A neural network for image anomaly detection with deep pyramidal reporesentations and dynamic routing

Pankaj Mishra
Dipartimento di Scienze Matematiche, Informatiche e Fisiche, Università degli Studi di Udine
Via delle Scienze 206, 33100 Udine, Italy
E-mail: mishra.pankaj@spes.uniud.it

Claudio Piciarelli
Dipartimento di Scienze Matematiche, Informatiche e Fisiche, Università degli Studi di Udine
Via delle Scienze 206, 33100 Udine, Italy
E-mail: claudio.piciarelli@uniud.it

Gian Luca Foresti
Dipartimento di Scienze Matematiche, Informatiche e Fisiche, Università degli Studi di Udine
Via delle Scienze 206, 33100 Udine, Italy
E-mail: gianluca.foresti@uniud.it

Image anomaly detection is an application-driven problem where the aim is to identify novel samples, which differ significantly from the normal ones. We here propose a deep reconstruction-based pyramidal approach, in which image features are extracted at different scale levels to better catch the peculiarities that could help to discriminate between normal and anomalous data. The features are dynamically routed to a reconstruction layer and anomalies can be identified by comparing the input image with its reconstruction. Unlike similar approaches, the comparison is done by using structural similarity and perceptual loss rather than trivial pixel-by-pixel comparison. The proposed method performed at par or better than the state-of-the-art methods when tested on publicly available datasets such as CIFAR10, COIL100 and MVTec.

1. Introduction

With the rise in modern IT infrastructure and new ways of high-speed data collection devices, an emerging requirement is to identity anomalies (also called *novelties* or *outliers*) in the collected data.

Anomaly detection is defined as the identification of samples which vary significantly from a reference set of normal data. It is a widespread problem, marking strong presence in the field of medical imaging, financial transactions, defect detection, video surveillance etc. A system which can perform such a task accurately and autonomously is in high demand.

Depending on the availability of data and labels, anomaly detection can roughly be classified in three main setups: fully supervised, semi-supervised and unsupervised.[1] *Fully Supervised* setup is used when labels are available for both normal and anomalous instances. In such cases, the problem reduces to a standard classification task, but with highly imbalanced datasets.[2,3] In *Semi Supervised* setup only data belonging to the normal class is available and the goal is to identify novel instances as normal or anomalous — this is why this approach is often called "novelty detection". Finally, the *Unsupervised* setup (also called "outlier detection") is similar to a clustering problem: in this case no labels are available, and the dataset must be split in the two classes of normal and anomalous data.

In this work we focus on the topic of image anomaly detection, which has many practical applications, e.g. in visual industrial inspection for quality assessment. Traditionally, image anomaly detection has been tackled with sparse-coding algorithms which try to learn the manifold of the normal data, either at global level or at feature level. Few GAN based approaches have also been tested,[4–6] where the authors try to reconstruct the normal image with a randomly generated data and expect higher reconstruction losses for the anomalous data. Most of the deep learning approaches are based on an autoencoder-like networks, where the normal data is first compressed to a low-dimensional latent space capturing the essential properties of the image (the so-called *causal factors*[7]) and then reconstructed back to its original aspect. In this case the network is expected to learn the appearance of normal images, and thus should fail in reconstructing the anomalous ones.

These approaches are typically best suited for global anomaly detection, in which the anomalous image is entirely different from normal ones (e.g. identify an orange when the normal data consists of apple images), but they are less efficient in finding local anomalies (e.g. an apple with a small rotten stain among good ones). Hence, to tackle with this problem, we propose a soft-self-attention based pyramidal approach for feature extraction and a *dynamic routing*[3] of these features to reconstruct the images. This will guarantees to capture better equivariant causal features of the images. Moreover, most of the reconstruction-based methods use a pixel-wise loss in order to compare reconstructions and input data. This assumes an independence among the pixels, which is not ideally the case. Hence, we adopted a more sophisticated loss function which considers the inter-relationship between the pixels and a perception-based loss computed by an another pre-trained network.

Finally, we tested our proposed network and method on one of the first real world datasets for image anomaly detection published by MVTec,[8] as well as CIFAR10 and COIL100 datasets. We found that the proposed model performed at par or better when compared with various state-of-the-art methods.

2. Related Work

Anomaly detection works have been proposed in several application fields, such as detection of anomalous network activity in intrusion detection systems,[9] medical image analysis for tumor detection,[10] structural integrity check in hazardous or inaccessible environments,[11] traffic analysis,[12] fault-prevention in industrial sensing systems.[13] Image-based anomaly detection is not a new topic in the industrial domain and many classical machine learning methods have been used to perform this task, such as Bayesian networks, rule-based system, clustering algorithms etc.[14] With the attempt to exploit the new methods in computer vision, especially deep learning, anomaly detection is posing a challenging problem and the research community is trying to solve this high real-time application problem using modern state-of-the art deep learning approaches.

As discussed above, anomaly detection either consists in outlier or novelty detection. When the training data contains the outliers (anomalies), outlier detection is about estimating the region where training data is mostly concentrated, consequently segregating the deviant observations.[15] Most of the time such methods employ clustering techniques, where extracted or engineered features are clustered and the high density regions are considered as normal while low densities reflect valuable information about some malicious action, system failure, deviation from normal behavior etc.

On the other hand, novelty detection is the task of identifying a completely novel or unseen data which is different from the previously observed data.[16] Since a novel sample is not necessarily an anomaly or a deviant sample, usually novelty detection methods use a novelty score and a threshold to declare a novel sample as anomaly.

In the case of deep learning approaches, some works deal with the supervised case, in which labeled anomalies do exist, but the dataset is heavily imbalanced. This is a very common scenario in actual industrial applications. These methods use classical generic techniques such as under-sampling the dominant class or over-sampling the smaller class, either by synthetic creation of data or data duplication. In either case the idea is to apply a pre-processing step to obtain a balanced dataset before passing through the deep network.[17] While early works on unsuper-

vised approach tried to use techniques such as Deep Belief Networks[18] for medical images or Restricted Boltzmann Machines[19] for network traffic analysis, more recent research includes autoencoders and generative models.[7] These methods try to learn a single class and are trained to minimize the reconstruction error. Also, some of these methods try to learn the density of the latent space and maximize its log-likelihood.[4,20,21] The limits of these methods are either the limited reconstruction capacity of the models or complex training procedures (especially in the case of GANs). While these models can perform well with global anomalies detection, they sometimes fail at finding small anomalies which are present locally in the images (like the anomalies presented in the real-world MVTec dataset). In contrast the fully unsupervised methods depend on the intrinsic features (causal features) of the data and attempts were made to learn the manifold of the normal class using GAN networks. At the time of testing the generator is inverted, which provides the comparison between the latent spaces of normal and anomalous samples.[5,22]

3. Proposed Model

Capturing the *causal factors* from which image can be reconstructed is the fundamental approach for a deep anomaly detection task. We propose a reconstruction-based approach, which takes a low-dimensional feature representation of images extracted through a pyramidal pooling layer and then dynamically routed to two *squashed* (see eq. 3) vectors. These two vectors store the *instantiation parameters*[3] of the underlying image. The first vector is always used for the reconstruction of the images. Later reconstruction losses are calculated and back-propagated. Our method employs single class (normal data) semi-supervised training, which is a typical industrial scenario, where labeled anomaly images are costly to obtain.

Compared to other deep anomaly methods, our main contribution consists in the addition of a pyramidal level in the network structure, to extract the causal features at different resolution scales. This is the way to increase the probability to extract the fine features at a scale in which the anomaly is particularly evident. In addition, we also adopted a better way to compare the input and reconstructed images. Most methods adopt trivial pixel-by-pixel comparison, e.g. MSE loss, which implicitly assumes the unrealistic hypothesis of independence between the pixels. We instead propose a high-level *Perceptual Loss*, and *Structural Similarity Index (SSIM)* loss, which quantify the image degradation at an higher abstraction level.

3.1. *Network Architecture*

We propose a network to learn the manifold of the normal class. The model aims to extract fine causal features of normal images that can be dynamically routed to capture the instantiation parameters. The model is trained to minimize the training losses over normal data and as a consequence it will result in higher losses in presence of anomalies. The network architecture is shown in Figure 2 and its composing blocks are here detailed:

- **SE-Resnet18** - A pre-trained Resnet18 network, with Squeeze-and-Excitation soft self attention (SE)[23] is used for the deep feature extraction. The network was trained over the imageNet dataset. All 5 layers of the Resnet18 has been used preceded by SE block, see Figure 1. The primary idea is that the network is able to extract generic low-level causal features of images. A pre-trained network also gives the benefits of transfer learning as the real world datasets are mostly related to imageNet dataset. Our detailed SE-Resnet18 architecture can be seen in appendix.

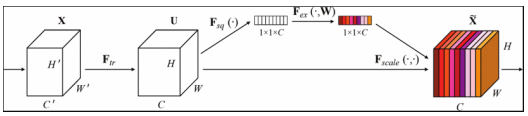

Fig. 1. Squeeze Excitation (SE) Block.[23]

The Squeeze Excitation block strengthens the representation power of CNN by encoding the spatial relationship throughout its feature hierarchy. It focuses on the channel relationship, that adaptively calculates channel-wise feature excitations by explicitly modelling the inter-dependencies between channels. For a transformation $F_{tr} : \mathbf{X} \rightarrow \mathbf{U}$, $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$, $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$, (a set of convolution operations, see Figure 1), the SE block performs a feature calibration: the *squeeze* operation is first applied on the features $\mathbf{U}$, which results in the aggregation of spatial dimensions $H \times W$ to produce a channel-only descriptor. This is followed

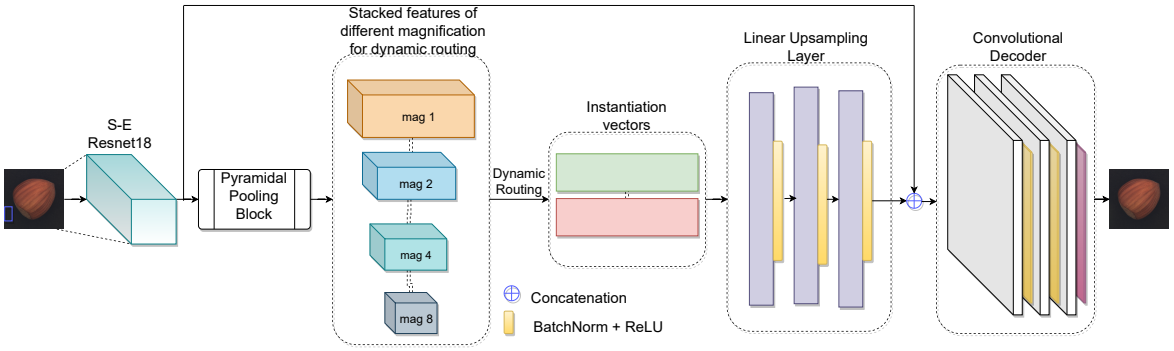

Fig. 2. Proposed network architecture. The network consists of the first levels of a SE-Resnet18 network for feature extraction, followed by a pyramidal pooling layer that extracts features at magnification 1, 2 4 and 8, which are then stacked in a matrix in $\mathbb{R}^{n\times 8}$. These features are then dynamically routed to two instantiation vectors in $\mathbb{R}^{64}$. The first vector is always passed to a linear upsampling layer and a final transposed convolutional decoder. The features obtained form the upsampling layer are concatenated with the output features of resnet18 before passing to final decoder.

by *excitation* operation learned for each channel using sigmoid activation of two linear layers sandwiching ReLU. Then the feature maps **U** are reweighted to generate the output of the SE block, which is subsequently passed to the other network layers. Mathematically, squeeze is defined as:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j) \quad (1)$$

Where u_c are the outputs of operations F_{tr}. The squeeze operation uses average pooling to create global embedding for the channels. The excitation operation instead is defined as:

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z)) \quad (2)$$

Here, the output of the squeeze block is converted into the sigmoid excitation by multiplying the learned weights W_1, then it passes through ReLU function δ, multiplies with the output of another set of learned weights W_2 and then passes through the sigmoid function at the end. W_1 reduces the dimensionality of the channels by a factor r (=2 in our case), which later brings back to original numbers using W_2. Sigmoid guarantees the positive values of channel weights. Finally, the channel features F_{tr} are multiplied by the weights obtained from the excitation operation. The entire operation can be seen as a self-attention mechanism on the channels using global information of the entire receptive field.

- **Pyramidal Pooling Layer** - This block takes input from the SE-Resnet18 block and applies and adaptive average pooling at different scales, respectively 1, 2, 4, 8, immediately followed by a convolution and batch normalization. The output features, corresponding to each magnification, are then stacked in a final vector $\in \mathbb{R}^{n\times 8}$.
- **Dynamic Routing** - It is a novel algorithm given by Hinton in his Capsule Networks paper.[3] The main intuition behind the algorithm is that *"It determines in statistical way, how the features from the previous layer will pass to the next layer"*. The routing algorithm can be seen in algorithm 1.
  Dynamic routing algorithm passes the statistically coherent features to the next level, where the features are stored in the two vectors. These vectors store the instantiation parameters (causal features) of images. Since the norm of these vectors represent the probability of the presence of an entity in the current input, a nonlinear *squashing function*, ensures that the vector lengths remain between 0 and 1 i.e. $\|v_j\| \in [0, 1]$. This non-linearity implicitly helps in the discriminative learning. Mathematically, squashing is defined as:

$$v_j = \frac{||s_j||^2}{a + ||s_j||^2} \frac{s_j}{||s_j||} \quad (3)$$

  where v_j is the vector output of capsule j and s_j is the total input.
- **Instantiation Vectors** - These are two vectors $\in \mathbb{R}^{64}$. The stacked features from the pyramidal pooling layer are stored in these vectors through dynamic routing. While training, the first vector is always passed further through the upsampling and decoder layers. While testing, the max length vec-

tor out of the two vectors is passed through the upsampling and decoder layers. The idea behind this approach is that dynamic routing will pass through the first vector only the features that are coherent for the reconstruction of normal images. This is in contrast with a traditional approach where all the features contribute to the output computation, which can be accomplished using a single instantiation vector (and thus no routing at all). In section 6 we propose an ablation study to prove that the two-vectors approach always performs better than the single-vector one.

Algorithm 1 Dynamic Routing Algorithm[3]

function ROUTING($\hat{u}_{j|i}, r, l$)
  For all features i in layer l and features j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all features i in layer l : $c_i \leftarrow softmax(b_i)$
    for all features j in layer $(l+1)$: $s_j \leftarrow \sum_i c_{ij}\hat{u}_{j|i}$
    for all features j in layer $(l+1)$: $v_j \leftarrow squash(s_j)$
    for all features i in layer l and features j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$
  end for
  return v_j
end function

- **Upsampling Layer** - An upsampling layer consists of three linear layers and is used to upsample the instantiation parameters from $\mathbb{R}^{64}$ to $\mathbb{R}^{512\times mf\times mf}$, where "mf" is the multiplying factor equals to the width of output features from SE-Resnet18.
- **Decoder** - Decoder is made of transposed convolutional layers, which take concatenated features from the SE-Resnet18 and upsampling layers with dimensions $\mathbb{R}^{batch\times n\times 8\times 8}$ and transforms them into the reconstructed image of same size as of the input image.

3.2. *Objective and Losses*

As stated before, the proposed model uses a reconstruction-based approach, in which the aim is to produce a network output similar to the input. In order to measure the similarity between the original image and its reconstruction, we considered three possible loss functions:

- *MSE Loss* - Mean Square Error (MSE) loss is a pixel-level loss, which assumes independence between pixels. MSE loss is computed between the input and the reconstructed images,i.e. $\|X-\hat{X}\|_2$, where X is the input and $\hat{X}$ is the output of the network (reconstructed image). MSE loss is often used in many reconstruction-based works, however the pixel-level independence assumption is unrealistic in real-world images.
- *Perceptual Loss* - Perceptual loss[24] is a more sophisticated loss trying to catch visually meaningful differences in images. It is an MSE loss computed between the high-level feature representations obtained by a pre-trained VGG11 network using its first four layers. The loss is defined as $\|F(X)-F(\hat{X})\|_2$, where F is the transformation function applied through the trained four layers of VGG11 network. The trained network is only used for the calculation of loss and the weights are not updated during training.
- *Structural Similarity Index* - The Structural Similarity Index (SSIM)[25] is used to measure the image similarity by considering visual properties that are lost in the standard MSE approach. The important feature in this loss calculation is that it takes care of perceptual phenomena, including both luminance and contrast, and it is defined as:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (4)$$

where, μ_x, μ_y, are the average values of input and reconstruction image, σ_x^2, σ_y^2 are the variance of input and reconstructed image, σ_{xy} is their covariance and c_1, c_2 are the two constants used for numerical stability.

The overall proposed objective function minimizes the total loss, defined as a weighted sum of the three image comparison losses:

$$TotalLoss = \|X-\hat{X}\|_2 + \lambda_1\|F(X)-F(\hat{X})\|_2 + \lambda_2 SSIM(X-\hat{X})$$

where X is the input image, $\hat{X}$ is the network output reconstructed images, and F is the non-linear function computed by the first four layers of a pre-trained

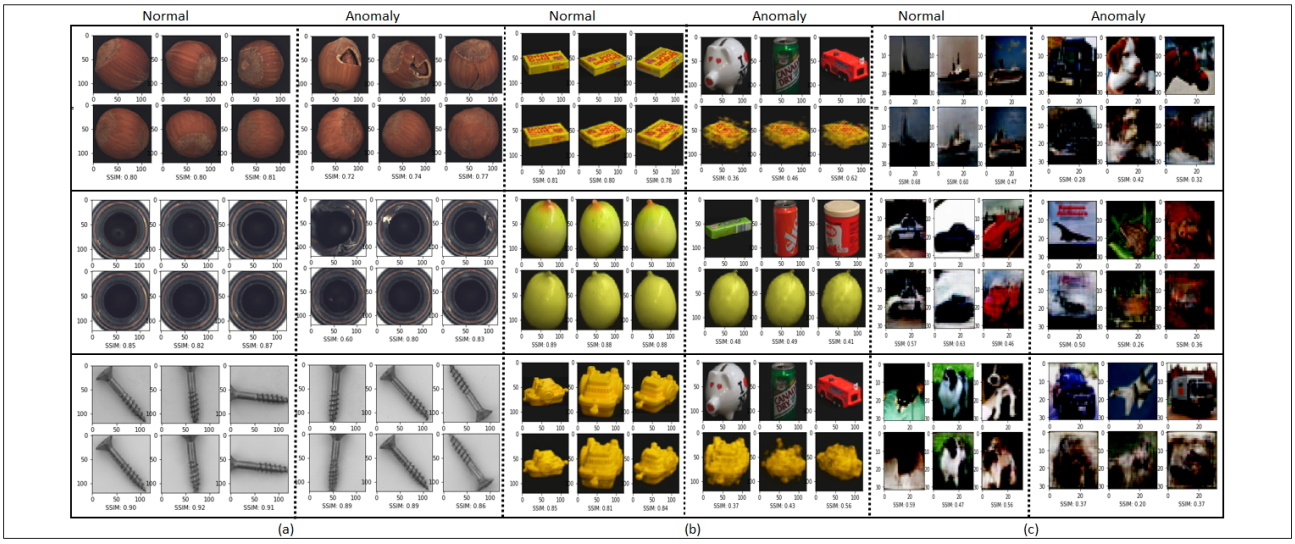

Fig. 3. (a) reconstruction of normal and anomaly classes for Hazelnut, bottle, and screw from MVTec dataset; (b) reconstruction of normal and anomaly classes for object 1, object2 and object3 from the COIL100 dataset; (c) reconstruction of normal and anomaly classes for the car, ship and dog from the CIFAR10 dataset.

VGG11 network. λ_1 and λ_2 are weighing factor between three losses, all the experiments discussed in section 4 are obtained with $\lambda_1 = \lambda_2 = 1$.

4. Results

The proposed model has been tested using publicly available datasets. Tests have been done on CIFAR10 and Coil100[26] datasets. Although these datasets are not specifically designed for the anomaly detection task, they are useful to show the ability of the system to discriminate between one class, considered normal, and the other ones, considered anomalies. In addition to this, the proposed model has also been tested on the real-world MVTec anomaly detection dataset,[8] which is a recently published dataset specifically for anomaly detection tasks (see figure 3).

- **CIFAR10**: It contains 60,000 images of ten classes with size 32×32 pixel. 50,000 images are for training while 10,000 images are for testing. For this study, we treated one of the class as normal and rest as anomaly. The results presented here are averaged over all the classes in several runs, in which each one of the original classes is chosen to be the normality model.
- **Coil-100**: The dataset has been taken from the Columbia Object Image Library. It contains 7,200 color images of 100 objects, having dimension 128×128. Each object is kept on an automated turntable, to capture the different pose of an object with a fixed color camera. The images were taken at a fixed pose interval of 5 degrees. For each object a total of 356 images were recorded. The results presented are averaged over all 100 objects (means 100 normality models, one for each object). While training one of the object is treated as the normal while all others as anomaly. The images were resized to 120×120, this is to maintain the same network structure used for the MVTec dataset. As the number of images in this dataset is limited, we used the training strategy of,[27] and split the training and testing data to 80% : 20%.
- **MVTec Dataset**: MVTec recently published a real world anomaly detection dataset. It contains 5354 high-resolution color images, of different texture and objects categories. It has normal and anomalous images which showcases 70 different types of anomalies of different real world products. It also contains 3 products images in gray-scale, as gray scale images are very common in industrial practices. To this dataset, all the images were first resized into 20×20 dimension, before passing to the model. This size has been chosen keeping in view that the structural integrity of the images such that anomalies are still visible by human eye.

The model is trained to learn the manifold of the normal class. The model weight is initiated with or-

thogonal initialization except the resent block, which was pretrained on imageNet, and the VGG11 block, which was pretrained on imageNet and kept fixed. The architectural hyper-parameters can be referenced in table 1.

Table 1. Training hyperparameters.

Adam learning Rate	0.001
weight decay	0.0001
batch size	120
Epochs	400

Table 2 shows the results obtained for CIFAR10 dataset. Training has been done considering one class as the normal and rest as the anomaly, and the same has been repeated over all the classes. The achieved results have been compared with standard methods such as one-class support vector machines and kernel density estimators, as well as with deep learning approaches such as denoising autoencoders, variational autoencoders,[28] Pix-CNN[29] and Latent Space Autoregression.[4] The comparative results have been taken from.[4] Performance is measured using the Area Under ROC Curve (AUC) metric. The proposed model superseded the results of the most of the state-of-the-art models in 7 out of 10 classes, and it has the best average result.

Table 3 shows the results on the MVTec dataset over all the 16 categories, comprising both textures (carpet, grid, leather, etc.) and objects (bottle, cable, capsule, etc). Our results are compared with other deep learning anomaly detection algorithms such as autoencoders with L2 norm loss and structural similarity loss,[25] the GAN-based approach AnoGAN,[21] and CNN feature dictionary.[30] The comparative results have been taken from.[8] Performance is compared again using the AUC metric. The proposed method achieves the best results on 8 out of 16 categories, and it reaches the best average result.

Table 4 shows the result of the coil100 dataset. The result is averaged over the 100 classes, where each class was considered as normal alternatively and rest as anomaly. Our proposed model achieved an AUC score of 0.998 surpassing GPND,[27] OCGAN[31] and DCAE,[32] which recorded 0.979, 0.995, and 0.908 respectively. Out of 100 classes more than 50 classes achieved AUC score of 1.

Table 4. COIL 100 results. The AUC score is averaged over all the 100 classes.

SoA Models	**Reference**	**AUC**
GPND	*NIPS 2018*[27]	0.979
OCGAN	*CVPR2019*[31]	0.995
DCAE	*MLSDA14*[32]	0.908
Ours		**0.998**

5. Conclusions

The model proposed in this work uses a deep pyramidal feature extraction and dynamically routes features through two instantiation vectors for the anomaly detection task. Anomalies are identified by the means of the network, which learn the manifold of the normal images in its instantiation vectors and then reconstruct them, ideally modeling an identity function. As the network is trained on normal data only, it is assumed that it will fail in reconstructing anomalous images, which can be detected by higher losses. The main contributions of this work consist in the usage of a multi-scale pyramidal approach that extract latent features at different resolutions, and the usage of a high-level loss function, to better compare images at feature level, rather than at pixel level. Moreover, differing from many works that have been evaluated on basic datasets only such as MNIST or FMNIST, the proposed model was tested on MVTec, a real-world dataset of defective products. Achieved results are promising and often outperform other state-of-the-art methods.

6. Ablation Study

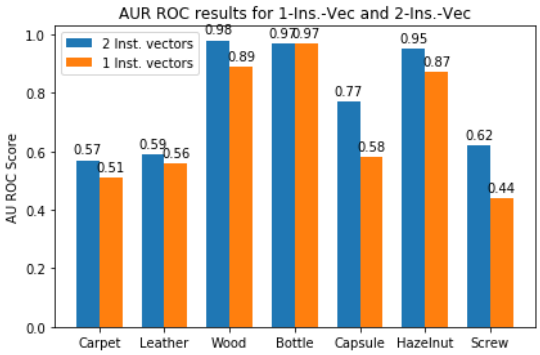

Fig. 4. Comparison of AUC for one and two instantiation vectors respectively.

Table 2. AUC results of anomaly detection using CIFAR10. Each row shows the normal class on which the model has been trained. Comparative results taken from literature.[4]

	CIFAR Results							
Class	**OC SVM**	**KDE**	**DAE**	**VAE**	**Pix CNN**	**GAN**	**LSA**	**Ours**
0	0.630	0.658	0.718	0.688	0.788	0.708	0.735	**0.751**
1	0.440	0.520	0.401	0.403	0.428	0.458	**0.580**	0.550
2	0.649	0.657	0.685	0.679	0.617	0.664	0.690	**0.708**
3	0.487	0.497	0.556	0.528	0.574	0.510	0.542	**0.609**
4	0.735	0.727	0.740	0.748	0.511	0.722	0.761	**0.805**
5	0.500	0.496	0.547	0.519	0.571	0.505	0.546	**0.645**
6	0.725	**0.758**	0.642	0.695	0.422	0.707	0.751	0.729
7	0.533	0.564	0.497	0.500	0.454	0.471	0.535	**0.651**
8	0.649	0.680	0.724	0.700	0.715	0.713	0.717	**0.771**
9	0.508	0.540	0.389	0.398	0.426	0.458	**0.548**	0.532
Mean	0.586	0.610	0.590	0.586	0.551	0.592	0.641	**0.675**

Table 3. AUC results of anomaly detection using the MVTec dataset. Comparative results taken from literature.[8]

MVTec Results					
Class	**AE (SSIM)**	**AE (L2)**	**AnoGAN**	**CNN Feat. Dict.**	**Ours**
Carpet	**0.665**	0.495	0.490	0.625	0.573
Grid	0.690	**0.775**	0.510	0.450	0.587
Leather	0.460	0.440	0.515	**0.670**	0.591
Tile	0.505	**0.770**	0.510	0.705	0.560
Wood	0.830	0.735	0.680	0.835	**0.978**
Bottle	0.875	0.795	0.690	0.530	**0.968**
Cable	0.610	0.555	0.525	0.605	**0.871**
Capsule	0.605	0.620	0.580	0.405	**0.771**
Hazelnut	0.535	0.885	0.495	0.485	**0.949**
Metal nut	0.540	**0.725**	0.495	0.645	0.672
Pill	0.600	0.615	0.620	0.455	**0.732**
Screw	0.505	0.685	0.345	0.430	**0.619**
Toothbrush	0.740	**0.985**	0.565	0.515	0.956
Transistor	0.515	0.710	0.665	0.575	**0.872**
Zipper	**0.800**	**0.800**	0.590	0.535	0.738
Mean	0.632	0.706	0.552	0.564	**0.762**

An ablation study has been done to justify our choice of two vectors for the instantiation parameters. Our hypothesis is that using a single vector, and thus no dynamic feature routing, will lead to worse results, since all the features are forced to contribute to the same instantiation vector. On the other hand, with two vectors, features are allowed at testing time to accumulate at the second vector if they don't agree, as in the case of anomalies. The proposed model reconstruction capabilities have been tested by comparing the SSIM of the reconstructed images.

The ablation study has been made using the MVTec dataset, maintaining similar hyperparameters (Table 1). For the comparison, three products (Bottle, Capsule, Hazelnut) and three textures(Carpet, Leather, Wood) have been chosen from the dataset. The comparative results can be seen in Figure 4. In all the categories, the two vectors approach performed better than one vector with average of 9% improvement.

References

1. X. Gu, L. Akoglu and A. Rinaldo, Statistical analysis of nearest neighbor methods for anomaly detection,

Advances in Neural Information Processing Systems, 2019, pp. 10921–10931.
2. C. Piciarelli, P. Mishra and G. L. Foresti, Image anomaly detection with capsule networks and imbalanced datasets, *International Conference on Image Analysis and Processing*, Springer2019, pp. 257–267.
3. S. Sabour, N. Frosst and G. E. Hinton, Dynamic routing between capsules, *Advances in neural information processing systems*, 2017, pp. 3856–3866.
4. D. Abati, A. Porrello, S. Calderara and R. Cucchiara, Latent space autoregression for novelty detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 481–490.
5. S. Akcay, A. Atapour-Abarghouei and T. Breckon, Ganomaly: Semi-supervised anomaly detection via adversarial training, *Proc. Asian Conference on Computer Vision*, (Springer, 2018).
6. A. Klushyn, N. Chen, R. Kurle, B. Cseke and P. van der Smagt, Learning hierarchical priors in vaes, *Advances in Neural Information Processing Systems 32*, (Curran Associates, Inc., 2019), pp. 2866–2875.
7. I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning* (MIT Press, 2016). `http://www.deeplearningbook.org`.
8. P. Bergmann, M. Fauser, D. Sattlegger and C. Steger, Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9592–9600.
9. M. Ahmed, A. N. Mahmood and J. Hu, A survey of network anomaly detection techniques, *Journal of Network and Computer Applications* **60** (2016) 19 – 31.
10. M. L. Antonie, O. R. Zaïane and A. Coman, Application of data mining techniques for medical image classification, *Proceedings of the Second International Conference on Multimedia Data Mining*, *MDMKDD'01*2001, pp. 94–101.
11. C. Piciarelli, D. Avola, D. Pannone and G. L. Foresti, A vision-based system for internal pipeline inspection, *IEEE Transactions on Industrial Informatics* **15**(6) (2019) 3289–3299.
12. C. Piciarelli, C. Micheloni and G. L. Foresti, Trajectory-based anomalous event detection, *IEEE Transaction on Circuits and Systems for Video Technology* **18**(11) (2008) 1544–1554.
13. P. Chen, S. Yang and J. A. McCann, Distributed real-time anomaly detection in networked industrial sensing systems, *IEEE Transactions on Industrial Electronics* **62**(6) (2015) 3832–3842.
14. V. Chandola, A. Banerjee and V. Kumar, Anomaly detection: A survey, *ACM Computing Surveys* **41**(3) (2009) 15:1–15:58.
15. R. Chalapathy and S. Chawla, Deep learning for anomaly detection: A survey arXiv:1901.03407v2 [cs.LG], (2019).
16. D. Miljković, Review of novelty detection methods, *The 33rd International Convention MIPRO*, IEEE2010, pp. 593–598.
17. M. Buda, A. Maki and M. A. Mazurowski, A systematic study of the class imbalance problem in convolutional neural networks, *Neural Networks* **106** (oct 2018) 249–259.
18. D. Wulsin, J. Blanco, R. Mani and B. Litt, Semi-supervised anomaly detection for eeg waveforms using deep belief nets, *2010 Ninth International Conference on Machine Learning and Applications*, 2010, pp. 436–441.
19. U. Fiore, F. Palmieri, A. Castiglione and A. De Santis, Network anomaly detection with the restricted boltzmann machine, *Neurocomput.* **122** (2013) 13–23.
20. M. Nadeem, O. Marshall, S. Singh, X. Fang and X. Yuan, Semi-supervised deep neural network for network intrusion detection, *2016 KSU conference on cybersecurity education, research and practice*, 2016.
21. T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth and G. Langs, Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, *International Conference on Information Processing in Medical Imaging*, Springer2017, pp. 146–157.
22. L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt and M. Kloft, Image anomaly detection with generative adversarial networks, *European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer2018, pp. 3–17.
23. J. Hu, L. Shen and G. Sun, Squeeze-and-excitation networks, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 7132–7141.
24. J. Johnson, A. Alahi and L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, *European conference on computer vision*, Springer2016, pp. 694–711.
25. P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger and C. Steger, Improving unsupervised defect segmentation by applying structural similarity to autoencoders, *International joint conference on computer vision, imaging and computer graphics theory and applications*, 2019.
26. S. A. Nene, S. K. Nayar and H. Murase, Columbia object image library (coil-100), Tech. Report Technical Report CUCS-006-96, Columbia University (1996).
27. S. Pidhorskyi, R. Almohsen and G. Doretto, Generative probabilistic novelty detection with adversarial autoencoders, *Advances in neural information processing systems*, 2018, pp. 6822–6833.
28. D. P. Kingma and M. Welling, auto-encoding variational bayes, *International Conference on Learning Representations*, 2014.
29. A. Van den Oord, N. Kalchbrenner, L. Espeholt,

O. Vinyals, A. Graves *et al.*, Conditional image generation with pixelcnn decoders, *Advances in neural information processing systems*, 2016, pp. 4790–4798.
30. P. Napoletano, F. Piccoli and R. Schettini, Anomaly detection in nanofibrous materials by cnn-based self-similarity, *Sensors* **18**(1) (2018) p. 209.
31. P. Perera, R. Nallapati and B. Xiang, Ocgan: One-class novelty detection using gans with constrained latent representations, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2898–2906.
32. M. Sakurada and T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, *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.

7. Appendix

In this section we present the detailed network structure of the proposed model.

- **Decoder Structure** - As the image size of the datasets used for the study are different, the network structure has to adapt accordingly, mostly the decoder part to reconstruct the images of the same size to that of input image. The detailed network architecture can be seen in Table 6
- **Upsampling Layer** - Upsampling layer uses the linear layer to upsample the instantiation parameters from $\mathbb{R}^{64}$ to $\mathbb{R}^{512\times mf\times mf}$, where "mf" is the multiplying factor equals to the width of output features from SE-Resnet18. A detailed structure can be seen in the Table 5.
- **Pyramidal Pooling Layer** - It takes input from the SE-Resent18 block and applies adaptive average pooling. The detailed network structure is shown in Figure 5.

Table 5. Detailed Upsampling Layer structure

Upsampling Layer
Linear in:64,out:128
Batch norm
ReLU
Linear in:128,out:512
Batch norm
ReLU
Linear in:128,out:512*mf^2

Table 6. Detailed Decoder structure for CIFAR10, COIL100 and MVTec dataset

Decoder	
CIFAR10	***MVTec/Coil100***
ConvTranspose2d in:64,out:16:k:5,s:1p:1	ConvTranspose2d in:1024,out:16:k:3,s:2p:1
Batch norm ReLU	Batch norm ReLU
ConvTranspose2d in:16,out:32:k:5,s:1	ConvTranspose2d in:16,out:32:k:3,s:2p:1
Batch norm ReLU	Batch norm ReLU
ConvTranspose2d in:32,out:32:k:6,s:1	ConvTranspose2d in:32,out:32:k:4,s:2
Batch norm ReLU	Batch norm ReLU
ConvTranspose2d in:32,out:32:k:6,s:1	ConvTranspose2d in:32,out:3:k:4,s:2,p:1
Batch norm ReLU	Tanh
ConvTranspose2d in:32,out:32:k:5,s:1	
Batch norm ReLU	
ConvTranspose2d in:32,out:3:k:5,s:1	
Tanh	

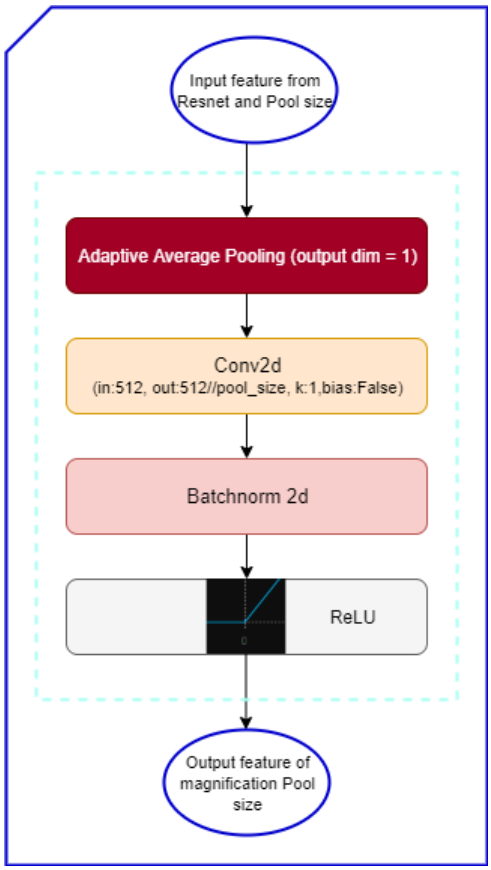

Fig. 5. Detailed structure of Pyramidal Pooling Layer