Detection of anomalies with possible localization

Elisabeth Lykken Hamre Fernando Velilla Hurtado Moiz Ahmad Abassi
Politecnico di Torino
Corso Duca degli Abruzzi 24, Turin, Italy
s350844@studenti.polito.it s354393@studenti.polito.it s349710@studenti.polito.it

Abstract

Detection and localization of anomalous regions in images is of high importance for several computer vision task. In production for instance, defects should be detected before a product reach costumers. This paper address unsupervised anomaly detection by using different architectures of convolutional autoencoders, trained exclusively on normal images. In testing, the reconstruction error is used to (i) classify images as anomalous or normal and (ii) localize the anomalous regions with pixel-wise heat-maps. The approach is evaluated on both normal and anomaly images. The robustness of the model is tested by a domain shift. Our best architectures seem to be detecting and localizing large defects well, but fail on smaller defects. Although state-of-the-art implementations have better scores, this paper provides a course of action for future work on the topic.

1. Introduction

For this project we have studied unsupervised anomaly detection (binary-class classification) and localization. Automatic anomaly detection is important in industrial domains where the volume produced is so large that human inspection of defects is inefficient. In addition, defects can be too small for humans to detect, or it can be time-demanding. Typically, anomaly detection is set up to discover regions in the data which deviates from the "normal" appearance of its surrounding area which should follow a similar pattern/structure. For this project, the model is trained on normal images under controlled conditions. We evaluate the model's capability to detect and localize the anomalies on a standard test-set. We have tried several architectures, the most important ones are mentioned. Additionally, the robustness of the model is tested by performing a domain-shift on the standard test-set, and we try to mitigate the performance-drop by also performing shifts on the training-set .

2. Dataset

For this project, we have used the anomaly detection set "MVTec AD" [2, 3], chosen because it is a benchmarking for anomaly detection methods. The dataset consists of 5000 images, divided into fifteen categories of objects/textures. Each category has a set of anomaly-free training images and a test-set with both defect-free and various defect-types. Specifically, we have looked at the "capsule-category". The training-set of this category consists of 219 images and the test-set of 132 images. In the test-set, about 18% of the images are normal. The capsule-category has various defects, such as cracks, faulty imprints, pokes, scratches and squeezes/deformations We split the training-set into a training and validation-set. The split is 90/10. Before feeding the images to the model, we perform some pre-processing. Input images are resized from 1024x1024 resolution to 256x256 resolution, and added data-augmentation such as random rotations ($degrees = 5$) to introduce variability on the training-set that reduces overfitting. Link to data-set: https://www.mvtec.com/company/research/datasets/mvtec-ad

3. Method

3.1. Model architecture

3.1.1 Autoencoder

The implemented models are autoencoders. As said by MathWorks, autoencoders can outperform traditional engineering techniques in several applications, such as anomaly detection [7]. An autoencoder is built out of two smaller networks, an encoder and a decoder. It outputs a reconstruction of it's input. The **encoder** downsamples the input during training, and learns a set of features—the latent representation. Meanwhile, the **decoder** is trained to reconstruct the input based on the features in the latent representation. The difference between the input-image and the reconstructed image is used to set a reconstruction-score, a threshold for classification. The idea is that when the model

is trained on normal images, the model will fail in reproducing an anomalous image, and therefore the reconstruction-score will be much higher for anomaly images.

3.1.2 The implemented model I

Model architecture I is visualized in Figure 3, and is a 2-dimensional autoencoder. It is inspired by the autoencoder used in the paper "Unsupervised anomaly detection in brain MRI: Learning abstract distribution from massive healthy brains" [6]. The input to the model is RGB-images of size 256x256. The encoder has one convolutional layer, further it has series of six residual downsampling blocks, as seen in Figure 4. The downsampling-blocks reduces the spatial dimensions of the input image while increasing the feature channels. The bottleneck layer is a latent vector of size 512. The decoder is symmetrical to the encoder, it decreases the feature channels and increase the spatial dimensions back to the input size to reconstruct the image. The output layer is a convolutional layer with a sigmoid as the activation function.

3.1.3 The implemented model II

The second architecture was based on the SOTA paper "Self-Supervised Training with Autoencoders for Visual Anomaly Detection" [1], which proposed a self-supervised framework that treats the autoencoder as a non-linear orthogonal projection onto the manifold of normal samples. The model (see Figure 1) integrates Stacked Dilated Convolutions (SDC), utilizing dilation rates of 1,2,4,8,16,32 to approximate higher-order spatial dependencies without losing resolution. Furthermore, to prevent the network from neglecting the reconstruction of corrupted regions, the standard skip connections are enhanced with Convex Combination Modules (CCM), which acts as an attention mechanism (see Figure 2). The final layer uses a sigmoid activation function.

Another critical enhancement by this architecture was applying a **patch transformation** strategy [1]. First, for each normal sample x, first, over a randomly sampled patch we select a real-valued mask $M \in [0,1]^{h \times w \times 3}$, using an elastic deformation technique. Then, gaussian distortion (with random parameters) is applied to this mask resulting in varying shapes. Finally, these masks are used to smoothly merge the original patches with a new content given by replacement patches from the publicly available Describable Textures Dataset (DTD). Given an input x and a replacement y, we create a corrupted image $\hat{x}$ according to the formula: $\hat{x} := M \odot y + \bar{M} \odot x$, where $\bar{M} := 1 - M$ denotes a corresponding complement of the mask M.

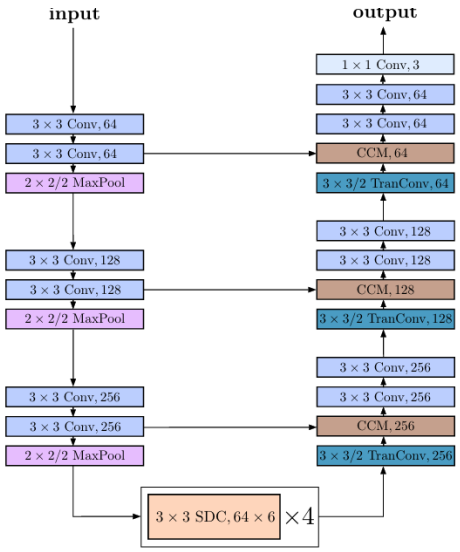

Figure 1. Architecture of Model II. Image from [1].

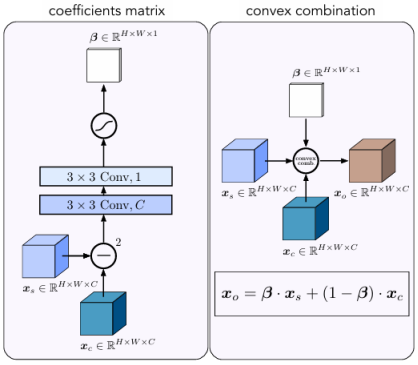

Figure 2. Convex Combination Module (CCM). Image from [1].

3.2. Anomaly score I

The anomaly score is set based on a weighted sum of the Mean Square Error (MSE) and a Structural Similarity Index Measure (SSIM) [10]. The weighting-parameter is called α. The value of α ranged from 0.1 to 0.5, so the SSIM had higher emphasis:

$$Combined\ Loss = \alpha \cdot MSE + (1 - \alpha) \cdot SSIM \quad (1)$$

- **MSE**: measures the pixel-wise difference between the input image and the reconstructed image.

$$MSE(x, \hat{x}) = (\hat{x} - x)^2 \quad (2)$$

- **SSIM**: Measures structural similarity between images. μ_x and $\mu_{\hat{x}}$ are the average pixel-values for images x and $\hat{x}$, σ_x and $\sigma_{\hat{x}}$ are the variance of image x and $\hat{x}$, $\sigma_{x\hat{x}}$ is the covariance between image x and $\hat{x}$. C_1 and C_2 are small constants to avoid zero-division.

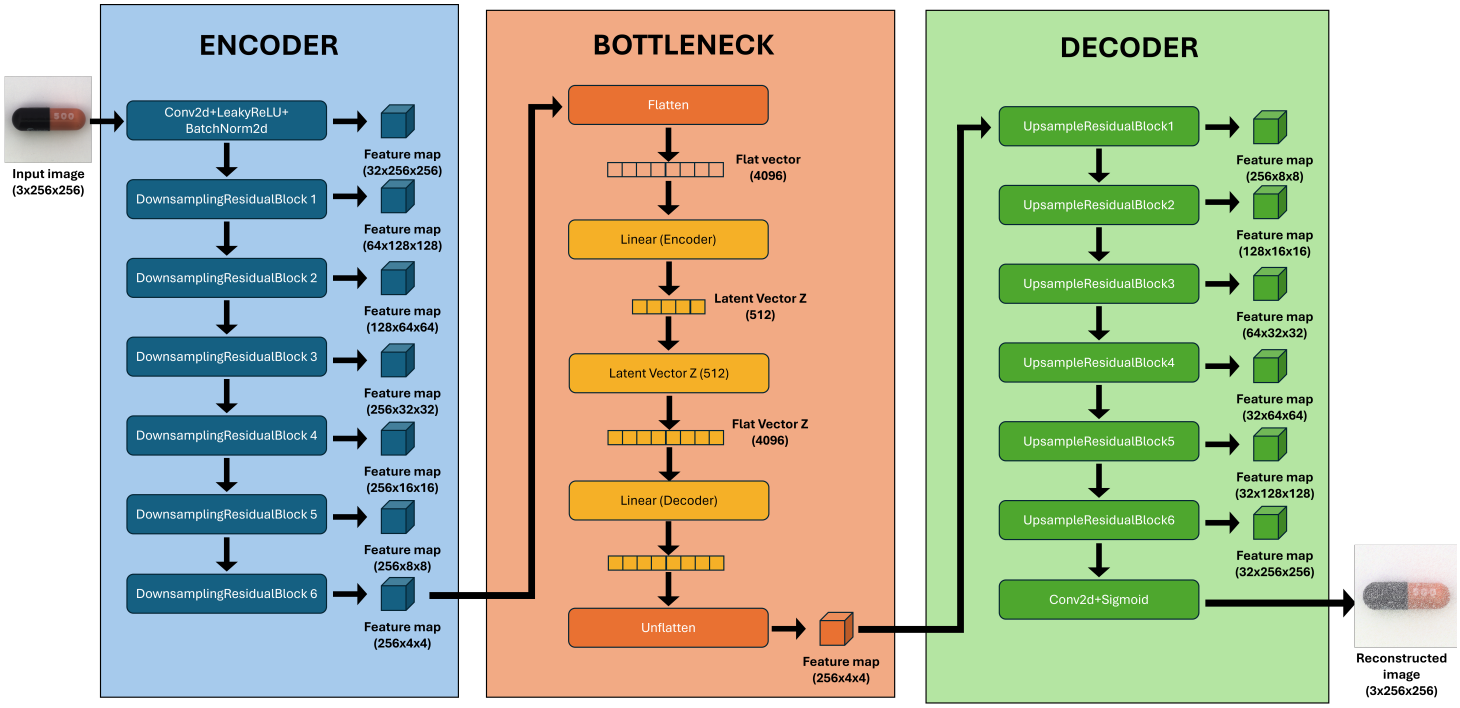

Figure 3. Architecture of Model 1.

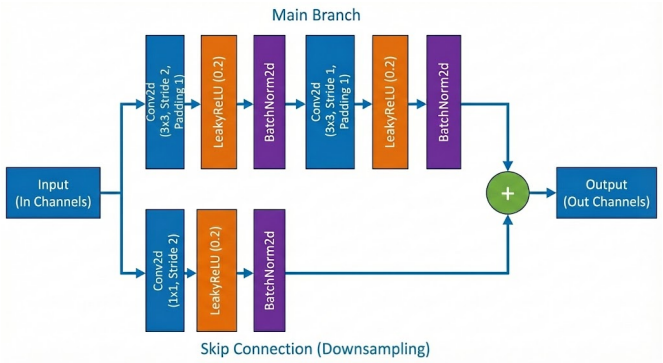

Figure 4. The downsampling residual block repeated in the encoder. The upsampling residual block in the decoder is symmetrical to this block. Image is generated with AI.

$$SSIM(x, \hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)} \tag{3}$$

For MSE we have used the built-in PyTorch function, while the SSIM is a function from the torchmetrics library.

3.2.1 Anomaly score II

To implement the self-supervised training method developed in [1], we first define the core variables used in the data generation and loss calculation. An original, anomaly-free image is denoted as x, while its partially modified version, which contains artificial corruptions, is represented by $\hat{x}$. The specific regions within $\hat{x}$ that have been modified are identified through a real-valued mask $M \in [0, 1]^{h \times w \times 3}$, with its complement $\bar{M} = 1 - M$ marking the corresponding uncorrupted areas of the image. Finally, the element-wise tensor multiplication between these masks and the image data is represented by the operator $\odot$.

The model was optimized by minimizing the following objective function:

$$\mathcal{L}(\hat{x}, x, M) = \frac{(1-\lambda)}{\|\bar{M}\|_1} \cdot \|\bar{M} \odot (f_\theta(\hat{x}) - x)\|_2^2 + \frac{\lambda}{\|M\|_1} \cdot \|M \odot (f_\theta(\hat{x}) - x)\|_2^2 \tag{4}$$

The loss function in Equation 4 is composed of two primary terms balanced by the hyperparameter λ. It's first term encourages the model to accurately replicate the uncorrupted parts of the image ($\bar{M} \odot x$). The second term forces the network to repair or replace the corrupted patches ($M \odot \hat{x}$), by recreating the original content x from the surrounding context.

3.3. Threshold

To classify an image as anomaly or normal we use a threshold. The threshold is set by the N-sigma rule [5]. We calculate the anomaly score on each image in the validation

set. Then we find the mean-value, μ, and standard deviation, σ, from those anomaly scores:

$$Threshold = \mu + (k \cdot \sigma) \tag{5}$$

The parameter k decides how conservative the threshold is. A higher value for k will make the threshold higher, and therefore the anomaly-score has to be higher for an image to be classified as anomalous. In other words, for a higher k/threshold we have to be more certain that the image is anomalous to be classified as anomalous. We used $k = 0.0001$ in the end.

3.4. Evaluation protocol and metrics

3.4.1 Evaluation metrics

Image-level ROC-AUC: ROC-AUC measures the trade-off between the True Positive Rate, (TPR), and the False Positive Rate (FPR) [8]. This metric is calculated for the entire image, often by taking the maximum value or the sum of scores from its internal anomaly map (measures how well the model performs at **classification**).

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN} \tag{6}$$

A score of 1.0 indicates perfect separation, while 0.5 indicates random guessing.

Pixel-wise ROC-AUC : This metric measures the model's ability to accurately locate specific anomalous regions within an image. Using the test set, a separate score is calculated and evaluated for every individual pixel in the image, where each pixel predicted score is compared against its corresponding pixel in a **ground truth** binary mask. It evaluates the **segmentation/**localization accuracy.

Heatmap visualizations: To create the anomaly maps, first the pixel-wise squared difference of the original image with respect to the reconstruction is calculated, which is subsequently averaged over the color channels to produce a difference map (gray-scale image). In the last step we apply a gaussian filter on the input to produce a final anomaly heatmap.

F1-score: Indicates a model's capability to find all of the positive instances (Recall or TPR, (6)) while ensuring that the positives are correctly classified (precision, (7)). This is an important metric for imbalanced datasets.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$F1 = \frac{Precision \cdot Recall}{Precision + Recall} \tag{8}$$

The F1-metric outputs a score between 0 and 1, the higher the number the better the model is at classifying the images correctly.

4. Experiments

4.1. Hyper-parameter configuration

The selection of hyper-parameters was conducted through an iterative experimental process, guided by validation loss convergence and qualitative inspection of reconstructions.

Table 1 details the specific configurations for the four main architectures. The Adam optimizer and MSE+SSIM loss function offered the best stability. Other hyperparamter-configurations were tested but omitted.

4.2. Model I configurations

During the project, we iterated on the model architecture and pre-processing strategy four times to optimize performance.

- **Architecture and pre-processing 1:** The baseline model (Figure 3) utilizes a flattened latent vector. In architecture 1, the bottleneck maintaines a volumetric shape ($256 \times 64 \times 64$) instead of flattening it, aiming to preserve spatial structure. No pre-processing was used other than resizing.
- **Architecture and pre-processing 2:** This architecture introduced two significant modifications: expanding the input depth to 4 channels and integrating the Convolutional Block Attention Module (CBAM).
- **Architecture and pre-processing 3:** Same as figure 3, but latent vector dimension was changed to 128, and base channels to 64 (initial number of filters for the first convolution). Data augmentations only included random rotations and resize.

4.3. Model II configurations

Three different configurations were obtained which enhanced the results of Model I. The first one, named *"Config e2"* was obtained using an initial learning rate of 10^{-3} which changed dynamically with the use of a scheduler (configured with $patience = 5$ and $rate = 0.5$) and with a λ of 0.9, which gave more importance to the correction of anomalies in the image. The second and third one (*"Config e4"* and *"Config e5"*) were obtained with an initial learning rate of 10^{-4} ($patience = 10$ and $rate = 0.5$).

4.4. Training process

The loss curves obtained in the training process for the best architecture of model I obtained (*"Config h2"* of Architecture 3) can be found in Figure 5. It shows a steady

Table 1. Hyperparameter configurations across the four primary evaluated architectures.

Parameter	**Arch 1 (c1/m7)**	**Arch 2 (2a/b/c)**	**Arch 3 (h1/2)**	**Model II**
Input Resolution	1024x1024	512x512	256x256	256x256
Input Channels	3 (RGB)	**4 (RGB+L)**	3 (RGB)	3 (RGB)
Bottleneck Spatial	$256 \times 64 \times 64$	256×6412	128	3x3 SDC, 64X6
Optimizer	SGD	Adam	Adam	Adam
Learning Rate	10^{-3}	10^{-4}	10^{-4}	10^{-4} with scheduler
Batch Size	1	5	16 (Eff. 32)	4 (Eff. 32)
Loss Function	MSE	MSE	MSE + SSIM	Anomaly score II
Training Epochs	50	20	100	200
Data Augmentation	Only norm.	Rotation	Rotation	Rotation
Special Features	None	**CBAM**	$C_{base} = 64, z_{dim} = 128$	$\lambda = 0.9$

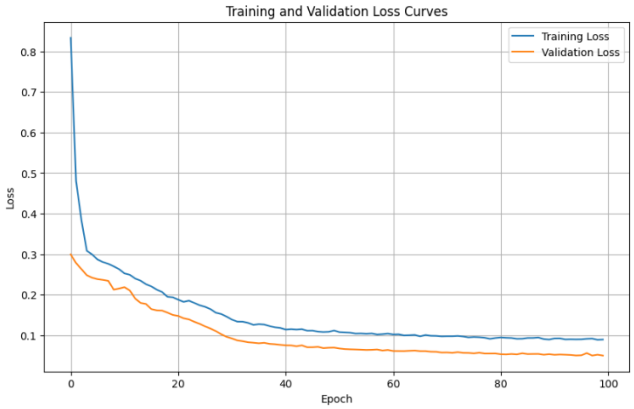

Figure 5. Loss-curve of training and validation for configuration h2.

decrease in the loss of the training and validation sets, with the latter being smaller throughout the epochs

To stabilize convergence a conservative training rate of 1×10^{-4} (Adam) was used to ensure fine-grained weight updates. Second, to bypass GPU VRAM limitations that capped physical batch sizes at 16, we implemented **gradient accumulation**. This technique allowed us to simulate a larger effective batch size, which improved generalization without exceeding hardware constraints.

In the case of Model II, gradient accumulation and scheduling was used, furthermore, to obtain sharper reconstructions we extended the training to 200 epochs. The curves of training/validation losses were overlapping from approximately 125epochs.

4.5. Determining a threshold

The methods we have tried for setting a threshold includes:

- **False Positive Rate:**Calculate the reconstruction loss/anomaly score of each image in the validation-set. Set the threshold such that e.g. 95% of the normal images in the validation-set will have loss lower than this score. All images with a higher score will be classified as anomalous. This should lead to some normal images being misclassified. We tried diffrent scores, from 97% to 50%.

- **Sigma-N Rule**: as described in section 3.3. We tried k-values in the range from 5 down to $1e-05$.

- **Interquartile Range (IQR):**This approach uses quartiles to define the threshold [4]. It is designed so that outliers in the validation set does not get a large impact on the threshold. It still calculates the loss for all validation images.

4.6. Testing with a second test-set, Model I

We also made a smaller test-set, test-set 2, where we removed images with small anomalies. This test-set had a total of 68 images, both good and anomalous samples. We tested with the same threshold on both test-sets.

4.7. Robustness under Domain Shift

In the project we also performed a domain shift on the test-set to check the robustness of our model. The domain-shift is performed by adding data augmentations from Torchvision to the test-set. The data augmentations includes rotation, zooming in, change of lighting conditions, resolution change and applying gaussian noise as "sensor-noise". The details of the test-augmentations are described in Table 2. To mitigate the performance-drop we tried augmenting the training-set with similar, but not identical data augmentations, and training the model with the augmentations seen in table 3. Then we tested on the shifted test-set.

Table 2. Augmentations on the test-set (Domain-shift)

Operation	**Configuration Details**
Resize	img.input-size $\times 1.3$ (Buffer Zone)
RandomAffine	deg: 5°, scale: [1, 1.1], fill: 200
Resolution	downscale up to 0.5 of input-size, rescale to org.size
ColorJitter	brightness: 0.3, contrast: 0.3
Gaussian Noise	mean=0, std=0.03

Table 3. Augmentations I and II on the train-set

Operation	Configuration Details
Augmentation I	
ColorJitter	brightness: 0.15, contrast: 0.15, hue: 0.05
Gaussian Noise	mean=0, std=0.01
Augmentation II	same ColorJitter and Noise as I, additionally
Resolution	downscale up to 0.8 of input-size, rescale to org.size

5. Results

5.1. Image-level classification

We evaluated the model's ability to distinguish between normal and anomalous images using ROC-AUC. The F1-score was used for the h2-configuration to see the difference between the two test-sets.

5.1.1 Architecture 1

Table 4 shows the results corresponding to the ROC-AUC for the first architecture implemented. The results were disappointing—well below state-of-the-art benchmarks, additionally, the model was reconstructing the anomalies in the image—input images are almost equal to their reconstructions, as seen in Fig.6, increasing the number of false negatives.

Table 4. Performance of architecture 1, image-wise ROC-AUC.

Config	Res.	ROC-AUC	Loss-Func.
Config 1	1024x1024	0.436	MSE

5.1.2 Architecture 2

Config 2a yielded a ROC-AUC of 0.61, as seen in Table.5. Anomalies were reconstructed.

Table 5. Performance of architecture 2. Image-wise ROC-AUC.

Config	Res.	ROC-AUC	Loss-Func.
Config 2a	512x512	0.61	MSE

5.1.3 Architecture 3

Table 6 summarizes the performance metrics for architecture 3. This architecture demonstrated successfully the core capability to replicate only normal characteristics of the image—anomalous features weren't reconstructed.

Table 6. Performance of architecture 3. Image-wise ROC-AUC

Config	Res.	ROC-AUC	Loss-Func.
Config h1	256x256	0.61	MSE+SSIM ($\alpha = 0.2$)
Config h2	256x256	0.65	MSE+SSIM ($\alpha = 0.1$)

5.1.4 Model II

"Config e4" achieved a peak image-level ROC-AUC of 0.70. While *"Config e2"* performs poorly at global classification, it excels in localization with a superior pixel-wise ROC-AUC of 0.83. Ultimately, *"Config e5"* is the preferred choice for its dual-task proficiency in both detection and segmentation.

Table 7. Performance of model II. Image and pixel-wise ROC-AUC.

Config	Res.	Image ROC-AUC	Pixel ROC-AUC
Config e5	256x256	**0.70**	**0.84**
Config e4	256x256	**0.70**	0.52
Config e2	256x256	0.56	**0.83**

5.2. Localization

The models ability to localize anomalies is based on a qualitative visual inspection of the heatmaps for model I, additionally pixel-wise ROC-AUC for model II.

5.2.1 Architecture 1

As seen in the heat-map in Figure 6, images are reconstructed very similarly to its original counterpart, even replicating the anomalous regions. For that reason, the pixel-wise difference between both images will be lower than expected.

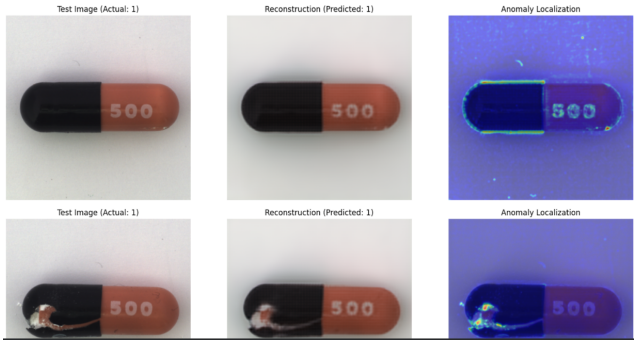

Figure 6. Input image, reconstruction and error heatmap of config 1

5.2.2 Architecture 2

Some samples exhibit minor "halo" artifacts around capsule boundaries, seen in Figure 7. This indicates that while internal texture is well-reconstructed, the model faces challenges with high-frequency edge transitions.

5.2.3 Architecture 3

Architecture 3 did not reconstruct anomalies as seen in Figure 8. The heatmaps significantly highlights the areas with

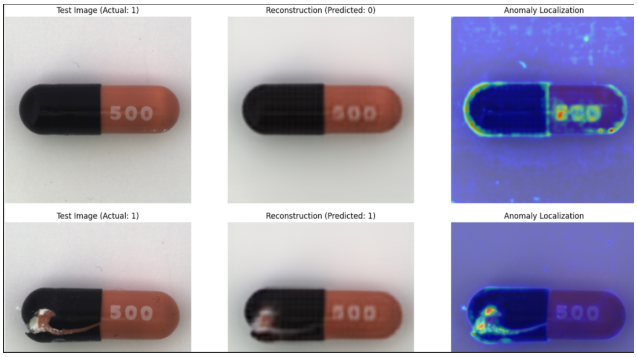

Figure 7. Input image, reconstruction and heatmap of Config 2a

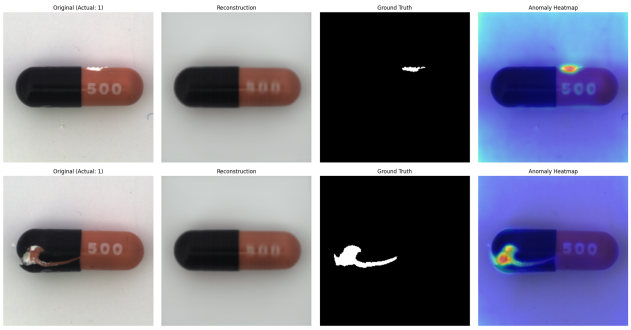

Figure 8. Input image, reconstruction, mask and heatmap of Config h2

defects in the image. In this case, also ground truth masks were plotted next to the heatmaps to facilitate assessing the behavior of the model.

5.2.4 Model II

The pixel-wise ROC-AUC score of 0.83 of *"Config e2"* and 0.84 of *"Config e5"* indicates they clearly detect anomaly regions and the amount of false positive pixels that are classified is minimal compared with *"Config e4"* and the other models. Visually we can confirm this analysis with the heatmap in Figure 9 and the comparison with the ground-truth masks.

5.3. Results on threshold

We had a hard time determining a threshold for all our different architectures and trained models, regardless the method used we had similar results for the threshold. In the end we went with the sigma-N rule with $k = 0.0001$, for the h2-configuration seen in Table.6. With this, we achieved the confusion matrices seen in Table.8.

Table 8. Classification performance for the *h2-configuration* across different threshold factors (k).

Thresh.	TP	TN	FP	FN	Acc.	Prec.	Rec.	F1
$k = 0.0001$	59	17	6	50	57.6%	90.8%	54.1%	67.8%
$k = 1$	35	19	4	74	40.9%	89.7%	32.1%	47.3%

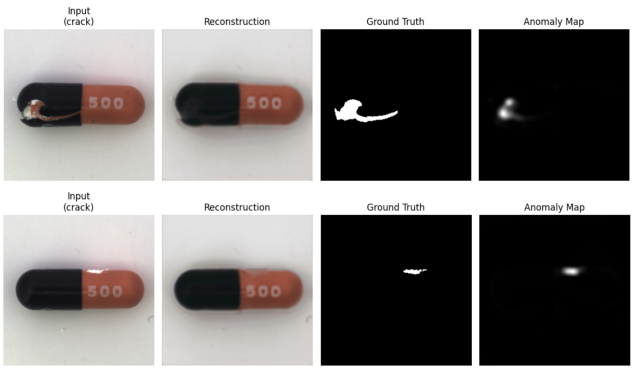

Figure 9. Input image, reconstruction, mask and heatmap of Config e5.

When increasing k, e.g. to $k = 1$, we got more FN's and less TP's in the confusion matrix,the model incorrectly categorized anomalous samples as normal. This highlights that we needed a very conservative k-value to be able to "catch" anomalies.

5.4. Results with the second test-set

The confusion matrix of test-set 2 with the h2-configuration is shown in Table.9.The F1-score improved to 79.5%. The F1-score is evaluated due to it being threshold-dependent.

Table 9. Classification performance for the *h2-configuration* on Test-Set 2

Dataset	TP	TN	FP	FN	Acc.	Prec.	Rec.	F1
Test-Set 2	33	18	5	12	75.0%	86.8%	73.3%	79.5%

5.5. Robustness under domain shift

Model II performance under the domain-shift is shown in Table 10. As expected, ROC-AUC score dropped due to the model's inability to recreate images outside its normal distribution, but we can see when training with augmented data that the performance increase again.

Table 10. Performance under domain shift. Image-wise ROC-AUC.

Config	Res.	ROC-AUC	Shifted-data
Baseline	256x256	0.70	None
Config e5	256x256	0.58	Test
Config ds1	256x256	0.63	Train I +Test
config ds2	256x256	0.64	Train II + Test

5.6. Performance comparison to other papers

The highest image-wise ROC-AUC-score we obtained was 0.70 for the e4-configuration. This is below SOTA benchmarks for the MVTec AD capsule-class obtained in other papers. For instance, in [11], they report image-wise

ROC-AUC scores over 0.80, and in [9] they even report ROC-AUC scores over 0.90.

6. Discussion

6.1. Influence of Architecture

Our experiments demonstrated that the choice of architecture for the autoencoder, especially the bottleneck size, significantly impacts performance. A to large latent space allows the model to learn an identity mapping of the input-image, and anomalies are reconstructed. The whole idea to use an autoencoder for this type of anomaly detection, is that the autoencoder is not able to reconstruct anomalies, because this will lead to a higher anomaly-score between input-images and reconstructed-images when the input-image contains an anomaly. Therefore it is important that the autoencoder can not reconstruct anomalies. That's why Model II was introduced, with the intention to project anomalous samples into the manifold of normal samples, penalizing the reconstruction of anomalous features in the training phase, which improved AUC-ROC both image and pixel-wise.

6.2. The problem of determining a threshold

During testing, we saw that we need a low k-value to be able to catch anomalies, as seen from Table.8. Combined with the results from testing on the second test-set, seen in Table.9, we suspect that the issue might be in how we calculate the loss. The loss function is described in section.3.2. It calculates loss per-pixel, but average the pixel-wise error over the whole image. This might lead to a small defect being "lost" in the average of normal pixels. Consequently, the distribution of anomaly-scores for anomalous images significantly overlap with the the normal image distribution's anomaly-scores. This overlap can make the selection of the parameter k highly sensitive; a higher threshold will detect only large anomalies, while a lower threshold will have trouble distinguishing between anomalous and normal images due to their similar anomaly-scores. This assumption is verified by our model performing better on the second test-set, where images with small defects were removed manually.

6.3. Error Analysis

Qualitative analysis of the generated heatmaps and reconstructed images reveals distinct failure modes:

- **False Negatives:** The model frequently fails to detect low-contrast anomalies, such as faint imprints or subtle color shifts on the capsule body. In addition, these anomalies are also small in size. We believe we fail to detect these anomalies due to the pixel-wise reconstruction score being "lost" when the average of the whole image is calculated, and therefore the anomaly is not detected with the threshold.
- **False Positives:** A recurring issue is the "edge effect." for the first two architectures. The autoencoder struggles to reconstruct boundaries between the capsule and the background. Spatial misalignments or blurriness in the reconstruction result in large pixel-wise errors along the object. This generates a "halo" of high error in the heatmap, which the thresholding logic classifies as a defect. For architectures in model I, it can also be a case of thresholding whether an image is classified as false negative or false positive, as the threshold is set low, almost at the "normal" anomaly-score.

6.4. Limitations and Robustness

A major limitation of our unsupervised approach is its lack of semantic understanding of "defects." The model relies entirely on the assumption that anomalies fall outside the learned manifold of normal data. As shown in our domain-shift experiments, the model is sensitive to rotational changes and different lighting. Because the training data are tightly controlled, the model fails to generalize to shifts. Retraining on shifted data mitigates this, it highlights the model's inability to learn invariant object representations without explicitly have seen the type of data before.

7. Conclusion

In this project we have studied unsupervised anomaly detection and localization with convolutional autoencoders trained exclusively on normal images. We have used the reconstruction-error as an image-wise classification score and pixel-wise heat-maps for localization. Our best configurations obtained an image-level ROC-AUC score of 0.70 on the standard testing-set, and a pixel-wise ROC-AUC score of 0.84. While this is below the state-of-the-art performance for classification of the MVTec AD data-set, this is expected as we implemented our own model and did not rely on e.g. a pretrained backbone. A key challenge for us was the selection of a threshold; we struggled to find a balance between capturing small-defect areas, where a high threshold gave many false negatives, and classifying most images as anomalous, in other words many false positives. As anticipated, we observed a performance drop under a domain-shift on the test-set. The ROC-AUC score decreased to 0.61 compared to 0.70 in the case of Model II. When also training the model with data-augmentations, the ROC-AUC increased to 0.67, indicating that exposure to shifted data can recover some of the losses. We suspect that our main issue with model I was the loss-function, further work could be to explore other implementations of this. For instance the ones seen in the papers with state-of-the-art architectures, [9, 11].

References

[1] Alexander Bauer, Shinichi Nakajima, and Klaus-Robert Müller. Self-supervised training with autoencoders for visual anomaly detection, 2024.

[2] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The MVTec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision*, 129(4):1038–1059, 2021. doi: 10.1007/s11263-020-01400-4.

[3] 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 (CVPR)*, pages 9584–9592, 2019. doi: 10.1109/CVPR.2019.00982.

[4] Ch. Sanjeev Kumar Dash, Ajit Kumar Behera, Satchidananda Dehuri, and Ashish Ghosh. An outliers detection and elimination framework in classification task of data mining. *Decision Analytics Journal*, 6:100164, 2023.

[5] Jin Uk Ko, Kyumin Na, Joon-Seok Oh, Jaedong Kim, and Byeng D. Youn. A new auto-encoder-based dynamic threshold to reduce false alarm rate for anomaly detection of steam turbines. *Expert Systems with Applications*, 189:116094, 2022.

[6] Guoting Luo, Wei Xie, Ronghui Gao, Tao Zheng, Lei Chen, and Huaiqiang Sun. Unsupervised anomaly detection in brain mri: Learning abstract distribution from massive healthy brains. *Computers in Biology and Medicine*, 154:106610, 2023. doi: 10.1016/j.compbiomed.2023.106610.

[7] MathWorks. Autoencoders, 2026. [Online]. Available: `https://se.mathworks.com/discovery/autoencoder.html`. Accessed: 2026-01-08.

[8] MathWorks. Roc curve and performance metrics, 2026. [Online]. Available: `https://se.mathworks.com/help/stats/performance-curves.html`. Accessed: 2026-01-11.

[9] Lu Wang, Dongkai Zhang, Jiahao Guo, and Yuexing Han. Image anomaly detection using normal data only by latent space resampling. *Applied Sciences*, 10(23), 2020.

[10] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[11] Edward K.Y. Yapp and Ngoc C.N. Doan. Anomaly detection on mvtec ad using vq-vae-2. *Procedia CIRP*, 130:1809–1814, 2024. 57th CIRP Conference on Manufacturing Systems 2024 (CMS 2024).