ReConPatch : Contrastive Patch Representation Learning for Industrial Anomaly Detection

Jeeho Hyun Sangyun Kim Giyoung Jeon Seung Hwan Kim
Kyunghoon Bae Byung Jun Kang*

LG AI Research

Abstract

Anomaly detection is crucial to the advanced identification of product defects such as incorrect parts, misaligned components, and damages in industrial manufacturing. Due to the rare observations and unknown types of defects, anomaly detection is considered to be challenging in machine learning. To overcome this difficulty, recent approaches utilize the common visual representations pre-trained from natural image datasets and distill the relevant features. However, existing approaches still have the discrepancy between the pre-trained feature and the target data, or require the input augmentation which should be carefully designed, particularly for the industrial dataset. In this paper, we introduce ReConPatch, which constructs discriminative features for anomaly detection by training a linear modulation of patch features extracted from the pre-trained model. ReConPatch employs contrastive representation learning to collect and distribute features in a way that produces a target-oriented and easily separable representation. To address the absence of labeled pairs for the contrastive learning, we utilize two similarity measures between data representations, pairwise and contextual similarities, as pseudo-labels. Our method achieves the state-of-the-art anomaly detection performance (99.72%) for the widely used and challenging MVTec AD dataset. Additionally, we achieved a state-of-the-art anomaly detection performance (95.8%) for the BTAD dataset.

1. Introduction

Anomaly detection in industrial manufacturing is key to identify the defects in products and maintain their quality. Anomalies can include incorrect parts, misaligned components, or damage to the product. Machine learning approaches for anomaly detection have been widely studied with an increasing demand for the automation in industrial applications. The main concern of these approaches is to learn how to discriminate anomalies from normal cases based on previously collected data. However, anomaly detection is particularly challenging because defects are rarely observed and unknown types of defects can occur. Such situation, in which the majority of cases are marked as normal and abnormal cases are scarce in the collected data, has lead to the improvements in one-class classification.

The key concept of one-class classification for anomaly detection is to train a model to learn a distance metric between data and detect anomalies at a large distance from the nominal data. In an effort to learn the metric, reconstruction-based approaches have been proposed to detect anomalies by measuring the reconstruction errors using auto-encoding models [8, 26, 33] or generative adversarial networks (GANs) [28, 32]. As the variety of data is not sufficiently rich to estimate a reliable nominal distribution from scratch, recent works have shown that leveraging the common visual representation, obtained from a natural image dataset [11], can result in high anomaly detection performance [3, 7]. Although pre-trained models can provide rich representations without adaptation, such representations are not sufficiently distinguishable to identify subtle defects in industrial images. The distribution shift between natural and industrial images also makes it difficult to extract anomaly-specific features. For improvements in anomaly detection performance, it is essential to train a model to learn a representation space that effectively discriminates borderline anomalies.

To alleviate the distribution shift between the pre-trained and the industrial datasets, prominent features for anomaly detection can be distilled by training a student model to reproduce the representation of the pre-trained model using a teacher supervision [5]. Attaching a normalizing flow [12] at the end of the pre-trained model is another approach to exploit the pre-trained representation and estimate the distribution of normality [30]. Unfortunately, existing methods still require extensive handcrafted input augmentation, such as random crop, random rotation, or color jitter. Particu-

*Correspondence to: bj.kang@lgresearch.ai

larly in case of industrial images, data augmentation should be carefully designed by the user expertise.

In this paper, we introduce unsupervised metric learning framework for anomaly detection by enhancing the arrangement of the features, *ReConPatch*. Contrastive learning-based training schemes present weaknesses in terms of modeling variations within nominal instances, which may increase the false-positive rate of the anomaly detection. To this end, ReConPatch utilizes the contextual similarity [18] among features obtained from the model as a pseudo-label for the training. Specifically, our method efficiently adapts feature representation by training only a simple linear transformation, as opposed to training the entire network. By doing so, we are able to learn a target-oriented feature representation which achieves higher anomaly detection accuracy without input augmentation, making our method a practical and effective solution for anomaly detection in various industrial settings.

2. Related Work

Unsupervised machine learning approaches in anomaly detection using neural networks have been widely analyzed. Deep Support Vector Data Description (SVDD) trains a neural network to map each datum to the hyperspherical embedding and detect anomalies by measuring the distance from the center of the hypersphere [31]. Patch SVDD has been developed as a patch-wise extension of Deep SVDD, utilizing the features of each spatial patch from the convolutional neural network (CNN) feature map to enhance localization and enable fine-grained examination [40]. The reconstruction-based approach assumes that normal data can be accurately reconstructed or generated by training a model using a nominal dataset, whereas abnormal data cannot. Based on this assumption, an anomaly score is calculated as the error between the original input and the reconstructed input. Auto-encoding models are used for the reconstruction model [8, 26, 33]. With the improvements in GANs, several approaches have also shown the effectiveness of GANs in anomaly detection [28, 32]. When training a model from scratch, variety and abundance should be guaranteed, which is mostly not available for anomaly detection.

To alleviate the shortage of data in anomaly detection, several attempts have been made to utilize a common visual representation pre-trained with a rich natural image dataset [11]. Previous studies that use such pre-trained model measures the distance between the representations of input data and their nearest neighbors to detect the anomalies [3] and compares hierarchical sub-image features to localize anomalies [7]. To efficiently compare the input with training set, a memory bank is introduced to store the representatives [7].

DifferNet [30] provides a normalizing flow [12] that is helpful in training a bijective mapping between the pre-trained feature distribution and the well-defined density of the nominal data, which is used to identify the anomalies. A condition normalizing flow using positional encoding is proposed by CFLOW-AD [13]. As the normalizing flow is trained to map features to the nominal distribution, this method is vulnerable to the outliers in the training dataset.

PatchCore proposes a locally aware patch feature and efficient greedy subsampling method to define the coreset [29]. The coupled-hypersphere-based feature adaptation (CFA) trains a patch descriptor that maps features onto the hypersphere, which is centered on the nearest neighbor in the memory bank [19]. PaDiM estimates a Gaussian distribution of patch features at each spatial location to detect and localize out-of-distributions (OODs) as anomalies [9]. PNI is developed to train a neural network to predict the feature distribution of each spatial location and its neighborhoods [2].

3. Method

Our proposed method, ReConPatch, focuses on learning a representation space that maps features extracted from nominal image patches to be grouped closely if they share similar nominal characteristics in an unsupervised learning manner. Although previous work [29] has shown the effectiveness of selecting representative nominal patch features using a pre-trained model, this model still presents a representation biased to the natural image data, which has a gap with the target data. The main concept of our proposed approach is to train the target-oriented features that spread out the distributions of patch features according to the variations in normal samples, and gathers the similar features.

3.1. Overall structure

As shown in Fig. 1, our framework consists of the training and the inference phases. In the training phase, we first collect the feature map at layer l, $\Phi_l(x) \in \mathbb{R}^{C \times H \times W}$ for each input x in the training data using the pre-trained CNN model. The feature maps have different spatial resolutions at the feature hierarchy of the CNN, so they are interpolated to have the same resolution before being concatenated. Patch-level features $\mathcal{P}(x, h, w) \in \mathbb{R}^{C'}$[1] then generated by aggregating the feature vectors of the neighborhood within a specific patch size s in the same approach employed in PatchCore [29]. Adaptive average pooling is used for the local aggregation.

ReConPatch utilizes two networks to train representations of the patch-level features. One of these is a network for patch-level feature representation learning, which is trained using the relaxed contrastive loss $\mathcal{L}_{RC}$ in Eq. 7. The representation network is composed of a feature rep-

[1] C and C' can be different according to the aggregation.

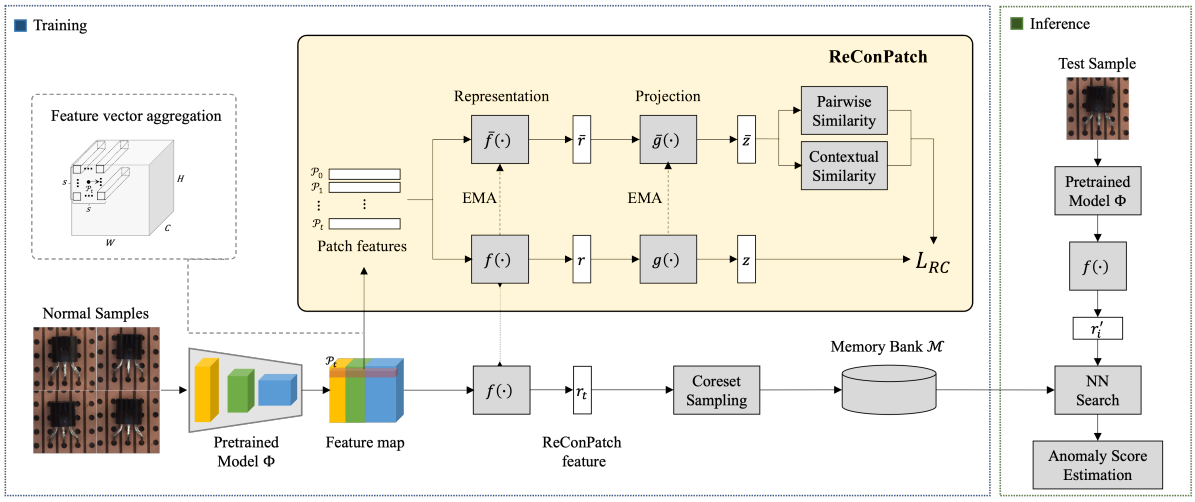

Figure 1. Overall structure of the anomaly detection using ReConPatch. ReConPatch consists of two networks to train representations of the patch-level features, which includes the feature representation layer f, $\bar{f}$ and projection layer g, $\bar{g}$ respectively. Upper networks $(\bar{f}, \bar{g})$ are used to calculate pairwise and contextual similarities between patch-level feature pairs, while the bottom networks (f, g) used for the representation learning of patch-level features is trained through relaxed contrastive loss $\mathcal{L}_{RC}$.

resentation layer f and the projection layer g respectively. When computing the $\mathcal{L}_{RC}$, pseudo-labels should be provided for every pair of features. The other network is used to calculate pairwise and contextual similarities between patch-level feature pairs. In addition, the similarity calculation network is gradually updated by an exponential moving average (EMA) of the representation network. To distinguish the two networks, the layers in the latter network is denoted as $\bar{f}$ and $\bar{g}$ respectively.

After training the representation, the patch-level features extracted from the pre-trained CNN are transformed into target-oriented features using the feature representation layer f [6]. The representative features are selected using the coreset subsampling approach based on the greedy approximation algorithm [35] and stored in a memory bank. In the inference phase, the features of a test sample are extracted using the same process as training, and the anomaly score is calculated by comparing the features with the nominal representative in the memory bank.

3.2. Patch-level feature representation learning

The objective of ReConPatch is to learn target-oriented features from patch-level features, thereby enabling more effective discrimination between normal and abnormal features. To accomplish this goal, a patch-level features representation learning approach is applied to aggregate highly similar features while repelling those with low similarity. However, such training requires labeled pairs to indicate the degree of proximity between patch-level features. To address this issue, we utilize the similarity between patch-level features using the pairwise similarity and the contextual similarities as pseudo-labels. The similarity is high,

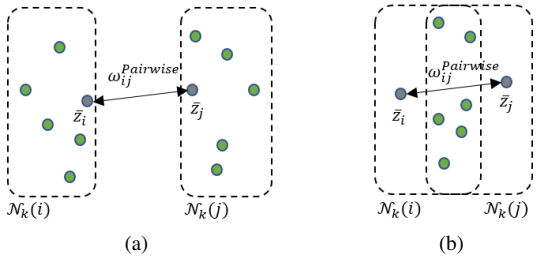

Figure 2. Illustrative examples of similarity measures in the representation space. The pairwise similarity $\omega_{ij}^{Pairwise}$ between $\bar{z}_i$ and $\bar{z}_j$ is identical in both (a) and (b). In (a), the k-nearest neighbors $\mathcal{N}_k(i)$ and $\mathcal{N}_k(j)$ do not enclose each other. Therefore, $\omega_{ij}^{Contextual}$ has a lower value, and the $\bar{z}_i$ and $\bar{z}_j$ pair should become apart. By contrast, as $\mathcal{N}_k(i)$ and $\mathcal{N}_k(j)$ enclose each other in (b) case, $\omega_{ij}^{Contextual}$ takes a higher value, so that $\bar{z}_i$ and $\bar{z}_j$ pair should attract each other.

then the pair is pseudo-labeled as positive and vice versa.

For two arbitrary patch-level features p_i and p_j obtained by $\mathcal{P}(x, h, w)$, let the projected representation be $\bar{z}_i = \bar{g}(\bar{f}(p_i))$ and $\bar{z}_j = \bar{g}(\bar{f}(p_j))$. The pairwise similarity between two features, $\omega_{ij}^{Pairwise}$, is then provided by

$$\omega_{ij}^{Pairwise} = e^{-\|\bar{z}_i - \bar{z}_j\|_2^2/\sigma} \tag{1}$$

where σ is the bandwidth of the Gaussian kernel, which can be adjusted to tune the degree of smoothing in the similarity measure [17, 18]. We note that Eq. 1 is used to measure the Gaussian kernel similarity between p_i and p_j, which is widely used to measure anomaly scores. However, the pairwise similarity is insufficient to consider the relationships

among groups of features. As depicted in Fig. 2, for example, cases (a) and (b) have the same pairwise similarity. In (a) case, $\bar{z}_i$ and $\bar{z}_j$ belong to different groups of features; therefore, they should be separated. By contrast, in (b), they belong to the same group and should be gathered.

This leads to the simultaneous measure of contextual similarity, which consider the neighborhood of an embedding vector. Let k-nearest neighborhood of the feature index i is given as a set of indices, $\mathcal{N}_k(i) = \{j | d_{ij} \leq d_{il}\}$ where l is k-th nearest neighbor and d_{ij} denotes the Euclidean distance between the two embedding vectors ($d_{ij}{=}\|\bar{z}_i{-}\bar{z}_j\|_2^2$). Two patch-level features can be regarded as contextually similar if they share more nearest neighbors in common [21]. The contextual similarity $\widetilde{\omega}_{ij}^{Context}$ between two patch-level features p_i and p_j is then defined as

$$\widetilde{\omega}_{ij}^{Contextual} = \begin{cases} \frac{|\mathcal{N}_k(i) \cap \mathcal{N}_k(j)|}{|\mathcal{N}_k(i)|}, & \text{if } j \in \mathcal{N}_k(i) \\ 0, & otherwise. \end{cases} \quad (2)$$

In addition, the approach developed in this study adopts the idea of query expansion, which is widely used to improve the information retrieval, by expanding the query to the neighbors of neighbors [18, 21]. $\widetilde{\omega}_{ij}^{Context}$ is redefined by averaging the similarities over the set of k-nearest reciprocal neighbors.

$$\mathcal{R}_k(i) = \{j | j \in \mathcal{N}_k(i) \text{ and } i \in \mathcal{N}_k(j)\} \quad (3)$$

$$\hat{\omega}_{ij}^{Contextual} = \frac{1}{\left|\mathcal{R}_{\frac{k}{2}}(i)\right|} \sum_{l \in \mathcal{R}_{\frac{k}{2}}(i)} \widetilde{\omega}_{lj}^{Contextual}. \quad (4)$$

Because $\hat{w}_{ij}^{Context}$ is asymmetric, the contextual similarity is finally defined as the average bi-directional similarity of a pair, which is given by

$$\omega_{ij}^{Contextual} = \frac{1}{2}\left(\hat{\omega}_{ij}^{Contextual} + \hat{\omega}_{ji}^{Contextual}\right). \quad (5)$$

The final similarity between two patch-level features p_i and p_j is then defined as a linear combination of two similarities with a quantity $\alpha \in [0, 1]$,

$$\omega_{ij} = \alpha \cdot \omega_{ij}^{Pairwise} + (1 - \alpha) \cdot \omega_{ij}^{Contextual}. \quad (6)$$

Patch-level features do not have explicit labels because each patch image is correlated with neighboring patches. Moreover, the goal is to obtain unique target-oriented features rather than clearly distinguishing them. Thus, relaxed contrastive loss [17] was adopted, in which inter-feature similarity is considered as pseudo-labels. Let $\delta_{ij}{=}\|z_i - z_j\|_2/(\frac{1}{N}\Sigma_{n=1}^{N}\|z_i - z_n\|_2)$ denote the relative distance between embedding vectors in a mini-batch. The relaxed contrastive loss is given by

$$\mathcal{L}_{RC}(z){=}\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}\delta_{ij}^2{+}(1{-}\omega_{ij})\max(m{-}\delta_{ij},0)^2 \quad (7)$$

where z is the embedding vectors inferred by $g(f(p))$, N is the number of instances in a mini-batch, and m is the repelling margin. ω_{ij} in Eq. 7 determines the weights of the attracting and repelling loss terms.

While representation learning networks f and g are trained with relaxed contrastive loss, the similarity calculation network $\bar{f}$ and $\bar{g}$ are slowly updated with an the EMA of the parameters in f and g respectively. Fast training of the similarity calculation network reduces the consistency of the relationships between the patch-level features, leading to unstable training. Let $\theta_{\bar{f},\bar{g}}$ be the parameters of the similarity calculation network and $\theta_{f,g}$ be the parameters of the representation learning network. $\theta_{\bar{f},\bar{g}}$ is then updated by

$$\theta_{\bar{f},\bar{g}} \leftarrow \gamma \cdot \theta_{\bar{f},\bar{g}} + (1 - \gamma) \cdot \theta_{f,g} \quad (8)$$

where γ is the hyper-parameter that adjusts the rate of momentum update.

3.3. Anomaly detection with ReConPatch

Anomaly scores are calculated in the same manner as in the case of PatchCore [29]. After training, the coreset is subsampled from the newly trained feature representation $f(\cdot)$ using the greedy approximation algorithm [35] and stored in memory bank $\mathcal{M}$. The coreset takes a role of the representative feature, which is used to compute the anomaly score. The pixel-wise anomaly score is then obtained by calculating the distance between the representation layer output, $f(p_t)$, and its nearest coreset r^* within the memory bank,

$$r^* = \underset{r \in \mathcal{M}}{\operatorname{argmin}} \mathcal{D}(f(p_t), r), \quad (9)$$

$$s_t = \left(1{-}\frac{e^{s'_t}}{\sum_{r' \in \mathcal{N}_b(r^*)} e^{\mathcal{D}(f(p_t), r')}}\right)\mathcal{D}(f(p_t), r^*), \quad (10)$$

where $\mathcal{N}_b(r^*)$ is the set of b-nearest neighbors of r^* in the memory bank. In addition, the image-wise anomaly score is computed as the maximum score over the anomaly scores calculated for every patch feature in the image.

The accuracy of anomaly detection can be further improved by score-level ensemble from multiple models. To compensate different score distribution of each model, we normalize each score using the modified z-score [1], normalization is necessary to evenly fuse the score levels of each model. The anomaly score is normalized to the modified z-score [1], defined as

$$\bar{s}_t = \frac{s_t - \widetilde{s}}{\beta \cdot MAD}, \quad (11)$$

where $\widetilde{s}$ and MAD are the median value of the anomaly scores and the Mean Absolute Deviation over the entire dataset for training. β is a constant scale factor, which is set to 1.4826 in our method, assuming the anomaly score is normally distributed.

Method	Ours-25%	Ours-10%	Ours-1%
Detection	99.24	99.27	**99.49**
Segmentation	98.01	**98.07**	**98.07**

Table 1. Ablation study results on the coreset subsampling percentage for our proposed ReConPatch model with a WideResNet-50 backbone on the MVTec AD dataset.

Dimension	1024	512	256	128	64
PatchCore	**99.1**	98.66	98.45	98.54	97.75
ReConPatch	99.49	**99.56**	99.53	99.52	99.14

Table 2. Ablation study results for the f layer dimension on the MVTec AD dataset using PatchCore [29] and proposed ReConPatch model with a WideResNet-50 backbone.

4. Experiments and analysis

4.1. Experimental setup

Dataset In this study, we used the MVTec AD [4] dataset and BTAD [25] dataset for our experiments. MVTec AD dataset is widely used as an industrial anomaly detection benchmark. It consists of 15 categories, with 3,629 training images and 1,725 test images. BTAD dataset is composed of RGB images representing three distinct industrial products. The dataset consists of 1,799 images for training and 741 images for testing. The training dataset includes only normal images, whereas the test dataset includes both normal and anomalous images. Each category in the test dataset has labels for normal and abnormal images, and anomaly ground truth mask labels for segmentation evaluation.

Metrics To evaluate the performance of our proposed model, anomaly detection and segmentation performance is compared using the area under the receiver operation characteristic (AUROC) curve metric, following [7, 9, 19, 29]. For detection performance evaluation, we measure the image-level AUROC by using the model output anomaly score and the normal/abnormal labels of the test dataset. For segmentation, we measures the pixel-level AUROC using the anomaly scores obtained from the model output for all pixels and the anomaly ground truth mask labels.

Implementation details. For the single model, ImageNet pre-trained WideResNet-50 [42] is employed as the feature extractor. The f layer output size is set to 512, and the coreset subsampling percentage is set to 1%. Our proposed ReConPatch is trained for 120 epochs per each category. Without specific instructions, hierarchy levels[2] 2 and 3 are used with a patch size of $s = 3$ to generate the patch-level features. Particularly for the segmentation evaluation in Table 5, hierarchy levels 1, 2, and 3 were used with a patch size of $s = 5$, which is identified as the best performance through the ablation study in section 4.2. In addition, for the comparison with PNI [2] using WideResNet-101, hierarchy levels 2 and 3 were used with a patch size of $s = 5$.

[2]Hierarchy levels denote residual blocks in WideResNet architecture, which is same in [29].

Metric	Detection	Segmentation
WRN-50, $s = 3$, 512 dim, layer (2+3), Imagesize 224		
AUROC	99.56	98.07
WRN-50, $s = 5$, 512 dim, layer (2+3), Imagesize 224		
AUROC	98.84	97.82
WRN-50, $s = 5$, 512 dim, layer (1+2+3), Imagesize 224		
AUROC	98.7	**98.18**

Table 3. Ablation study results with adding more hierarchy levels and larger patch size for our proposed ReConPatch model on the MVTec AD dataset.

Method	Class → / ↓ Aug. Method	Object	Texture	Average
PatchCore	w/o Aug.	99.17	98.96	99.10
	w/ Aug.	94.86	96.09	95.48
	Diff.	9.94	2.87	3.62
ReConPatch	w/o Aug.	99.44	99.81	99.56
	w/ Aug.	97.65	99.47	98.56
	Diff.	1.79	0.34	1.00

Table 4. Ablation study results for data augmentation on MVTec AD dataset using PatchCore [29] and proposed ReConPatch.

For the ensemble model, ImageNet pre-trained WideResNet-101 [42], ResNext-101 [39], and DenseNet-201 [15] are used as feature extractors for comparison with the PatchCore [29]. The f layer output size was set to 384, and we applied a coreset subsampling with percentage of 1% to all models in the ensemble. We trained ReConPatch for 60 epochs for each category. Hierarchy levels 2 and 3 were used for feature extraction in each model, and a patch size of $s = 3$ was applied to generate the patch-level features. Furthermore, to compare with PNI [2] using 480×480 image size, different parameters were applied. The f layer output size was set to 512, and a patch size of $s = 5$ was used. In this case, we trained each category for 120 epochs. ReConPatch was trained using AdamP [14] optimizer with a cosine annealing [22] scheduler. The learning rate was set to 1e-5 for a single model and 1e-6 for the ensemble model, with a weight decay of 1e-2. In the models using a 480×480 image size, the learning rate was specifically set to 1e-6. We provide the hyperparameter setup in Appendix B.

4.2. Ablation study

In this study, we aim to investigate the optimal configuration of ReConPatch through ablation studies. The first ablation was performed to determine the optimal core-

Backbone	WRN-101		WRN-50					
Image size	480×480	480×480	256×256	224×224	224×224	224×224	224×224	224×224
↓ Class\Method →	PNI [2] (w/ refine)	**Ours**	CFLOW-AD [13]	SPADE [7]	PaDiM [9]	PatchCore [29]	CFA [19]	**Ours**
Bottle	(**100**, **98.87**)	(**100**, 98.78)	(**100**, **98.76**)	(-, 98.4)	(-, 98.3)	(**100**, 98.6)	(**100**, -)	(**100**, 98.2)
Cable	(**99.76**, **99.1**)	(99.66, 98.86)	(97.59, 97.64)	(-, 97.2)	(-, 96.7)	(99.5, 98.4)	(99.8, -)	(**99.83**, **99.3**)
Capsule	(99.72, **99.34**)	(**99.76**, 99.24)	(97.68, 98.98)	(-, **99**)	(-, 98.5)	(98.1, 98.8)	(97.3, -)	(**98.8**, 97.61)
Hazelnut	(**100**, **99.37**)	(**100**, 99.07)	(99.98, 98.82)	(-, **99.1**)	(-, 98.2)	(**100**, 98.7)	(**100**, -)	(**100**, 98.94)
Metal nut	(**100**, **99.29**)	(**100**, **99.29**)	(99.26, **98.56**)	(-, 98.1)	(-, 97.2)	(**100**, 98.4)	(**100**, -)	(**100**, 95.76)
Pill	(**96.89**, **99.03**)	(96.21, 98.66)	(96.82, **98.95**)	(-, 96.5)	(-, 95.7)	(96.6, 97.4)	(**97.9**, -)	(97.49, 95.35)
Screw	(99.51, **99.6**)	(**99.84**, 99.59)	(91.89, 98.1)	(-, 98.9)	(-, 98.5)	(98.1, **99.4**)	(97.3, -)	(**98.52**, 98.79)
Toothbrush	(99.72, 99.09)	(**100**, **99.16**)	(99.65, 98.56)	(-, 97.9)	(-, 98.8)	(**100**, 98.7)	(**100**, -)	(**100**, **98.88**)
Transistor	(**100**, **98.04**)	(**100**, 96.18)	(95.21, 93.28)	(-, 94.1)	(-, 97.5)	(**100**, 96.3)	(**100**, -)	(**100**, **99.65**)
Zipper	(99.87, **99.43**)	(**99.89**, 99.25)	(98.48, 98.41)	(-, 96.5)	(-, 98.5)	(99.4, **98.8**)	(99.6, -)	(**99.76**, 98.56)
Object classes	(**99.55**, **99.12**)	(99.54, 98.81)	(97.66, 98.01)	(-, 97.57)	(-, 97.79)	(99.17, **98.35**)	(99.19, -)	(**99.44**, 98.1)
Carpet	(**100**, **99.4**)	(**100**, 99.29)	(98.73, **99.23**)	(-, 97.5)	(-, 99.1)	(98.7, 99)	(97.3, -)	(**99.6**, 98.75)
Grid	(98.41, **99.2**)	(**99.5**, 98.73)	(99.6, 96.89)	(-, 93.7)	(-, 97.3)	(98.2, 98.7)	(99.2, -)	(**100**, **99.04**)
Leather	(**100**, **99.56**)	(**100**, 99.48)	(**100**, **99.61**)	(-, 97.6)	(-, 99.2)	(**100**, 99.3)	(**100**, -)	(**100**, 96.02)
Tile	(**100**, **98.4**)	(**100**, 97.15)	(**99.88**, 97.71)	(-, 87.4)	(-, 94.1)	(98.7, 95.6)	(99.4, -)	(99.78, **98.92**)
Wood	(**99.56**, **97.04**)	(99.47, 95.16)	(99.12, 94.49)	(-, 88.5)	(-, 94.9)	(99.2, 95)	(**99.7**, -)	(99.65, **98.9**)
Texture classes	(99.59, **98.72**)	(**99.79**, 97.96)	(99.47, 97.59)	(-, 92.94)	(-, 96.92)	(98.96, 97.52)	(99.12, -)	(**99.81**, **98.33**)
Average	(99.56, **98.98**)	(**99.62**, 98.53)	(98.26, 97.87)	(85.5, 96)	(95.3, 97.5)	(99.1, 98.1)	(99.2, **98.2**)	(**99.56**, 98.18)

Table 5. Anomaly detection and segmentation performance on the MVTec AD dataset. (image-level AUROC, pixel-level AUROC)

set subsampling percentage. To this end, we compared anomaly detection and segmentation AUROC metrics using three subsampling percentages: 25%, 10%, and 1%, which were the same percentages used in PatchCore [29]. The pre-trained WideResNet-50 [42] backbone was used as the baseline for this experiment and the ourput dimension of the f layer is set to 1024. The results are presented in Table 1. We observe that the subsampling percentage of 1% provides the best performance. In addition, experiments to analyze the performance according to the feature dimension were performed by changing various output dimension of the f layer (1024, 512, 256, 128, and 64). The experiments were conducted with coreset subsampling set to 1%. The results are presented in Table 2, indicating that the highest performance was achieved with the dimension of 512. We note that even with 64 dimension, ReConPatch outperforms PatchCore with 1024, which supports the dimension reduction capability of our method.

Table 3 shows the results of an ablation study using more hierarchy levels and larger patch size on the MVTec AD [4] dataset with our proposed ReConPatch model. This study aims to improve segmentation performance by utilizing more diverse and coarse information on the patch features. The results indicates that when the patch size is increased to 5 and hierarchy levels 1, 2, and 3 are used, the segmentation performance increased up to 98.18% with small decrease in detection performance.

Real-world scenarios can present a variety of environmental conditions that can affect the quality of images. These conditions may include geometric changes, lighting changes, defocusing, and other factors that can impact the accuracy and reliability of image data. Table 4 shows that

Ensemble Backbone	WRN-101 & RNext-101 & DenseN-201			
Image size	480×480	480×480	320×320	320×320
Method	PNI [2] (w/ refine)	**Ours**	PatchCore [29]	**Ours**
Detection	99.63	**99.72**	99.6	**99.67**
Segmentation	**99.06**	98.67	98.2	**98.36**

Table 6. Comparison of ensemble model anomaly detection (image-level AUROC) and segmentation (pixel-level AUROC) performance on the MVTec AD dataset.

ReConPatch is robust to these environmental changes by learning patch-level feature representations. To simulate real-world scenarios, we randomly applied rotation, translation, color jitter (brightness and contrast), and Gaussian blur. While PatchCore's image-level AUROC decreased to 3.62 under these conditions, ReConPatch's only slightly decreased to 1.0.

4.3. Anomaly detection on MVTec AD

In this section, we evaluate the anomaly detection performance of our proposed method on the MVTec AD dataset by comparing it with previous works that used the same pre-trained model and image size [7, 9, 19, 29]. We also include the performance of concurrent methods PNI [2] and CFLOW-AD [13] in Tables 5. In case of PNI [2], a WideResNet-101 model with an image size of 480×480 was used. To improve its performance, a refinement network was included, which was trained in a supervised manner using artificially created defect dataset. For CFLOW-AD [13], a WideResNet-50 model with an image size of

Class	VT-ADL [25]	SPADE [7]	PaDiM [9]	FastFlow [41]	PyramidFlow [20]	CFA [19]	RD4AD [10]	RD++ [36]	PNI [2]	PatchCore [29]	**Ours**
1	(97.6, **99**)	(91.4, 97.3)	(99.8, 97)	(99.4, 97.1)	(**100**, 97.4)	(98.1, 95.9)	(96.3, 96.6)	(96.8, 96.2)	(-, 97.4)	(98, 96.9)	(99.7, 96.8)
2	(71, 94)	(71.4, 94.4)	(82, 96)	(82.4, 93.6)	(88.2, **97.6**)	(85.5, 96)	(86.6, 96.7)	(**90.1**, 96.4)	(-, 97)	(81.6, 95.8)	(87.7, 96.6)
3	(82.6, 77)	(99.9, 99.1)	(99.4, 98.8)	(91.1, 98.3)	(99.3, 98.1)	(99, 98.6)	(**100**, **99.7**)	(**100**, **99.7**)	(-, 99)	(99.8, 99.1)	(**100**, 99)
Avg.	(83.7, 90)	(87.6, 96.9)	(93.7, 97.3)	(91, 96.3)	(**95.8**, 97.7)	(94.2, 96.8)	(94.3, 97.7)	(95.6, 97.4)	(-, **97.8**)	(93.1, 97.3)	(**95.8**, 97.5)

Table 7. Anomaly detection and segmentation performance on the BTAD [25] dataset. (image-level AUROC, pixel-level AUROC)

256×256 is used. The evaluation results used in CFLOW-AD were the best performances obtained for each category when using the image size of 256×256.

For the single-model performance comparison, we performed the same pre-processing as described in previous work [7, 9, 19, 29]. Specifically, we resized each image to 256×256 and then center-cropped to 224×224. For the ensemble model, the same pre-processing was used as in [29], each image was resized to 366×366 and then center-cropped to 320×320. In addition, to compare with PNI [2], we resized each image to 512×512 and then center-cropped to 480×480. No data augmentation was applied to any category.

The performance of the ReConPatch in Tables 5 was obtained using 1% coreset subsampling and f layer dimensions of 512, which is determined according to Table 2. Table 5 compares the anomaly detection and segmentation performance of a single model for each category of the MVTec AD [4] dataset, evaluated with image-level AUROC. Our proposed ReConPatch achieved an image-level AUROC of 99.56%, which outperformed CFA [19] (at 99.3%). Furthermore, ReConPatch provided higher performance than the state-of-the-art PNI [2] with WideResNet-101 [42], which achieved the performance of 99.62%.

Our proposed approach focused on improving the anomaly detection performance. As a result, the segmentation performance may not be as high as its detection performance. However, we achieved a higher performance of 98.18% compared to PatchCore [29], indicating that the addition of ReConPatch feature in the f layer contributed to the improved segmentation performance.

Table 6 presents the performance of our ensemble model, which was evaluated using the modified z-score in Eq. 11 for each output from WideResNet-101 [42], ResNext-101 [39], and DenseNet-201 [15] models. Our model achieved state-of-the-art performance in anomaly detection task with AUROC of 99.72% on the MVTec AD dataset using an image size of 480×480. We note that our model still outperforms the PNI [2] using a smaller image size of 320×320, achieving an AUROC of 99.67% compared to AUROC of 99.63% Furthermore, we outperformed PatchCore [29] in terms of anomaly segmentation performance, with an improved performance of 98.36% AUROC.

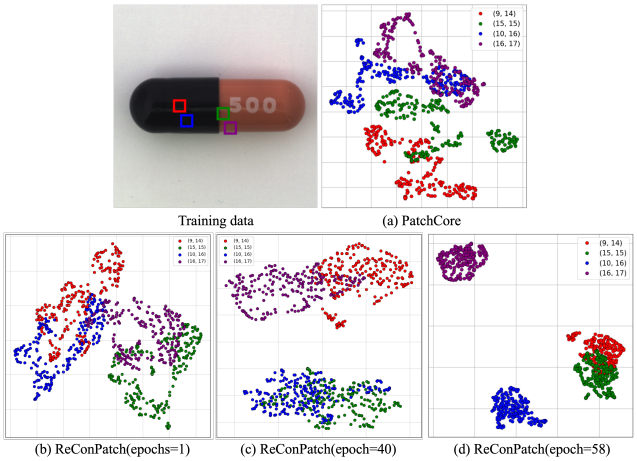

Figure 3. An illustrative comparison of features mapped by (a) PatchCore and (b) (c) (d) ReConPatch using the MVTec AD dataset. The scatter plot describes the feature space of each method, colored according to the pixel position.

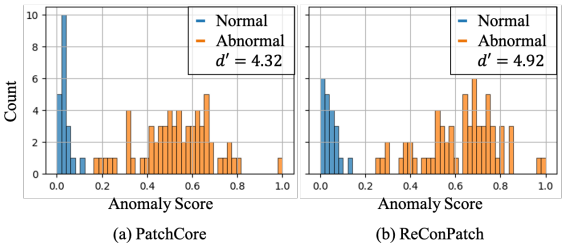

Figure 4. The histogram of the anomaly score of the normal and abnormal data for the bottle class. ReConPatch shows high discriminability, as shown in d' measure.

4.4. Anomaly detection on BTAD

To verify the capability of anomaly detection and segmentation in other dataset, we compare the performance of our model with contemporary methods using BTAD dataset [25]. For BTAD dataset, we use the pre-trained WideResNet-101 model as a feature extractor and image size of 480×480 for ReConPatch, which achieve our best performance. Table 7 shows the image-level AUROC and the pixel-level AUROC on BTAD dataset. Our model achieves a state-of-the-art performance in anomaly detection, with an AUROC of 95.8%. Furthermore, in anomaly segmentation, our model outperforms PatchCore [29] with a higher AUROC of 97.5%.

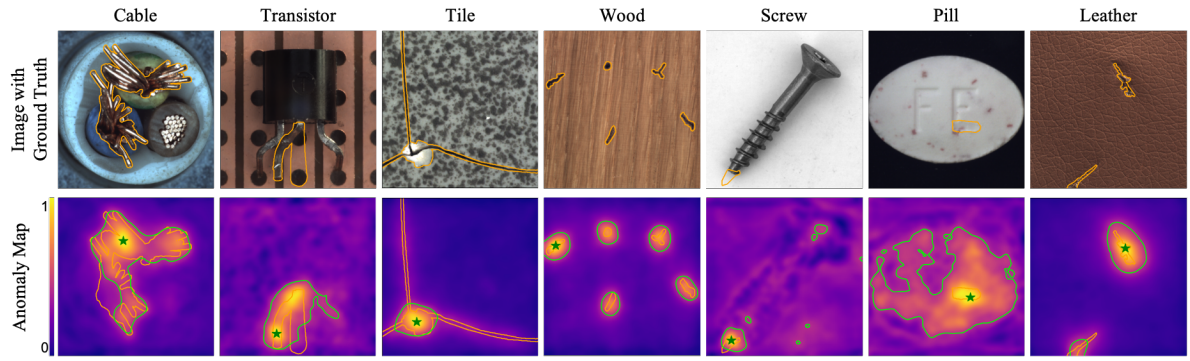

Figure 5. Examples of images with anomalies (top) and measured anomaly score maps (bottom) on MVTec AD dataset. The orange line depicts the ground truth of the anomalies and the green line depicts thresholds optimizing F1 scores of anomaly segmentation. The green star indicates the maximal location of the anomaly score in the heatmap.

4.5. Qualitative analysis

To assess the impact of ReConPatch learning on the feature space, we contrast the feature space of PatchCore and ReConPatch using the MVTec AD dataset. Our visualization, depicted in Figure 3, employs UMAP [24] for effective 2D representation of high-dimensional patch features, with color coding indicating spatial positions. The visualization attests that ReConPatch's training encourages proximity of features with similar positions. Building on findings in prior research [2, 13], which demonstrated the value of positional information, we hypothesize that ReConPatch's performance enhancement arises from implicit positional information learning. We also visualize the feature map along the training, which indicates the features are trained to map similar position to be gather.

ReConPatch's reconfigured feature space yields more distinct histogram distributions of image-level anomaly scores compared to PatchCore. In Figure 4, we observe this effect on the MVTec AD dataset's bottle class. ReConPatch compresses the score distribution for normal data while pushing the abnormal data's distribution further from the normal one, a contrast to PatchCore [29]. We gauge the distribution separability using the d' discriminability index [34] between normal and abnormal data:

$$d' = \frac{|\mu_{abnormal} - \mu_{normal}|}{\sqrt{(\sigma^2_{abnormal} + \sigma^2_{normal})/2}}. \tag{12}$$

Here, the patch features mirror those of locally aware patch features in PatchCore. ReConPatch, as detailed in Section 3.2, leverages target-oriented features through patch-level representation training, enhancing discrimination between normal and abnormal attributes. Performance-wise (Table 5), ReConPatch achieves an image-level AUROC of 99.56

We present anomaly score maps overlaid on input images (Figure 5) with ground truth annotations. Higher values in the anomaly map indicate probable anomalies. A threshold optimized via F1 scores governs the green line. Our analysis focuses on 4 superior classes (cable, transistor, tile, wood) and 3 inferior classes (metal nut, pill, leather). Despite intricate ground truth cases, ReConPatch consistently identifies anomaly locations. While inferior class anomaly maps may exhibit noise, the green star pinpointing maximal anomaly score aligns with ground truth anomalies, affirming our method's robust performance in anomaly detection.

5. Conclusion

In this paper, we introduce the ReConPatch to learn a target-oriented representation space, which can effectively distinguish the anomalies from the normal dataset. ReConPatch effectively trains the representation by applying the metric learning with softly guided by the similarity over the nominal features. Applying the contrastive learning with two similarity based pseudo soft labels, ReConPatch shows the state-of-the-art performance on the MVTec anomaly detection dataset. We also provide the anomaly detection performance on the additional BTAD dataset, where ReConPatch also achieves the best performance. We believe that ReConPatch would contribute to the improvements in anomaly detection since it shows high performance without extensive data augmentation and enables dimension reduction without significant loss of performance. Furthermore, we expect to improve the performance in the pixel-level abnormal detection by considering the correlation among the neighboring features.

References

[1] Vaibhav Aggarwal, Vaibhav Gupta, Prayag Singh, Kiran Sharma, and Neetu Sharma. Detection of spatial outlier by using improved z-score test. pages 788–790, 2019. 4

[2] Jaehyeok Bae, Jae-Han Lee, and Seyun Kim. Image anomaly detection and localization with position and neighborhood information. *arXiv preprint arXiv:2211.12634*, 2022. 2, 5, 6, 7, 8

[3] Liron Bergman, Niv Cohen, and Yedid Hoshen. Deep nearest neighbor anomaly detection. *arXiv preprint arXiv:2002.10445*, 2020. 1, 2

[4] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019. 5, 6, 7, 1, 2, 3, 4

[5] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4183–4192, 2020. 1

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 3

[7] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020. 1, 2, 5, 6, 7

[8] Diana Davletshina, Valentyn Melnychuk, Viet Tran, Hitansh Singla, Max Berrendorf, Evgeniy Faerman, Michael Fromm, and Matthias Schubert. Unsupervised anomaly detection for x-ray images. *arXiv preprint arXiv:2001.10883*, 2020. 1, 2

[9] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV*, pages 475–489. Springer, 2021. 2, 5, 6, 7, 1

[10] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9737–9746, 2022. 7

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 2

[12] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *5th International Conference on Learning Representations*, 2017. 1, 2

[13] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 98–107, 2022. 2, 6, 8

[14] Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights. *arXiv preprint arXiv:2006.08217*, 2020. 5, 1

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 5, 7

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, pages 1–1, 2019. 1

[17] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Embedding transfer with label relaxation for improved metric learning. pages 3967–3976, June 2021. 3, 4

[18] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Self-taught metric learning without labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7431–7441, 2022. 2, 3, 4

[19] Sungwook Lee, Seunghyun Lee, and Byung Cheol Song. Cfa: Coupled-hypersphere-based feature adaptation for target-oriented anomaly localization. *IEEE Access*, 10:78446–78454, 2022. 2, 5, 6, 7

[20] Jiarui Lei, Xiaobo Hu, Yue Wang, and Dong Liu. Pyramidflow: High-resolution defect contrastive localization using pyramid normalizing flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14143–14152, 2023. 7

[21] Christopher Liao, Theodoros Tsiligkaridis, and Brian Kulis. Supervised metric learning to rank for retrieval via contextual similarity optimization. *arXiv preprint arXiv:2210.01908*, 2022. 4

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5, 1

[23] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. 1

[24] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 8, 2

[25] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, pages 01–06. IEEE, 2021. 5, 7, 4, 6

[26] Duc Tam Nguyen, Zhongyu Lou, Michael Klar, and Thomas Brox. Anomaly detection with multiple-hypotheses predictions. In *International Conference on Machine Learning*, pages 4800–4809. PMLR, 2019. 1, 2

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 1

[28] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. *Advances in neural information processing systems*, 31, 2018. 1, 2

[29] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2022. 2, 4, 5, 6, 7, 8, 1, 3

[30] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1907–1916, 2021. 1, 2

[31] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018. 2

[32] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3379–3388, 2018. 1, 2

[33] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014. 1, 2

[34] Adrian J Simpson and Mike J Fitter. What is the best index of detectability? *Psychological Bulletin*, 80(6):481, 1973. 8

[35] Samarth Sinha, Han Zhang, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, and Augustus Odena. Small-GAN: Speeding up GAN training using core-sets. 119:9005–9015, 13–18 Jul 2020. 3, 4

[36] Tran Dinh Tien, Anh Tuan Nguyen, Nguyen Hoang Tran, Ta Duc Huy, Soan Duong, Chanh D Tr Nguyen, and Steven QH Truong. Revisiting reverse distillation for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24511–24520, 2023. 7

[37] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. 1

[38] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 1

[39] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 5, 7

[40] Jihun Yi and Sungroh Yoon. Patch svdd: Patch-level svdd for anomaly detection and segmentation. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[41] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows. *arXiv preprint arXiv:2111.07677*, 2021. 7

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5, 6, 7, 1, 2