k-NNN: Nearest Neighbors of Neighbors for Anomaly Detection

Ori Nizan
Technion—Israel Institute of Technology
snizori@campus.technion.ac.il

Ayellet Tal
Technion—Israel Institute of Technology
ayellet@ee.technion.ac.il

Abstract

Anomaly detection aims at identifying images that deviate significantly from the norm. We focus on algorithms that embed the normal training examples in space and, when given a test image, detect anomalies based on the features' distance to the k-nearest training neighbors. We propose a new operator that takes into account the varying structure & importance of the features in the embedding space. Interestingly, this is achieved by considering not only the nearest neighbors but also the neighbors of these neighbors (k-NNN). Our results demonstrate that by simply replacing the nearest neighbor component in existing algorithms with our k-NNN, while leaving the rest of the algorithms unchanged, the performance of each algorithm is improved. This holds true for both common homogeneous datasets, such as specific flowers, as well as for more diverse datasets.

1. Introduction

Anomaly detection aims at finding patterns in the data that do not conform to the expected "behavior" [3]. It has numerous applications, most notably in manufacturing, surveillance, fraud detection, medical diagnostics, autonomous cars, and detecting outliers. Out of the variety of anomaly detection methods [10, 13, 14, 47], we focus on those that rely on the *k-Nearest-Neighbor (k-NN)* operator [5, 15, 20, 38, 39]. These methods learn the embedding of *normal* images or patches. Given a test image, the distance of its embedding to its k-nearest (training) neighbors is computed and this distance determines whether the test image is anomalous or not. The underlying assumption is that anomalous features should reside farther away from normal features than normal features from each other. Thus, a point is considered anomalous when its average distance to its k nearest neighbors exceeds a certain threshold.

A major disadvantage of this approach is that the structure and the importance of the features in the embedding space is not taken into account when looking for anomalies. Figure 1 shows such a case, in which the normal set varies and consists of flowers that belong to different classes, with different inner distances (the flowers are not classified beforehand). A flower that does not resemble any of the normal flowers (in the red rectangle) will not be detected as anomalous by the *k-NN* operator, because its distance to its nearest flowers is less than the distances between normal flowers in different regions of the embedding space. Our operator, which implicitly takes the structure of the space into account, will detect this flower as anomalous.

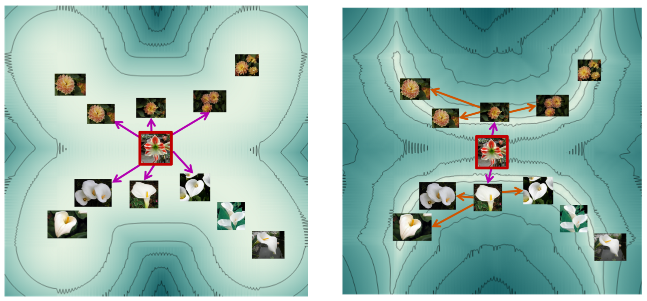
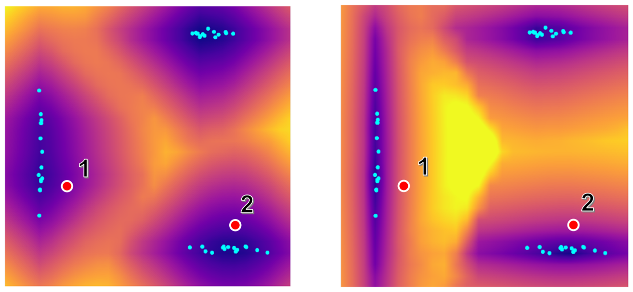

(a) *k-NN* (b) *k-NNN*

Figure 1. **Neighbors of neighbors illustration.** A common approach is to base the anomaly score on the distances between a test image to its k-nearest neighbors (*k-NN*) in the feature space. In this figure, the yellow and the white flowers are considered normal, whereas the flower in the red rectangle is anomalous. The background color represents the anomaly score. (a) Since the normal set is diverse, *k-NN*-based methods might fail, since the distances of the anomalous image to its neighbors is smaller than the distances between similar images to each other (e.g., between the white flowers). (b) Our *k-NNN* operator sets better distances between neighbors, which reflect the diversity of the normal examples. It will correctly detect the anomalous flower as such.

The structure of the embedding space is important also when the normal set is homogeneous, as illustrated in Figure 2 for a synthetic example. The 2D embedding of the normal training points lie on three lines, two of which are parallel and one is perpendicular to them. Two anomalous points, marked as 1 and 2 (in red), lie above the horizontal line and to the right of the vertical line, respectively. Their *5-NN* distance is the same as that of the normal points between themselves, and thus they might not be identified as



(a) *k-NN*: 1,2 are normal (b) *k-NNN*: 1,2 are anomalous

Figure 2. ***k-NNN* benefit.** The cyan points represent the 2D embedding of normal images. The heat maps show the *5-NN* distance of each point on the plane; the yellower a region, the more anomalous it is. The distance between the red anomalous points to their 5-nearest cyan neighbors is equal to the distances between the cyan points themselves. Thus, the classical *k-NN* operator fails to detect them as anomalous. Differently, our *k-NNN* operator, which uses the neighbors statistics, detects them correctly.

anomalous by *k-NN*-based methods. Similar cases are likely to occur when there are not enough training samples.

We propose a novel operator, termed the *k-nearest neighbors-of-neighbors (k-NNN)*, which addresses this problem, as illustrated in Figures 1-2. It differentiates between regions and considers the more indicative features at certain regions as more influential. For instance, in Figure 2 the feature that makes point 2 anomalous is its y feature, whereas the feature that makes point 1 anomalous is its x feature. We show how to efficiently realize this idea of considering regions differently, by simply looking at the neighbors of neighbors of a test point. Intuitively, the neighbors of neighbors provide information about regions, which balances between a global view of the dataset and a more local view, which is based only on the immediate neighbors.

To consider the feature importance, we analyze the directions associated with the anomalies. The classical *Principal component analysis (PCA)* analyzes datasets of high-dimension features, while preserving the maximum amount of information. We observe that for anomaly detection, Eigen vectors associated with small Eigen values matter more than those of large values. Furthermore, as shown in Figure 2, the difference between an anomalous point and its nearest neighbor(s) is perpendicular to the direction of the large Eigen vector(s). Intuitively, this is so since anomalies are characterized by features not present in the dataset. We show how to utilize this observation within our operator.

To demonstrate the benefit of our approach, we replace the *k-NN* operator used in several anomaly detection algorithms with our *k-NNN* operator. We show how this modification manages to improve the results of each algorithm on a variety of datasets.

Hence, this paper makes two contributions:

1. It introduces a novel, general, efficient and accurate operator—the *k-NNN* operator, which provides an "in-between" look at the data, between local and global. It benefits both diverse and homogeneous normal sets.

2. It proposes a novel normalization scheme that prioritizes the small Eigen values, effectively handling the challenges associated with small datasets. Additionally, it shows how to address the data high dimensionality, even when the number of neighbors is limited.

2. Related work

Anomaly detection. Anomaly detection is important to discover potentially dangerous situations, in the manufacturing industry for detecting product faults, in medicine for diagnosing diseases etc. It is a highly challenging task due to image structure, varying environmental conditions, imbalanced datasets, and data diversity. Hence, this task has attracted a huge amount of research. We refer the reader to a couple of comprehensive and excellent surveys [11, 12, 49].

Hereafter, we consider methods that detect whether or not an image is anomalous and do not aim to segment it. They may be categorized to three classes, as follows.

Reconstruction-based methods learn a set of basis functions on the training data. Given a test image, they attempt to reconstruct it using these functions. If the test image cannot be reconstructed, it is considered anomalous. The set of basis functions vary. Examples include K-means [21], K nearest neighbors (k-NN) [16], principal component analysis (PCA) [2] etc. Deep learning has been used as well [43, 53].

Distribution-based methods model the probability density function (PDF) of the distribution of the normal data [16, 29]. Given a test example, it is evaluated using the PDF. If the probability is small, it is considered anomalous. Deep learning can be applied as well [27, 54].

Classification-based methods are the most prevalent recently. They includes one-class methods [42, 44, 48] and self-supervised learning [6, 17, 18, 22]. Recently it was shown in [5] that a simple method, which is based on *k-NN*, outperforms such self-supervised methods.

Nearest neighbors, which we pursue in this paper, may be considered as reconstruction-based or as distribution-based, since it performs density estimation.

The k-NN operator. Nearest neighbor search has been utilized across a wide range of applications in computer vision. The *k-NN* operator has been found beneficial in classification and correspondence [4, 8, 46], intrusion detection [28], medical applications [26, 31], fault detection [5, 38], out of distribution detection [45] and more. In some applications approximation of the *k-NN* operator was studied [23, 30, 36]. We focus, however, on the exact *k-NN* operator in the context of anomaly detection. The most related works to ours

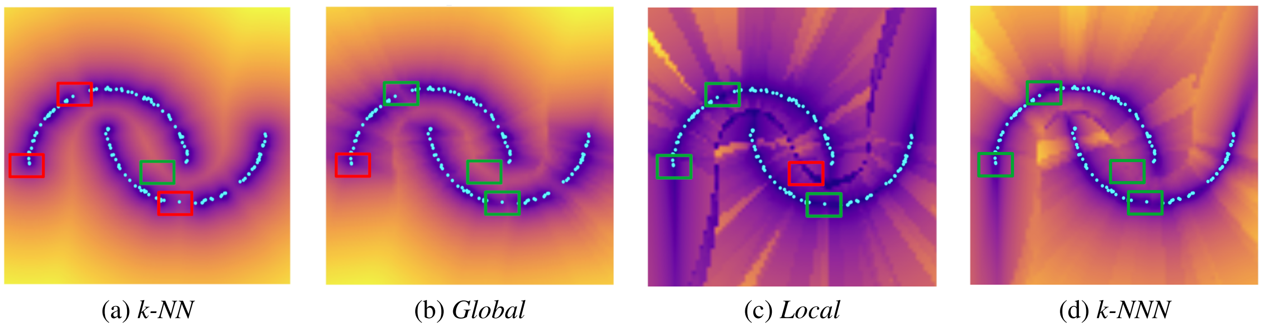


Figure 3. **Different types of normalization.** The cyan points represent the normal points in a 2D embedding space; they lie along two circular arcs. We expect that all the normal points will reside along these arcs, even if the arcs contain holes or terminate; we also expect that points that lie in other regions of the plane will be anomalous. The background color represents the anomaly score of each region according to the specific method: The more yellowish a region, the more anomalous it is. While our *k-NNN* (d) correctly classifies the plane (blue regions are only along the arcs), the local method erroneously considers as normal the region in-between the arcs (c), and the global and *k-NN* method erroneously detect regions along the arcs (in holes or beyond termination) as anomalous (a-b). In this figure green rectangles mark correct outcome (normal/anomalous) and red rectangles mark incorrect outcome.

are [15, 19, 33, 34, 38, 41, 50], which use nearest neighbors for anomaly detection.

3. Method

Given an image, our goal is to determine whether it is anomalous or not. This should be done in a semi-supervised manner, utilizing only a dataset of normal images, without anomalies. We follow the approach in which features are extracted during training, in order to represent normal images. During inference, a given test image is passed through the feature extractor and the k-nearest neighbors in the (training) feature space are found. An anomaly score is derived from the distances to these nearest neighbors.

Eigen-vector for anomaly detection. In order to take into account the shape of the embedding space, we estimate the space directions using its Eigen vectors. Recall that the greater the Eigen value, the larger the variance of the data in the direction of the corresponding Eigen vector. Our proposed normalization is based on our observation that small Eigenvalues tend to correspond to anomaly directions more than large Eigenvalues. This can be explained by the fact that a small variation means that normal images are close in that direction [40]. Thus, a small deviation in this direction is more likely to be an anomaly.

The neighbors of neighbors operator. One may consider a couple of setups. In a global setup, the Eigen vectors are determined for the whole training (normal) set during preprocessing. In a local setup, the Eigen vectors are calculated for a test point based on its k-nearest neighbors in the training set. We propose an "in-between" operator, which gathers more statistical information on the embedding space than the local operator and not as much as the global operator. In particular, for each neighbor we utilize the Eigen vectors (/values) of its neighbors. We elaborate on the realization of this idea hereafter.

Figure 3 illustrates the intuition behind our operator. The normal points, in cyan, lie along one of two circular arcs. Obviously, normal points should lie along these arcs, even in holes and beyond the arcs' termination, whereas anomalous points should reside elsewhere in the plane. In this figure, the plane is colored according to its normality/anomaly, as determined by each method. Blue regions are considered to be normal by the method, whereas yellow regions are considered anomalous. The green rectangles highlight regions where the specific method correctly classifies points as normal or anomalous. The red rectangles highlight regions in which the specific method fails to classify points. It can be seen that our method enjoys the benefits of all worlds—global & local. This result is analyzed and supported quantitatively in Section 5.

To realize our operator, during training (Figure 4), we first compute the feature vector of each training image, using any preferable embedding model. Then, we compute the k nearest neighbors in feature space for each point of the training data. From these neighbors, we compute the point's n Eigen vectors and their corresponding Eigen values and store this information. Hence, the Eigen vectors (/values) are relative to each individual training point, regardless of the test point.

At inference (Figure 5), given a test point and its feature vector, f, we find its k nearest neighbors among the training samples, f_i, $1 \leq i \leq k$. Each of these f_is is already associated with n Eigen vectors and Eigen values, v_{ij} and e_{ij}, $1 \leq j \leq n$, computed during training.

Following our observation, for a point to be considered normal, the difference vector between it and its neighbor should be parallel to the large Eigen vectors (parallel to the

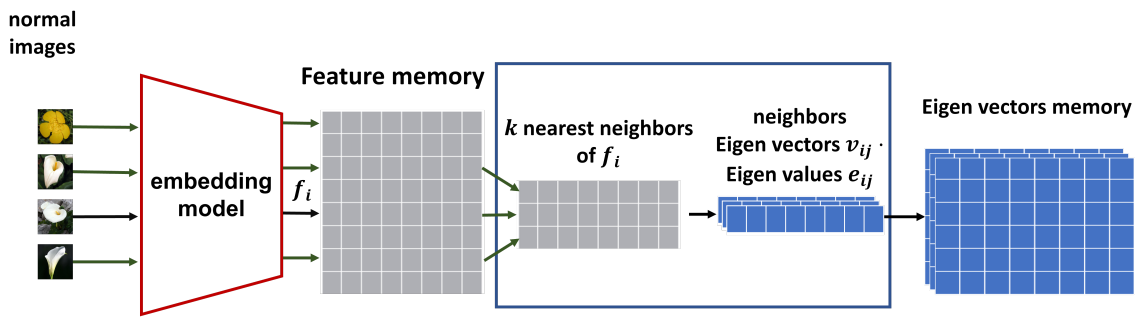

normal images
embedding model
f_i
Feature memory
k nearest neighbors of f_i
neighbors Eigen vectors v_{ij} · Eigen values e_{ij}
Eigen vectors memory

Figure 4. **k-NNN training.** Given training normal images, their embeddings, f_i, are computed. Then, the nearest neighbors of each image embedding is computed. The Eigen vectors and Eigen values, derived from their k neighbors, are computed and stored.

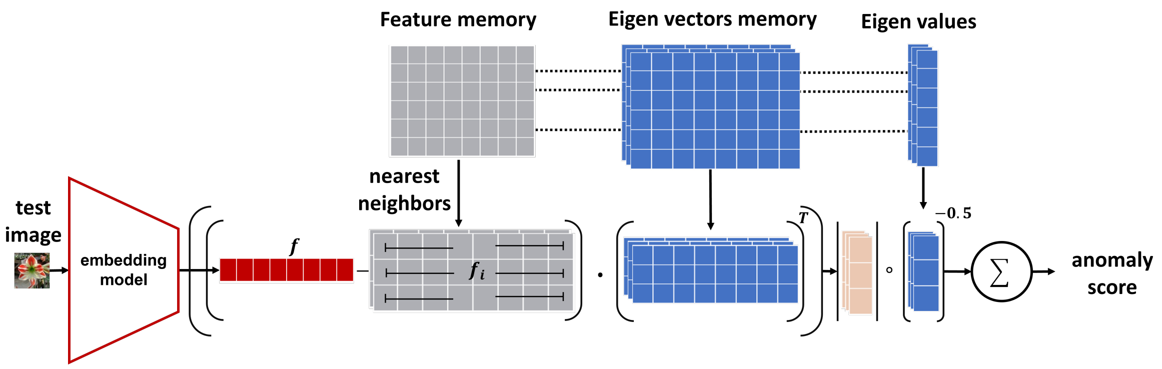


Figure 5. ***k-NNN* inference.** Given an input image, its embedding f is first computed. Its k nearest neighbors are found and the Eigen values & vectors of each neighbor are extracted from the memory. An anomaly score is calculated according to Eq. 1.

distribution of the normal embeddings). Reversely, for a point to be considered anomalous, this vector should be perpendicular to the large Eigen vectors. Thus, we calculate the anomaly score AS of a feature vector f as follows.

$$AS(f) = \sum_{i=1}^{k}\sum_{j=1}^{n} |(f - f_i) \cdot v_{ij}| \cdot \frac{1}{\sqrt{e_{ij}}}. \qquad (1)$$

In Eq. 1, the difference between the test feature vector and that of its neighbor is multiplied by the different Eigen vectors, specific to the i^{th} nearest neighbor. The more parallel these vectors are, the larger the value of this multiplication. Furthermore, this number is multiplied by square root of the inverse of the Eigen value, giving more weight to the small Eigen values. Figure 3(d) demonstrates that the *k-NNN* operator indeed classifies the plane properly.

Feature partition & re-ordering. Evaluating the Eigen vectors by neighbors-of-neighbors (let alone locally) means that only a small number of points in the neighborhood of f is used for estimating the Eigen vectors. This might be prohibitive since a feature vector of dimension N cannot be estimated by $k << N$ neighbors, as it will result in major loss of information.

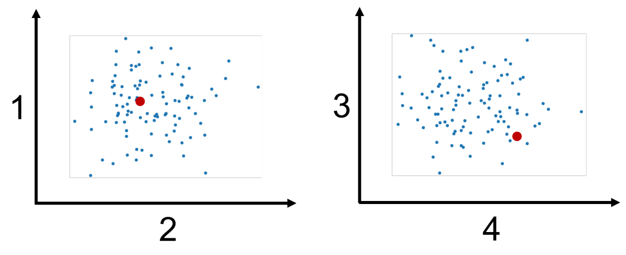
To address this problem, we propose to estimate the Eigen vectors in parts. We divide the vector features (entries) into equal-size sets and calculate the Eigen vectors for each set separately. In particular, we divide the feature vectors of dimension N into at least N/k sets. In the following we denote the number of sets by S and the dimension of the (sub)-feature vector of a set by L (e.g. if $S = N/k$ then $L = k$). In general, the more samples used to calculate the Eigen vectors, the larger L may be.

Specifically, given a feature vector of the test point , f, and its k nearest neighbors among the training samples, $f_i, 1 \leq i \leq k$, we partition f and f_i into parts, f_s & $f_{i,s}$, $1 < i < k, 1 < s < S$. We denote the Eigen vectors and Eigen values associated with f_i, which are similarly partitioned, by $v_{ij,s}$, $e_{ij,s}$, $1 \leq j \leq n$ $(n < L)$, respectively.

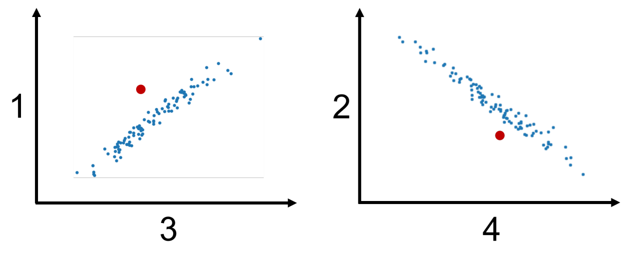
As before, we calculate the difference between f and each of its neighbors, however this time this is done per set. The anomaly score of f, AS, takes into account the results of all the sets, as follows:

$$AS(f) = \sum_{i=1}^{k}\sum_{j=1}^{n}\sum_{s=1}^{S} |(f_s - f_{i,s}) \cdot v_{ij,s}| \cdot \frac{1}{\sqrt{e_{ij,s}}}. \qquad (2)$$

The remaining question is how to partition the vectors into sets. The disadvantage of using independent sets is that



(a) Initial order: 1 & 2, 3 & 4



(b) Correlated order: 1 & 3, 2 & 4

Figure 6. **Reordering by correspondence.** Suppose we are given 100 feature vectors, each with 4 entries (features). In the graphs, every axis represents one feature. (a) shows that there is no correlation between features 1 and 2 (left), and similarly between features 3 and 4 (right). Thus, the anomalous red point cannot be distinguished from the normal cyan points. However, after the features are reordered according to their correlation, it is much easier to distinguish between the anomalous point and the normal ones (b).

the relations between the features in the different sets are not taken into account. This may be harmful when the anomalies depend on these relationships. Figure 6 illustrates such a synthetic case, for vectors of 4 features, where the red point is anomalous. In a partition of the features into sets $\{1, 2\}, \{3, 4\}$, as in (a), the red point is indistinguishable from the normal cyan points and thus will not be detected.

This problem can be mitigated by re-ordering the features before partitioning them into sets, based on the correlation between them. If properly done, every set will contain the features that are most correlated to one another, resulting in more meaningful Eigen vectors. Figure 6(b) illustrates the reordering effect, where the features are partitioned into sets $\{1, 3\}, \{2, 4\}$. When reordering is performed prior to splitting the features into sets, the red anomaly point is easily spotted and distinguished from the normal cyan ones, and is thus detected as anomalous.

We propose to apply the following procedure for feature re-ordering. First, the correlations between all pairs of entries of all the feature vectors in the training set are computed. To maximize the correlation within each set, we reorder the feature vector entries of all the vectors simultaneously. This is done in a greedy fashion as follows. The first entry remains in place; the second entry is switched with the one that is most correlated to the first. From now on, until the number of features in the set is L, the subsequent entry is chosen as the one that has the highest average correlation with its previous two entries. When L is reached, we start a new set, whose first entry is chosen as the one that is least correlated with the last two features of the previous set.

4. Experiments

We demonstrate the benefits of our method in two ways. In Section 4.1, we replace the *k-NN* component of state-of-the-art anomaly detection methods with our *k-NNN* operator and show improved results. In Section 4.2, we apply our operator to structured synthetic features. In both cases, we demonstrate that even when applying our method to features extracted by networks not specifically designed for anomaly detection, the results are excellent. For evaluation, we use the *AUROC* metric, which is commonly used for evaluating anomaly detection performance.

4.1. Improving anomaly detection methods

In the following, we replace the *k-NN* component of state-of-the-art *k-NN*-based anomaly detection methods with our *K-NNN* and evaluate the results on several datasets.

Networks & datasets. We examine three systems that use *k-NN*: (1) k-NN applied to the features of ResNet18 [47], (2) *SPADE* [15], and (3) *Panda* [38]. We use four datasets: (1) *MVTec* [7], which contains $5,354$ high-resolution images of different object and texture classes. It is divided into $3,629$ normal images for training and $1,725$ anomalous images for testing. The images contain more than 70 different types of defects (anomalies), such as scratches, dents, and structural changes. The ground truth was specified manually. (2) *Oxford flowers 102* [32], which contains 102 flower categories, with $1,020$ training and validation images and $6,149$ test images, where each class includes 40-258 images. (3) *Fashion MNIST* [51], which contains 10 categories, with $60,000$ training and validation images and $10,000$ test images; each class includes $6,000$ images. (4) *CIFAR10* [25], which contains 10 categories, with $50,000$ training and validation images and $10,000$ test images; each class includes $6,000$ images.

Results. Table 1 shows the results on MVTec. In this dataset, every class has anomalous examples of its own. Our method improves the mean performance of all three networks. Furthermore, it improves the performance for almost all the classes (except 3 classes for a single network).

Table 2 reinforces the above results. It shows improved performance on three additional datasets, which differ greatly from one another, in their size, type and diversity.

Table 3 evaluates our model on highly diverse normal

Classes	Feature +*k-NN*	Feature +*k-NNN*	[15]	[15] +*k-NNN*	[38]	[38] +*k-NNN*
carpet	0.896	**0.990**	0.928	**0.959**	0.843	**0.898**
grid	0.444	**0.777**	0.473	**0.663**	0.554	**0.723**
leather	0.792	**0.986**	0.954	**0.974**	0.960	**0.975**
tile	0.986	**0.993**	0.965	**0.970**	**0.985**	0.976
wood	0.636	**0.938**	0.958	**0.985**	**0.913**	0.906
bottle	0.971	**0.983**	0.972	**0.988**	**0.992**	0.982
cable	0.882	**0.934**	0.848	**0.899**	0.821	**0.863**
capsule	0.803	**0.919**	0.897	**0.941**	0.911	**0.919**
hazelnut	0.903	**0.991**	0.881	**0.966**	0.925	**0.968**
metal_nut	0.813	**0.913**	0.710	**0.857**	0.788	**0.860**
pill	0.738	**0.882**	0.801	**0.822**	0.757	**0.786**
screw	0.712	**0.840**	0.667	**0.839**	0.690	**0.805**
toothbrush	0.886	**0.969**	0.889	**0.953**	0.861	**0.914**
transistor	0.878	**0.936**	0.903	**0.929**	0.871	**0.902**
zipper	0.937	**0.964**	**0.966**	0.949	0.934	**0.951**
mean	0.819	**0.934**	0.854	**0.913**	0.854	**0.895**

Table 1. **Replacing the *k-NN* component by our *k-NNN* on MVTec.** Our *k-NNN* improves the mean performance of all networks, as well as the performance for almost all the classes.

Dataset	Feature +*k-NN*	Feature +*k-NNN*	[15]	[15] +*k-NNN*	[38]	[38] +*k-NNN*
CIFAR10	0.841	**0.871**	0.893	**0.922**	0.939	**0.943**
Fashion	0.935	**0.936**	0.911	**0.919**	0.954	**0.958**
Flowers	0.615	**0.895**	0.917	**0.919**	0.935	**0.944**

Table 2. **Replacing *k-NN* by our *k-NNN* on three additional datasets.** Our AUROC results outperform those of the 3 methods.

classes. We consider all classes from a specific dataset as normal, except one, without prior classification. Consequently, only images from that single class are considered anomalous. Despite the challenges posed by diverse normal sets, our *k-NNN* consistently improved networks' results, often showing even greater improvements in performance.

Table 4 further studies the issue of diversity. In MVTec, every class has its own normal and anomalous examples, hence it may be considered as a set of independent datasets. In this experiment, we gradually increase the number of classes, i.e. if the number of classes is 5, we consider all the normal (unclassified) images of the 5 classes as normal and all the anomalous (unclassified) images of these classes as anomalous. (Table 1 is the base case.) The table shows that generally the more diverse the normal class is, the more advantageous our method is. This is not surprising, as after all this is exactly what *k-NNN* is supposed to do—be adaptive to the structure of the feature space.

Dataset	Feature +*k-NN*	Feature +*k-NNN*	[15]	[15] +*k-NNN*	[38]	[38] +*k-NNN*
CIFAR10	0.662	**0.719**	**0.694**	0.667	0.599	**0.610**
Fashion	0.760	**0.780**	0.729	**0.739**	0.683	**0.698**
Flowers	0.612	**0.681**	0.624	**0.686**	0.668	**0.689**

Table 3. **Performance on diverse normal sets.** Images from all the categories, expect one, are considered normal, and images from that single class are anomalies. The AUROC results are averaged across all the classes, i.e. each class is considered anomalous once. When replacing *k-NN* by our *k-NNN* in various networks, our operator is usually more beneficial than for homogeneous sets.

#normal classes	Feature +*k-NN*	Feature +*k-NNN*	[15]	[15] +*k-NNN*	[38]	[38] +*k-NNN*
5	0.806	**0.890**	0.760	**0.938**	0.599	**0.783**
7	0.743	**0.852**	0.716	**0.913**	0.597	**0.817**
11	0.680	**0.760**	0.747	**0.901**	0.631	**0.809**
15	0.627	**0.757**	0.691	**0.884**	0.627	**0.814**

Table 4. **Performance when increasing the normal sets.** Out of the 15 highly diverse classes of MVTec, we use an increasing number of sets, where all their normal and anomalous images are considered as such. As before, no classification is performed beforehand. Our operator is especially beneficial on diverse sets.

4.2. Performance on synthetic benchmarks

We evaluate our method on synthetic, well-thought benchmarks commonly used for visualizing clustering and classification algorithms [1]. These datasets demonstrate the strength of our method in structured embedding spaces. The benchmarks are created by various random sampling generators, allowing us to control their size and complexity. Figure 7 illustrates the three benchmarks we use:

1. **Moons.** The points form two interleaving half circles.

2. **Circles.** The points are organized into two circles, sharing the same center point but having different radii.

3. **Swiss roll.** The points are structured in a shape that resembles a rolled-up Swiss roll pastry.

In our setup, we consider all the generated points as embeddings of normal examples. The farthest a point in the plane is from the specific distribution, the more anomalous it should be. For the training, half of the generated points were used. For the evaluation, the other half was considered as the true positives. We generated the negatives (anomalies) by uniformly sampling the plane. In our experiments we used 100-500 points for training and $5,000$ for testing.

Table 5 quantitatively demonstrates the benefits of our operator. It is important to note that we cannot directly

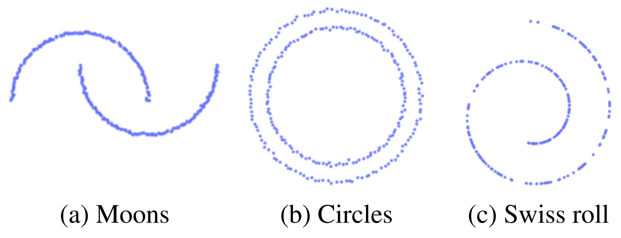


Figure 7. **Synthetic benchmarks**

Dataset	k-NN	Local	Global	*k-NNN*
Moons	0.8214	0.8408	0.8860	**0.9162**
Circles	0.8079	0.8123	0.8109	**0.8163**
Swiss roll	0.9496	0.9496	0.9794	**0.9816**

Table 5. **Performance on the synthetic dataset.** This table shows that our *k-NNN* method outperforms all variants of *k-NN*.

compare against other state-of-the-art methods because they compute the embedding as an integral part of the network, whereas in our case, the embedding is given. Figure 8 provides a qualitative illustration of the results. The classical *k-NN* erroneously identifies a wide area around the curve as normal, while the global *k-NN* identifies in-curve points as anomalous (e.g., the spaces in the spirals), and the local *k-NN* introduces anomalous curves, as observed in the case of the moons and the roll. In contrast, our method accurately captures the thin normal curves, including the holes in them and their continuation. Below, we define these methods.

5. Ablation study

5.1. *k-NN* methods

In this section we study variants of *k-NN* normalization methods and compare them to our *k-NNN* operator.

1. Baseline (no normalization). The Euclidean distance between feature vectors is used without normalizing the features, as in [9, 15, 24, 37]. The advantage of this approach is the speed and having only a single parameter (k, the number of neighbors). However, ignoring the shape of the local and global embedding space might harm performance. This is evident in Figures 3 and 8(a), where normal points within holes and in the continuation of the curve are erroneously detected as anomalous (the red rectangles in Figure 3(a)).

2. Global Eigen-vector normalization. At training time, all the points are used to calculate the Eigen vectors $v_1, v_2, \dots v_n$ and their associated Eigen values $e_1, e_2, \dots e_n$. At test time, given an image represented by its feature vector f, we compute its k nearest neighbors $f_1, f_2, \dots f_k$ from the training set and normalize $(f - f_i)$ based on the Eigen values. The anomaly score, AS, is then

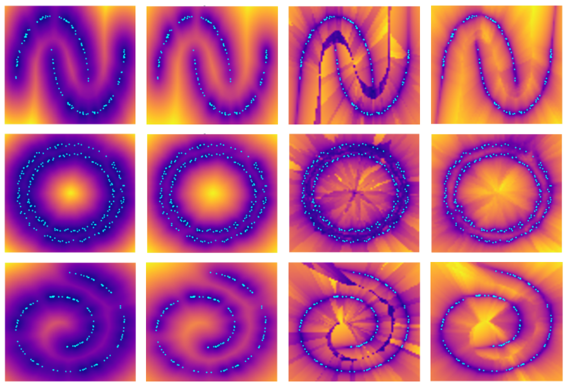


Figure 8. **Qualitative results on synthetic datasets.** Only our method captures the curve distributions accurately. That is to say, the only blue regions across all datasets are the thin curves, including the holes in them and their continuation.

calculated as follows:

$$AS(f) = \sum_{i=1}^{k} \sum_{j=1}^{n} |(f - f_i) \cdot v_j| \cdot \frac{1}{\sqrt{e_j}}. \tag{3}$$

As before, normalization is done using the square root of the inverse of the Eigen value, as we observed that lower Eigen values indicate anomalies. Unlike Eq. 1, all vectors in the embedding space are normalized in the same manner.

The global normalization offers advantages, as it is less affected by noise compared to considering only a subset of points, and it takes into account the relations between features. Figures 3 and 8(b) confirm that global normalization improves upon the baseline *k-NN*. However, global normalization ignores the different directions for different regions of the embedding space. This results in misclassification, as shown in the red square of Figure 3(b), where the continuation of the curve is considered anomalous, though it should be normal. The discrepancy arises from the global Eigen vector direction not matching the correct local direction.

3. Local normalization. In local normalization, we calculate the Eigen vectors for each test point in the embedding space locally, based on its k nearest neighbors in the train set. Then, as before, we calculate the difference between the test sample f and each of its neighbors, and apply Eq. 3.

Figures 3,8(c) show that indeed all the points along the curve are correctly detected as normal (e.g, see the green squares in Figure 3(c)). However, artifacts are created—the blue ”snakes” in-between the arcs of the moons (partially visible in the red rectangle), within the roll and between the circles. The points on these snakes should be classified as anomalous, whereas they are considered to be normal.

Evaluation with different embeddings.

In Table 6 we compare these approaches using a variety

Network	k-NN	Local	Global	*k-NNN*
Max				
Dino-Vits8 [10]	0.9357	0.9357	0.9510	**0.9556**
Resnet50 [47]	0.7690	0.7693	0.8040	**0.8095**
Resnet101 [47]	0.7690	0.7693	0.8040	**0.8095**
ResNext50 [52]	0.7690	0.7693	0.8040	**0.8095**
ResNext101 [52]	0.7690	0.7693	0.8040	**0.8095**
Mean				
Dino-Vits8 [10]	0.9194	0.9207	0.9358	**0.9379**
Resnet50 [47]	0.7282	0.7283	0.7205	**0.7350**
Resnet101 [47]	0.7256	0.7257	0.7414	**0.7423**
ResNext50 [52]	0.7282	0.7283	0.7222	**0.7382**
ResNext101 [52]	0.7284	0.7285	0.7374	**0.7427**

Table 6. **Comparison of various *k-NN* methods.** Our *k-NNN* operator outperforms all other nearest neighbor variants on MVTec. Furthermore, simply finding the embedding using Dino-Vits8 and then running our operator detects anomalous images pretty well.

of image embeddings. Specifically, we used ResNet [47], ResNeXt [52], and Dino-ViT-S/8 [10]). For each embedding, we applied the nearest neighbor variants to detect anomalies on MVTec. Our *k-NNN* outperforms all other variants, consistently with the results in Table 5, where the same experiment was performed on the 2D synthetic dataset of [35]. It is interesting to note that this very simple method already manages to detect anomalous images well.

5.2. Parameters and runtime

How many neighbors should be used? For clarity, throughout the paper, we did not elaborate on having two neighboring parameters: the number of neighbors of a given test image and the number of neighbors of the train images, which are pre-computed and stored (i.e., neighbors of neighbors). Table 7 shows typical results (15 classes from Table 4). It is beneficial to use a small number of neighbors and a large number of neighbors of neighbors. For instance, having 3 neighbors, each having 25 neighbors, is better than 75 direct neighbors *(k-NN)*, improving the performance from 0.616 to 0.757. Intuitively, a few nearby neighbors and enough of their neighbors suffice to provide good statistics of the nearby space. This justifies the key idea of the paper: *k-NN*, which considers only Euclidean distances, cannot capture the structure of the space, even if more neighbors are added. Conversely, our *k-NNN* captures the space structure and addresses the problem of an anomaly being closer to certain clusters than normal examples from each other. In our implementation we use 3 neighbors and 25 neighbors of neighbors across all datasets.

Sub-feature vector dimension. Another parameter that should be set is L, the dimension of the sub-feature vector used for the partition, which might affect the algorithm's

#neighbors	#neighbors-neighbors	performance
1	75	0.753
3	20	0.753
3	**25**	**0.757**
3	75	0.755
4	20	0.754
5	15	0.753
10	5	0.686
60	0	0.616
75	0	0.616
80	0	0.634

Table 7. **How many neighbors are needed?** Considering a few nearby neighbors and many of their neighbors (top) is advantageous to having many neighbors (bottom). This verifies the key idea of the paper—neighbors-of-neighbors (top) capture the structure of space much better than only neighbors do (bottom).

performance and runtime. We used $L = 5$, which experimentally exhibited the best performance. For instance, in Table 4 [15], when $L = 4$ the performance already decreased by 0.006 and similarly when using larger L.

Runtime. A key advantage of our method is that it requires no training. We utilize features generated by any network and apply our *k-NNN* operator. If the Eigen vectors are computed during pre-processing, the inference runtime is instantaneous. Specifically, computing the Eigen vectors and the partition during pre-processing takes approximately 0.074 seconds per image. Determining anomalies in a test image takes only 0.014 seconds. These experiments were performed on the CPU (*AMD EPYC 7763*).

Limitations. The disadvantage of our *k-NNN* is its running time during pre-processing and the memory needed to store the Eigen vectors. Furthermore, our operator has 3 hyper-parameters that need to be to tuned, in comparison to a single parameter in classical *k-NN*.

6. Conclusion

We introduced *k-NNN*, a novel nearest-neighbor operator that utilizes neighbors of neighbors statistics to capture feature space shape information. By computing and storing Eigen vectors of the train set neighbors, *k-NNN* improves anomaly score accuracy during inference through a novel normalization scheme. To handle high-dimensional vectors with limited and insufficient neighbors, we compute these Eigen vectors in parts using multiple feature sets.

Our study demonstrates that replacing the k-NN component of multiple anomaly detection networks with our K-NNN leads to significant improvements. This enhancement is observed in both homogeneous and diverse datasets.

Acknowledgements. We gratefully acknowledge the support of the Israel Science Foundation 2329/22.

References

[1] Scikit-learn. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets 6

[2] Abdi, H., Williams, L.J.: Principal component analysis. Wiley interdisciplinary reviews: computational statistics **2**(4), 433–459 (2010) 2

[3] An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE **2**(1), 1–18 (2015) 1

[4] Arbel, N., Tal, A., Zelnik-Manor, L.: Partial correspondence of 3d shapes using properties of the nearest-neighbor field. Computer & Graphics **82**, 183–192 (Aug 2019) 2

[5] Bergman, L., Cohen, N., Hoshen, Y.: Deep nearest neighbor anomaly detection. arXiv preprint arXiv:2002.10445 (2020) 1, 2

[6] Bergman, L., Hoshen, Y.: Classification-based anomaly detection for general data. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=H1lK_lBtvS 2

[7] Bergmann, P., Fauser, M., Sattlegger, D., Steger, C.: MVTec AD – A comprehensive real-world dataset for unsupervised anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 5

[8] Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008) 2

[9] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000) 7

[10] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. arXiv preprint arXiv:2104.14294 (2021) 1, 8

[11] Chalapathy, R., Chawla, S.: Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407 (2019) 2

[12] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 1–58 (2009) 2

[13] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020) 1

[14] Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15750–15758 (2021) 1

[15] Cohen, N., Hoshen, Y.: Sub-image anomaly detection with deep pyramid correspondences. ArXiv **abs/2005.02357** (2020) 1, 3, 5, 6, 7, 8

[16] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. Applications of data mining in computer security pp. 77–101 (2002) 2

[17] Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. International Conference on Learning Representations (2018) 2

[18] Golan, I., El-Yaniv, R.: Deep anomaly detection using geometric transformations. In: Advances in Neural Information Processing Systems. pp. 9758–9769 (2018) 2

[19] Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., Hengel, A.v.d.: Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1705–1714 (2019) 3

[20] Gu, X., Akoglu, L., Rinaldo, A.: Statistical analysis of nearest neighbor methods for anomaly detection. In: NeurIPS (2019) 1

[21] Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1), 100–108 (1979), http://www.jstor.org/stable/2346830 2

[22] Hendrycks, D., Mazeika, M., Dietterich, T.: Deep anomaly detection with outlier exposure. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=HyxCxhRcY7 2

[23] Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence **33**(1), 117–128 (2010) 2

[24] Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-based outliers: algorithms and applications. The VLDB Journal **8**(3), 237–253 (2000) 7

[25] Krizhevsky, A.: Learning multiple layers of features from tiny images pp. 32–33 (2009), https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf 5

[26] Li, C., Zhang, S., Zhang, H., Pang, L., Lam, K., Hui, C., Zhang, S.: Using the k-nearest neighbor algorithm for the classification of lymph node metastasis in gastric cancer. Computational and mathematical methods in medicine **2012** (2012) 2

[27] Li, D., Chen, D., Goh, J., Ng, S.k.: Anomaly detection with generative adversarial networks for multivariate time series. arXiv preprint arXiv:1809.04758 (2018) 2

[28] Liao, Y., Vemuri, V.R.: Use of k-nearest neighbor classifier for intrusion detection. Computers & security **21**(5), 439–448 (2002) 2

[29] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017) 2

[30] Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence **42**(4), 824–836 (2018) 2

[31] Medjahed, S.A., Saadi, T.A., Benyettou, A.: Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules. International Journal of Computer Applications **62**(1) (2013) 2

[32] Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: Indian Conference on Computer Vision, Graphics and Image Processing (Dec 2008) 5

[33] Norlander, E., Sopasakis, A.: Latent space conditioning for improved classification and anomaly detection. arXiv preprint arXiv:1911.10599 (2019) 3

[34] Pang, G., Shen, C., Cao, L., Hengel, A.v.d.: Deep learning for anomaly detection: A review. arXiv preprint arXiv:2007.02500 (2020) 3

[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011) 8

[36] Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge University Press (2011) 2

[37] Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 427–438 (2000) 7

[38] Reiss, T., Cohen, N., Bergman, L., Hoshen, Y.: Panda: Adapting pretrained features for anomaly detection and segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2806–2814 (2021) 1, 2, 3, 5, 6

[39] Reiss, T., Hoshen, Y.: Mean-shifted contrastive loss for anomaly detection. arXiv preprint arXiv:2106.03844 (2021) 1

[40] Rippel, O., Mertens, P., Merhof, D.: Modeling the distribution of normal data in pre-trained deep features for anomaly detection. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 6726–6733. IEEE (2021) 3

[41] Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., Gehler, P.: Towards total recall in industrial anomaly detection. arXiv preprint arXiv:2106.08265 (2021) 3

[42] Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M.: Deep one-class classification. In: International conference on machine learning. pp. 4393–4402. PMLR (2018) 2

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

[44] Schölkopf, B., Williamson, R.C., Smola, A., Shawe-Taylor, J., Platt, J.: Support vector method for novelty detection. Advances in neural information processing systems **12** (1999) 2

[45] Sun, Y., Ming, Y., Zhu, X., Li, Y.: Out-of-distribution detection with deep nearest neighbors. In: International Conference on Machine Learning. pp. 20827–20840. PMLR (2022) 2

[46] Tan, S.: An effective refinement strategy for knn text classifier. Expert Systems with Applications **30**(2), 290–298 (2006) 2

[47] Targ, S., Almeida, D., Lyman, K.: Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029 (2016) 1, 5, 8

[48] Tax, D.M., Duin, R.P.: Support vector data description. Machine learning **54**, 45–66 (2004) 2

[49] Tran, T.M., Vu, T.N., Vo, N.D., Nguyen, T.V., Nguyen, K.: Anomaly analysis in images and videos: A comprehensive review. ACM Computing Surveys **55**(7), 1–37 (2022) 2

[50] Winkens, J., Bunel, R., Roy, A.G., Stanforth, R., Natarajan, V., Ledsam, J.R., MacWilliams, P., Kohli, P., Karthikesalingam, A., Kohl, S., et al.: Contrastive training for improved out-of-distribution detection. arXiv preprint arXiv:2007.05566 (2020) 3

[51] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017) 5

[52] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017) 8

[53] Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14. pp. 649–666. Springer (2016) 2

[54] Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., Chen, H.: Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: International conference on learning representations (2018) 2