A Neural Network for Image Anomaly Detection with Deep Pyramidal Representations and Dynamic Routing

A neural network for image anomaly detection with deep pyramidal representations 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 PIADE, 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, COIL-100 and MVTec.

1. Introduction

Modern IT infrastructures are more and more oriented to the acquisition of enormous amounts of data, which cannot be manually analyzed and require proper algorithms to be processed. In this field, an emerging requirement is to identify anomalies in the collected data.

Anomaly detection is defined as the identification of samples which significantly differ from a reference set of normal data, under the assumption that the number of anomalies is much smaller than the number of normal samples. From a rigorous point of view the problem is not well defined, as there is no rigorous formalization of the properties this difference should have in order to be considered significant: the definition of anomaly is often dependent on an arbitrarily-defined threshold. However, from a practical point of view, anomaly detection is a widespread problem, marking strong presence in the fields of medical imaging,[1] network intrusion detection,[2] defect detection,[3] fault prevention,[4] video surveillance,[5] and many others. In medical imaging an anomaly could be a tumor tissue among several data of healthy patients, in quality-assurance industrial inspection it can be a defective product, in the surveillance videos of a shopping mall it can be the behavioral pattern of a thief compared to normal clients, etc.

This work is focused on the topic of deep neural

networks for image anomaly detection. The goal thus is, given a set of images representing the normality model, to classify new images either as normal or anomalous. We assume that there are no anomalies in the training set, thus following a semi-supervised approach (see section 2). The basic idea is to adopt a reconstruction-based strategy, in which a neural network learns how to encode the input images into a low-dimension latent space and then reconstructs them in order to minimize the difference between the original and reconstructed image. By training such a network on normal data only, the network will learn only the features that are useful for the reconstruction of the training set, and thus will fail at reconstructing anomalous images. The difference between input and reconstructed images can thus be used as an anomaly score.

Despite the reconstruction-based approach has been already used in literature, we here propose PIADE (Pyramidal Image Anomaly DEtector), a network architecture that includes several novel strategies to improve the overall system performance. First, we adopt a pyramidal approach[6] that analyzes the image features at different scale levels in parallel. This way, we improve the probability that relevant image features are extracted at the scale level in which the anomaly is more evident. Then, we borrow the idea of *dynamic routing* from capsule networks[7] to have a finer control on the features that are really useful for the task of anomaly detection. Moreover, most of the reconstruction-based methods use a pixel-wise loss functions in order to compare reconstructions and input data. This assumes an independence among the pixels, which is generally not true. Hence, we adopted a more sophisticated loss function which considers the inter-relationship between the pixels and a perception-based loss computed by a pre-trained network. Finally, while many papers[8–12] in this field have been evaluated on simple and not anomaly-oriented datasets such as MNIST,[13] we tested our proposed network and method on one of the first real-world datasets for image anomaly detection published by MVTec.[14] We found that the proposed model performed at par or better when compared with various state-of-the-art methods.

The rest of the paper is organized as follows: in section 2 we give a short overview of the most relevant works in the field of deep-learning-based anomaly detection, as well as a taxonomy of the proposed approaches based on the type of learning and the adopted strategies. In section 3 we describe in detail the proposed model, defining both the network structure and the loss functions used to train it. Section 4 shows the experimental results and compare the system performance with the state-of-the-art literature. Here we also evaluate the system on the novel and anomaly-oriented MVTec[14] dataset. Section 5 extends the experimental results with some ablation studies, in which the system performances are measured when some of the network components are turned off, in order to measure their influence on the final result. Conclusions are in section 6.

2. Related Works

Image-based anomaly detection is not a new topic, especially in the field of industrial inspection where it is used to identify defective products.[15] Several classical image processing and machine learning methods have been used to perform this task, such as Bayesian networks, rule-based systems, clustering algorithms, etc.[16,17] However, despite the recent trends in machine learning being almost entirely focused on deep neural networks,[18–26] deep anomaly detection is still a novel topic which has been mostly investigated only in the last few years.[27] The rest of this section is focused on deep learning strategies for anomaly detection.

Most of the proposed works can be roughly grouped in three fundamental approaches, differing in the type of adopted learning strategy, on the training data and on the expected outcomes:

- supervised learning;
- unsupervised learning;
- semi-supervised learning.

Some authors[28] use the terms "outlier detection" and "novelty detection" to distinguish between unsupervised and semi-supervised approaches. However, there is no general consensus on whether these terms should be used as synonyms or should denote different problems.[17] In the rest of this paper we use the most generic term "anomaly detection".

The supervised approach consists in training a standard binary classifier, in which the two classes represent the normal and the anomalous data, and a labeled training set is available.[29] The main difference with regular classification problems consists in

the training dataset being typically extremely imbalanced, as many real-world scenarios (e.g. industrial inspection) can produce a large number of normal samples but a very limited amount of anomalies. This requires proper data augmentation strategies such as over-sampling the anomaly class, either by synthetic data creation 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.[30,31] Few works have tried to deal with data imbalance directly rather than with data augmentation, as in the work by Piciarelli et al.[32] where the authors use a capsule network and define a proper anomaly metric which is efficient even with a 99:1 data imbalance. Also the work by Perera and Patel[33] can be considered a supervised approach since it requires an external dataset of negative examples, however in that case that dataset is just used to extract compact and descriptive features for the normal class, which are later analyzed with a traditional classifier.

The unsupervised approach considers the case of unlabeled training sets containing both normal data and anomalies. It consists in estimating the region where normal data is mostly concentrated, consequently separating the deviant observations. Since neural networks are not particularly well-suited for clustering tasks, the most common approach in this case is to use the neural network as a feature extractor, and the features are later analyzed with other machine learning tools such as one-class Support Vector Machines. In several papers the two steps are handled independently,[34–36] but some works proved that better results can be achieved if the feature extraction and classification tasks are jointly optimized, in order to extract the best features for subsequent anomaly detection.[11,37,38]

The most common approach to deep anomaly detection however is the semi-supervised one. In this case the assumption is that a labeled training set exists, but it contains only normal data. This is a frequent scenario in real-world applications, where large amounts of normal data can be acquired easily but anomalous ones are extremely rare. The task of the anomaly detection system is thus to learn a "normality" model from the training data, and subsequently classify as anomaly any sample that does not fit the model.[8,9,39–42] Some authors define this approach as unsupervised, especially if the system can work even in presence of a small number of anomalies in the training data. However, we believe that "robust semi-supervised" would be a better description of these approaches, since the anomalies in the training data are not directly used to learn anything, and the system is just robust enough to learn the normality model despite the anomalies in the training data.

The strategies to build the normality model in semi-supervised deep-learning works mostly fall in two categories: reconstruction-based methods and probabilistic methods. In the former case the network ideally acts as an identity function on normal data, trying to reproduce the input image in its output. The network typically has an autoencoder-like structure, where the image is first encoded into an intermediate low-dimension latent space, and then it is reconstructed back to its original dimensions. This structure forces the latent space to catch only the most representative features of the images. In this case, the normality model description is distributed across all the network weights, and a successfully trained network is able to correctly reconstruct normal images. On the other hand, it is assumed that anomalous images have different latent features and thus cannot be properly reconstructed. Anomaly detection is thus obtained by measuring the dissimilarity between the original and the reconstructed image.[43–46]

The probabilistic methods instead aim at explicitly building a probabilistic model of the latent space of normal images.[42] This can be obtained for example with variational autoencoders,[47] where the latent space is forced to have a Gaussian distribution. In this case, anomaly detection can be obtained directly in the latent space, comparing the latent features of the input image with the Gaussian model representing the normality. Another popular probabilistic approach is to rely on Generative Adversarial Networks (GAN). GAN models are composed of two networks: a generator that creates images starting from random samplings, and a discriminator trying to distinguish between generated and real images. The two networks are in competition, with the generator trying to create images that are more and more similar to the training ones, and the discriminator trying to distinguish them. Convergence is achieved when the generator is able to create novel, synthetic images that are indistinguishable from the training ones, and

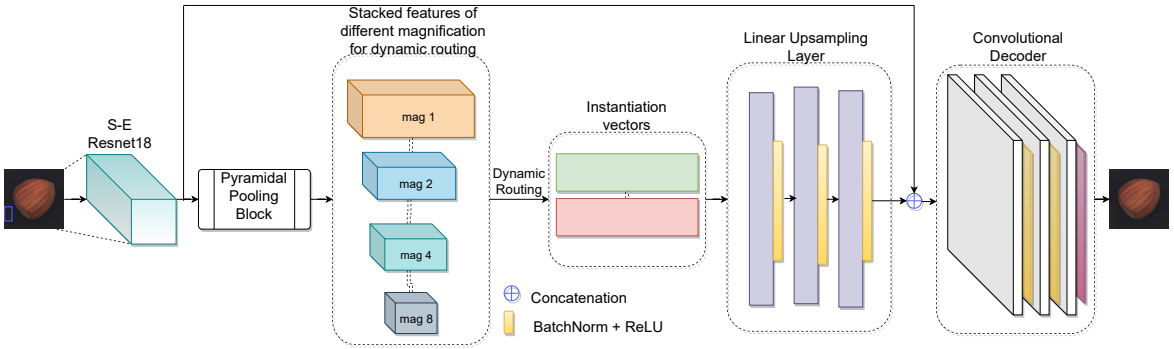

Fig. 1. Proposed PIADE network architecture. The network consists of the first levels of a SE-Resnet18 network for feature extraction, followed by a pyramidal pooling layer that extracts features at different scales. The features are then dynamically routed to two instantiation vectors in $\mathbb{R}^{64}$. The vectors are 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.

the discriminator has the same performances of a random guess. Anomaly detection can be performed by inverting the generator and projecting the tested image in the generator's sampling space, where it can be classified using a probabilistic model. The inversion is not trivial but feasible, as in the ADGAN model.[8] Other approaches try to avoid the inversion step by including the anomaly detector in the discriminator, as in AnoGAN.[41]

It must be noted that the reconstruction-based and probabilistic models can coexist, as for example in the GANomaly architecture by Akcay et al.,[9] in OCGAN by Perera et al.,[12] or in the work by Abati et al.,[40] where anomalies are identified by an analysis at both the image and latent space levels.

3. Proposed Model

As already introduced in Section 1, we propose to approach the problem of anomaly detection using PIADE, a semi-supervised, reconstruction-based deep neural network. The network is thus trained on normal data only, and its aim is to reconstruct the input image after its encoding in a low-dimension intermediate latent space. The rationale of this approach is that the latent space will model only the relevant features of normal images, and will be unsuitable to describe anomalous images. Anomalies can thus be detected by their poor reconstruction.

Compared to other deep anomaly methods, our main contributions consist in the addition of a pyramidal level in the network structure, to extract image features at different resolution scales. This way we increase the probability to extract the features at a scale in which the anomaly is particularly evident. We also use dynamic routing, as proposed by Hinton for capsule networks,[7] to better extract relevant features. In addition, we also adopt a better way to compare the input and reconstructed images. Most methods use trivial pixel-by-pixel comparisons, e.g. MSE loss, which implicitly assumes the unrealistic hypothesis of statistical independence between pixels. We instead propose a high-level image comparison loss which quantify the image degradation at an higher abstraction level. In Section 3.1 we describe the network architecture and in Section 3.2 we define the image comparison loss function.

3.1. *Network Architecture*

The PIADE network architecture is shown in Figure 1. It consists of an initial ResNet block to extract basic image features. These features are then pooled in the pyramidal pooling block, in order to represent them at different scales. Following the idea of capsule networks,[7] the pooled features are then dynamically routed to two instantiation vectors (more details below), in order to filter the best ones that are useful for later image reconstruction. Image is then reconstructed via a linear upsampling layer and a convolutional decoder in order to obtain an output with the same shape of the input data. We here give a detailed description of each block.

- **SE-ResNet18** - A pre-trained ResNet18 network is used for deep feature extraction. The network was trained over the imageNet dataset. All 5 con-

volutional layers of the ResNet18 have been used. The primary idea is that the network is able to extract generic 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.

In order to improve the quality of the extracted features, each convolutional block is followed by a Squeeze-and-Excitation (SE) block, as proposed by Hu et al.[48] (see Figure 2). The SE block is a form of attention mechanism among convolutional channels, that adaptively calculates channel-wise weights to to explicitly model the inter-dependencies between channels.

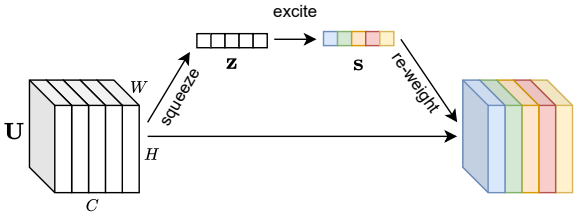

Fig. 2. Squeeze-Excitation (SE) Block.[48]

Formally, given the output tensor $\mathbf{U} \in \mathbb{R}^{W \times H \times C}$ of a convolutional layer, SE first squeezes it to a vector $\mathbf{z} \in \mathbb{R}^{1 \times 1 \times C}$ by aggregating the spatial dimensions according to the following equation:

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j). \tag{1}$$

Then, the excitation vector $\mathbf{s} \in \mathbb{R}^{1 \times 1 \times C}$, containing the channel weights, is computed as follows:

$$\mathbf{s} = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})) \tag{2}$$

where σ is an element-wise sigmoid function, δ is a ReLU activation function, and $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{4} \times C}$, $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{4}}$ are two learned matrices that model the excitation function itself. Then, the tensor $\mathbf{U}$ is channel-wise re-weighted using weights $\mathbf{s}$ (meaning that each channel $\mathbf{U}_c \in \mathbb{R}^{W \times H}$ is multiplied by the scalar $\mathbf{s}_c$) to generate the output of the SE block, which is subsequently passed to the other network layers. 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** - The idea behind the pyramidal pooling layer is that image features can be analyzed at different magnifications, and possibly relevant features that are well-visible at a given scale could be not well extracted by the network at another scale. The pyramidal pooling layer thus scales the input features at different magnification levels, thus increasing the possibility that features relevant for the anomaly detection task are actually extracted.

  Each pyramid layer consists in the application of an adaptive average pooling. For a given feature map with spatial size $W \times H$, standard average pooling creates a new map with size $W/k \times H/k$, and each element of the new map is the average of the corresponding elements in the original map, as shown in Figure 3. Adaptive average pooling works in a similar way, but the term k is automatically chosen to guarantee a fixed size output, independently from the input size.

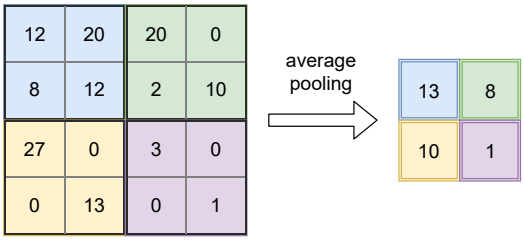

Fig. 3. Average pooling with $k = 2$.

In the proposed architecture we adopt four different pyramid levels, respectively with spatial output sizes of 8×8, 4×4, 2×2, and 1×1. The number of channels is left unchanged (in our case, the ResNet18 module final output uses 512 channels). Each pooling layer is followed by a convolution to reduce the number of channels (1×1 kernel, stride=1, 64 channels) and a batch normalization. The output features of each pyramid level are then flattened, concatenated and reshaped in a final set of 8-dimensional feature vectors.

- **Dynamic Routing** - Dynamic routing is a novel algorithm proposed by Hinton in his Capsule Networks paper.[7] In capsule networks, the belonging of a sample to a given class is represented by a vector (called instantiation vector) rather than a scalar value. The norm of the vector represents how much the sample belongs to a specific class, while the vector itself represents a specific instance of that class, hence the name. Each instantiation vector is defined as a sum of several features, and

dynamic routing is the algorithm that dynamically chooses which features must be routed to each vector. In other words, for each class, dynamic routing selects the best features that are more suited to describe that class.
Dynamic routing algorithm is shown in algorithm 1, where the `squash`() function forces the vector norms to be in the range $[0, 1]$:

$$\texttt{squash}(s_j) = \frac{||s_j||^2}{a + ||s_j||^2} \frac{s_j}{||s_j||} \tag{3}$$

In the proposed system, dynamic routing is used to route the features from the pyramidal pooling layer to two instantiation vectors, described below.

Algorithm 1 Dynamic Routing Algorithm[7]

function ROUTING($\hat{u}_{j|i}, r, l$)
  ▷ $\hat{u}_{j|i}$ are the features from layer i to j
  ▷ l is the current layer
  $\forall i \in l, \forall j \in l+1 : b_{ij} \leftarrow 0$
  for r iterations **do**
    $\forall i \in l : c_i \leftarrow \texttt{softmax}(b_i)$
    $\forall j \in (l+1) : s_j \leftarrow \sum_i c_{ij} \hat{u}_{j|i}$
    $\forall j \in (l+1) : v_j \leftarrow \texttt{squash}(s_j)$
    $\forall i \in l, \forall j \in (l+1) : b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$
  end for
  return v_j
end function

- **Instantiation Vectors** - In a standard capsule network classifier, each instantiation vector represents a class. In our case, we adapted this model to anomaly detection by using only two instantiation vectors: the first one represents the normal class, while the second one is just used to collect all the features that are discarded by the routing algorithm because not relevant enough to model the normal class. Observe that this is not the same as modeling the anomaly class because we do not have anomalies in the training set. In the proposed reconstruction-based method, these vectors are the low-dimension ($\mathbb{R}^{64}$ in our experiments) latent space in which images are mapped before reconstruction. For this reason, during training the first instantiation vector is always passed to the upsampling and decoding layers. While testing, the vector with maximum norm is decoded instead. This method is in contrast with traditional approaches 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 5 we propose an ablation study to prove that the two-vectors approach always performs better than the single-vector one.
- **Upsampling and decoding layers** - The upsampling layer consists of three linear layers and it 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. The 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 the same size as 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 its input. In order to measure the similarity between the original image and its reconstruction, we considered three possible loss functions:

- *MSE Loss* - Mean Squared Error (MSE) loss is a pixel-level loss, which assumes independence between pixels. MSE loss is computed as the average of the squared pixel-wise differences of the two images, and can be formally defined in terms of the Frobenius norm as $\frac{1}{WH}\|X - \hat{X}\|_F^2$, where X is the input and $\hat{X}$ is the output of the network (respectively the original and the reconstructed image), and W, H are the image width and height. 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[49] is a more sophisticated loss trying to catch visually meaningful differences in images. It is a MSE loss computed between the high-level image features obtained by a pre-trained VGG11 network using its first four layers. The loss is defined as $\frac{1}{WHC}\|F(X) - F(\hat{X})\|_F^2$, where F is the transformation function applied through the trained four layers of VGG11 network, and W, C, H are the size of the resulting feature map. 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)[50] 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 L, defined as a weighted sum of the three image comparison losses:

$$\begin{aligned} L(X,\hat{X}) = MSELoss(X,\hat{X}) + \\ \lambda_1 PercLoss(X,\hat{X}) + \lambda_2 SSIM(X,\hat{X}) \end{aligned} \quad (5)$$

where X is the input image and $\hat{X}$ is the image reconstructed by the network. λ_1 and λ_2 are weighing factor between three losses. All the experiments discussed in section 4 are obtained with $\lambda_1 = \lambda_2 = 1$, since we noted that small differences in these values do not lead to significant differences in the results. In the ablation study presented in section 5 we show the results with $\lambda_1 = 0$ and $\lambda_1 = \lambda_2 = 0$, thus disabling the perceptual and/or the SSIM components.

4. Experimental Results

In this section we present the experimental results obtained with PIADE. We first describe the datasets used for training and testing, then we describe the testing procedure and the adopted metrics to measure the system performance. Finally, comparative results are given, in order to evaluate the achieved results with other state-of-the-art works on anomaly detection.

4.1. *Datasets*

The proposed PIADE model has been tested using publicly available datasets. Tests have been done on CIFAR10[51] and COIL-100[52] datasets. Although these datasets are not specifically designed for anomaly detection tasks, they are useful to show the ability of the system to discriminate between one class, considered normal, and the other ones, considered anomalies. Since the performances of many previous works have been evaluated on these datasets, testing on CIFAR10 and COIL-100 is important for comparative results. In addition to this, the proposed model has also been tested on the real-world MVTec anomaly detection dataset,[14] which is a recently published dataset specifically for anomaly detection tasks (see figure 4).

- **CIFAR10**: It contains 60,000 images with size 32×32 pixel. Images are grouped in ten classes: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. 50,000 images are for training while 10,000 images are for testing. For this study, we treated one of the classes 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 pixels. Differing from CIFAR10, each class in COIL-100 represents a single object, but observed from different points of view: each object was kept on an automated turntable, and the images were taken at a fixed pose step of 5 degrees. For each object a total of 72 images were recorded. As in the case of CIFAR10, experimental results are averaged over 100 runs, each one with a different class chosen as normal. 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 Pidhorskyi et al.,[53] and split the training and testing data with a ratio of $80\% : 20\%$.
- **MVTec Dataset**: MVTec recently published a real-world anomaly detection dataset. It contains 5,354 high-resolution color images of different texture and objects categories. It has normal and anomalous images which showcase 70 different types of anomalies of different real-world products. It also contains 3 products images in grayscale, as

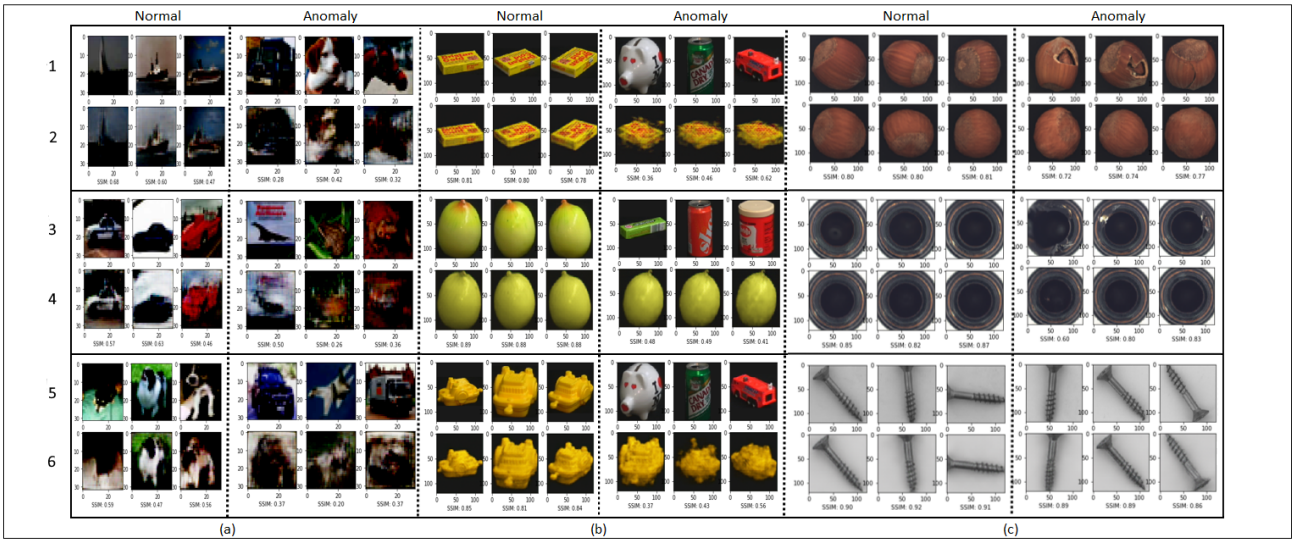

Fig. 4. Reconstructions of normal and anomaly images for (a) CIFAR10, (b) COIL-100, and (c) MVTec datasets. Rows 1, 3, and 5 show few examples taken from the normal class and from the anomalies. Rows 2, 4, and 6 show the reconstructed images. The network is unable to correctly reconstruct anomalous data. The normal classes shown for the CIFAR10 examples are respectively Ship (row 1), Car (row 3) and Dog (row 5). The normal classes for COIL100 are the first three objects of the dataset. The normal classes for MVTec are Hazelnut, Bottle and Screw.

grayscale images are very common in industrial practices. With this dataset, all the images were first resized to 120×120 pixel size before being passed to the model. Image anomalies are still visible at this resolution.

4.2. *Training and testing procedure*

The model is trained by minimizing the total loss $L(X, \hat{X})$ (equation 5). The same loss is also used in testing phase as an anomaly score. The model weights are initiated with orthogonal initialization except the ResNet block, which was pre-trained on imageNet, and the VGG11 block, which was pre-trained on imageNet and kept fixed. The architectural hyper-parameters are shown in table 1. The model has been implemented in Python using the Pytorch framework[54,a], the source code is available online[b]. All the experiments have been done on a dual Xeon E5-2660 CPU, 224 GB RAM, 1 Tesla K40 and 2 Titan XP GPUs. On such hardware, training took on average 20 minutes for each class of the MVTec dataset, and 30 minutes for each class of the CIFAR10 dataset. Inference time is however extremely low, requiring on average 0.012 seconds to classify a single MVTec image and 0.006 seconds for a CIFAR10 image.

Table 1. Training hyperparameters.

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

4.3. *Performance metrics*

In order to measure the system performance, we consider the total loss $L(X, \hat{X})$ (equation 5) as a measure of the degree of anomaly of an image. This is a sound approach since the loss measures the dissimilarity between input and reconstructed images, and the reconstruction-based approach assumes this dissimilarity will be high for anomalous images.

Once this anomaly score is computed, a threshold is required to convert it to a binary classification: any image with a score above the threshold will be considered anomalous, while the remaining ones will be classified as normal. Once this classification is done, standard statistics such as True Positives (TP),

[a] `https://pytorch.org/`
[b] `https://github.com/pankajmishra000/PIADE`

Table 2. AUC scores using the CIFAR10 dataset. Each row shows the normal class on which the model has been trained. Comparative results taken from literature.[40]

Class	OC SVM	KDE	DAE	VAE	Pix CNN	GAN	LSA	PIADE (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**

False Positives (FP), True Negatives (TN) and False Negatives (FN) can be computed. These raw values are then converted into suitable ratios: in particular, we considered the True Positive Rate (TPR, also known as Recall, or Sensitivity) and the False Positive Rate (FPR), defined as follows:

$$\begin{aligned} TPR &= TP/(TP+FN) \\ FPR &= FP/(FP+TN) \end{aligned} \tag{6}$$

Rather than choosing an arbitrary threshold, we followed the popular approach of computing the (TPR, FPR) pairs for every possible threshold: the plot of these values gives the well-known Receiver Operating Characteristic (ROC) Curve. Finally, the area under the ROC curve (AUC) is used as a performance measure that summarizes the overall quality of the achieved results. This approach has been adopted to perform comparative analysis with state-of-the-art methods both on CIFAR10 and COIL-100 datasets. Observe that the choice of (TPR, FPR) pairs, and thus the use of ROC curves, is typically more suited when the tested dataset is balanced, while on imbalanced datasets the Precision/Recall values are more suited, since they are not affected by the large abundance of True Negatives. However, despite dealing with anomalies, our datasets are balanced for testing: in CIFAR10 and COIL-100, anomalies belong to 9 classes out of 10, and thus can be safely chosen so that their total amount is comparable to the amount of normal data.

Also the MVTec dataset has enough test anomalies to be considered balanced for testing. In this case, however, we adopted the same testing procedure described in the original paper where the dataset is proposed.[14] Here, for each class and each tested method, we compute the Sensitivity (TPR) and Specificity (TNR) for all the possible thresholds, and we select the best pair (TPR_{best}, TNR_{best}) as the one that maximizes their sum $TPR + TNR$. The best algorithm for each class is defined as the one with the highest mean of the two values, i.e. with the highest $(TPR_{best} + TNR_{best})/2$.

4.4. *Comparative results*

Table 2 shows the results obtained for CIFAR10 dataset. Training has been done considering one class as normal and the rest as anomaly, and this procedure 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,[55] Pix-CNN[56] and Latent Space Autoregression.[40] The comparative results have been taken from the work by Abati et al.[40] Table 2 shows that the proposed model superseded the results of the state-of-the-art models in 7 out of 10 classes, and it has the best average result.

Table 3. AUC scores using the COIL-100 dataset.

Models	Reference	AUC
GPND	*NIPS 2018*[53]	0.979
OCGAN	*CVPR2019*[12]	0.995
DCAE	*MLSDA14*[57]	0.908
PIADE (ours)	–	**0.998**

Table 4. Results on the MVTec dataset. Each row shows the results achieved on a specific category. Each cell shows the best TNR (top) and TPR (bottom) values. The method with the highest mean of the two values is shown in bold. Comparative results taken from literature.[14]

Class	AE SSIM	AE (L2)	Ano Gan	CNN Feat. Dict.	PIADE (ours)
Carpet	**0.43** **0.90**	0.57 0.42	0.82 0.16	0.89 0.36	0.33 0.74
Grid	0.38 1.00	**0.57** **0.98**	0.90 0.12	0.57 0.33	0.67 0.60
Leather	0.00 0.92	0.06 0.82	0.91 0.12	**0.63** **0.71**	0.82 0.45
Tile	1.00 0.04	**1.00** **0.54**	0.97 0.05	0.97 0.44	0.97 0.19
Wood	0.84 0.82	1.00 0.47	0.89 0.47	0.79 0.88	**0.85** **0.93**
Bottle	0.85 0.90	0.70 0.89	0.95 0.43	1.00 0.06	**0.87** **1.00**
Cable	0.74 0.48	0.93 0.18	0.98 0.07	0.97 0.24	**0.73** **0.93**
Capsule	0.78 0.43	1.00 0.24	0.96 0.20	0.78 0.03	**0.60** **0.83**
Hazelnut	1.00 0.07	0.93 0.84	0.83 0.16	0.90 0.07	**0.89** **0.97**
Metal nut	1.00 0.08	0.68 0.77	0.86 0.13	0.55 0.74	**0.57** **0.78**
Pill	0.92 0.28	1.00 0.23	1.00 0.24	0.85 0.06	**0.89** **0.51**
Screw	0.95 0.06	0.98 0.39	0.41 0.28	0.73 0.13	**0.80** **0.67**
Toothbrush	0.75 0.73	**1.00** **0.97**	1.00 0.13	1.00 0.03	0.92 0.96
Transistor	1.00 0.03	0.97 0.45	0.98 0.35	1.00 0.15	**0.70** **0.90**
Zipper	1.00 0.60	0.97 0.63	0.78 0.40	0.78 0.29	**1.00** **0.65**

Table 3 shows the results on the COIL-100 dataset. Since it would be impractical to report the results for all the 100 classes in a table, we just show the final average AUCs for each method, computed over the 100 classes, where each class was alternatively considered as normal and rest as anomaly. Our proposed model achieved a final AUC score of 0.998, surpassing GPND,[53] OCGAN[12] and DCAE,[57] which recorded 0.979, 0.995, and 0.908 respectively. Out of 100 classes, more than 50 classes achieved AUC score of 1, denoting a 100% correct classification.

Table 4 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,[50] the GAN-based approach AnoGAN,[41] and CNN feature dictionary.[58] The comparative results have been taken from the work by Bergmann et al.[14] Performance is compared by computing the average of the best TPR and TNR for each class and for each model. The proposed method achieves the best results on 10 out of 15 categories. It is worth noting that PIADE performs poorly on the texture classes (Carpet, Grid, Leather, Tile). This is probably due to the ResNet module, which has been pre-trained to extract features that are meaningful to describe full objects rather than textures and patterns.

5. Ablation Studies

We here propose a set of ablation studies, in which the network is re-trained after the removal of specific parts in order to measure the influence of those parts on the network performance.

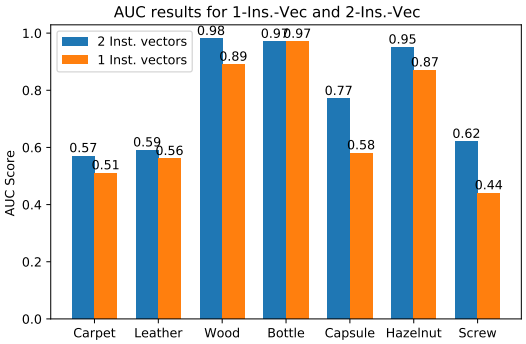

Fig. 5. Comparison of AUC for one and two instantiation vectors respectively.

The first ablation study has been done to justify the use of dynamic routing with two instantiation vectors. Our hypothesis is that using a single vector, and thus disabling 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 do not give a valid contribution to the image reconstruction task. 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 the same hyperparameters used for regular testing (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 5. In all the categories, the two instantiation vectors approach performed better than one vector with a 9% improvement on average.

Another ablation study has been done to study the effect of the Squeeze-Excitation attention module. In this case we used the ResNet with and without soft attention and found that the model with soft-attention performed better in comparison to vanilla ResNet. We took the same 7 objects of the previous experiment and measured the AUC. Results of SE-ResNet performed always better or at par with the vanilla ResNet, as shown in Figure 6.

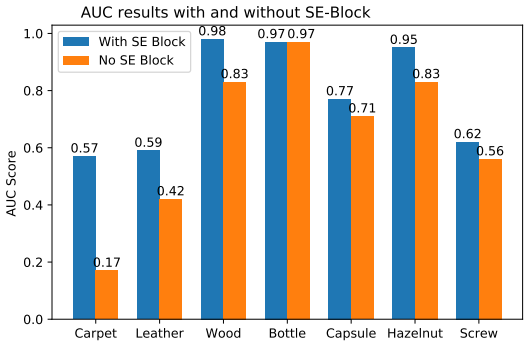

Fig. 6. Comparison of AUC with and without the Squeeze-Excitation attention module.

The last ablation study aims at testing the performance of the three loss functions described in section 3.2. In order to measure the system performances, three models have been trained with the following configurations: a) MSE loss only ($\lambda_1 = \lambda_2 = 0$); b) MSE + SSIM loss ($\lambda_1 = 0$); c) MSE + SSIM + Perceptual loss ($\lambda_1 = \lambda_2 = 1$). The network was trained on the MVTec dataset categories "bottle", "capsule" and "carpet", results are measured in terms of AUC. The results shown in Figure 7 clearly show that the model with all the three losses has superior results if compared to the other configurations.

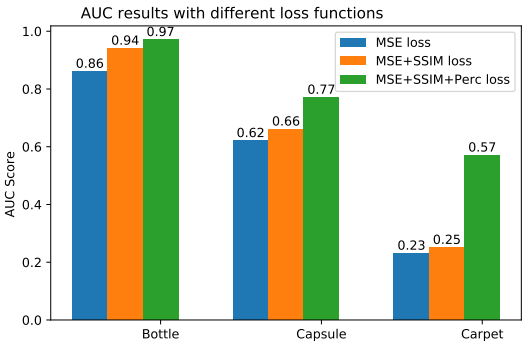

Fig. 7. Comparison of AUC with different loss function combinations.

6. Conclusions

In this work we proposed a reconstruction-based, semi-supervised deep neural network for image anomaly detection. The network is trained on normal data only, and it builds a "normality model" by mapping the input images in a low-dimension feature space, from which they can be correctly reconstructed. The inability of the network to reconstruct other images allows the identification of anomalies, which can be detected by their higher reconstruction error. Compared to other state-of-the-art works, the proposed models includes a pyramidal multi-scale approach to analyze image features at different scale levels, a dynamic routing layer inspired by the architecture of capsule networks, and a high-level image comparison loss. Moreover, the system has been tested not only on standard datasets such as CIFAR10 and COIL-100 (which have not been initially created for anomaly detection experiments), but also on the recently proposed MVTec dataset of anomalies in industrial images. Experimental results showed that the proposed model is at least at-par, and often outperforms other state-of-the-art works. Further ablation experiments prove the validity of the architectural choices on which the proposed network is based.

References

1. 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, 2001, pp. 94–101.
2. 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.
3. 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.
4. 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.
5. 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.
6. A. M. Soares, B. J. Fernandes and C. J. Bastos-Filho, Structured pyramidal neural networks, International Journal of Neural Systems **28**(5) (2018) 1750021.
7. S. Sabour, N. Frosst and G. E. Hinton, Dynamic routing between capsules, Advances in neural information processing systems, 2017, pp. 3856–3866.
8. 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, 2018, pp. 3–17.
9. S. Akcay, A. Atapour-Abarghouei and T. Breckon, Ganomaly: Semi-supervised anomaly detection via adversarial training, Proc. Asian Conference on Computer Vision, 2018.
10. C. Zhou and R. C. Paffenroth, Anomaly detection with robust deep autoencoders, Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (New York, NY, USA, 2017), pp. 665–674.
11. L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller and M. Kloft, Deep one-class classification, Proceedings of the 35th International Conference on Machine Learning, eds. J. Dy and A. Krause Proceedings of Machine Learning Research **80**, (PMLR, Stockholmsmässan, Stockholm Sweden, 2018), pp. 4393–4402.
12. 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.
13. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner et al., Gradient-based learning applied to document recognition, Proceedings of the IEEE **86**(11) (1998) 2278–2324.
14. 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.
15. D. Carrera, F. Manganini, G. Boracchi and E. Lanzarone, Defect detection in sem images of nanofibrous materials, IEEE Transactions on Industrial Informatics **13**(2) (2017) 551–561.
16. V. Chandola, A. Banerjee and V. Kumar, Anomaly detection: A survey, ACM Computing Surveys **41**(3) (2009) 15:1–15:58.
17. M. A. Pimentel, D. A. Clifton, L. Clifton and L. Tarassenko, A review of novelty detection, Signal Processing **99** (2014) 215 – 249.
18. I. Goodfellow, Y. Bengio and A. Courville, Deep Learning (MIT Press, 2016). `http://www.deeplearningbook.org`.
19. A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems, 2012, pp. 1097–1105.
20. O. M. Manzanera, S. K. Meles, K. L. Leenders, R. J. Renken, M. Pagani, D. Arnaldi, F. Nobili, J. Obeso, M. R. Oroz, S. Morbelli et al., Scaled subprofile modeling and convolutional neural networks for the identification of parkinson's disease in 3d nuclear imaging data, International journal of neural systems **29**(09) (2019) 1950010.
21. A. H. Ansari, P. J. Cherian, A. Caicedo, G. Naulaers, M. De Vos and S. Van Huffel, Neonatal seizure detection using deep convolutional neural networks, International journal of neural systems **29**(04) (2019) 1850011.
22. C. Hua, H. Wang, H. Wang, S. Lu, C. Liu and S. M. Khalid, A novel method of building functional brain network using deep learning algorithm with application in proficiency detection, International journal of neural systems **29**(01) (2019) 1850015.
23. A. Antoniades, L. Spyrou, D. Martin-Lopez, A. Valentin, G. Alarcon, S. Sanei and C. C. Took, Deep neural architectures for mapping scalp to intracranial eeg, International journal of neural systems **28**(08) (2018) 1850009.
24. U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan and H. Adeli, Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals, Computers in biology and medicine **100** (2018) 270–278.
25. U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, H. Adeli and D. P. Subha, Automated eeg-based screening of depression using deep convolutional neural network, Computer methods and programs in biomedicine **161** (2018) 103–113.
26. M. H. Rafiei and H. Adeli, A new neural dynamic classification algorithm, IEEE transactions on neural networks and learning systems **28**(12) (2017) 3074–3083.
27. B. R. Kiran, D. M. Thomas and R. Parakkal, An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos, Journal of Imaging **4**(2) (2018).
28. A. Géron, Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent

systems (O'Reilly Media, 2017).
29. C. Ieracitano, F. Pantó, N. Mammone, A. Paviglianiti, P. Frontera and F. C. Morabito, Toward an Automatic Classification of SEM Images of Nanomaterials via a Deep Learning Approach, Neural Approaches to Dynamics of Signal Exchanges eds. A. Esposito, M. Faundez-Zanuy, F. C. Morabito and E. Pasero (Springer Singapore, Singapore, 2020), Singapore, pp. 61–72.
30. 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.
31. S. K. Lim, Y. Loo, N.-T. Tran, N.-M. Cheung, G. Roig and Y. Elovici, Doping: Generative data augmentation for unsupervised anomaly detection with GAN, 2018 IEEE International Conference on Data Mining (ICDM), 2018, pp. 1122–1127.
32. C. Piciarelli, P. Mishra and G. L. Foresti, Image anomaly detection with capsule networks and imbalanced datasets, International Conference on Image Analysis and Processing, 2019, pp. 257–267.
33. P. Perera and V. M. Patel, Learning deep features for one-class classification, IEEE Transactions on Image Processing **28** (2019).
34. S. M. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning, Pattern Recognition **58** (2016) 121–134.
35. M. Gutoski, N. M. R. Aquino, M. Ribeiro, E. Lazzaretti and H. S. Lopes, Detection of video anomalies using convolutional autoencoders and one-class support vector machines, XIII Brazilian Congress on Computational Intelligence, **2017**2017.
36. C. Aytekin, X. Ni, F. Cricri and E. Aksu, Clustering and unsupervised anomaly detection with L2 normalized deep auto-encoder representations, 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–6.
37. Z. Ghafoori and C. Leckie, Deep multi-sphere support vector data description, Proceedings of the 2020 SIAM International Conference on Data Mining, SIAM2020, pp. 109–117.
38. R. Chalapathy, A. K. Menon and S. Chawla, Anomaly detection using one-class neural networks, arXiv preprint arXiv:1802.06360 (2018).
39. 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.
40. 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.
41. 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, 2017, pp. 146–157.
42. P. Oza and V. M. Patel, One-class convolutional neural network, IEEE Signal Processing Letters **26**(2) (2018) 277–281.
43. P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger and C. Steger, Improving unsupervised defect segmentation by applying structural similarity to autoencoders, arXiv preprint arXiv:1807.02011 (2018).
44. M. Sabokrou, M. Khalooei, M. Fathy and E. Adeli, Adversarially learned one-class classifier for novelty detection, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3379–3388.
45. J. Chen, S. Sathe, C. Aggarwal and D. Turaga, Outlier detection with autoencoder ensembles, Proceedings of the 2017 SIAM international conference on data mining, SIAM2017, pp. 90–98.
46. Y. Xia, X. Cao, F. Wen, G. Hua and J. Sun, Learning discriminative reconstructions for unsupervised outlier removal, Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1511–1519.
47. J. Sun, X. Wang, N. Xiong and J. Shao, Learning sparse representation with variational auto-encoder for anomaly detection, IEEE Access **6** (2018) 33353–33361.
48. 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.
49. J. Johnson, A. Alahi and L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, European conference on computer vision, 2016, pp. 694–711.
50. 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.
51. A. Krizhevsky, Learning multiple layers of features from tiny images, Master's thesis, Dept. of Comp. Sci., University of Toronto (2009).
52. 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).
53. S. Pidhorskyi, R. Almohsen and G. Doretto, Generative probabilistic novelty detection with adversarial autoencoders, Advances in neural information processing systems, 2018, pp. 6822–6833.
54. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy,

B. Steiner, L. Fang, J. Bai and S. Chintala, Pytorch: An imperative style, high-performance deep learning library, Advances in Neural Information Processing Systems 32, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett (Curran Associates, Inc., 2019), pp. 8024–8035.
55. D. P. Kingma and M. Welling, auto-encoding variational bayes, International Conference on Learning Representations, 2014.
56. 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.
57. 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.
58. P. Napoletano, F. Piccoli and R. Schettini, Anomaly detection in nanofibrous materials by cnn-based self-similarity, Sensors **18**(1) (2018) p. 209.