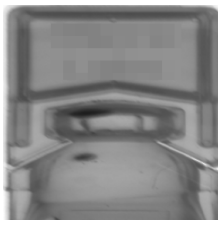
The biggest drawback in both situations is the non-parametric classifier's inability to delve deeper into the result's perceptual interpretation. Further architectures, such as GRD-Net [21], DRÆM [11], the described deep perceptual

autoencoder by N. Shvetsova et al. [45], or the proposed one by P. Bergmann et al. [46], have been researched and developed in order to accomplish the aim of identifying defects, within images, interpreting the inferred features. The mentioned architecture prove to have excellent performance, but with the drawback of extreme computational and architectural complexity, which makes them challenging to use in practical situations, particularly when accessed online.

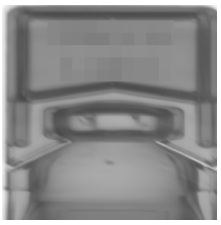
Considering the aforementioned, a promising avenue for future research is to investigate how to obtain information from the latent representation of the input example, as a sample of the stochastic process, that might be processed to produce a visual representation of the anomalous regions within the provided image. An excellent starting point can be taken from the work carried out by P. Esser et al. [35], where a discrete representation of the image through a vector quantized variational autoencoder (VQVAE), is used to generate high resolution synthetic images, introducing to this end the use of transformers for imagine encoding, which does not have a strong local correlation within images, in contrast to CNN. However, as the products in this work are the result of an industrial process, their shape and position vary very little. For this reason, a topological connection between the input image and the derived features may help identify anomalies that deviate from the distribution.

A Result Images

Below are shown examples taken from real cases analyzed during algorithm validation phase.

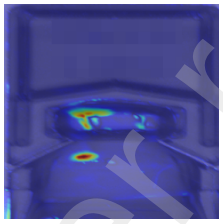


(a) Original vial region image (X)



(b) The rebuild image $\hat{X}$

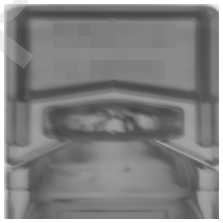
(c) The difference image



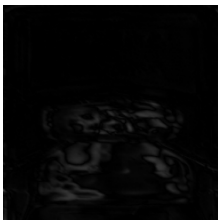
(d) The heatmap H superimposed

Figure 10: Stuck particle in the upper part of the product

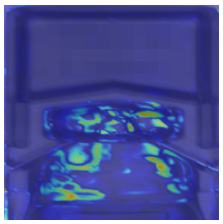
(a) Original vial region image (X)



(b) The rebuild image $\hat{X}$

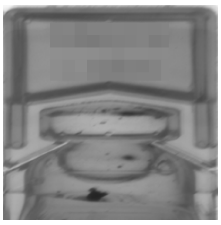


(c) The difference image



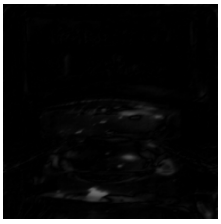
(d) The heatmap H superimposed

Figure 11: Stuck particle and anomalous liquid behavior that results in foam formation because of contamination

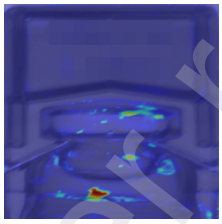


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

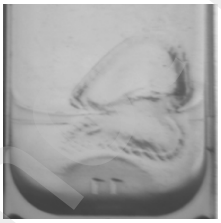


(c) The difference image

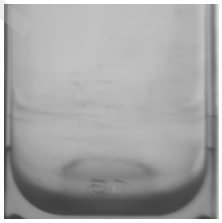


(d) The heatmap H superimposed

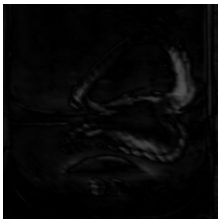
Figure 12: Stuck particle and anomalous liquid behavior that results in foam formation because of contamination



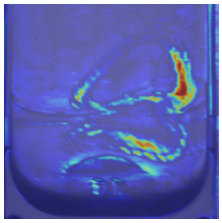
(a) Original vial region image (X)



(b) The rebuilt image $\hat{X}$

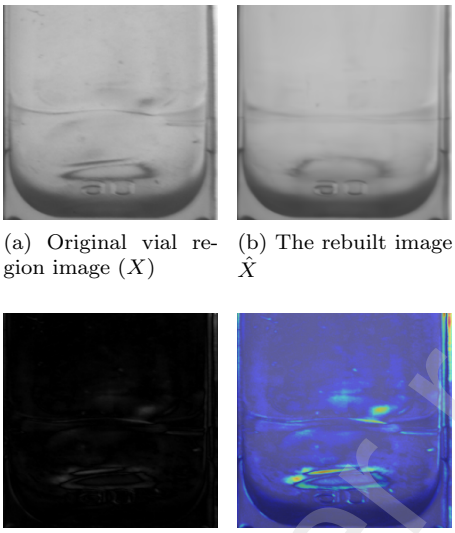


(c) The difference image



(d) The heatmap H superimposed

Figure 13: Lower body deformation



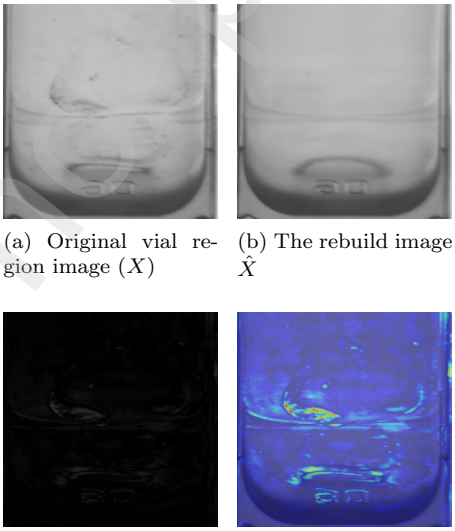
(a) Original vial region image (X)

(b) The rebuilt image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 14: Light lower body deformation



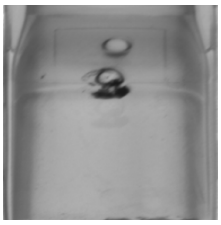
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

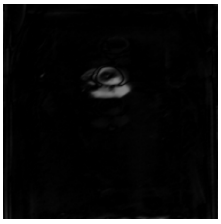
(d) The heatmap H superimposed

Figure 15: Lower body deformation

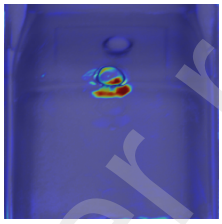


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

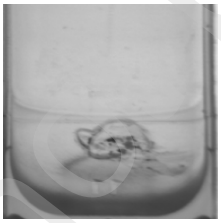


(c) The difference image



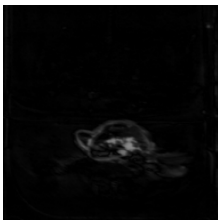
(d) The heatmap H superimposed

Figure 16: Black spot on upper part of the vial's body.

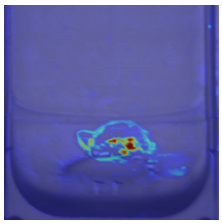


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

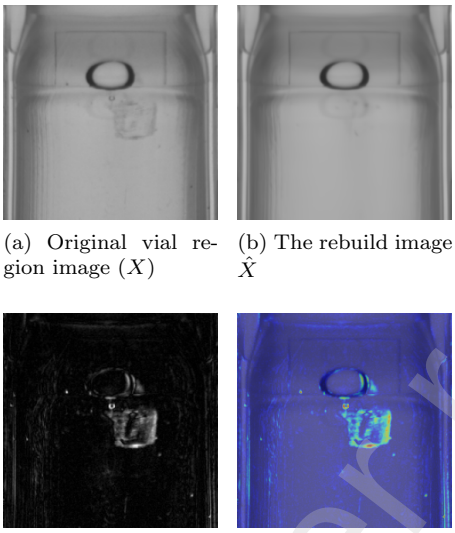


(c) The difference image



(d) The heatmap H superimposed

Figure 17: Burn in the lower part of the vial.



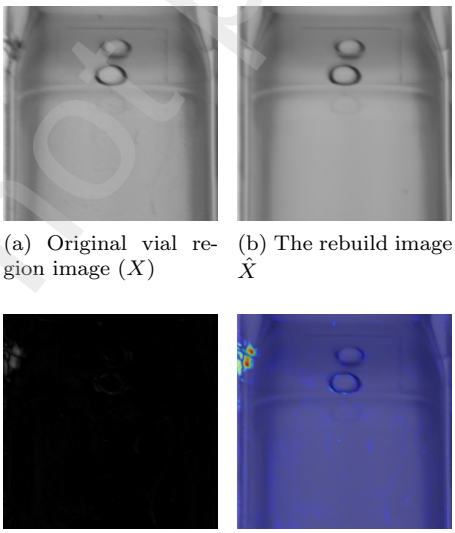
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 18: Light scratch near the product neck.



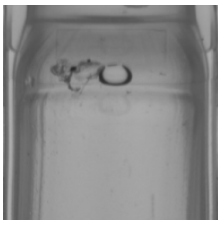
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

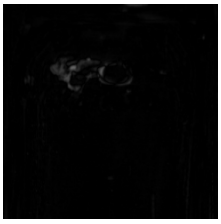
(d) The heatmap H superimposed

Figure 19: External imperfection caused by laser cutting of the product.

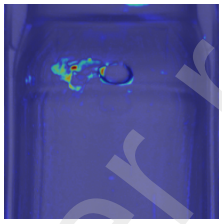


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

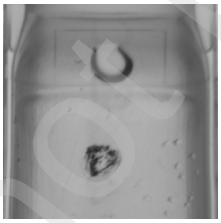


(c) The difference image



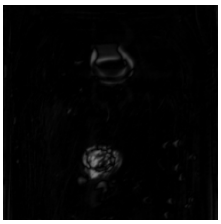
(d) The heatmap H superimposed

Figure 20: Deep scratch on the upper part of the vial's body.

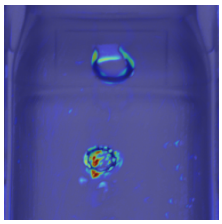


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

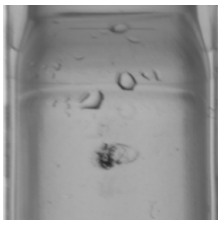


(c) The difference image



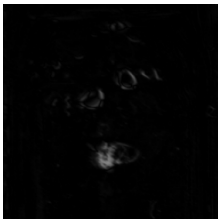
(d) The heatmap H superimposed

Figure 21: Burn on the vial's body. It is reproduced as a bubble, since the network does not know how to represent the defect features.

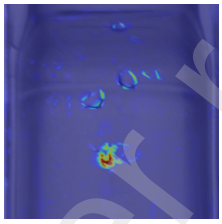


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

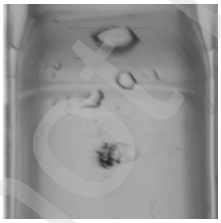


(c) The difference image



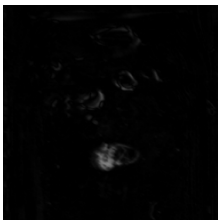
(d) The heatmap H superimposed

Figure 22: Black spot on the vial's body

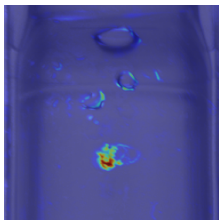


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

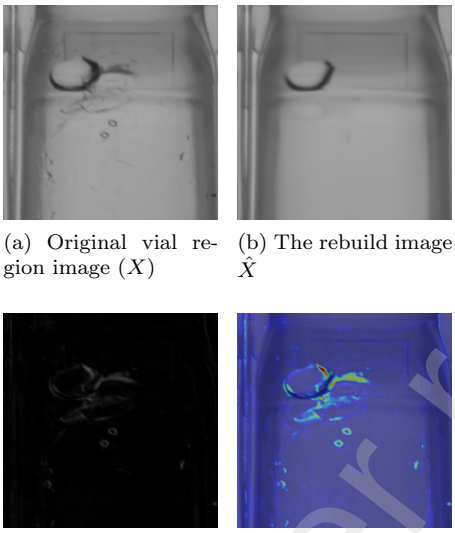


(c) The difference image



(d) The heatmap H superimposed

Figure 23: Black spot of the vial's body



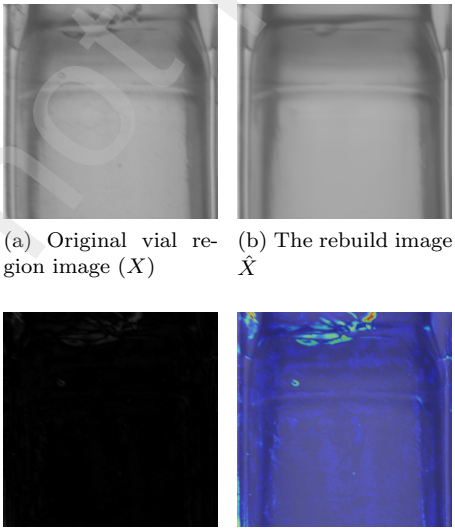
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 24: Bump on the lower part of the vial's neck, near to a big bubble stuck in the internal layer of the BFS surface.



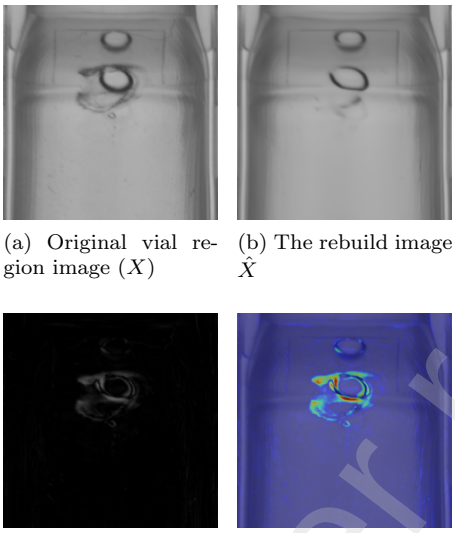
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 25: Light bump on the vial's neck



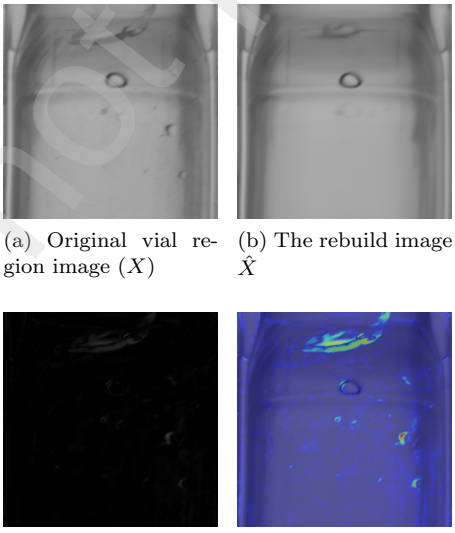
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 26: Heavy deformation near the neck region, overlapped to a bubble that the network is still able to reproduce.



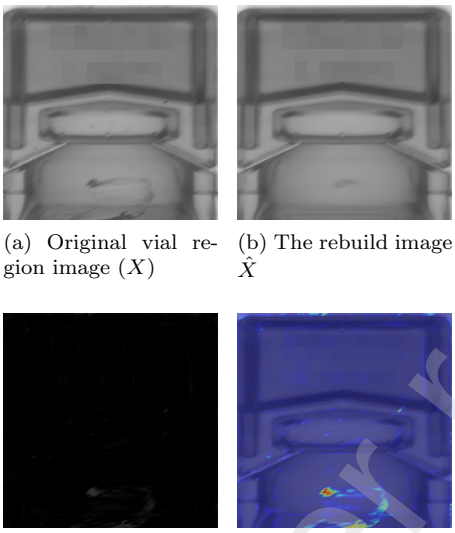
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 27: Scratch on the vial's neck.



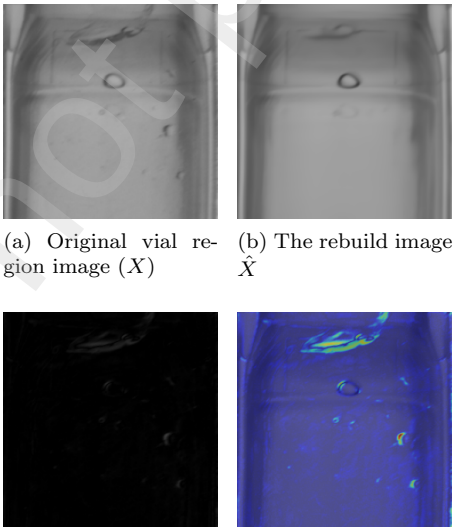
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 28: Light bump on the lower part of the vial's cap region.



(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure 29: Scratch on the vial's neck.

References

[1] Paul Mooijman, Cagatay Catal, Bedir Tekinerdogan, Arjen Lommen, and Marco Blokland. The effects of data balancing approaches: A case study. *Applied Soft Computing*, 132:109853, 2023.

[2] Hongzuo Xu, Guansong Pang, Yijie Wang, and Yongjun Wang. Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2023.

[3] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomalib: A deep learning library for anomaly detection, 2022.

[4] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020.

[5] Umberto Michelucci. An introduction to autoencoders. *CoRR*, abs/2201.03898, 2022.

[6] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[9] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pages 622–637. Springer, 2018.

[10] Samet Akçay, Amir Atapour-Abarghouei, and Toby P Breckon. Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[11] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Draem-a discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8330–8339, 2021.

[12] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[14] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*, pages 475–489. Springer, 2021.

[15] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2022.

[16] David Muhr, Michael Affenzeller, and Josef Küng. A probabilistic transformation of distance-based outliers, 2023.

[17] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows, 2021.

[18] Jaehyeok Bae, Jae-Han Lee, and Seyun Kim. Pni : Industrial anomaly detection using position and neighborhood information, 2023.

[19] Yixuan Zhou, Xing Xu, Jingkuan Song, Fumin Shen, and Heng Tao Shen. Msflow: Multi-scale flow-based framework for unsupervised anomaly detection, 2023.

[20] Berend Denkena, Marc-Andre Dittrich, Hendrik Noske, and Matthias Witt. Statistical approaches for semi-supervised anomaly detection in machining. *Production Engineering*, 14, 03 2020.

[21] Niccolò Ferrari, Michele Fraccaroli, and Evelina Lamma. Grd-net: Generative-reconstructive-discriminative anomaly detection with region of interest attention module. *International Journal of Intelligent Systems*, 2023:1–18, 09 2023.

[22] Chathurika S. Wickramasinghe, Daniel L. Marino, and Milos Manic. Resnet autoencoders for unsupervised feature learning from high-dimensional data: Deep models resistant to performance degradation. *IEEE Access*, 9:40511–40520, 2021.

[23] Simone Zini, Simone Bianco, and Raimondo Schettini. Deep residual autoencoder for blind universal jpeg restoration. *IEEE Access*, 8:63283–63294, 2020.

[24] Viet-Tuan Le and Yong-Guk Kim. Attention-based residual autoencoder for video anomaly detection. *Applied Intelligence*, 53(3):3240–3254, Feb 2023.

[25] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection, 2021.

[26] Xuan Xia, Xizhou Pan, Nan Li, Xing He, Lin Ma, Xiaoguang Zhang, and Ning Ding. Gan-based anomaly detection: A review. *Neurocomputing*, 493:497–535, 2022.

[27] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *arXiv preprint arXiv:1807.02011*, 2018.

[28] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.

[29] Shashanka Venkataramanan, Kuan-Chuan Peng, Rajat Vikram Singh, and Abhijit Mahalanobis. Attention guided anomaly localization in images. In *European Conference on Computer Vision*, pages 485–503. Springer, 2020.

[30] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. *Advances in neural information processing systems*, 31, 2018.

[31] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3379–3388, 2018.

[32] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019.

[33] Xin Xie, Yuhui Huang, Weiye Ning, Dengquan Wu, Zixi Li, and Hao Yang. Rdad: A reconstructive and discriminative anomaly detection model based on transformer. *International Journal of Intelligent Systems*, 37(11):8928–8946, 2022.

[34] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. IEEE, June 2021.

[35] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021.

[36] Yonglong Tian, Olivier J. Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data, 2021.

[37] Oliver Rippel, Patrick Mertens, and Dorit Merhof. Modeling the distribution of normal data in pre-trained deep features for anomaly detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6726–6733. IEEE, 2021.

[38] Liron Bergman, Niv Cohen, and Yedid Hoshen. Deep nearest neighbor anomaly detection. *arXiv preprint arXiv:2002.10445*, 2020.

[39] Paolo Napoletano, Flavio Piccoli, and Raimondo Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors*, 18(1):209, 2018.

[40] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020.

[41] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[42] Kilian Batzner, Lars Heckler, and Rebecca König. Efficientad: Accurate visual anomaly detection at millisecond-level latencies, 2023.

[43] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[45] Nina Shvetsova, Bart Bakker, Irina Fedulova, Heinrich Schulz, and Dmitry V. Dylov. Anomaly detection in medical imaging with deep perceptual autoencoders. *IEEE Access*, 9:118571–118583, 2021.

[46] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *International Journal of Computer Vision*, 130(4):947–969, Apr 2022.

Contents