About the Application of Autoencoders for Visual Defect Detection

Richárd Rádli
Faculty of Information Technology
University of Pannonia
Egyetem u. 10.
Veszprém, Hungary
radli.richard@virt.uni-pannon.hu

László Czúni
Faculty of Information Technology
University of Pannonia
Egyetem u. 10.
Veszprém, Hungary
czuni@almos.uni-pannon.hu

ABSTRACT

Visual defect detection is a key technology in modern industrial manufacturing systems. There are many possible appearances of product defects, including distortions in color, shape, contamination, missing or superfluous parts. For the detection of those, besides traditional image processing techniques, convolutional neural networks based methods have also appeared to avoid the usage of hand-crafted features and to build more efficient detection mechanisms. In our article we deal with autoencoder convolutional networks (AEs) which do not require examples of defects for training. Unfortunately, the manual and/or trial-and-error design of AEs is still required to achieve good performance, since there are many unknown parameters of AEs which can greatly influence the detection abilities. For our study we have chosen a well performing AE known as structural similarity AE (SSIM-AE), where the loss function and the comparison of the output with the input is implemented via the SSIM instead of the often used L1 or L2 norms. Investigating the performance of SSIM-AE on different data-sets, we found that its performance can be improved with modified convolutional structures without modifying the size of latent space. We also show that finding a model with low reconstruction error during training does not mean good detection abilities and denoising AEs can increase efficiency.

Keywords

autoencoder neural network, convolutional neural network, defect detection, unsupervised anomaly detection

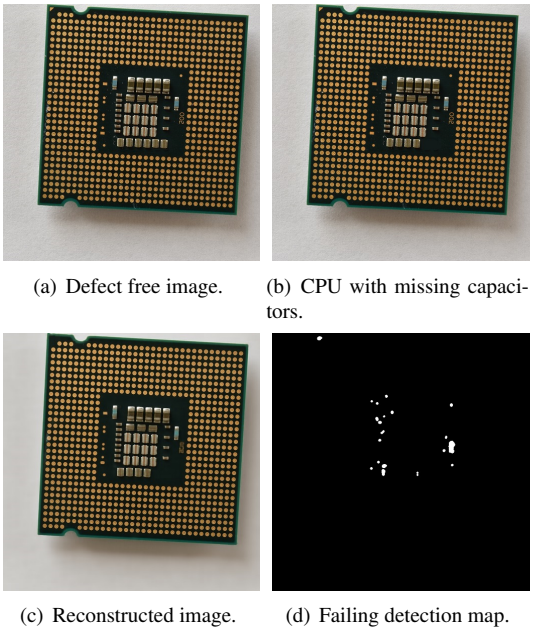
1 INTRODUCTION

There are several applications of neural networks in manufacturing systems [Wang2018], visual inspection of products is a critical step during their production. To have the greatest freedom in the handling and imaging of objects we need sophisticated methods to allow different distortions such as changes in lighting, position and orientation, especially in case of 3D objects. Due to this problem, in recent times, different deep learning solutions have been developed, autoencoder (AE) neural networks are very promising for anomaly detection in visual quality inspection, surveillance, or medical image analysis.

As we will discuss, there are several types of autoencoders, it is not clear which one suits best unsupervised defect detection and segmentation. Recent improvements in generative networks mainly focused on creating photo-realistic "natural" images as faces, nature, cityscapes, etc. In defect detection, while we have the advantage of the narrower image domain (we can train separate AEs for each class of products), reconstruction errors are much less tolerable, otherwise the succeeding detection phase may fail. Moreover, there is a large set of parameters which significantly influence detection abilities. Our purpose is to investigate the applicability of a well-performing autoencoder (namely SSIM-AE) for the fault detection on repetitive and partly repetitive structures. A great advantage of using AEs is that unsupervised training and detection is possible without making efforts to collect samples with anomalies. However, there is no guarantee that images with errors will not be reconstructed making detection unsuccessful since many times errors are reproduced in the decoded image. Fig. 1 shows a good example for failing anomaly detection with the otherwise well performing SSIM-AE [Bergmann2019]. Input image has a missing capacitor and the AE reproduced the image with the same defect.

First, we overview the different approaches for visual anomaly detection mainly focusing on AE networks, then, in Section 3, the SSIM-AE and its variants are introduced. Section 4 details the selected data-sets,

while in Section 5, we explain our experiments and discuss the performance of SSIM-AEs with different settings for the pixel-wise segmentation of different errors. Finally, we summarize our paper in the last section.



(a) Defect free image. (b) CPU with missing capacitors.

(c) Reconstructed image. (d) Failing detection map.

Figure 1: Illustration of failed detection of missing components from a CPU with SSIM-AE at resolution 512×512.

2 ALTERNATIVES FOR VISUAL INSPECTION

Optical defect detection approaches can be grouped in the following main categories: Template matching [Buniatyan2017], [Zhou2019] local feature based methods [Gai2016],[Xi2017], statistical texture models [Cogranne2014],[Cao2015], neural networks [Weimer2016], [Tuluptceva2019], [Tuluptceva2020], [Alaverdyan2020], [Nagy2021]. There can be overlap among these groups, and each category have some advantages, but there is no doubt that due to the generality, robustness and accuracy of deep learning convolutional AE approaches, they are getting more focus in recent years. In our article we discuss only AE networks having the main advantage of unsupervised training and the ability to handle various imaging conditions.

2.1 About Autoencoders

An autoencoder is a special kind of feed-forward neural network containing three main sequential components: the encoder network $E : \mathbb{R}^{k\times h\times w} \to \mathbb{R}^d$, the latent space $\mathbb{R}^d$, and the decoder network $D : \mathbb{R}^d \to \mathbb{R}^{k\times h\times w}$.

$$\hat{\mathbf{x}} = D(E(\mathbf{x})) = D(\mathbf{z}), \tag{1}$$

where $\mathbf{x}$ is the input image, $\mathbf{z}$ is the latent information, and $\hat{\mathbf{x}}$ is the output of the network.
AEs were originally used for dimensionality reduction, feature extraction, but nowadays they are very popular tools for generative modeling. The similarity of principal component analysis (PCA) and AEs is well know, moreover, in many scenarios AEs outperformed linear or kernel PCAs [Sakurada2014].
The low-level neural structure in AEs can be convolutional or can utilize extreme learning machines [Kasun2013].
They can also be categorized according to their regularization techniques. The loss function (J), to learn a specific task, can be generally formed this way:

$$\mathbf{J}(\mathbf{x},\theta) = \frac{1}{n}\sum \|\mathbf{x}-\hat{\mathbf{x}}\| + R(\mathbf{z}), \tag{2}$$

where $\theta = (w_E, b_E, w_D, b_D)$ are the weights and biases of the encoder and decoder networks and n is the number of elements. In sparse AEs R is based on the Kullback-Leibler divergence or on the L1 norm, the purpose is to make the most of the hidden unit's activations close to zero. Contractive AEs apply the Frobenius norm on the derivative of $\mathbf{z}$ as a function of $\mathbf{x}$ to make the model resistant to small perturbations; and in an information theoretic-learning autoencoder Renyi's entropy is used.
Variational AEs (VAEs) impose constraints on the latent variables in a different way, they estimate posteriori probability $p(\mathbf{z}|\mathbf{x})$ with an assumption of a prior knowledge $p(\mathbf{z})$ being a normal Gaussian distribution. Adversarial AEs (AAEs) use generative adversarial networks (GANs) to perform variational inference. VAEs and AAEs are both generative models, and both are based on maximum likelihood. The difference between VAEs and AAEs can be characterized that while VAEs apply explicit rules on $\mathbf{z}$, AAEs control its distribution implicitly.
Denoising autoencoder (DAE) is an extension of the basic autoencoder, which is trained to reconstruct corrupted input data. In [Vincent2010] not only Gaussian noise, but also salt-and-pepper noise and zero-masking noise was also investigated. C denotes the corrupting function generating $\tilde{\mathbf{x}}$. Eq. 1 now becomes:

$$\hat{\mathbf{x}} = D(E(C(\mathbf{x}))) = D(E(\tilde{\mathbf{x}})) = D(\mathbf{z}). \tag{3}$$

The applied loss puts an emphasis on the corrupted areas by proper weighting:

$$\mathbf{J}(\mathbf{x},\tilde{\mathbf{x}},\theta) = \alpha \sum_{i\in\mathscr{C}} (\mathbf{x_i}-\hat{\mathbf{x}}_\mathbf{i})^2 + \beta \sum_{i\notin\mathscr{C}} (\mathbf{x_i}-\hat{\mathbf{x}}_\mathbf{i})^2, \tag{4}$$

where $\mathscr{C}$ denotes the indexes of corrupted components. Feature matching autoencoder [Dosovitskiy] enforces the features of the input and its reconstruction to be equal:

$$\mathbf{J}(\mathbf{x},\tilde{\mathbf{x}},\theta) = w_{im}\sum(\mathbf{x}-\hat{\mathbf{x}})^2 + w_{ft}\sum(F(\mathbf{x})-F(\hat{\mathbf{x}}))^2, \tag{5}$$

where F is a feature extractor which can be fixed or also trained. In [Dosovitskiy], besides using features for evaluation, adversarial discriminator was also utilized.
We guide people, with more interest in the overview of AEs, to papers [Zhai2018] and [Yuan2019].

2.2 Autoencoders for Defect Detection

In general, trained AEs can generate very similar outputs to their inputs, if the later fits the trained model or with other words, the image models could learn the inherent features properly. That is if the difference between the two is above a threshold Th, we set the detection map to 1:

$$\mathbf{m} = \begin{cases} 1, & \text{if } ||\mathbf{x}-\hat{\mathbf{x}}|| > Th, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The applied $||.||$ can be various (e.g. L1, L2, SSIM [Bergmann2019]).
In [Beggel2019] they deal with the problem of training AEs when the training set is contaminated with a small fraction of outliers. Their proposed adversarial autoencoder imposes a prior distribution on the latent representation, placing anomalies into low likelihood regions, thus potential anomalies can be identified and rejected already during training.
[Tuluptceva2020] proposes perceptual deep autoencoders where relative-perceptual-L1 loss [Tuluptceva2019], robust to low contrast and noise, is applied for training and also to predict the abnormality for new inputs. The applied progressive growing technique helped the efficiency of training: in the beginning of the training, the loss function computes the dissimilarity between the low-resolution images using the features from the coarse layers of the network, whereas, as the training advances, the "level" of this information is increased by including deeper and deeper features. The addition of the new layers to the autoencoder with the gradual increase of the depth of the features entailed in the calculation of the perceptual loss was synchronized.
Siamese networks are good for supervised anomaly detection, since they can learn discriminating features efficiently [Nagy2021]. In [Alaverdyan2020] an unsupervised siamese convolutional autoencoder is proposed to detect anomalies in brain MRI images. Anomalies are detected by a one class SVM in latent space. The role of the siamese network is to regularize the latent space: $R(z)$ (Eq. 2) is the cosine distance of two independent samples in the two siamese branches in the latent space. Since each voxel of the MRI data is handled independently (with its own SVM), thus anomalies are learnt for every location.
The approach we utilize in our study is from [Bergmann2019], which applies structural similarity (SSIM) in the loss function. It was reported that it outperforms many other AEs and reaches the state-of-art on some data-set. It is discussed in details in the next section.

3 SSIM AUTOENCODERS FOR DEFECT DETECTION

In our understanding SSIM-AE [Bergmann2019] is a kind of feature matching AE (Eq. 5) with the following loss function:

$$\mathbf{J}(\mathbf{x},\theta) = \frac{1-SSIM(\mathbf{x},\hat{\mathbf{x}})}{2} = \frac{1-l(\mathbf{x},\hat{\mathbf{x}})^{\alpha}c(\mathbf{x},\hat{\mathbf{x}})^{\beta}s(\mathbf{x},\hat{\mathbf{x}})^{\gamma}}{2}. \tag{7}$$

l stands for the luminance measure, estimated by comparing the patches' mean intensities, c for the contrast responsible for the local variance, and s evaluates the covariance of patches. We set all three exponents to 1, and other inner variables (not detailed here) as follows: $c_1 = 0.01$, $c_2 = 0.03$, and the window size of the patches $K = 11$. SSIM is not only utilized in the loss but also in the comparison of the input and outputs:

$$\mathbf{m} = \begin{cases} 1, & \text{if } \frac{1-SSIM(\mathbf{x},\hat{\mathbf{x}})}{2} > Th, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Since SSIM already includes the luminance of pixel values, $w_{im} = 0$ in Eq. 5, colors are simply neglected. The SSIM index was originally designed to create an accurate perceptual quality metrics for the comparison of two $K \times K$ image patches [Wang2004]. In our case not the perceptual information makes it appealing to use it in the loss function but its efficient extraction of structural information: SSIM-AE is forced to represent, beside luminance, local variance and covariance. According to [Bergmann2019] SSIM-AE could significantly outperform the FM-AE of [Dosovitskiy] and a tested VAE. The code available at the Internet[1] is not fully identical to the description in their paper. In our experiments we followed the paper except for setting the stride of overlapping windows when fusing the cropped images for reconstruction (32 instead of 30).
The structure of the encoder is given in Table 1, the decoder has the opposite structure, as generally.

[1] https://github.com/plutoyuxie/AutoEncoder-SSIM-for-unsupervised-anomaly-detection-

Layer	Output Size	Kernel	Stride	Padding
Input	128x128x3			
Conv1	64x64x32	4x4	2	1
Conv2	32x32x32	4x4	2	1
Conv3	32x32x32	3x3	1	1
Conv4	16x16x64	4x4	2	1
Conv5	16x16x64	3x3	1	1
Conv6	8x8x128	4x4	2	1
Conv7	8x8x64	3x3	1	1
Conv8	8x8x32	3x3	1	1
Conv9	1x1xd	8x8	1	0

Table 1: Architecture of the encoder of SSIM-AE [Bergmann2019] for input image $128 \times 128 \times 3$. $d = 100$ for the texture images, $d = 500$ for the CPUs.

Layer	Output Size	Kernel	Stride	Padding
Input	128x128x3	3x3		
Conv1	64x64x32	3x3	2	1
Conv2	64x64x32	3x3	1	1
Conv3	64x64x32	3x3	1	1
Conv4	32x32x64	3x3	2	1
Conv5	32x32x64	3x3	1	1
Conv6	32x32x64	3x3	1	1
Conv7	16x16x128	3x3	2	1
Conv8	16x16x128	3x3	1	1
Conv9	16x16x128	3x3	1	1
Conv10	8x8x256	3x3	2	1
Conv11	8x8x256	3x3	1	1
Conv12	8x8x256	3x3	1	1
Conv13	1x1xd	8x8	1	0

Table 2: Architecture of the encoder of SSIM-AEe for input image $128 \times 128 \times 3$. $d = 100, 500$ for the texture images, $d = 500$ for the CPUs.

In our experiments we investigated several questions: Can the results be improved by increasing the number of layers but keeping the size of the latent space? How does SSIM-AE perform in case of less periodic structures? What happens if some components are added or removed from the original images?
We made two modifications to the original SSIM-AE:

1. In one modification (named SSIM-AEe) we increased the number of convolutional layers as depicted in Table 2. We kept the latent layers untouched.

2. In SSIM-DAE we followed Eq. 3 to build a denoising AE trained with masked blocks (see Section 5 for details). The purpose of our SSIM-DAE was to implement an implicit regularization of the network to reconstruct patches from a larger vicinity. There is low probability that during decoding both the target and source areas are all affected by defects. But either one is distorted $||\hat{x} - \tilde{x}||$ will have high value.

4 DATA-SETS

The visually observable defects can be various such as faulty color, contamination, geometrical distortion, missing parts, or superfluous parts. We assume that our objects are almost 2D (like the surface of a rectangular object without significant bulges or holes), and images have variations in translation, orientation, and lighting conditions.
In our study, we used five data-sets. Two of them, provided by [Bergmann2019], contain regular patterns of woven fabric textures. Each of these two sets include 100 defect-free images, and 50 test images with various defects. Ground truth images were also provided in binary maps. We have chosen these textures, since they are good representatives of roughly regular repetitive patterns. Since we focus on convolutional AEs, the radius of convolutions and the number of layers can have a great impact on the generation abilities depending on the texture size.
The other data-sets involve 60 genuine images of the backside of a CPU of size 37.50 mm $\times$ 37.50 mm. We separated 48 images for training purposes and the remaining 12 for various tests. Three different test cases were generated for the CPUs:

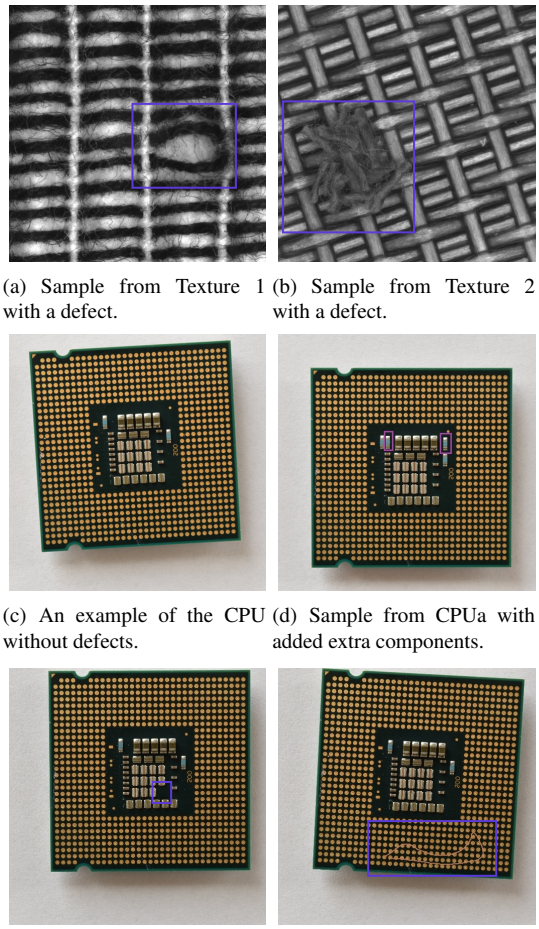
- CPUa: we added extra components with visually realistic appearance;
- CPUc: the test images are loaded with synthetic contamination;
- CPUm: some components (such as pins or capacitors) are removed.

Table 3 summarizes the mean and standard deviation of the defected areas for all 5 data-sets. In case of textures, anomalies are significantly larger.

	Texture 1	Texture 2	CPUa	CPUc	CPUm
mean	8.00	5.52	0.16	0.69	0.31
standard deviation	4.56	2.20	0.07	0.26	0.28

Table 3: Mean and standard deviation of defected areas (in percentage) in the different test sets.

The selection of the CPU images is twofold: first, such and similar images are typical in the production of electronic components; second, the image of the CPU contains regularly repetitive (outward areas with contacts) and regular but much less repetitive regions (central capacitor area). Besides, the CPU images contain more finer details than the textures.
All images are of size 512×512, their illustration with defects is in Fig. 2. Ground truth images were also provided as binary maps.



(a) Sample from Texture 1 with a defect.
(b) Sample from Texture 2 with a defect.
(c) An example of the CPU without defects.
(d) Sample from CPUa with added extra components.
(e) Sample from CPUm with a missing capacitor.
(f) Sample from CPUc with synthetic contamination.

Figure 2: Illustration of test images with various defects.

5 TRAINING, TESTING, AND EVALUATION

During training we followed the method and hyper-parameters described in [Bergmann2019]. They first resized images to 256 × 256, then image cropping was applied to the size of 128 × 128. Networks were trained on NVIDIA RTX 2070 GPU for 200 epochs with ADAM optimizer, initial learning rate was 2×10^{-4} with 10^{-5} weight decay. The latent space was set to $d = 100$ and $d = 500$ for Texture 1 and Texture 2, and $d = 500$ for the CPU, considering its structural complexity and finer details. Furthermore, the window size of the SSIM was adjusted to 11.

In [Bergmann2019], for both texture data-sets, they increased the variability of training data by generating 10,000 images using various augmentation procedures such as rotation, cropping, and flipping (both vertically and horizontally). In our study, we did the same, except in the case of the CPU tests, where no flip was involved to keep the asymmetric structure.

To train the SSIM-DAE we added noise to the input images: small, homogeneous masks were added to the 48 images of the original training data-set. The size of masks were set to 20×20, being greater than the size of one capacitor in the middle of the CPU; the color was set to the dominant color of the actual (128×128) crop. Each training image contained one masks, $\alpha = \beta = 1$ in Eq. 4.

For evaluation, we have chosen the standard receiver operating characteristic (ROC) curves and the area under curve (AUC) values. We defined the true positive rate (TPR) as the rate of pixels that were correctly classified as defect, and false positive rate (FPR) as the ratio of pixels that were misclassified as defect. The process of testing was accomplished by the reconstructions of image patches of 128 × 128, by moving over the test images (with stride of 32). Residual maps were created using SSIM (Eq. 8), then ROC curves and AUC values were computed thresholding with increasing SSIM values.

To monitor the reconstruction abilities of the different networks, we also computed the SSIM and mean squared error (MSE) of reconstructions for defect-free testing images (last columns of Table 5).

5.1 Results and Discussion

Four kinds of AEs (SSIM-AE, SSIM-AEe, SSIM-DAE, and SSIM-DAEe) were compared in 5 test cases (Texture 1, Texture 2, CPUa, CPUc, CPUm) as described before. Results are summarized in Table 4 and Table 5 (best values are in bold), ROC curves are illustrated in Fig. 4 and Fig. 5.

For Texture 1 and Texture 2 SSIM-AE performed very well, neither increasing the number of convolutions (SSIM-AEe), nor increasing the size of the latent space (from $d = 100$ to $d = 500$) could make (significant) achievements. In general, Texture 1 and 2 are missing fine details (see Fig. 2 a and b) making deeper convolutions useless.

In case of the CPU tests we got different results. Comparing SSIM-AE to SSIM-AEe we see that increasing the number of convolutional layers could increase the accuracy of reconstruction but it could not help the ability to detect anomalies (except for CPUm). That is if a network is better in the reconstruction of defect-less items, it does not mean it is also better to mishandle defected inputs (thus to detect anomalies).

Since the CPU images had more details, SSIM-AEe could learn the image models more accurately than SSIM-AE (even with the same latent space): it is shown by MSE and SSIM values, measured between error-less inputs and their reconstructions, in Table 5. Since CPUa contained small anomalies, detection went quite well (AUC is between 0.9822 and 0.9952), while CPUc had the worst results. Finally, we found that the larger number of layers and the denoising type training of SSIM-DAEe resulted in the best performance for all

CPU test cases (bold in table).

	Texture 1	Texture 2
SSIM-AE	**0.9490**	0.9710
SSIM-AEe	0.9445	0.9697
SSIM-AE, $d = 500$	0.9186	**0.9773**
SSIM-AEe, $d = 500$	0.9118	0.9709

Table 4: AUC values when testing AE methods on data-set Texture 1 and Texture 2.

	CPUa	CPUc	CPUm	SSIM	MSE
SSIM-AE	0.9930	0.9071	0.9262	0.7263	90.0472
SSIM-AEe	0.9822	0.8939	0.9557	0.8195	66.5257
SSIM-DAE	0.9914	0.9059	0.9651	0.7851	88.7726
SSIM-DAEe	**0.9952**	**0.9499**	**0.9815**	**0.9133**	**57.3972**

Table 5: Testing AE methods for different distortions on the CPU data-set. AUC values are given in the first three columns, SSIM and MSE is measured as the reconstruction error of error free test samples. SSIM is on the [-1,1] scale, MSE stands for Mean Squared Error.

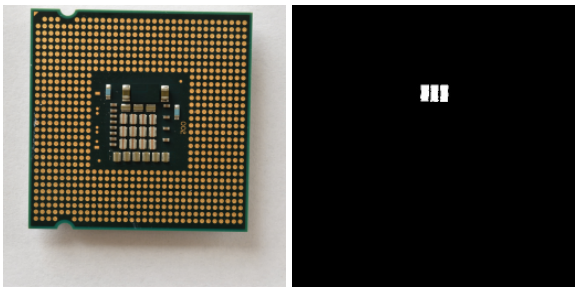
6 CONCLUSION

In case of visual quality inspection we are often not satisfied with the statistical definition of normal patterns and anomalies. For many products the strict topological structures should be retained during production, thus, the otherwise normal, missing or superfluous parts are not acceptable.
We investigated the performance of different SSIM autoencoders for defect detection on repetitive and semi-repetitive structures on 5 data-sets. By introducing modifications to the proposed AE of [Bergmann2019] we found, that:

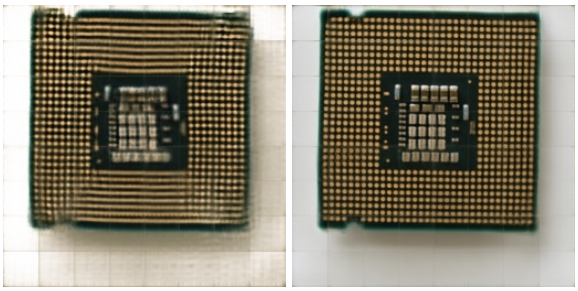
1. If an AE can reproduce (anomaly free) images with higher fidelity it does not strictly induce it is better in anomaly detection. It is an important observation, since AE's main application area is unsupervised anomaly detection, where no examples of defects are available to design and train optimal neural networks.

2. Unsupervised AEs are not reliable for (at least small) contamination or superfluous parts, DAEs may fit many defect detection tasks better.

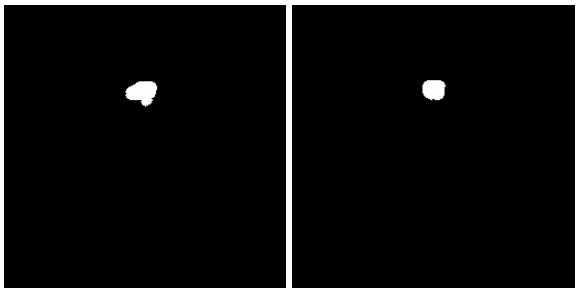
The proposed SSIM-DAEe outperformed other variants in our CPU experiments without the growth of the latent space, while in case of textures, the DAEe could not increase detection accuracy. (We also outperformed those feature matching and VAE approaches which are used as reference methods in [Bergmann2019]). Some illustrations of the results on CPU images are given in Fig. 3 for the SSIM-AE and SSIM-DAEe.
We believe current AEs are very far from their limit for defect detection, most recent regularization techniques consider the probability density functions with less emphasis on strict structural features.
No systematic design is known to optimally set the encoder, decoder, and latent parts, or the type of regularization. In future we plan to make more tests on the industrial data-sets of [Bergmann2021], and to build more structure awareness in the neural models investigating structural and contextual attention similar to [Yu2018].



(a) Input test image with missing capacitor and ground truth.



(b) Reconstruction with SSIM-AE and SSIM-DAEe.

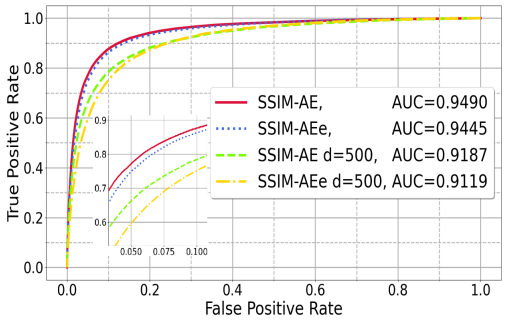


(c) Corresponding detection maps with SSIM threshold set to 0.8425

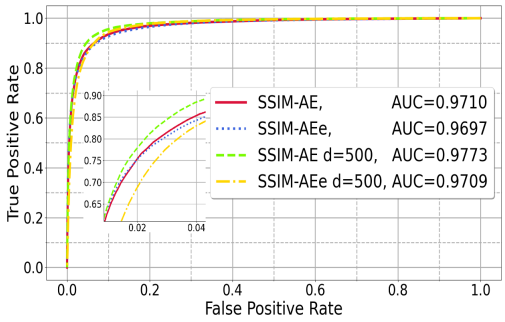
Figure 3: Illustration of the detection performance of SSIM-AE and SSIM-DAE for a CPU with missing capacitor.

7 ACKNOWLEDGMENTS

We are grateful to the NVIDIA corporation for supporting our research with GPUs obtained by the NVIDIA GPU Grant Program. We acknowledge the financial support of the projects 2018-1.3.1-VKE-2018-00048 under the ÚNKP-19-3 New National Excellence Program, 2020-4.1.1-TKP2020 under the Thematic Excellence Program, and the Hungarian Research Fund grant OTKA K 135729.

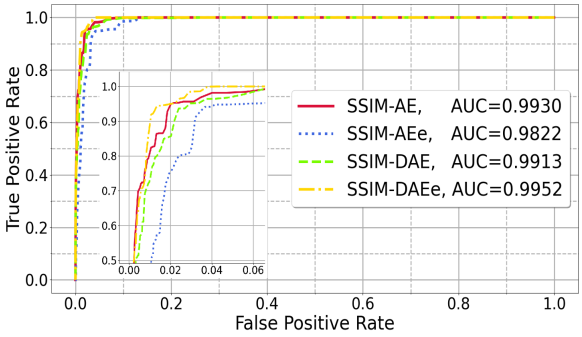


(a) AEs on Texture 1 at resolution 256 × 256.

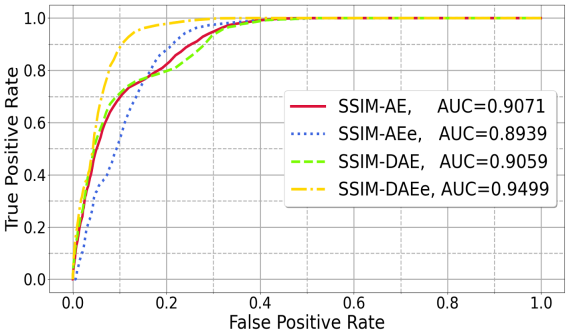


(b) AEs on Texture 2 at resolution 256 × 256.

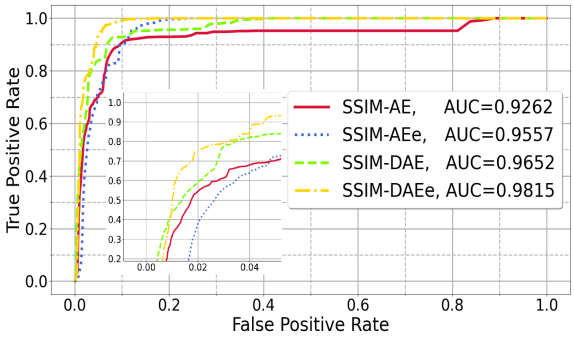
Figure 4: The performance of the AEs on Texture 1 and Texture 2.



(a) The performance of different AEs on CPUa.



(b) The performance of different AEs on CPUc.



(c) The performance of different AEs on CPUm.

Figure 5: The performance of different AEs on CPUa, CPUc, and CPUm tests.

8 REFERENCES

[Alaverdyan2020] Alaverdyan, Z., Jung, J., Bouet, R., Lartizien, C. (2020). Regularized siamese neural network for unsupervised outlier detection on brain multiparametric magnetic resonance imaging: application to epilepsy lesion screening. Medical image analysis, 60, 101618.

[Beggel2019] Beggel, L., Pfeiffer, M., Bischl, B. (2019, September). Robust anomaly detection in images using adversarial autoencoders. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 206-222). Springer, Cham.

[Bergmann2019] Bergmann, P.; Löwe, S.; Fauser, M.; Sattlegger, D. and Steger, C. (2019). Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders. VISAPP, pages 372-380.

[Bergmann2021] Bergmann, P., Batzner, K., Fauser, M., Sattlegger, D., Steger, C. (2021). The MVTec Anomaly Detection data set: A Comprehensive Real-World data set for Unsupervised Anomaly Detection. International Journal of Computer Vision, 1-22.

[Buniatyan2017] D. Buniatyan, T. Macrina, D. Ih, J. Zung, and H. S. Seung, Deep learning improves template matching by normalized cross correlation. arXiv preprint arXiv:1705.08593, 2017.

[Cao2015] Cao, J., Zhang, J., Wen, Z., Wang, N., and Liu, X. (2015). Fabric defect inspection using prior knowledge guided least squares regression. Multimedia Tools and Applications, 76:4141-4157.

[Cogranne2014] R. Cogranne and F. Retraint, Statistical detection of defects in radiographic images using an adaptive parametric model. Signal Processing, vol. 96, pp. 173-189, 2014.

[Dosovitskiy] Dosovitskiy, A., Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. arXiv preprint

arXiv:1602.02644.

[Gai2016] Gai, S. (2016). New banknote defect detection algorithm using quaternion wavelet transform. Neurocomputing, 196, 133-139.

[Kasun2013] Kasun L.L.C.,Zhou H.,Huang G.-B.,et al. Representational learning with ELMs for big data[J]. IEEE Intelligent Systems,2013,28(6):31-34.

[Nagy2021] A. Nagy and L. Czúni, Detecting Object Defects with Fusioning Convolutional Siamese Neural Networks. In Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP, pages 157-163

[Sakurada2014] Sakurada, M., Yairi, T. (2014, December). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis (pp. 4-11).

[Santana2016] Santana, E., Emigh, M., Principe, J. C. (2016, July). Information theoretic-learning auto-encoder. In 2016 International Joint Conference on Neural Networks (IJCNN) (pp. 3296-3301). IEEE.

[Tuluptceva2019] Tuluptceva, N., Bakker, B., Fedulova, I., Konushin, A. (2019, November). Perceptual image anomaly detection. In Asian Conference on Pattern Recognition (pp. 164-178). Springer, Cham.

[Tuluptceva2020] Tuluptceva, N., Bakker, B., Fedulova, I., Schulz, H., Dylov, D. V. (2020). Anomaly detection with deep perceptual autoencoders. arXiv preprint arXiv:2006.13265.

[Xi2017] Xi, J., Shentu, L., Hu, J., Li, M. (2017). Automated surface inspection for steel products using computer vision approach. Applied optics, 56(2), 184-192.

[Yu2018] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T. S. (2018). Generative image inpainting with contextual attention. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5505-5514).

[Yuan2019] Yuan, F.-N Zhang, L. Shi, J.-T Xia, X. Li, G. (2019). Theories and Applications of Auto-Encoder Neural Networks: A Literature Survey. Jisuanji Xuebao/Chinese Journal of Computers. 42. 203-230. 10.11897/SP.J.1016.2019.00203.

[Vincent2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P. A., Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(12).

[Wang2004] Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612.

[Wang2018] Wang, J., Ma, Y., Zhang, L., Gao, R. X., Wu, D. (2018). Deep learning for smart manufacturing: Methods and applications. Journal of Manufacturing Systems, 48, 144-156.

[Weimer2016] Weimer, D., Scholz-Reiter, B., Shpitalni, M. (2016). Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. CIRP Annals, 65(1), 417-420.

[Zhai2018] Zhai, J., Zhang, S., Chen, J., He, Q. (2018, October). Autoencoder and its various variants. In 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 415-419). IEEE.

[Zhou2019] X. Zhou, Y. Wang, C. Xiao, Q. Zhu, X. Lu, H. Zhang, J. Ge, and H. Zhao, Automated visual inspection of glass bottle bottom with saliency detection and template matching. IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 11, pp. 4253-4267, 2019.

Last page should be fully used by text, figures etc. Do not leave empty space, please.

Do not lock the PDF – additional text and info will be inserted, i.e. ISSN/ISBN etc.