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

Niccolò Ferrari[1,2*], Nicola Zanarini[2], Michele Fraccaroli[1], Alice Bizzarri[1], Evelina Lamma[1]

[1*]Department of Engineering, University of Ferrara, Via Saragat 1, Ferrara, 44122, Italy.
[2]Bonfiglioli Engineering, Via Amerigo Vespucci 20, Ferrara, 44124, Italy.

*Corresponding author(s). E-mail(s): niccolo.ferrari@unife.it;

Abstract

Industrial visual inspection in pharmaceutical production requires high accuracy under strict constraints on cycle time, hardware footprint, and operational cost. Manual inline inspection is still common, but it is affected by operator variability and limited throughput. Classical rule-based computer vision pipelines are often rigid and difficult to scale to highly variable production scenarios. To address these limitations, we present a semi-supervised anomaly detection framework based on a generative adversarial architecture with a residual autoencoder and a dense bottleneck, specifically designed for online deployment on a high-speed Blow-Fill-Seal (BFS) line. The model is trained only on nominal samples and detects anomalies through reconstruction residuals, providing both classification and spatial localization via heatmaps. The training set contains 2,815,200 grayscale patches. Experiments on a real industrial test kit show high detection performance while satisfying timing constraints compatible with a 500 ms acquisition slot.

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

1 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.

2 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

3 Introduction

Industrial applications increasingly require machine learning solutions for demanding tasks such as inline visual anomaly detection. In real deployments, models must satisfy stringent constraints on hardware resources and throughput while preserving high inspection accuracy. As a consequence, technical feasibility is as relevant as raw model performance.

Anomaly detection for machine vision is particularly relevant in the pharmaceutical sector, where non-destructive quality controls must be both accurate and fast. For this reason, pharmaceutical manufacturers are investing in integrating deep learning systems into production-line automation. In practice, system design must balance the following constraints:

1. Accuracy (missed rejects and false rejects)
2. Hardware constraints
3. Total cost of ownership

Accuracy has a dual impact: it directly affects patient safety and, in parallel, business outcomes. Hardware and cost are correlated but not equivalent; hardware choices influence footprint, maintainability, and integration complexity in addition to direct expenditure.

In many production sites, quality control still relies on manual inspection. This process is vulnerable to non-systematic errors, including oversight and attention loss, and therefore limits both consistency and throughput. Cosmetic inspection and particle detection are especially challenging because decisions rely on perceptual cues that are difficult to formalize analytically.

Classical procedural algorithms are typically developed ad hoc from limited sets of compliant and anomalous examples. They rely on handcrafted thresholds, heuristic pattern matching, and many tunable parameters. Although this may appear flexible, the resulting systems are usually tightly coupled to a specific product and image configuration, hence difficult to scale or transfer. In addition, oversimplified analytical models often fail to separate process noise from true defects (e.g., moving bubbles vs. foreign particles), making robust deployment difficult.

Deep learning is currently the state of the art for real-time anomaly detection in machine vision. Supervised classification is mature and widely studied; however, it is most effective when classes are balanced [28].

In industrial production, compliant samples largely outnumber anomalous ones. This imbalance makes supervised training problematic. A practical alternative is semi-supervised learning, where the model is trained only on nominal data and anomalies are identified as out-of-distribution patterns [3, 43]. Most methods fall into two broad families: *reconstruction-based* and *embedding similarity-based*.

3.1 Reconstruction-based

This family is based on the assumption that a model trained only on conforming samples cannot faithfully reconstruct anomalous regions. It has been extensively studied because it enables robust reconstruction subspaces without defective training samples. Representative models include autoencoders (AE) [5, 26], variational autoencoders (VAE) [22, 23], generative adversarial networks (GAN) [18], and GAN-based variants such as GANomaly [1, 2] and DRÆM [45]. Since anomalous regions are out of distribution with respect to training data, they are typically reconstructed poorly and can be detected by thresholding residuals or similarity measures such as SSIM [39]. These architectures are usually computationally heavy in both training and inference, but they produce compact and manageable latent representations.

3.2 Embedding similarity-based

This family uses deep neural networks, often pre-trained on large-scale datasets, to extract embedding vectors that summarize visual content. The underlying idea is that expressive backbones, such as residual networks [19], can provide robust feature representations even when the target dataset is limited. Regularity can then be modeled with methods such as PaDiM [12], PatchCore [29, 34], and normalizing-flow approaches including FastFlow [44], PNI [4], and MSFlow [46]. In many formulations, nominal data are approximated with multivariate Gaussian models [13]. The main drawback is interpretability: the anomaly score is a distance in feature space rather than a direct reconstruction error [12]. In addition, memory requirements can become prohibitive on embedded or on-board hardware because latent-space storage often grows with dataset size.

3.3 Adopted solution

Based on the considerations above, we developed an inline real-time anomaly detection system for cosmetic inspection of BFS plastic vial strips filled with liquid. The main

objective was not only high detection accuracy, but also feasibility under strict 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 proposed architecture is adapted from previous work [16] to satisfy hardware limits and acceptance-test constraints. The model uses a single GAN with a residual autoencoder (RAE) and a dense bottleneck (DRAE). As in DRAEM, Perlin noise is randomly superimposed on input images during training, not to synthesize defects, but to improve denoising behavior and reduce the tendency of the model to copy the input.

The training dataset contains 2,815,200 images obtained from 782 strips of 5 vials each. Every strip was acquired 10 times with 16 frames per acquisition, plus 2 ranked images. Each image was divided into 20 patches: 5 vial regions, each split into 4 sub-regions.

In summary, the contribution of this work is:

1. A generative network based on a residual autoencoder (RAE) [24, 40, 47] with a fully-connected bottleneck, embedded in a GAN architecture [1].
2. A preprocessing pipeline that partitions each vial into four logical regions for patch-level analysis.
3. Training-time augmentation on nominal samples, including randomized Perlin-noise masking to improve robustness against out-of-distribution perturbations.
4. A multi-stage evaluation protocol from raw patches to vial- and run-level aggregation, aligned with industrial acceptance criteria.
5. Online deployment of the inference pipeline in the machine control software through C++ TensorFlow APIs.

4 Related Work

The implemented approach is primarily based on GRD-Net [16], which is inspired by DRÆM [45]. This family belongs to reconstruction-based anomaly detection, where models reconstruct nominal inputs and expose out-of-distribution regions through reconstruction mismatch [25, 41]. Representative architectures include autoencoders (AE) [5, 8, 17, 26], variational autoencoders (VAE) [23, 38], and GAN-based methods [1, 2, 31, 35]. In these methods, the encoder maps the image into a compact latent space [5]. SSIM [8] and pixel-wise reconstruction errors [9] are commonly used both for training and for anomaly scoring at inference. Transformer-based reconstruction models

have also been proposed [27, 42], including hybrid convolutional-transformer designs [15]. A key advantage of reconstruction-based methods is interpretability through residual maps; a typical drawback is computational cost. Contrastive extensions have shown additional performance gains [37].

A second major family is *embedding similarity-based* anomaly detection. These approaches extract feature vectors for image-level anomaly detection [7, 33] or patch-level localization [30]. Compared with reconstruction-based methods, they are often lighter at inference because they mainly require an encoder. However, they are less interpretable: scores are distances between inferred embeddings and the nominal embedding distribution. Early work such as SPADE [11] modeled normality with hypersphere-based representations. Later methods include PaDiM [12], PatchCore [34], and normalizing-flow variants [4, 14, 44]. These techniques can deliver high throughput, but memory-bank requirements may become prohibitive on constrained industrial hardware. Moreover, reconstruction-based methods can better capture certain logical defects, as discussed by Batzner et al. [6].

5 Methods

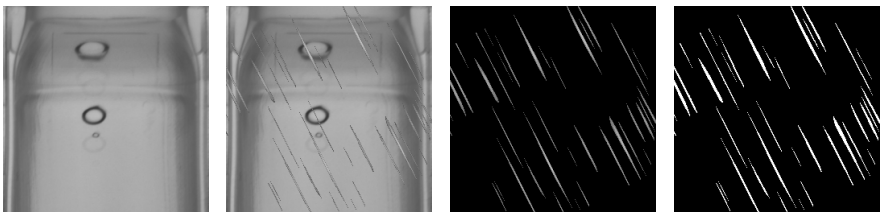
To describe our approach, we first summarize the GRD-Net [16] architecture.

5.1 GRD-Net

Inspired by DRÆM [45], GRD-Net [16] is composed of two networks. The first is a GAN with a fully-convolutional residual autoencoder (CRAE). The second is a U-Net with skip connections that takes the original image X and the reconstructed image $\hat{X}$ as input. During training, a perturbation function P_q superimposes Perlin noise on X with probability q, generating a perturbed image X_n and a binary map M. In M, perturbed pixels are set to 1.0 and unperturbed pixels to 0.0. The original X is unchanged, while X_n is generated as:

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

In this design, perturbation acts as an augmentation mechanism and forces the autoencoder to solve a denoising task in addition to feature compression. It



(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

Fig. 1 (a) Original vial-region image X. (b) Same image with superimposed Perlin noise X_n. (c) Isolated Perlin noise map N. (d) Binary mask M of the superimposed region.

also produces the supervision map used by the second network for anomaly-region prediction.

The superimposed noise forces the network to reconstruct masked content and therefore emphasizes essential structure learning. Similar principles were investigated, for example, in masked autoencoding approaches [21].

In the initial project, a Region of Interest (ROI) attention map was also used during training to focus learning on predefined areas that vary across samples.

5.1.1 Residual Autoencoders

An autoencoder (AE) is trained to reconstruct an input image, ideally filtering out nuisance noise. It is composed of an encoder (E), which maps input features into a latent representation z, and a decoder (D), which reconstructs the input from z. To succeed, the model must preserve informative structure while discarding non-relevant variability. This is typically encouraged by constraining the latent dimensionality. Residual architectures are widely adopted because they mitigate optimization degradation (e.g., vanishing gradients) [20]. Although historically dominant in supervised tasks, they are now broadly used in unsupervised and semi-supervised settings [16, 24, 40, 47].

5.1.2 Generative Adversarial Networks

GANs were originally proposed for realistic synthetic data generation [18]. They consist of a generator and a discriminator trained in competition: the discriminator distinguishes real from generated samples, while the generator learns to fool it. As in GANomaly [1], the discriminator in our setting compares reconstructed and original images.

5.1.3 GRD-Net architecture

The first component of GRD-Net is the Generator (G), composed of an Encoder (G_E) that maps X to a latent space z, a Decoder (G_D) that reconstructs $\hat{X}$ from z, and a second Encoder ($G_{\hat{E}}$) that maps $\hat{X}$ to $\hat{z}$. The autoencoder corresponds to G_E+G_D. The Discriminator (C) is a convolutional encoder followed by a dense layer that performs binary real/fake classification.

The generator objective is composed of three terms: adversarial, contextual, and encoder-consistency losses.

The adversarial term $\mathcal{L}_{adv}$ stabilizes optimization by matching feature activations from C. Let $\mathcal{F}_C$ denote the output of the last convolutional layer of C:

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

where $\mathcal{L}_2$ is the ℓ_2 loss.

The contextual (reconstruction) loss combines pixel-level and structural terms:

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

where $\mathcal{L}_1$ is the ℓ_1 loss and $\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(X, \hat{X})$.

The encoder-consistency loss, as in Akcay et al. [1], minimizes the distance between z from X and $\hat{z}$ from $\hat{X}$:

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

The final generator objective is:

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

where w_a and w_b weight the contextual subterms, and w_1, w_2, w_3 weight the three components of $\mathcal{L}_{gen}$.

5.2 Application specific optimizations

Starting from the framework described in section 5.1, we introduced application-specific augmentations: Perlin-noise perturbation, random rotation in $[-\frac{\pi}{8}, \frac{\pi}{8}]$ rad, and random vertical flip. Horizontal flips and rotations larger than $\frac{\pi}{2}$ were excluded because they can generate unrealistic anomalous patterns.

These augmentations improve generalization but also increase training entropy, which can reduce optimization stability. To counter this effect, we introduced a *Noise Loss*:

$$\mathcal{L}_{nse} = w_4 \cdot \mathcal{L}_2(|(1.0 - \beta)\hat{M} \cdot \hat{X} - M \cdot \overset{*}{X}|, \beta N), \tag{6}$$

where $\overset{*}{X}$ denotes the post-perturbation input generated by $P_q(X)$:

$$\begin{aligned} &\overset{*}{X} = (1 - M) \cdot X + (1.0 - \beta)M \cdot X + \beta N, \\ &\beta \sim \mathcal{U}(0.5, 1.0), \\ &N = \text{Perlin noise isolated}, \\ &M = \begin{cases} 1, & \text{if the pixel belongs to the noise region } N, \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{7}$$

where $\overset{*}{X}, M, N, \beta = P_q(X)$, and N is the isolated Perlin-noise map applied to a black image with the same size as X. Equation 7 defines the superimposition mechanism. Parameter q is the probability of applying Perlin noise; therefore, with probability $1-q$, $\overset{*}{X} = X$, $M = \mathbb{0}$, and $N = \mathbb{0}$. Perlin noise introduces non-Gaussian perturbations that better resemble real defects and simultaneously masks non-rectangular regions [21, 32]. This discourages trivial identity mapping, a common weakness of vanilla autoencoders on small defects. Moreover, the ℓ_1 term in the contextual loss was replaced with Huber loss to improve stability around the origin:

$$\mathcal{L}_{con} = w_a \cdot \mathcal{L}_{Huber}(X, \hat{X}) + w_b \cdot \mathcal{L}_{\text{SSIM}}(X, \hat{X}), \tag{8}$$

Starting from DRÆM [45] and GANomaly [1], we used an iterative branch-and-bound tuning process and obtained the following configuration:

$$\begin{aligned} w_a &= 2.0 \\ w_b &= 1.0 \\ w_1 &= 1.0 \\ w_2 &= 50.0 \\ w_3 &= 1.0 \\ w_4 &= 3.0, \end{aligned} \tag{9}$$

Although $\mathcal{L}_{\text{SSIM}}$ provides the strongest contribution to reconstruction quality, it can be unstable on high-entropy images. Increasing w_a mitigated this behavior at the cost of slower convergence.

5.2.1 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 5.1.1, residual network are widely used [20, 24, 40, 47], 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 [16].

5.2.2 Classification and segmentation

This application does not require full-resolution semantic segmentation; patch-level anomaly classification and heatmap localization are sufficient for process integration. Therefore, only the generative branch is used in training and inference. Patch partitioning allows coarse spatial localization of anomalies. The anomaly score is defined as:

$$\phi = 1 - \text{SSIM}(X, \hat{X}), \tag{10}$$

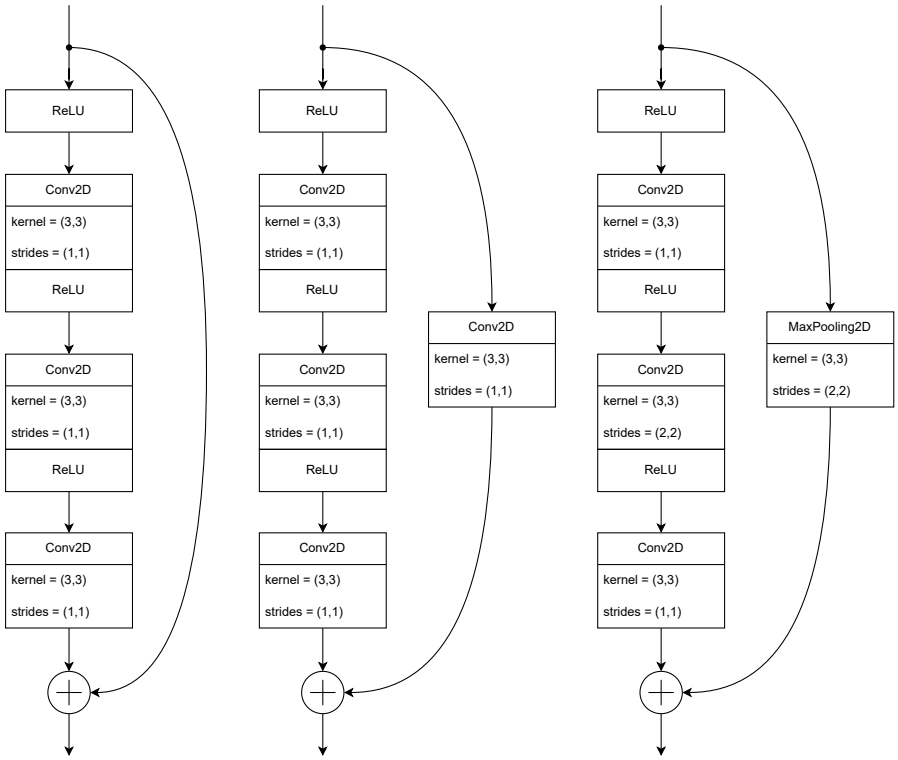
and the heatmap is computed as the absolute difference between input and reconstruction:

$$H = |X - \hat{X}|\Big|_0^1, \tag{11}$$

where $\Big|_0^1$ denotes min-max normalization in $[0, 1]$:

$$H = a + b \cdot \frac{|X - \hat{X}| - \min(|X - \hat{X}|)}{\max(|X - \hat{X}|) - \min(|X - \hat{X}|)}, \; a = 0 \wedge b = 1. \tag{12}$$

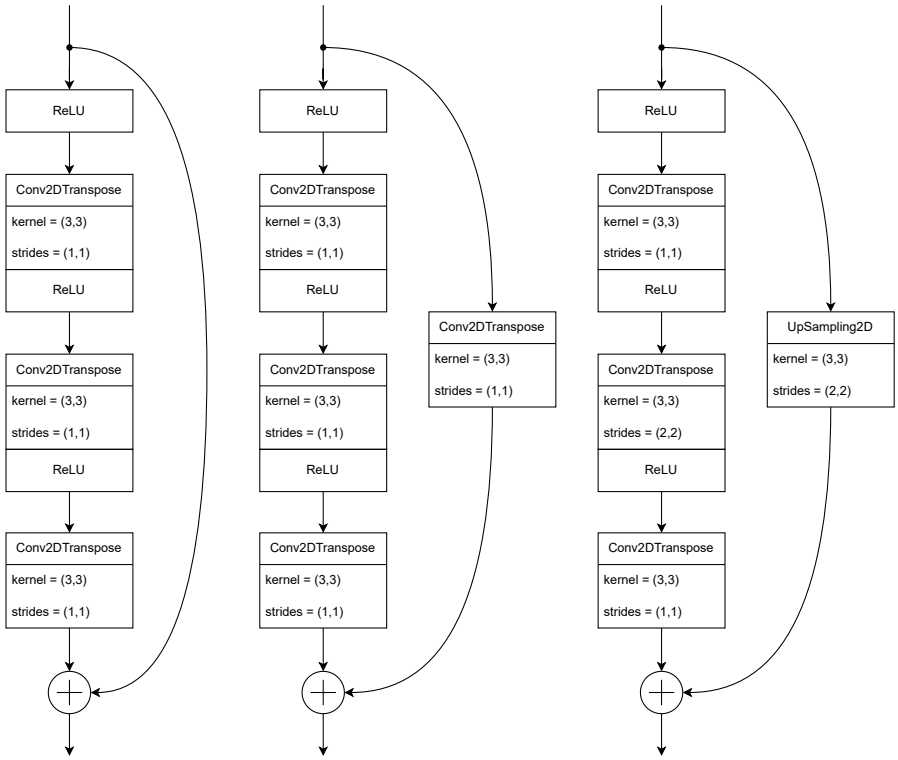
The optimal threshold ϕ_t is tuned on a dedicated calibration subset containing both nominal and defective samples, strictly disjoint from the final test kit used for reporting. During inference, if one patch is classified as reject, the corresponding product is



(a) Residual encoder

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

flagged as reject. In that case, 11 is used to generate the heatmap of the anomalous subregion.



(a) Residual decoder

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

6 Experiments

We conducted a set of experiments to optimize detection performance while preserving inference latency within the 500 ms slot between two consecutive acquisitions. For completeness, we first summarize the industrial context and the hardware/software environment.

Product

The inspected product (Figure 4[1]) is a BFS strip of 5 vials filled with 10 ml of liquid excipient. During handling, bubbles are frequently generated in the liquid region, especially around the *meniscus*. In addition, the narrow *neck* can trap liquid, and droplets are often visible above the liquid level. As shown in Figure 5, each strip contains 5 vials, and each vial is partitioned into 4 logical regions: *flag* (upper, red), top *body* (blue), liquid *body* (green), and *bottom* (yellow).

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

(a) The product

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

Train dataset acquisition

The production acquisition setup was replicated at laboratory scale. Both setups used a telecentric lens to mitigate geometric distortion, especially on side and bottom vial regions. Each strip was acquired 10 times with a semicircular motion focused on the upper side of the product, yielding 16 frames per acquisition. For each frame set, two additional images were generated via a *rank* filter by selecting the minimum and maximum gray-level responses. 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

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

the rank A[3] corresponds to the largest gray value within the mask. As a consequence, moving artifacts are suppressed in lighter-rank images and emphasized in darker-rank images. This increases the variability of nominal samples. After patch extraction, the dataset contained 2,815,200 images, split into 90% training and 10% validation. 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:

$$N = f \cdot v \cdot r = 3 \cdot 5 \cdot 4 = 60, \tag{13}$$

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

Training was run for 10 epochs because of the very high numerosity of the training set, using the pipeline described in section 5, and specifically in subsections 5.2, 5.2.1, and 5.2.2. The residual backbone follows the ResNet v2 design principles [20]. This section reports:

1. The hardware and software environment
2. The architecture of the adopted network
3. The training phase
4. Performance on positive and negative examples in the test dataset
5. The results in terms of time and throughput
6. Experiments on the generative network

[3] In this case $A = 16$
[4] More specific details about machine hardware cannot be provided due to the NDA and company policies

6.1 Hardware

The hardware, as already mentioned in subsection 3.3 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

6.2 Network architecture

Encoder

As mentioned in subsection 5.2.1, the network is residual based on ResNet version 2 architecture. Each stage is composed of 3 residual blocks:

1. The first block (A in Figure 6) concatenates three consecutive 3×3 convolutions with one 1×1 convolution. Spatial size $H_i \times W_i$ is preserved.
2. The second block (B in Figure 6) concatenates three consecutive 3×3 convolutions with the input from block A. Spatial size $H_i \times W_i$ is preserved.
3. The third block (C in Figure 6) concatenates three consecutive 3×3 convolutions with a downsampling branch. The middle convolution halves both H_i and W_i.

The encoder uses 4 stages, yielding a final embedding size of 16×16 with 1024 channels.

Bottleneck

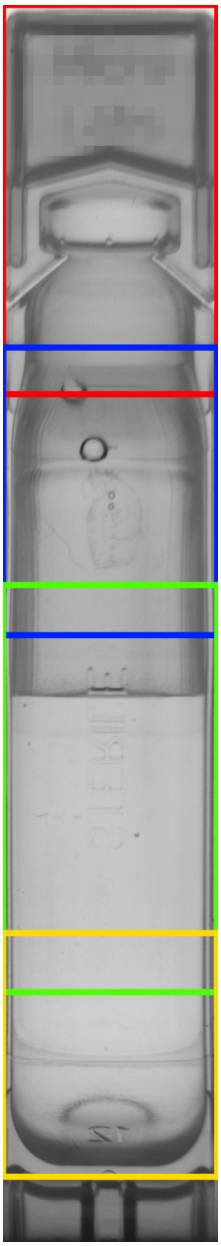
The bottleneck is fully-connected with a size of 64 features, as in figure 7.

Decoder

The decoder is the inverse architecture of the encoder, described above. As in the encoder, each stage is composed of 3 residual blocks:

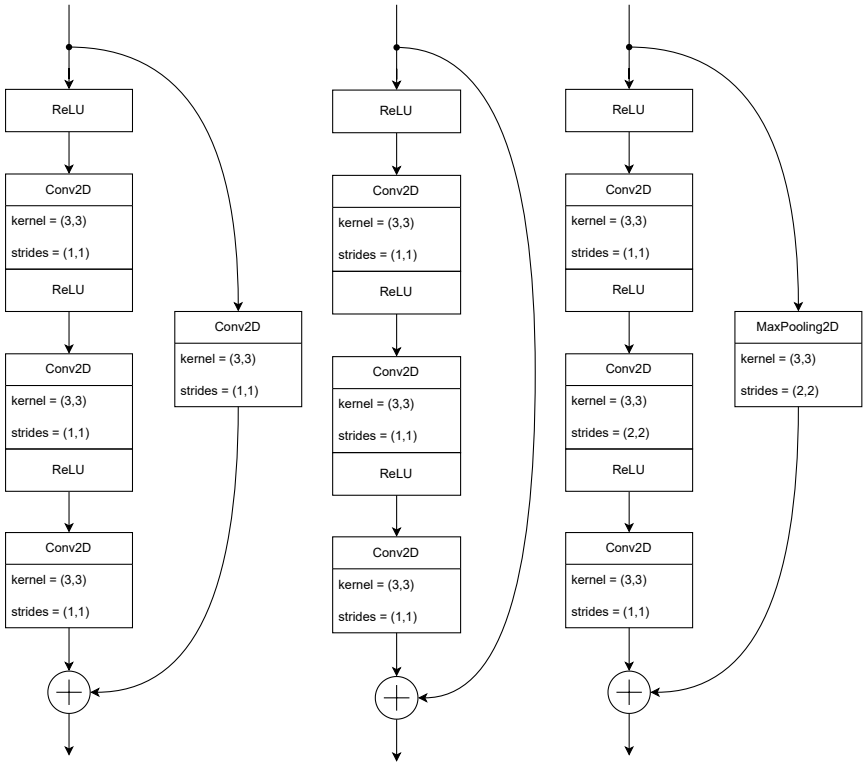
1. The first block (A in Figure 8) concatenates three consecutive transposed convolutions with 3×3 kernel and one 1×1 convolution. Spatial size $H_i \times W_i$ is preserved.
2. The second block (B in Figure 8) concatenates three consecutive transposed convolutions with the input from block A. Spatial size $H_i \times W_i$ is preserved.
3. The third block (C in Figure 8) concatenates three consecutive transposed convolutions with an upsampling branch. The middle convolution doubles both H_i and W_i.

The decoder uses 4 stages to recover a final output size of 256×256 with a single channel and sigmoid activation.



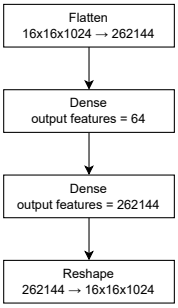
(a) The vial regions

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



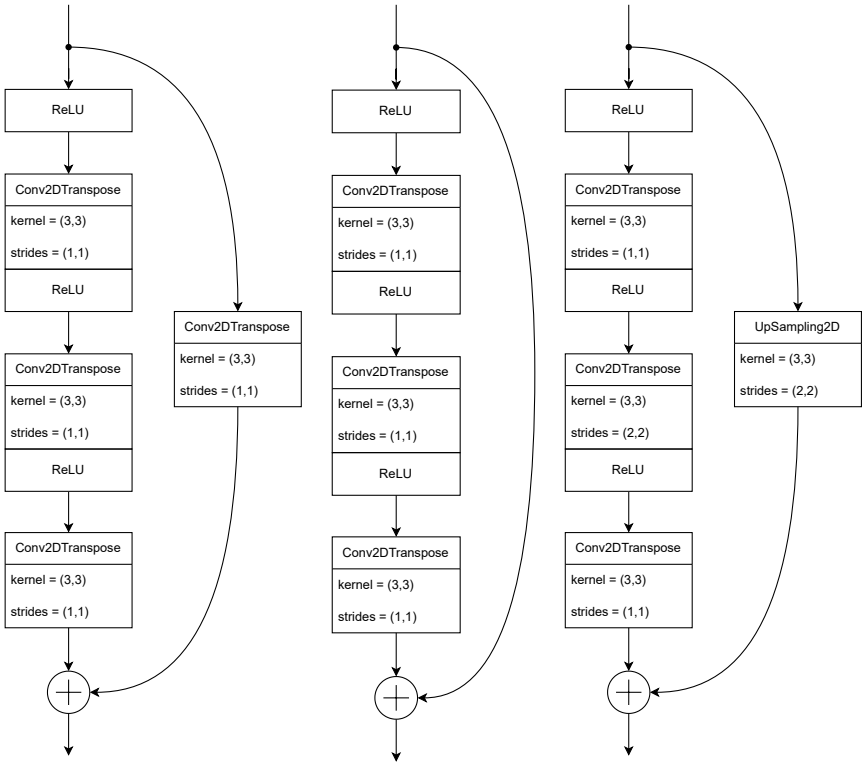
(a) ResEnc Black A (b) ResEnc Block B (c) ResEnc Block C

Fig. 6 Three residual blocks in the encoder (ResEnc). One stage consists of three convolutional blocks in the order A, B, C.



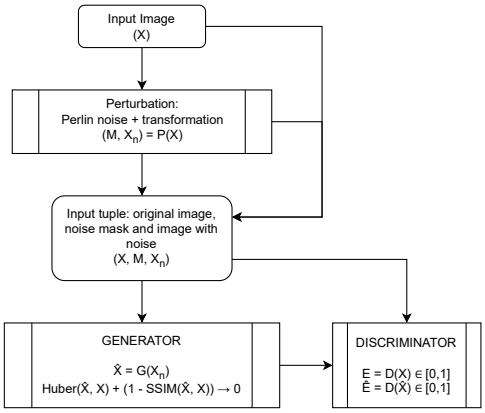
(a) Dense bottleneck

Fig. 7 The bottleneck stage.



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

Fig. 8 Three residual blocks in the decoder (ResDec). One stage consists of three transposed-convolution blocks in the order A, B, C.



(a) Train flowchart

Fig. 9 The training flowchart.

6.3 Training phase

Training follows a standard GAN schedule with alternating optimization of generator and discriminator. The initial learning rate is 1.5×10^{-4}, with cosine-decay restarts every 2,533,680 steps and restart amplitude set to $\frac{1}{3}$ of the previous maximum. Adam is used for both networks. Batch size is 32, and perturbation probability is set to $q = 0.75$. The training flowchart is shown in Figure 9.

6.4 Results

Experiments were performed on the client-provided knapp test kit. The set contains 141 defective products and 120 nominal products. Since the class distribution is relatively balanced, we report accuracy, true positive ratio (TPR), true negative ratio (TNR), balanced accuracy, and inference time statistics. Thresholds are calibrated on a dedicated validation subset, and the final test kit is used only for performance reporting. In this manuscript, we report point estimates; confidence intervals will be included in an extended statistical analysis. The analysis is presented at three aggregation levels:

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

A representative qualitative subset is reported in Appendix A.

Due to NDA constraints on the proprietary dataset and pipeline, this manuscript focuses on absolute deployment performance and engineering feasibility. A full baseline comparison against representative methods (e.g., PaDiM, PatchCore, FastFlow) on public benchmarks with the same preprocessing and evaluation protocol is planned as follow-up work.

Acceptance policy

For industrial validation, the machine must perform at least as well as human inspectors on the same knapp test kit. Each product is acquired 10 times; each acquisition is defined as one run. A nominal product is considered correctly classified if it is predicted as nominal in at least 7 runs out of 10. A defective product is considered correctly classified if it is predicted as defective in at least 7 runs out of 10. Otherwise, it is counted as misclassified.

Overall results

Overall results are reported in table 1. A dedicated threshold is estimated for each region due to region-specific pixel distributions. The mean inference time per frame is computed 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, μ_{t_b} is the mean time per batch, v is the number of vials per strip, and r is the number of regions per vial.

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
Balanced accuracy	0.9530	0.9515	0.9883	0.9984	0.9762
Mean inference time (ms)	0.1689	0.1689	0.1689	0.1689	-

Table 1 Overall results using the equation 10 as anomaly score.

Per product aggregation

Whole-strip classification is designed to minimize GMP[5] risk: if at least one region is classified as reject, the entire product is rejected. This policy reduces false acceptance at the expense of increased false rejection; therefore, threshold retuning is required to satisfy project constraints. To this end the final score function per vial becomes:

$$\theta_v = \max(\left\{1.0 - \text{SSIM}(X_i, \hat{X}_i)\right\}), \; i \geq 0 \; \wedge \; i < f \cdot v \cdot r, \tag{14}$$

where i is the patch index.

Per run aggregation

For final validation, the algorithm must match or exceed the manual-inspection benchmark measured by the client on the same knapp kit. Each product is acquired 10 times and is considered correctly classified if the label is confirmed in at least 7 runs out of 10 (70%). Results are reported 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
Balanced accuracy	0.9581
Mean inference time	0.4873

Table 2 Per strip results using the equation 14 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
Balanced accuracy	0.9638

Table 3 Per run results using the equation 14 as anomaly score.

7 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.

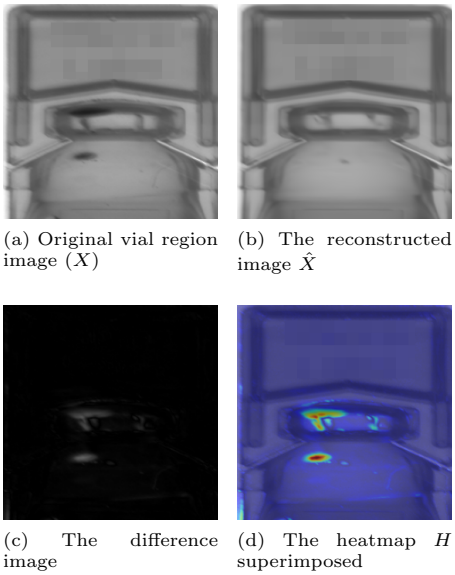
Several solutions have been proposed in the embedding similarity-based setting, including PaDiM [12], PatchCore [34], and EfficientAD [6]. In these methods, distances between inferred embeddings and nominal embeddings collected during training are projected back to image space to obtain anomaly maps; with calibrated thresholds, this enables pixel-level classification. The reference representation can be organized in multiple forms, such as nominal clusters or probabilistic models. 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 [16], DRÆM [45], the described deep perceptual autoencoder by N. Shvetsova et al. [36], or the proposed one by P. Bergmann et al. [10], 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. [15], 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.



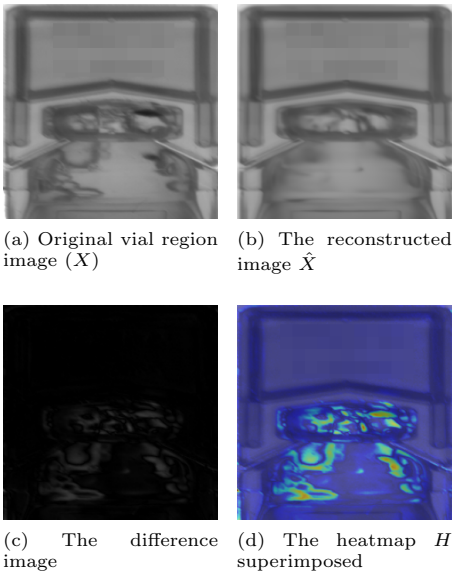
(a) Original vial region image (X)

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

(c) The difference image

(d) The heatmap H superimposed

Fig. 10 Stuck particle in the upper part of the product



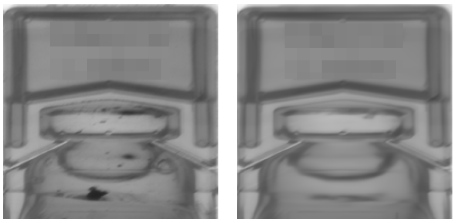
(a) Original vial region image (X)

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

(c) The difference image

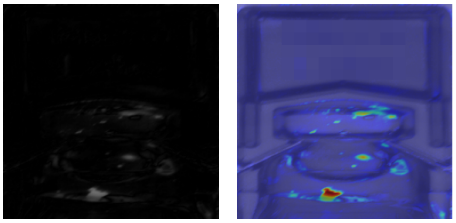
(d) The heatmap H superimposed

Fig. 11 Stuck particle and anomalous liquid behavior that results in foam formation because of contamination



(a) Original vial region image (X)

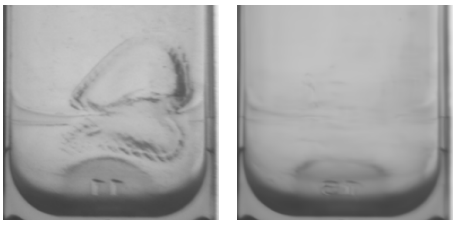
(b) The reconstructed image $\hat{X}$



(c) The difference image

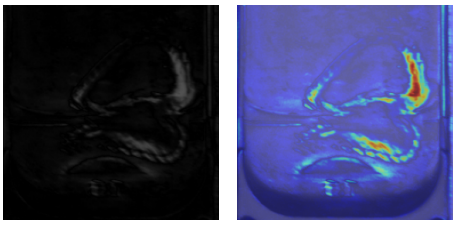
(d) The heatmap H superimposed

Fig. 12 Stuck particle and anomalous liquid behavior that results in foam formation because of contamination



(a) Original vial region image (X)

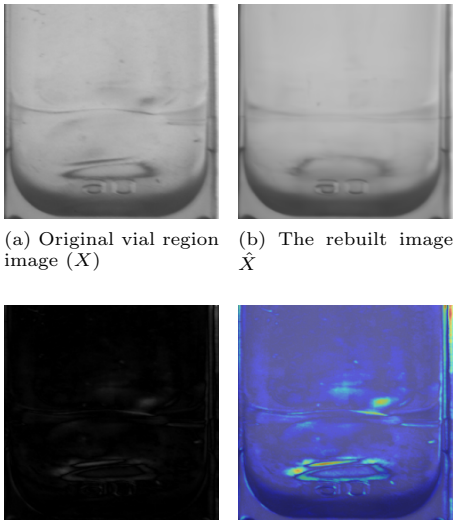
(b) The rebuilt image $\hat{X}$



(c) The difference image

(d) The heatmap H superimposed

Fig. 13 Lower body deformation



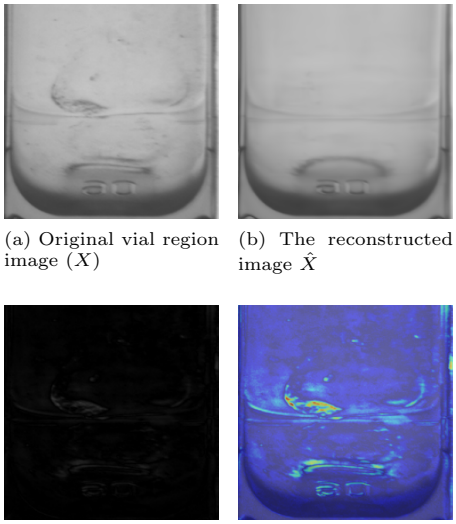
(a) Original vial region image (X)

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

(c) The difference image

(d) The heatmap H superimposed

Fig. 14 Light lower body deformation



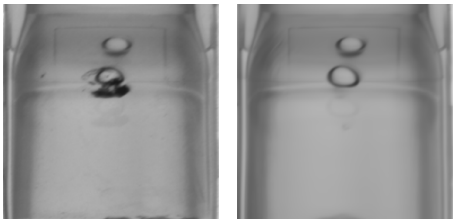
(a) Original vial region image (X)

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

(c) The difference image

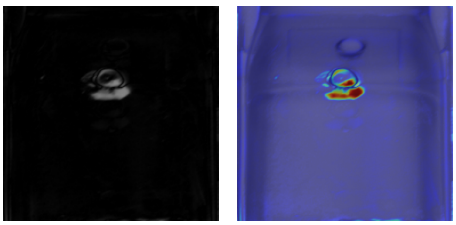
(d) The heatmap H superimposed

Fig. 15 Lower body deformation



(a) Original vial region image (X)

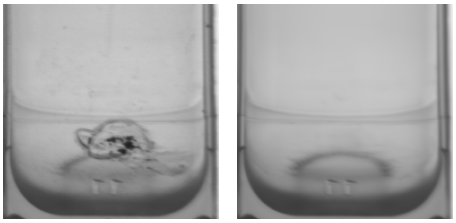
(b) The reconstructed image $\hat{X}$



(c) The difference image

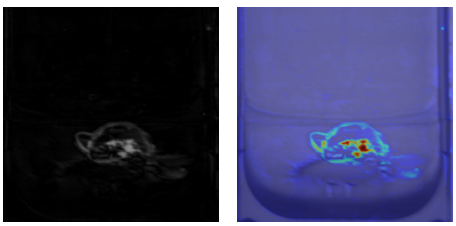
(d) The heatmap H superimposed

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



(a) Original vial region image (X)

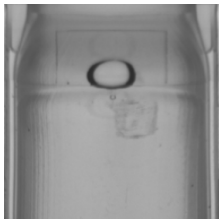
(b) The reconstructed image $\hat{X}$



(c) The difference image

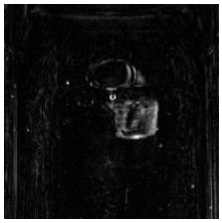
(d) The heatmap H superimposed

Fig. 17 Burn in the lower part of the vial.

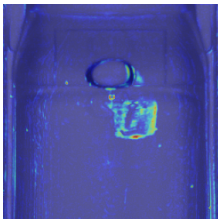


(a) Original vial region image (X)

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



(c) The difference image

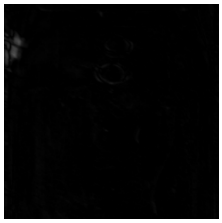


(d) The heatmap H superimposed

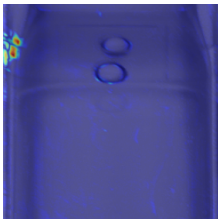
Fig. 18 Light scratch near the product neck.

(a) Original vial region image (X)

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

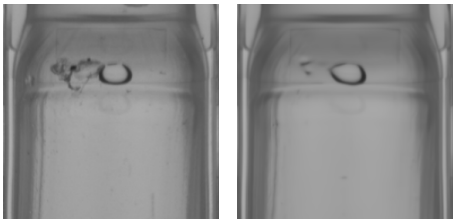


(c) The difference image



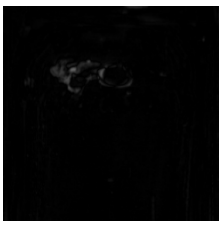
(d) The heatmap H superimposed

Fig. 19 External imperfection caused by laser cutting of the product.

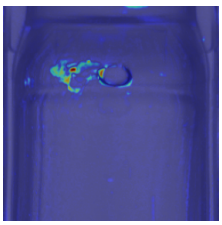


(a) Original vial region image (X)

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



(c) The difference image

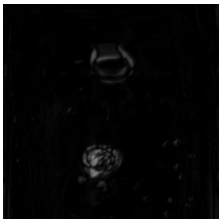


(d) The heatmap H superimposed

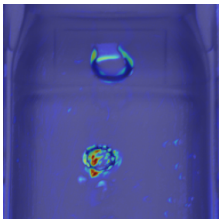
Fig. 20 Deep scratch on the upper part of the vial's body.

(a) Original vial region image (X)

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

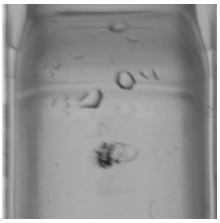


(c) The difference image



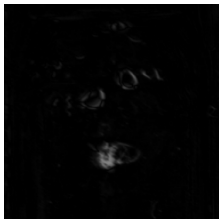
(d) The heatmap H superimposed

Fig. 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.

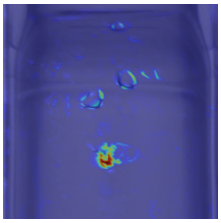


(a) Original vial region image (X)

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

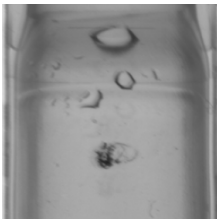


(c) The difference image



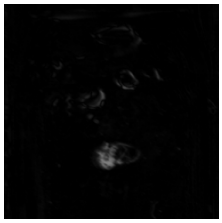
(d) The heatmap H superimposed

Fig. 22 Black spot on the vial's body

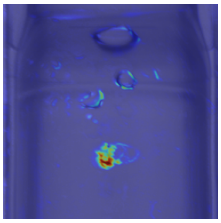


(a) Original vial region image (X)

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

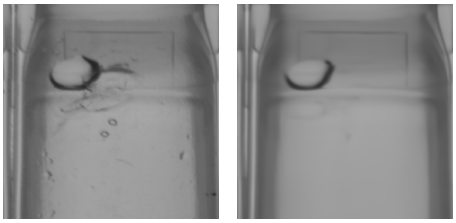


(c) The difference image



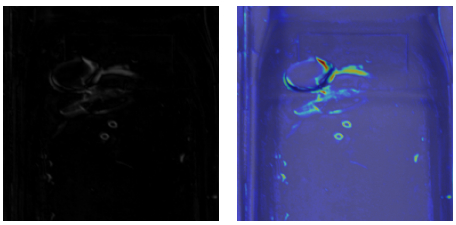
(d) The heatmap H superimposed

Fig. 23 Black spot of the vial's body



(a) Original vial region image (X)

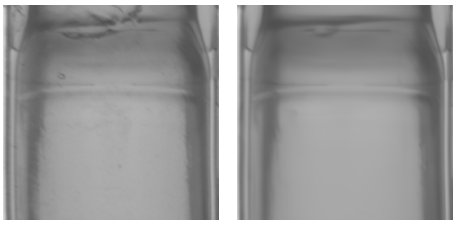
(b) The reconstructed image $\hat{X}$



(c) The difference image

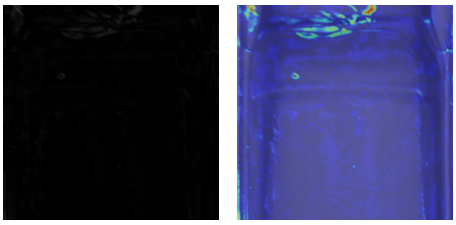
(d) The heatmap H superimposed

Fig. 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)

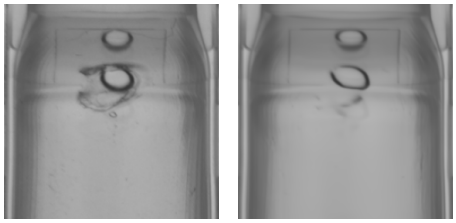
(b) The reconstructed image $\hat{X}$



(c) The difference image

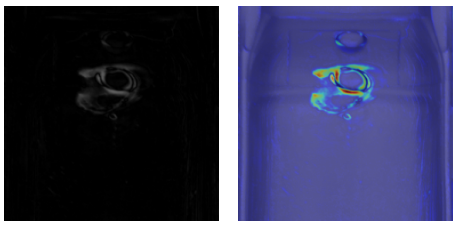
(d) The heatmap H superimposed

Fig. 25 Light bump on the vial's neck



(a) Original vial region image (X)

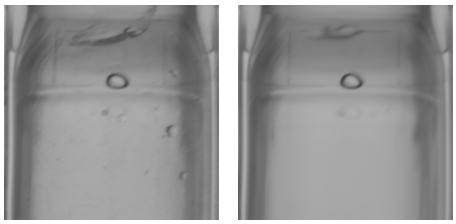
(b) The reconstructed image $\hat{X}$



(c) The difference image

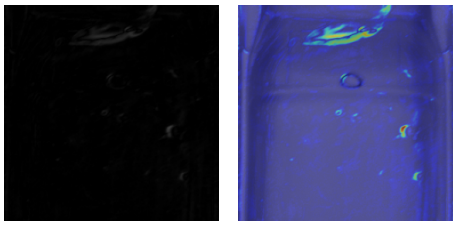
(d) The heatmap H superimposed

Fig. 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)

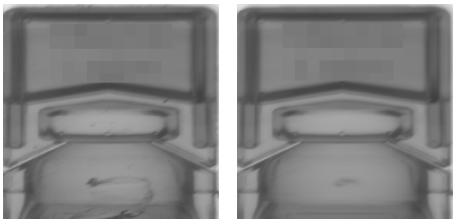
(b) The reconstructed image $\hat{X}$



(c) The difference image

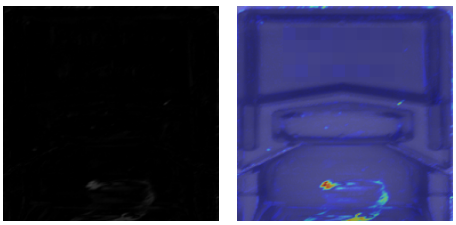
(d) The heatmap H superimposed

Fig. 27 Scratch on the vial's neck.



(a) Original vial region image (X)

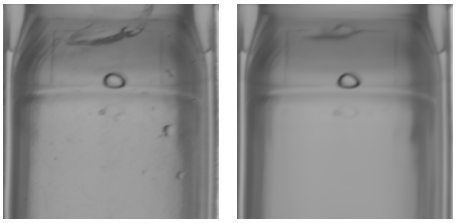
(b) The reconstructed image $\hat{X}$



(c) The difference image

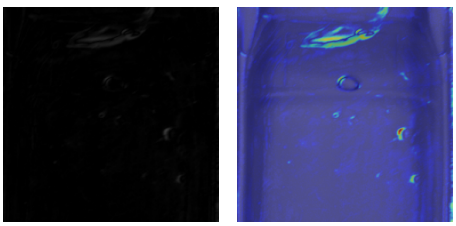
(d) The heatmap H superimposed

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



(a) Original vial region image (X)

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



(c) The difference image

(d) The heatmap H superimposed

Fig. 29 Scratch on the vial's neck.

Acknowledgements. 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.

Declarations

Funding 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.

Conflict of interest/Competing interests The authors declare that they have no competing interests.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Data availability The datasets generated and/or analyzed during the current study are not publicly available due to NDA and commercial confidentiality constraints, but may be available from the corresponding author on reasonable request and where contractually permitted.

Materials availability Not publicly available due to NDA and commercial confidentiality constraints.

Code availability Not publicly available due to NDA and proprietary constraints.

Author contribution All authors contributed to the study conception and design. Material preparation, data collection, analysis, and manuscript drafting/revision were performed by the authors. All authors read and approved the final manuscript.

References

[1] Akcay S, Atapour-Abarghouei A, Breckon TP (2018) Ganomaly: Semi-supervised anomaly detection via adversarial training. In: Asian conference on computer vision, Springer, pp 622–637

[2] Akçay S, Atapour-Abarghouei A, Breckon TP (2019) Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. In: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–8

[3] Akcay S, Ameln D, Vaidya A, et al (2022) Anomalib: A deep learning library for anomaly detection. arXiv:2202.08341

[4] Bae J, Lee JH, Kim S (2023) Pni : Industrial anomaly detection using position and neighborhood information. arXiv:2211.12634

[5] Bank D, Koenigstein N, Giryes R (2020) Autoencoders. CoRR abs/2003.05991. URL https://arxiv.org/abs/2003.05991, 2003.05991

[6] Batzner K, Heckler L, König R (2023) Efficientad: Accurate visual anomaly detection at millisecond-level latencies. arXiv:2303.14535

[7] Bergman L, Cohen N, Hoshen Y (2020) Deep nearest neighbor anomaly detection. arXiv preprint arXiv:200210445

[8] Bergmann P, Löwe S, Fauser M, et al (2018) Improving unsupervised defect segmentation by applying structural similarity to autoencoders. arXiv preprint arXiv:180702011

[9] Bergmann P, Fauser M, Sattlegger D, et al (2019) Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9592–9600

[10] Bergmann P, Batzner K, Fauser M, et al (2022) Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. International Journal of Computer Vision 130(4):947–969. https://doi.org/10.1007/s11263-022-01578-9, URL https://doi.org/10.1007/s11263-022-01578-9

[11] Cohen N, Hoshen Y (2020) Sub-image anomaly detection with deep pyramid correspondences. arXiv preprint arXiv:200502357

[12] Defard T, Setkov A, Loesch A, et al (2021) Padim: a patch distribution modeling framework for anomaly detection and localization. In: International Conference on Pattern Recognition, Springer, pp 475–489

[13] Denkena B, Dittrich MA, Noske H, et al (2020) Statistical approaches for semi-supervised anomaly detection in machining. Production Engineering 14. https://doi.org/10.1007/s11740-020-00958-9

[14] Dinh L, Sohl-Dickstein J, Bengio S (2016) Density estimation using real nvp. arXiv preprint arXiv:160508803

[15] Esser P, Rombach R, Ommer B (2021) Taming transformers for high-resolution image synthesis. arXiv:2012.09841

[16] Ferrari N, Fraccaroli M, Lamma E (2023) Grd-net: Generative-reconstructive-discriminative anomaly detection with region of interest attention module. International Journal of Intelligent Systems 2023:1–18. https://doi.org/10.1155/2023/7773481

[17] Gong D, Liu L, Le V, et al (2019) Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 1705–1714

[18] Goodfellow I, Pouget-Abadie J, Mirza M, et al (2020) Generative adversarial networks. Communications of the ACM 63(11):139–144

[19] He K, Zhang X, Ren S, et al (2015) Deep residual learning for image recognition. CoRR abs/1512.03385. URL http://arxiv.org/abs/1512.03385, 1512.03385

[20] He K, Zhang X, Ren S, et al (2015) Deep residual learning for image recognition. arXiv:1512.03385

[21] He K, Chen X, Xie S, et al (2021) Masked autoencoders are scalable vision learners. CoRR abs/2111.06377. URL https://arxiv.org/abs/2111.06377, 2111.06377

[22] Kingma DP, Welling M (2013) Auto-encoding variational bayes. https://doi.org/10.48550/ARXIV.1312.6114, URL https://arxiv.org/abs/1312.6114

[23] Kingma DP, Welling M (2019) An introduction to variational autoencoders. CoRR abs/1906.02691. URL http://arxiv.org/abs/1906.02691, 1906.02691

[24] Le VT, Kim YG (2023) Attention-based residual autoencoder for video anomaly detection. Applied Intelligence 53(3):3240–3254. https://doi.org/10.1007/s10489-022-03613-1, URL https://doi.org/10.1007/s10489-022-03613-1

[25] Mattia FD, Galeone P, Simoni MD, et al (2021) A survey on gans for anomaly detection. arXiv:1906.11632

[26] Michelucci U (2022) An introduction to autoencoders. CoRR abs/2201.03898. URL https://arxiv.org/abs/2201.03898, 2201.03898

[27] Mishra P, Verk R, Fornasier D, et al (2021) Vt-adl: A vision transformer network for image anomaly detection and localization. In: 2021 IEEE 30th International Symposium on Industrial Electronics (ISIE). IEEE, https://doi.org/10.1109/isie45552.2021.9576231, URL http://dx.doi.org/10.1109/ISIE45552.2021.9576231

[28] Mooijman P, Catal C, Tekinerdogan B, et al (2023) The effects of data balancing approaches: A case study. Applied Soft Computing 132:109853. https://doi.org/https://doi.org/10.1016/j.asoc.2022.109853, URL https://www.sciencedirect.com/science/article/pii/S1568494622009024

[29] Muhr D, Affenzeller M, Küng J (2023) A probabilistic transformation of distance-based outliers. arXiv:2305.09446

[30] Napoletano P, Piccoli F, Schettini R (2018) Anomaly detection in nanofibrous materials by cnn-based self-similarity. Sensors 18(1):209

[31] Pidhorskyi S, Almohsen R, Doretto G (2018) Generative probabilistic novelty detection with adversarial autoencoders. Advances in neural information processing systems 31

[32] Prabhakar C, Li HB, Yang J, et al (2023) Vit-ae++: Improving vision transformer autoencoder for self-supervised medical image representations. arXiv preprint arXiv:230107382 URL https://arxiv.org/abs/2301.07382, arXiv:2301.07382 [cs.CV]

[33] Rippel O, Mertens P, Merhof D (2021) Modeling the distribution of normal data in pre-trained deep features for anomaly detection. In: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, pp 6726–6733

[34] Roth K, Pemula L, Zepeda J, et al (2022) Towards total recall in industrial anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 14318–14328

[35] Sabokrou M, Khalooei M, Fathy M, et al (2018) Adversarially learned one-class classifier for novelty detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3379–3388

[36] Shvetsova N, Bakker B, Fedulova I, et al (2021) Anomaly detection in medical imaging with deep perceptual autoencoders. IEEE Access 9:118571–118583. https://doi.org/10.1109/access.2021.3107163, URL http://dx.doi.org/10.1109/ACCESS.2021.3107163

[37] Tian Y, Henaff OJ, van den Oord A (2021) Divide and contrast: Self-supervised learning from uncurated data. arXiv:2105.08054

[38] Venkataramanan S, Peng KC, Singh RV, et al (2020) Attention guided anomaly localization in images. In: European Conference on Computer Vision, Springer, pp 485–503

[39] Wang Z, Bovik AC, Sheikh HR, et al (2004) Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing 13(4):600–612

[40] Wickramasinghe CS, Marino DL, Manic M (2021) Resnet autoencoders for unsupervised feature learning from high-dimensional data: Deep models resistant to performance degradation. IEEE Access 9:40511–40520. https://doi.org/10.1109/ACCESS.2021.3064819

[41] Xia X, Pan X, Li N, et al (2022) Gan-based anomaly detection: A review. Neurocomputing 493:497–535. https://doi.org/https://doi.org/10.1016/j.neucom.2021.12.093, URL https://www.sciencedirect.com/science/article/pii/S0925231221019482

[42] Xie X, Huang Y, Ning W, et al (2022) Rdad: A reconstructive and discriminative anomaly detection model based on transformer. International Journal of Intelligent Systems 37(11):8928–8946. https://doi.org/https://doi.org/10.1002/int.22974, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22974,

https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.22974

[43] Xu H, Pang G, Wang Y, et al (2023) Deep isolation forest for anomaly detection. IEEE Transactions on Knowledge and Data Engineering pp 1–14. https://doi.org/10.1109/TKDE.2023.3270293

[44] Yu J, Zheng Y, Wang X, et al (2021) Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows. arXiv:2111.07677

[45] Zavrtanik V, Kristan M, Skočaj D (2021) Draem-a discriminatively trained reconstruction embedding for surface anomaly detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 8330–8339

[46] Zhou Y, Xu X, Song J, et al (2023) Msflow: Multi-scale flow-based framework for unsupervised anomaly detection. arXiv:2308.15300

[47] Zini S, Bianco S, Schettini R (2020) Deep residual autoencoder for blind universal jpeg restoration. IEEE Access 8:63283–63294. https://doi.org/10.1109/access.2020.2984387, URL http://dx.doi.org/10.1109/ACCESS.2020.2984387