Unsupervised anomaly detection in industrial image data with autoencoders

Dennis Rotondi[1,†], Leonardo Lavalle[1,†]

[1]*Student of Artificial Intelligence and Robotics, DIAG, Sapienza University of Rome*

Abstract

Anomaly detection is a rapidly growing field, and Autoencoders have become a widely adopted method for identifying anomalous samples through the use of a threshold on the reconstruction error. In this study, we introduce a novel extension of Convolutional Autoencoders to address the challenge of handling datasets with multiple categories, such as the MVTec AD dataset containing industrial products. Previous solutions for this benchmark have utilized separate models for different sub-categories, like objects and textures. Our approach instead utilizes a single model, trained on all classes simultaneously, with the incorporation of denoising and contractive strategies to improve anomaly detection performance. Additionally, a classification module is integrated on the latent space to determine the optimal threshold for each class, resulting in a significant improvement of over 1.5 times compared to the baseline model.

HIGHLIGHTS

- implementation from scratch of the simple Convolutional AE for the task using *PyTorch-Lightning*;
- extension of it with Contractive and Denoising advanced approaches;
- built a (*novelty*) Mixer AE that selects a different threshold according to each object;
- with structural loss and side-task training we got results comparable with the highest score recorded of other AE works that use different models for different sub-categories;
- run a performance test using an ensemble of all our implemented solutions.

1. Introduction

Anomaly detection is the process of identifying data points that deviate markedly from the rest of the data, those points are called outlying observation or "outlier" [1], for this reason it is also referred to as outlier detection. It has been a popular area of research in various fields for over 60 years, and nowadays its importance is increasing due to its countless applications in areas that are getting more and more interest such as risk management, compliance, cybersecurity, financial surveillance, and healthcare. In fact, the global anomaly detection market is expected to grow by 4.23 billion dollars during 2022-2026 as reported in [2]. This study identifies the rising incidence of internal threats and cyber frauds as one of the prime reasons driving the anomaly detection market growth during the next few years. In general, the increasing demand from institutions in spending on retail security systems will lead to sizable demand in the market. In a manufacturing context, anomalies are exposed variations from the norm, and they can have both positive and negative implications. For example, anomalous data can reveal problems, like a technical error on the production line. But, an anomaly can also highlight an opportunity, by revealing a way to improve a manufacturing process and saving money for the company. Industry 4.0 has led to a dramatic increase in the number of efficient software solutions and analytics programs available to help companies collect, measure, and manage data from all aspects of their operations [3, 4]. Anomaly detection involves identifying unusual or unexpected events, which can be challenging due to the rarity and unpredictability of such occurrences. This type of problem is distinct from others because it deals with a minority of data points that do not conform to the expected pattern or behavior. To be effective, a model for anomaly detection must be able to handle the unknown nature of these outlier data points, the diversity of anomaly types, and the imbalanced distribution of anomalies compared to normal instances [5]. Therefore, that makes it difficult if not impossible to gather a large number of labeled abnormal examples: we cannot store (memorize) all of them. At this point should be clear that a supervised approach is not ideal and it's here that deep neural networks come to the aid. They are well-suited to this problem because they can use complex combinations of linear and non-linear functions to learn expressive features of a fixed number of instances keeping the use of memory contained. It's plenty of strategies that involve deep learning for this task [5], in this work we'll focus on the process of learning the distribution representations of data instances by having seen only nominal (not abnormal) ones and optimizing a generic feature learning

EAI - AI for Visual Perception in HCI & HRI, Final Project

[†]These authors contributed equally.

rotondi.1834864@studenti.uniroma1.it (D. Rotondi);
lavalle.1838492@studenti.uniroma1.it (L. Lavalle)

objective function (the loss) that is not primarily designed for anomaly detection. The general formulation is easily compacted with Eq. (1)

$$\Theta^*, W^* = \arg\min_{\Theta, W} \sum_{x \in X} \ell(\psi(\phi(x; \Theta); W)) \tag{1}$$

$$S_x = f(x, \phi_{\Theta^*}, \psi_{W^*}) \tag{2}$$

here x is a sample drawn from the distribution of the normality X; ϕ a function that works on the input and produces its embedding, that is given to a function ψ, which outcome aims to minimize a loss function ℓ. After this optimization, the anomaly score S_x of Eq. (2) allows deciding whether an arbitrary sample x is anomalous or not according to a threshold. We'll see this framework specifically in the domain of Autoencoders (AE) [6], where ϕ, ψ take the name of encoder and decoder functions, and ℓ (often also S_x) is the reconstruction loss, which quantifies the differences between the original and reconstructed samples. The modality in which we can detect anomalies is very easy: if the reconstruction error is too high, the input it's targeted as an outlier (because the Autoencoder learned to represent patterns not existing in this sample and cannot codify it well), otherwise it belongs to the nominal distribution. For this project we got assigned the MVTec AD industrial inspection benchmark [7]. This dataset consists of 5354 high-resolution images of five unique textures and ten unique objects from different domains. Approximately one-third of these are used for the testing procedure 467 good and 1258 defective. There are as many as 73 different types of anomalies in the form of defects or structural deviations in the objects or textures: Figure 1 collects some examples. This task is especially

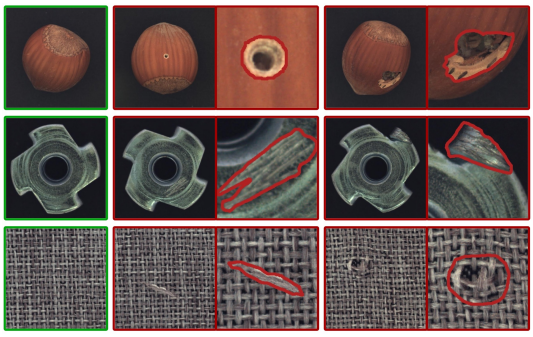

Figure 1: Two objects and one texture from the MVTec AD dataset. For each of them, one defect-free image and two images that contain anomalies are displayed. Taken from [7].

hard, as errors can vary from subtle changes such as thin scratches to larger defects like missing components, and indeed simple Autoencoders struggle because the feature space could be highly nonlinear and map similar samples to far regions in the latent space. In particular, we want the representation to be sensitive to changes in the directions that go from a normal instance to another and at the same time insensitive to changes in the directions that goes from a normal sample to an anomalous one. To this purpose, several types of advanced architectures have been introduced to learn richer and more expressive feature representations like Contractive [8], Denoising [9] or Variational Autoencoders [10]. Moreover, we decided to explore new techniques that come up to our mind with the aim of smoothing the learned space by adding training tasks that we'll describe in Section 3.

2. Related Works

As premised, the renewed interest in outlier detection of the last years in conjunction with the explosion of deep learning techniques thanks to the recent powerful hardware, had seen these artificial intelligence solutions applied to the problem not only by learning the distribution of the normality. The MVTec AD benchmark has been released only a few years ago (2019) and its extension just in 2021 [11], when most of the modern SOTA computer vision models were already been developed. For this reason, it's worth mentioning that actually the best results are provided by framework with no Autoencoders involved. Above all PatchCore [12] of Amazon Science (*facis de necessitate virtutem*), which exploiting a memory bank of nominal patch features (feature dictionary) basically solves the anomaly detection task in this industrial environment (over 99% AUROC) retrieving at inference time the nearest neighbors. Another very popular paradigm here, with similar excellent performance, is via Student Teacher Knowledge Distillation [13, 14], the idea is that a "student" model is trained to mimic the behavior of a "teacher" (pre-trained) model; the anomaly is identified when the distance between the two predictions is large. Recently, reconstruction-based methods have not been widely discussed due to their poor reconstruction quality and low performance. However, they have the advantage of not requiring additional, costly training samples for unsupervised training, which makes them more practical and faster in manufacturing decision-making. Among them, Autoencoders are preferred over GANs [15], because despite the fact that they follow the same principle and therefore their performance are comparable in their simplest form, GANs inference is computationally much more expensive. A mixed approach has been proposed in Adversarial Autoencoders [16], where a discriminator is exploited to better control the latent space distribution and the reconstruction of the normal images. What we are going to develop is founded on the basis of the Autoencoder works of [17, 18], here the authors introduced the idea of anomaly detection through the reconstruction anomaly score relying on robust features insensible to

outliers. While we are going to leverage side tasks on the feature representation, during the training we also want to focus our attention on the structure of the input [19, 20, 21] that have proven to work very well, instead of pixel-by-pixel metrics. We restrict ourselves to deterministic Autoencoder frameworks because according to the literature [11, 19] the probabilities obtained from VAEs and other stochastic models might fail to model the true likelihood of the training data and, in any case, do not report significant improvements over the use of Denoising and Contractive ones. We propose a novel architecture called Mixer-Autoncoder, which aims to be dataset independent in the thresholding process and can handle multiple categories without the need of multiple models for specific sub-categories (objects and textures), as seen in previous works such as [19, 20]. Eventually, we want to explore on this benchmark the power of an ensemble [22] to understand whether in this way it's possible to neglect the bias in reconstruction introduced by these simple models or not.

3. Methods

There are numerous approaches that have been proposed for unsupervised anomaly detection, and the field is quite varied as explained in Section 2. Autoencoders are only one of the many and they belong to the reconstruction-based methods family: namely, those models that reconstruct data points from a lower-dimensional latent space and identify anomalies as the data points that are poorly reconstructed. We have begun implementing a basic Autoencoder, which will serve as the foundation for more complex variations. Although the task may seem straightforward, there are numerous intricate details involved that we will delve into in this section.

3.1. Convolutional Autoencoder

For this project, we will focus on Convolutional Autoencoders (CAEs), which are popular in unsupervised anomaly detection due to their simplicity and ability to process data with spatial structure, such as images. Deep Convolutional Autoencoders are particularly well-suited for the MVTec dataset because they use convolutional layers, which allow them to learn translation-invariant features, capture spatial relationships in the data, and learn hierarchical features through multiple layers of convolutions. In addition to these benefits, CAEs are more efficient than regular Autoencoders because they have fewer parameters. In Figure 2 a general overview of this architecture. Reconstruction-based methods consist of two phases: (i) the first one where the model attempt to reconstruct defect-free training samples through a bottleneck: the latent space representation z, which can be

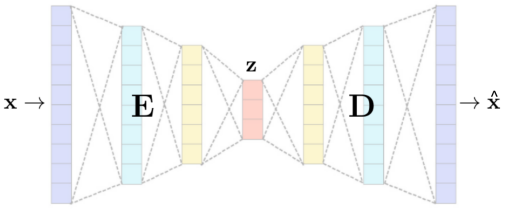

Figure 2: Convolutional Autoencoders classic representation, z is the latent representation, x is the input and $\hat{x}$ its reconstruction. The squares can be interpreted as convolutional kernels.

in $\mathbb{R}^n$ or Cartesian's products of it. Eq. (1) in this domain can be written using the following relations:

$$z = \phi(x; \Theta) = E(x; \Theta) \tag{3}$$

$$\hat{x} = \psi(z; W) = D(z; W) \tag{4}$$

which are referred to as the Encoder and Decoder functions respectively. The parameters of these functions are set to minimize the reconstruction loss, which is typically defined as the Mean Square Error (MSE):

$$\ell(x) = ||x - D(E(x; \Theta); W)||_2^2 \tag{5}$$

when the image values are scaled between 0 and 1 and each pixel is treated as a probability, this error becomes Binary Cross Entropy (BCE); (ii) the second one is the prediction phase, in which test images are compared to their reconstructions using a similarity function that produces the "anomaly score". This score may or may not be equal to the loss. Since Autoencoders fail to reproduce images that differ from the data that was observed during training, the higher the anomaly score the higher the probability that the image is an outlier. The overall performance of all systems depends on the crucial decision of setting a threshold value for the anomaly score, which determines when something should be classified as an anomaly or not. Although there is no ideal threshold that can consistently solve the anomaly detection task, there are three different approaches to determining it: by *manually* tune the threshold; using an *optimization metric*: you can use an optimization metric such as the area under the ROC curve to choose the threshold that maximizes the performance of the model; with an *heuristic approach*: you can use a heuristic approach such as setting the threshold to be the average reconstruction error μ plus a multiple of the standard deviation σ of the reconstruction errors. This last approach is the one we use because it is simple to implement and very effective. In particular, we keep track of the mean and the standard deviation of the reconstruction error of each batch and then compute their average at the end of the epoch. We

make this computation for each *i-th* training epoch and we update the threshold according to Eq. (6):

$$T_i = (1 - \alpha) \cdot T_{i-1} + \alpha \cdot (\mu + (\beta \cdot \sigma)) \tag{6}$$

Here, the variable α determines the extent to which the newly computed thresholds change the previous one T_{i-1}, while β is a variable that can only take values between $(-1, 1)$. This determines the amount of the standard deviation that is added to or subtracted from the mean value. Its negative configurations explicitly introduce a conservative behavior that gives more importance to the Recall over the Precision metric increasing the number of false positives, a design choice that is preferable when the risk is to put defective products on the market.

3.1.1. Implementation details

CAE serves as the foundation for all our implemented architectures. Unlike previous works such as [11, 19, 20] which employed specialized Convolutional AEs on subsets of MVTec, what we need here is to develop a comprehensive solution that can handle all categories. To achieve this, we design a structure that is large enough to model all categories while keeping the number of parameters limited, making it a preferable solution over more advanced methods. Therefore we construct from scratch our custom CAE using the best practices and well-working parameters outlined in [23, 24, 25]. The architecture layers of the proposed method are detailed in Table 1. A modification of this CAE that performed better in the more advanced Mixer-AE solution, discussed in Section 3.4, is presented in Table 2. It can be seen that for each version, the most suitable *input image* size, with which the best behaviors were observed, was highlighted.

Input Image: **256x256x3**

Layer	Kernel / Stride / Padding	In C	Out C
Conv1	4x4 / 2 / 1	3	16
Conv2	4x4 / 2 / 1	16	32
Conv3	4x4 / 2 / 1	32	64
Conv4	4x4 / 2 / 1	64	128
Conv5	4x4 / 2 / 1	128	256
Conv6	4x4 / 2 / 1	256	512
Conv7	4x4 / 1 / 0	512	1024
MLP	1024x*ld* / None / 0	1024	*ld*

Table 1
In our proposed autoencoder architecture, the encoder is composed of a series of convolutional layers and a final MLP, as depicted by the values provided. In particular: "In C" and "Out C" is the number of input channel and output channels (filters) respectively; *ld* is the latent dimension (128 produced the best result). The decoder is constructed as the reverse of the encoder, using ConvTranspose layers instead of Conv. The activation functions employed here are Leaky Rectified Linear Units with a slope of 0.2, applied after each Conv together with the batch normalization. In the final decoder layer, there is "tanh" activation to produce values in the range of the normalized input image (-1,1).

Input Image: **224x224x3**

Layer	Kernel / Stride / Padding	In C	Out C
Conv_i	3x3 / 1 / 1	i	i+1
Conv_i	3x3 /1 / 1	i+1	i+1
MaxPool_i	2x2 / None / 0	i+1	i+1
ConvT_j	2x2 / 2 / 1	j	j
Conv_j	4x4 / 2 / 1	j	j+1
Conv_j	4x4 / 2 / 1	j+1	j+1

Table 2
In this variation, we have discarded the MLP implementing a Full Convolutional model. The "convolutional triplet" is iteratively repeated for every value of i in the list [3, 64, 128, 256, 512, 1024] (except the last, i+1 is the next in the sequence), with the exception of the last iteration where MaxPool is not applied. Similarly, the "transpose convolutional" one is repeated for every value of j in the list [1024, 512, 256, 128, 64, 3] (except the last, j+1 is the next in the sequence), but in the final iteration, ConvT is not used. All the normalization and activation functions remain invariant w.r.t. the previous version.

3.2. Denoising AE

While the initial version of our CAE has demonstrated promising performance, one key aspect was that the anomalous images were reconstructed too well. In terms of latent space, this means that it may not be as compact as desired and may not be sufficiently sensitive to deviations from the normal data distribution. For this reason, we need the introduction of Denoising Autoencoders (DAE), a particular AE extension which is trained to reconstruct the original input image (or "clean" the original input image) from a corrupted version of it. This corruption is introduced by adding noise randomly to the input image. The original authors of DAE applied a zero-pixel noise parameterized by the desired proportion v of "destruction", namely for each input x of d components, a fixed number $v * d$ of them are chosen at random, and their value is forced to 0, while the others are left untouched. However, we've found better outcomes using *random white noise* that can be adjusted in amount using a specific hyperparameter $v = 0.3$ Eq. (7).

$$x_{Noise} = \text{clamp}(x + v * W) \tag{7}$$

W has the same size of x and each element w of W is distributed according to the gaussian $\mathcal{N}(0, 1)$. The clamp function is the standard used in computer graphics that ensures to have the image values in the normalized range (-1,1). The feature space benefits from this learning process because the model is exposed to the same image(s)

corrupted with different realizations of noise. If the model is able to correctly reconstruct the image in all cases, we can conclude that the same features are being extracted from samples that are not too dissimilar in the image space. As a result, during the prediction phase, even small anomalies may be treated as mere noise and filtered out. This leads to a better anomaly score and also makes the model less sensitive to anomalies at the latent space level, which is an ideal outcome. It is peculiar that the effectiveness of this strategy, which resulted in a 25% improvement in our results, was not acknowledged in other works, despite its simplicity.

3.3. Contractive AE

Another solution developed in the last decades that we decided to compare to Denoising AE are Contractive Autoencoders (COAE) [8]. They are another type of AE neural network that is specifically designed to encode and reconstruct data in a manner that is resistant to small perturbations in the input. This property, known as "contractivity", is achieved through the incorporation of a regularization term in the loss function that penalizes large changes in the encoded representation for small changes in the input. These networks have been shown to learn more robust and faithful representations of the data. The loss function becomes then:

$$\ell(x) = \|x - \hat{x}\|_2^2 + \lambda \cdot \|J_E(x)\|_F^2 \tag{8}$$

Where the second addendum is the Frobenius norm of the Jacobian of the Encoder function E weighted by a factor λ. In all of our experiments, we set this term around 0.0001. This relatively small value allows the mean squared error (or another reconstruction loss) to play a more significant role in the loss function, while still providing some regularization. Denoising and Contractive models are not mutually exclusive: it's certainly possible to use them separately but we can (and we did) combine them together trying to take advantage of the best each one has to offer.

3.4. Mixer AE

After dealing with the latent space problem (DAE and COAE), we observed that the MVTec AD dataset exhibited significant variance in the anomaly scores of its various classes (which include both objects and textures) and this justifies why all the authors that've tackled this problem with AEs decided to treat some categories separated as anticipated. We found that these scores were often quite distinct between objects (Figure 3), suggesting that different threshold values may be appropriate for different classes instead of using a single one for all of them. Initially, we had intended to use additional tasks, such as

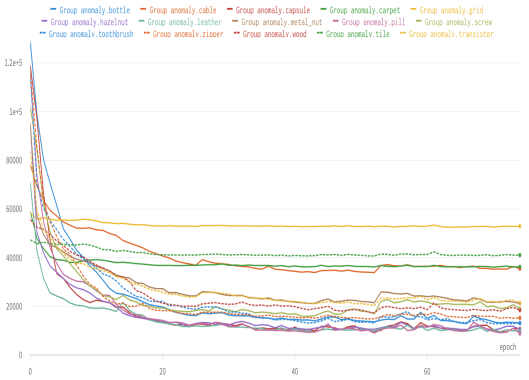

Figure 3: Anomaly scores of MVTec classes during training.

guessing the input's class label from its latent representation, to smooth the latent space and learn the most important features for each class. However, we eventually realized that we could also use this classification training to accurately select the appropriate threshold value for each class. Therefore, *we've introduced a novel extension to our model architecture that has been never used before for anomaly detection*, the **Mixer Autoencoder**, which allows class-specific thresholding. It takes its name from the Mixer Audio that is used in radio and in recording studios. In a similar manner to how audio mixers adjust the levels of various audio channels to produce a desired output sound, our Mixer Autoencoder modulates the threshold for making a final prediction based on the input object. This allows the model to effectively "mix" and adjust the relative importance of different features for different classes, leading to more accurate predictions. A general overview of the Mixer Autoencoder architecture is depicted in Figure 4. Here, there are in sequence the AE, the classifier of the latent space representation h and at the end of them we have the *Mixer* module which takes as input the anomaly score of the AE and the predicted class c of the classifier so to ideally output the anomaly prediction choosing the best threshold possible $thresholds[c]$ as summarized by Algorithm 1. The classi-

Algorithm 1 MIXER AE inference logic.

```
for x ∈ Images do
    h, x̂ ← AE( x )
    c ← classifier( h )
    if AnomalyScore( x, x̂ ) > thresholds[c] then
        prediction( x ) ← anomaly
    else
        prediction( x ) ← normal
    end if
end for
```

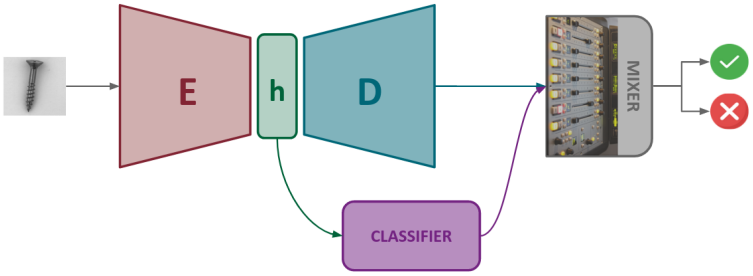

Figure 4: Visual representation of the Mixer Autoencoder architecture. E and D are the Encoder and Decoder functions implemented using convolutions and de-convolutions. The Classifier module takes the latent representation h in input and outputs the class to which apply the threshold.

fier module, which identifies the class of the input based on its latent representation, is a multi-layer perceptron (MLP) with three layers and dropout regulation in the case of the CAE described in Table 1. In the variant described in Table 2, a single convolutional block (3x3/1/0) is also used before the MLP (this variant performed best in all our experiments). In any case this is a very small network with less than 1M parameters that uses ReLU activation. At implementation level, this is a natural extension of the previously described models, the update threshold mechanism is performed as a generalization of the single Eq. (6) iterating on all the classes.

3.5. Structural Loss

The similarity functions described so far are based on per-pixel comparison of the input with its reconstruction, but recently is been pointed out [26] the disadvantages of these functions in Autoencoder frameworks in favor of metrics that incorporate image spatial information of local patch regions using structural similarity [27]. For this reason, we adopted as an anomaly score the Structural Similarity Index Measure (*SSIM*) metric. This trial is inspired by the human visual perception system, which is able to identify differences between two scenes based on structural information such as luminance, contrast and patterns. It is hoped that a metric that replicates this behavior will perform better on texture classes, where the MSE has difficulty [11]. The SSIM of two images x and y can be computed with Eq. (9).

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} \qquad (9)$$

These three functions output the similarity between the two images based on luminance, contrast and structure metrics respectively. The details of these can be found in [27], including the fact that $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ denote the relative importance of each of the metrics. There are variations of SSIM, such as MSSIM [21] which applies the metrics locally rather than globally and computes the average, and CW-SSIM [20] that is calculated using a complex wavelet transform, among others.

3.6. Ensemble

Ensemble methods have been shown to be effective for anomaly detection [28], as it can be challenging to design a single model that can accurately identify all types of anomalies. Indeed, by aggregating the predictions of multiple Autoencoder models, it is possible to improve the overall performance of the system reducing the bias and poor reconstruction power of these models. We decided to employ the *voting* ensemble method, the most simple and intuitive one. In particular, the output class is chosen according to the weighted majority paradigm:

$$y_{voting}(x) = \arg\max_{c} \sum_{m \in M} w_m \cdot I(y_m(x) = c) \qquad (10)$$

with $w_m > 0$ and $\sum_{m \in M} w_m = 1$. M is the number of detectors involved while c represents all the classes (in our case only c_1=anomaly and c_2=normal), x is the input image and I is a function that returns 1 if the expression inside it is True. The weights w_m represent the prior probability of each classifier. In our experiments, we set these weights normalized according to different performance metrics. As in the previous cases, this isn't the key to heaven and it may be necessary to experiment with different configurations to find the approach that works best for our dataset.

3.7. Training Setup

To conclude this section, we will provide an overview of the training procedure for our final model, the Mixer Autoencoder: it incorporates both the contractive and denoising strategy. The Convolutional Autoencoder and classifier have been trained from scratch in *PyTorch-Lightning*, using Adam as an optimizer since it works much better than SGD in this domain for its adaptive

property. We set Adam epsilon at 1e-6 for numerical stability and the learning rate at 0.001. The selected training scheduler is Reduce-On-Plateau, which is quite effective and lowers the learning rate when the performance metric has stopped improving. Network weights are initialized from a Gaussian distribution $\mathcal{N}(0, 0.02)$, as suggested in [29]. We've also applied Early Stopping for regulation on anomaly detection *F1 Score* of the validation set for 20 epochs. The computation of the loss uses batches of size 128. The loss is a weighted sum of the main error (MSE or SSIM depending on the experiment, weighted 1), the Crossentropy loss for the Mixer multi-class classification problem (weighted 3 to quickly decrease and improve performance), and the Frobenius contribution (weighted according to Eq. (8)).

4. Results

The main task of our project was to get the proper reconstruction in a way that anomalies do not appear in the decoder output or at least that the latent space is not able to encode this information, even if it's not possible to get apparently the difference by visual inspection (the same principle under the modern deep fake architectures). We cannot stress enough how the dataset was complex and how each object and texture of the dataset has a different degree of difficulty in being reconstructed (*that's why the Mixer strategy is needed*), especially for high-frequency patterns, which appear, for example, in zip and tile: resulting in rather blurry image in particular when L2 norm is used (Figure 5). Although this is a huge

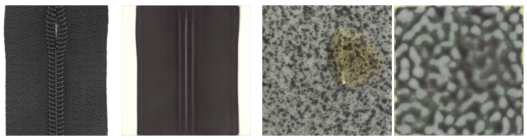

Figure 5: Zip and Tile classes *blurred* reconstruction.

problem for anomaly detection, the heterogeneity of the dataset makes the Mixer object classification task very easy, indeed we have been able to achieve **0.99 F1 score** in most of our trainings, making the thresholding mechanism possible. Increasing the number of parameters the risk is to make the model too good in understanding its job with a sparse latent space, while reducing them could lead the AE to reconstruct no more than the shape (see Figure 6). In the simple CAE described in Section 3.1, increasing the dimensionality of the latent space is directly proportional to a lower reconstruction error. While we noticed that, with Contractive and Denoising AEs, we have been able to improve generalization power preventing anomalies to be decoded perfectly as in Figure 7. This property is captured in the results of our models reported

model	anomaly detection		
	P	**R**	**F**
baseline	.674	.310	.425
DE - *AE* (L2)	.390	.758	.515
CO - *AE* (L2)	.744	.500	.598
CO+DE - *AE* (L2)	.690	.677	.684
CO+DE+MIX - *AE* (L2)	**.780**	.761	.771
CO+DE+MIX - *AE* (SSIM)	.726	**.956**	**.825**
CO+DE+MIX - *AE* (MSSIM)	.734	.858	.791
ENSEMBLE	.722	.812	.765

Table 3
Performance evaluation P (Precision), R (Recall) and F (F1 score) of the best experiment using each of the different solutions described in Section 3. The reported values represent the average of those for all classes in the dataset. The *baseline* is a simple Convolutional AE with L2 loss. DE stands for *denoising*, CO for *contractive* and MIX for *mixer* strategy. Inside the brackets we put the applied reconstruction loss. The base architecture used for all the models is the one in Table 2.

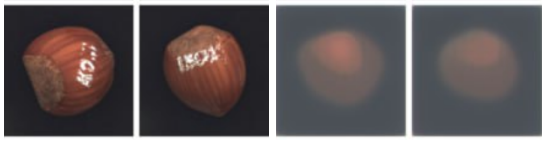

Figure 6: Anomalous hazelnuts reconstruction using a model with only 10M parameters.

in Table 3. Those are presented as Precision, Recall and F1 score but we found that in literature [19], before the problem was solved on this benchmark, was common to use for each dataset category the ratio of correctly classified samples of anomaly-free and anomalous images. Therefore in Table 4 we presented our best scores compared to the ones reported in 2019. This metric is not really fair since the validation dataset is very unbalanced for each different class in terms of normal and anomalous samples. However, we got only fewer points using a single model instead of two for the different sub-classes and therefore much fewer parameters! We also tried some augmentation techniques (flip, crop, rotation) with the hope of generalizing better image reconstruction but with poor results, thus we have shelved the idea.

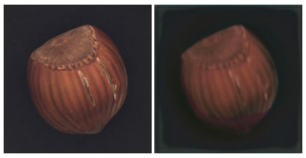

Figure 7: Anomalous hazelnut reconstruction using a model with 30M parameters and a high noise at training time (30%). The scratches have been almost removed.

category	model			
	AE (SSIM)	AE (L2)	GAN	**Ours** MIXER L2
Carpet	0.43	0.57	**0.82**	0.61
	0.90	0.42	0.16	0.58
Grid	0.38	0.57	**0.90**	0.66
	1.00	0.98	0.12	0.73
Leather	0.00	0.06	**0.91**	0.12
	0.92	0.82	0.12	**0.98**
Tile	**1.00**	**1.00**	0.97	0.33
	0.04	0.54	0.05	**0.95**
Wood	0.84	**1.00**	0.89	**1.00**
	0.82	0.47	0.47	0.77
Bottle	0.85	0.70	**0.95**	0.45
	0.90	0.89	0.43	**0.93**
Cable	0.74	0.93	**0.98**	0.36
	0.48	0.18	0.07	**0.77**
Capsule	0.78	**1.00**	0.96	0.09
	0.43	0.24	0.20	**0.95**
Hazelnut	**1.00**	0.93	0.83	0.43
	0.07	0.84	0.16	**0.98**
Metal nut	**1.00**	0.68	0.86	0.14
	0.08	0.77	0.13	**1.00**
Pill	0.92	**1.00**	**1.00**	0.96
	0.28	0.23	0.24	**0.43**
Screw	0.95	**0.98**	0.41	0.49
	0.06	**0.39**	0.28	**0.39**
Tootbrush	0.75	**1.00**	**1.00**	**1.00**
	0.73	**0.97**	0.13	0.83
Transistor	**1.00**	0.97	0.98	0.27
	0.03	0.45	0.35	**0.90**
Zipper	**1.00**	0.97	0.78	0.25
	0.60	0.63	0.40	**0.72**
overall performance	0.77	0.82	**0.88**	0.79
	0.49	**0.58**	0.22	0.48
	0.63	**0.70**	0.55	0.64

Table 4
Comparisons between three evaluated methods reported in [19] and our best trained model. For each category, the ratio of correctly classified samples of anomaly-free (top row) and anomalous images (bottom row) is given. At the end we have the average results of all classes and the final mean of these last two values.

5. Conclusions

Unluckily the ensemble didn't perform well as expected due to the very different methodologies applied that each time favorite a subset of all the categories. It's interesting to notice that for the heavily unbalanced metric proposed by the authors, no model is predominant over the other and we have been able to achieve results very close to SOTA AEs (obtained with multiple models) using our single Mixer Architecture (*CO+DE+MIX AE-L2*) that has the third highest F1 score under the same architecture trained with SSIM and MSSIM. This is because if we see each class independently with the MSE loss it's possible to threshold better between the same class, while structure base metrics improve the general reconstruction at the price of a more balanced Precision-Recall ratio.
We are very satisfied with our work and results: it's fascinating that we are approaching a significant real-world problem with techniques and reasonings acquired in class. Numerous proposed solutions have been documented in the literature for MVTec AD, however, it is still worth conducting an investigation on Autoencoders which are among the most basic and straightforward. We firmly believe that with higher hardware capabilities to sustain bigger models, we could even have achieved better results, for example utilizing larger batch sizes would have enhanced the statistical threshold computation, which is a crucial component as we have frequently observed.
However, one should ask, is it worth it? AEs have been proven to be very effective in specific domains, nonetheless, when the categories are vastly dissimilar, they encounter difficulty in generalizing sufficient features to effectively reconstruct them all. "Autoencoders are useful for some things, but turned out not to be nearly as necessary as we once thought."[1]. If the objective is to effectively solve a problem utilizing them, which would entail an increase in memory requirements, a potential solution to consider is the replacement of them with more recent deep learning methods that have been developed in recent years and possess well-established techniques.

References

[1] F. E. Grubbs, Procedures for detecting outlying observations in samples, Technometrics 11 (1969) 1–21. URL: https://doi.org/10.1080%2F00401706.1969.10490657. doi:10.1080/00401706.1969.10490657.

[2] reportlinker, Global anomaly detection market 2022-2026, Report Linker (2022). URL: https://www.reportlinker.com/p06359939/Global-Anomaly-Detection-Market.html?utm_source=GNW.

[3] P. Tanuska, L. Spendla, M. Kebisek, R. Duris, M. Stremy, Smart anomaly detection and prediction for assembly process maintenance in compliance with industry 4.0, Sensors 21 (2021). URL: https://www.mdpi.com/1424-8220/21/7/2376. doi:10.3390/s21072376.

[4] G. P. Tancredi, G. Vignali, E. Bottani, Integration of digital twin, machine-learning and industry 4.0 tools for anomaly detection: An application to a food plant, Sensors 22 (2022). URL: https://www.mdpi.com/1424-8220/22/11/4143. doi:10.3390/s22114143.

[1] Quote from Ian Goodfellow

[5] G. Pang, C. Shen, L. Cao, A. V. D. Hengel, Deep learning for anomaly detection, ACM Computing Surveys 54 (2022) 1–38. URL: https://doi.org/10.1145%2F3439950. doi:10.1145/3439950.

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

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

[8] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: Explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, Omnipress, Madison, WI, USA, 2011, p. 833–840.

[9] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 1096–1103. URL: https://doi.org/10.1145/1390156.1390294. doi:10.1145/1390156.1390294.

[10] D. P. Kingma, M. Welling, An introduction to variational autoencoders, Foundations and Trends® in Machine Learning 12 (2019) 307–392. URL: https://doi.org/10.1561%2F2200000056. doi:10.1561/2200000056.

[11] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, C. Steger, The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection, International Journal of Computer Vision 129 (2021) 1038–1059. URL: https://doi.org/10.1007/s11263-020-01400-4. doi:10.1007/s11263-020-01400-4.

[12] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, P. V. Gehler, Towards total recall in industrial anomaly detection, CoRR abs/2106.08265 (2021). URL: https://arxiv.org/abs/2106.08265. arXiv:2106.08265.

[13] M. Rudolph, T. Wehrbein, B. Rosenhahn, B. Wandt, Asymmetric student-teacher networks for industrial anomaly detection, 2022. URL: https://arxiv.org/abs/2210.07829. doi:10.48550/ARXIV.2210.07829.

[14] S. Yamada, S. Kamiya, K. Hotta, Reconstructed student-teacher and discriminative networks for anomaly detection, 2022. URL: https://arxiv.org/abs/2210.07548. doi:10.48550/ARXIV.2210.07548.

[15] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, U. Schmidt-Erfurth, f-anogan: Fast unsupervised anomaly detection with generative adversarial networks, Medical Image Analysis 54 (2019) 30–44. URL: https://www.sciencedirect.com/science/article/pii/S1361841518302640. doi:https://doi.org/10.1016/j.media.2019.01.010.

[16] S. Pidhorskyi, R. Almohsen, D. A. Adjeroh, G. Doretto, Generative probabilistic novelty detection with adversarial autoencoders, CoRR abs/1807.02588 (2018). URL: http://arxiv.org/abs/1807.02588. arXiv:1807.02588.

[17] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, MLSDA'14, Association for Computing Machinery, New York, NY, USA, 2014, p. 4–11. URL: https://doi.org/10.1145/2689746.2689747. doi:10.1145/2689746.2689747.

[18] C. Zhou, R. C. Paffenroth, Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 665–674. URL: https://doi.org/10.1145/3097983.3098052. doi:10.1145/3097983.3098052.

[19] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, C. Steger, Improving unsupervised defect segmentation by applying structural similarity to autoencoders, CoRR abs/1807.02011 (2018). URL: http://arxiv.org/abs/1807.02011. arXiv:1807.02011.

[20] A. Bionda, L. Frittoli, G. Boracchi, Deep autoencoders for anomaly detection in textured images using CW-SSIM, in: Image Analysis and Processing – ICIAP 2022, Springer International Publishing, 2022, pp. 669–680. URL: https://doi.org/10.1007%2F978-3-031-06430-2_56. doi:10.1007/978-3-031-06430-2_56.

[21] Z. Wang, E. Simoncelli, A. Bovik, Multiscale structural similarity for image quality assessment, in: The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers, 2003, volume 2, 2003, pp. 1398–1402 Vol.2. doi:10.1109/ACSSC.2003.1292216.

[22] J. Chen, S. Sathe, C. Aggarwal, D. Turaga, Outlier Detection with Autoencoder Ensembles, 2017, pp. 90–98. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611974973.11. doi:10.1137/1.9781611974973.11.

[23] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, CoRR abs/1505.04597 (2015). URL: http://arxiv.org/abs/1505.04597. arXiv:1505.04597.

[24] Y. Zhang, A better autoencoder for image: Convolutional autoencoder, 2018.

[25] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, C. Steger, Improving unsupervised defect segmentation by applying structural similarity to autoencoders, CoRR abs/1807.02011 (2018). URL: http://arxiv.org/abs/1807.02011. arXiv:1807.02011.

[26] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, C. Steger, Improving unsupervised defect segmentation by applying structural similarity to autoencoders, CoRR abs/1807.02011 (2018). URL: http://arxiv.org/abs/1807.02011. arXiv:1807.02011.

[27] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (2004) 600–612. doi:10.1109/TIP.2003.819861.

[28] A. Chiang, E. David, Y.-J. Lee, G. Leshem, Y.-R. Yeh, A study on anomaly detection ensembles, Journal of Applied Logic 21 (2017) 1–13. URL: https://www.sciencedirect.com/science/article/pii/S1570868316301240. doi:https://doi.org/10.1016/j.jal.2016.12.002.

[29] J. Zhu, T. Park, P. Isola, A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, CoRR abs/1703.10593 (2017). URL: http://arxiv.org/abs/1703.10593. arXiv:1703.10593.