SoftCore: Unsupervised Anomaly Detection with Noisy Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Although mainstream unsupervised anomaly detection (AD) algorithms perform well in academic datasets, their performance is limited in practical application due to the ideal experimental setting of clean training data. Training with noisy data is an inevitable problem in real-world anomaly detection but is seldom discussed. This paper considers label-level noise in sensory anomaly detection for the first time. To solve this problem, we proposed a memory-based unsupervised AD method, SoftCore, which efficiently denoises the data at the patch level. Noise discriminators are utilized to generate outlier scores for patch-level noise elimination before coreset construction. The scores are then stored in the memory bank to soften the anomaly detection boundary. Compared with existing methods, SoftCore maintains a strong modeling ability of normal data and alleviates the overconfidence problem in coreset. Comprehensive experiments in various noise scenes demonstrate that SoftCore outperforms the state-of-the-art AD methods on the MVTec AD benchmark, and is comparable to those methods under the setting without noise.

1 Introduction

Detecting anomalies by only nominal images without annotation is an appealing topic, especially in industrial applications where defects can be extremely tiny and hard to collect. Unsupervised anomaly detection (UAD) is proposed to solve this problem and has been largely explored. Recent methods[1; 2; 3; 4] usually model the AD problem as a one-class learning problem where a clean nomial training set is provided to extract representative nominal features. To determine whether a sample differs from the standard dataset, most previous unsupervised AD methods have to measure the distance between the test sample and the standard dataset distribution. For instance, Defard et al. [2] described the CNN features of standard data with a multivariate Gaussian distribution and learned the Gaussian parameters. Roth et al. [3] established a memory bank using the CNN features of standard data and utilized the nearest neighbor search to find the distance between the test sample and the normal data. Even though recent methods have achieved excellent performance, they all rely on the clean training set to extract nominal characteristics for later comparison with anomalous features. If the standard normal dataset is polluted with noisy data, i.e., the abnormal samples, the estimated boundary will be unreliable, and the classification for abnormal data will have low accuracy. In general, current unsupervised AD methods are not designed for and are not robust to noisy data.

However, in real-world practice, it's inevitable that there are noises that sneak into the standard normal dataset, especially for industrial manufacturing, where a large number of products are produced daily.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

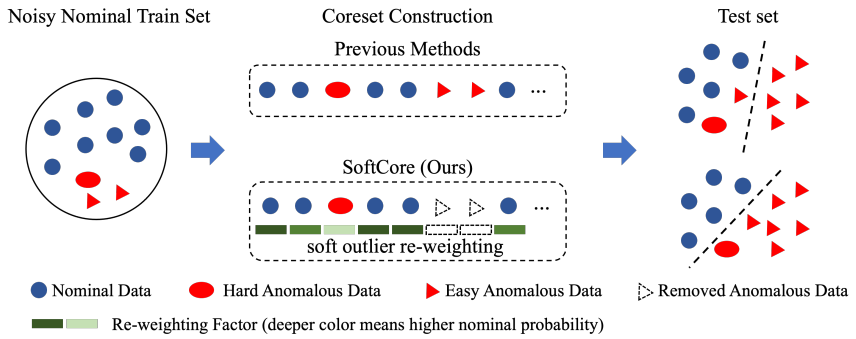

Figure 1: Illustration of SoftCore. Unlike previous methods that construct coreset without considering the negative effect of noisy data, SoftCore wipes off easy noisy data to formulate a clean training set and alleviates hard noisy data's impact by soft-reweighting.

This noise usually comes from inherent data shift or human misjudgment. Meanwhile, no existing unsupervised AD method can be directly applied to solve the problem of abnormal detection with noisy data. As in Fig. 1, noisy samples easily misinform AD algorithms, so algorithms misclassify similar anomaly samples in the test set and generate wrong locations.

In this paper, we first propose a more practical problem setting, namely the abnormal detection with noisy data, which is a more valuable scenario but seldom investigated. Our solution is inspired by one of the recent state-of-the-art methods, PatchCore [3]. PatchCore proposed a method to subsample the original CNN features of the standard normal dataset with the nearest searching and establish a smaller coreset as a memory bank. However, the coreset selection and classification process are vulnerable to polluted data. As shown in Fig. 3, the performance of PatchCore decreases sharply with the increasing noise rate. In this regard, we propose a patch-level selection strategy to wipe off the noisy image patch of noisy samples. Compared to conventional sample-level denoising, the normal image patches of a noise sample are exploited in coreset. Meanwhile, the denoising algorithm assigns an outlier score factor to each patch to be selected into coreset. Based on the patch-level denoising, we propose a novel AD algorithm with better noise robustness named SoftCore. Considering noisy samples are hard to be removed completely, SoftCore utilizes the local outlier factor to re-weight the coreset examples, which makes the coreset become a soft core. Patch-level denoising and re-weighting the coreset samples are proved effective in revising misaligned knowledge and alleviating the overconfidence in coreset in inference. Extensive experiments in various noise scenes demonstrate that SoftCore outperforms the state-of-the-art AD methods on MVTec Anomaly Detection (MVTecAD) [5] benchmark.

Our main contributions are summarized as follows: **1)** To the best of our knowledge, we are the first to study the image sensory anomaly detection with noisy data, which is a more practical setting but seldom investigated. **2)** We propose a patch-level denoising strategy for coreset memory bank, which largely improves the data usage rate compared to conventional sample-level denoising. **3)** We propose a novel SoftCore method to classify normal and abnormal samples based on the proposed denoising strategy, which assigns a outlier score factor to each patch in the coreset. SoftCore utilizes the local outlier factor to re-weight each patch in the coreset. Re-weighting the coreset samples alleviates the overconfidence problem on coreset. **4)** We set a baseline for unsupervised AD with noisy data, which outperforms most of existing unsupervised AD methods under the setting of noisy data, as well as comparable to these methods under the setting without noise.

2 Related Work

2.1 Learning with Noisy Labels

Noisy label recognition is becoming an emerging topic for supervised learning but has rarely been explored in unsupervised anomaly detection. For classification, some research [6; 7] propose to

filter noisy pseudo-labeled data with a high confidence threshold. Li et al. [8] selects noisy-labeled data with a mixture model and trains in a semi-supervised manner. Kong et al. [9] relabel harmful training samples. For object detection, multi-augmentation [10], teacher-student [11], or contrastive learning [12] are adopted to alleviate noise with the help of the expert model's knowledge. However, current noisy label recognition methods all rely on labeled data to co-rectify noisy data. In comparison, we target to improve the model's noise robustness in an unsupervised manner without introducing labor annotations.

2.2 Unsupervised Anomaly Detection

Training with agent tasks, also known as self-supervised learning, is a viable solution when there is no category and shape information of anomalies. Sheynin et al. [13] employ transformations such as horizontal flip, shift, rotation, and gray-scale change after a multi-scale generative model to enhance the representation learning. However, Li et al. [14] mention that naively applying existing self-supervised tasks is sub-optimal for detecting local defects. So they propose a novelty agent task named CutPaste, which simulates an abnormal sample by clipping a patch of a standard image and pasting it back at a random location. Similarity, DRAEM [15] synthesizes anomalies through Perlin Noise. Nevertheless, the inevitable discrepancy between the synthetic anomaly and the real anomaly disturbs the criteria of the model and limits the generalization performance. The gap between anomalies is usually larger than that between anomaly and normal. This is why AD methods deceived by some noisy samples can still work well when handling other kinds of anomalies.

Knowledge distillation is used in anomaly detection in an ingenious way. It is from a theory [16] that the representations of unusual patches are different between a pretrained teacher model and a student model, which tried its best to simulate teacher output in the training stage. Based on this theory, Salehi et al. [17] propose that considering multiple intermediate outputs in distillation and using a smaller student network lead to a better result. This improvement shows that restricting the generalization ability of the student model helps distinguish anomalies. Reverse distillation [18] uses a reverse flow which avoids the confusion caused by the same filters and prevents the propagation of anomaly perturbation to the student model. However, too long a training stage limits its usage.

Feature modeling here specifically refers to the direct estimation of the output features of the extractor, including distribution estimation [2], reversible transformation [19; 20] and storage operations [1; 3]. PaDiM [2] utilize multivariate Gaussian distributions to estimate the patch embedding of nominal data. In the inference stage, the embedding of irregular patches will be out of distribution. It is a simple but efficient method, but Gaussian distribution is inadequate for more complex data cases. So to enhance the estimation of density, DifferNet [19] and CFLOW [20] leverage the reversible normalizing flows based on multi-scale representation. Hou et al. [1] proposed that the granularity of division on feature maps is closely related to the reconstruction capability of the model for both normal and abnormal samples. So a multi-scale block-wise memory bank is embedded into an autoencoder network as a model of past data. PatchCore [3] is a more explicit but valuable memory-based method, which stores the sub-sampled patch features in the memory bank and calculates the nearest neighbor distance between the test feature and the coreset as an anomaly score. Although PatchCore is outperformance in the typical setting, it is overconfident in the training set, which leads to poor noise robustness. Except above, there are other kinds of related methods in AD, such as Pre-trained Model Adaption [21] and Image Reconstruction [4; 22].

3 The Proposed Method

3.1 Overview

Unsupervised anomaly detection aims to automatically classify and segment defect areas by only training on nominal images. During the training process, the implicit label information is utilized since we assume that all training images are nominal. In this paper, we focus on the situation where this assumption does not always hold.

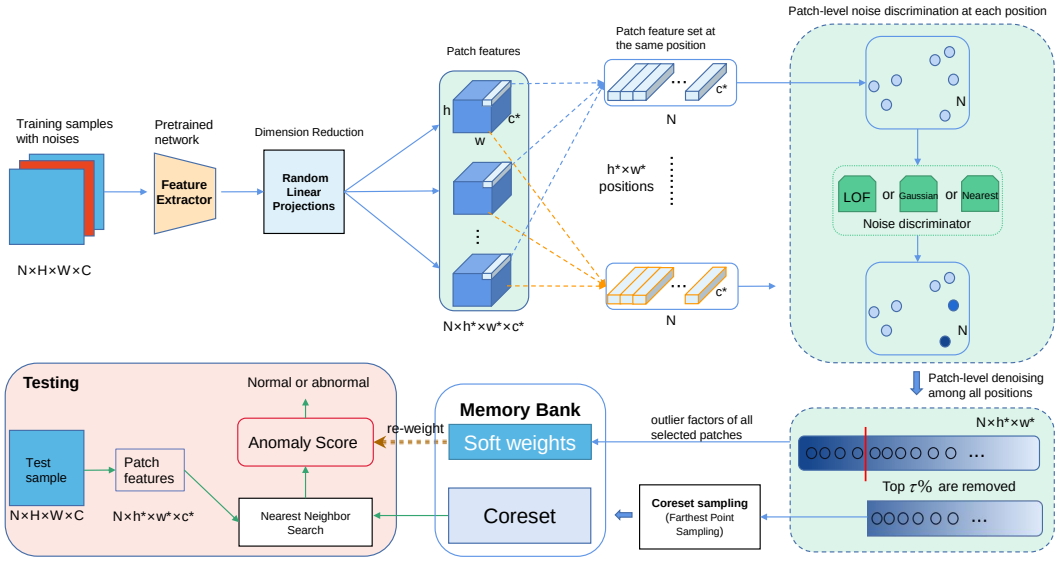

Figure 2: Overview of the proposed method. In the training phase, the noises are distinguished at patch level at each position of the feature map by a noise discriminator. The deeper color a patch node has, the higher probability that it is a noise patch. After achieving outlier scores for all patches, the top $\tau\%$ patches with the highest outlier score are removed. The coreset is a subset of remaining patches after denoising. Different from other methods, our memory bank consists of the samples in coreset and their outlier scores which are stored as soft weights. Soft weights will be further utilized to re-weight the anomaly score in inference.

The target of image-level denoising is to find $\mathcal{X}_{noise}$ from $\mathcal{X}$, where $\mathcal{X} = \{x_i : i \in (1, ..., N), x_i \in \mathbb{R}^{C \times H \times W}\}$ denotes training images (channels C, height H, width W), $\mathcal{X}_{nominal}$ and $\mathcal{X}_{noise}$ denotes nominal and noisy images in training set respectively. Patch-based unsupervised anomaly methods, such as [3], have three main processes: feature extraction, coreset selection with memory bank construction, and anomaly detection. One of the important assumptions in [3] is that the training set only contains nominal images, and the coreset should have full coverage of all training data. During the test, An incoming image will directly search in the memory bank for similar features, and the anomaly score is the dissimilarity with the nearest patches. Unfortunately, in real-world applications, noise always exists. The searching process may collapse if the assumed clean full coverage memory bank contains noise. Therefore, we propose SoftCore which filters noisy data by a noise discriminator before coreset construction and softens the searching process for down-weighting the hard unfiltered noisy samples.

Following convention in [3], we use $\phi_i \in \mathbb{R}^{c^* \times h^* \times w^*}$ as the feature map (channels c^*, height h^*, width w^*) of image $x_i \in \mathcal{X}$, $\phi_i(h, w) \in \mathbb{R}^{c^*}$ as the patch at (h, w) on the aggregated feature map with dimension c.

3.2 Noise Discriminative Coreset Selection

With increasing training images, the online memory bank can become exceedingly large and infeasible for inference and GPU/CPU storage. [3] uses min-max facility location to select coreset without considering noisy samples during selection. Therefore, we propose three noise reduction methods for coreset selection to fill in the blanks of noise unsupervised anomaly detection.

3.2.1 Nearest Neighbor

We set Nearest neighbor distance as our baseline [23] where a large distance means an outlier. With the assumption that the amount of noisy samples X_{noise} is much less than clean samples $X_{nominal}$, the neighbor is defined as the feature memory of the corresponding position of each patch instead of

the whole coreset as in PatchCore. Given a batch of images, $\phi \in \mathbb{R}^{b^* \times c^* \times h^* \times w^*}$ represents batch feature where b^* means batchsize. $\phi_i(h,w) \in \mathbb{R}^{c^*}$ is patch feature at position (h,w) on the feature map of image i with dimension c. Each patch's nearest neighbor distance $\mathcal{W}_i^{nn}$ is defined as:

$$\mathcal{W}_i^{nn}(h,w) = \min_{b \in b^*}(\phi_i(h,w) \cdot \phi_b(h,w)^T), \tag{1}$$

We first calculate patch distance, then take the minimum among batch dimensions (neighbor) and keep top τ patches sorted by W^{nn}. This method can discriminate apparent outliers but suffer from uneven distribution of different clusters, where some clusters can have large inter-distance and lead to mistakenly threshed as noisy data. To treat all clusters equally, we propose another multi-variate Gaussian method to calculate the cleanness score without the interference of different clusters' densities.

3.2.2 Multi-Variate Gaussian

With Gaussian's normalizing effect, all clean images' characteristics can be treated equally. To apply Gaussian distribution on image characteristics dynamically, we calculate the inlier probabilities on the batch dimension for each patch $\phi_i(h,w)$, similar to 3.2.1. The multi-variate Gaussian distribution $\mathbb{N}(\mu_{h,w}, \Sigma_{h,w})$ can be formulated that $\mu_{h,w}$ is the batch mean of $\phi_i(h,w)$ and sample covariance $\Sigma_{h,w}$ is:

$$\Sigma_{h,w} = \frac{1}{b^*-1}\sum_{b=1}^{b^*}(\phi_b(h,w) - \mu_{h,w})(\phi_b(h,w) - \mu_{h,w})^T) + \epsilon I, \tag{2}$$

where the regularization term ϵI makes $\sum_{h,w}$ full rank and invertible [2]. Finally, with the estimated multi-variate Gaussian distribution $\mathcal{N}(\mu_{h,w}, \sum_{h,w})$, Mahalanobis distance is calculated as the noisy magnitude $\mathcal{W}_i^{mvg}(h,w)$ of each patch:

$$\mathcal{W}_i^{mvg}(h,w) = \sqrt{(\phi_i(h,w) - \mu_{(h,w)})^T \Sigma_{(h,w)}^{-1}(\phi_i(h,w) - \mu_{(h,w)})}, \tag{3}$$

High Mahalanobis distance means high anomalous score, and we only keep top τ patches sorted by $\mathcal{W}_i^{mvg}(h,w)$. Even though Gaussian distribution normalizes and captures the essence of image characteristics, small feature clusters may be overwhelmed by large feature clusters. In the scenario of a prominent feature cluster and a small cluster in a batch, the small cluster may be out of 1-, 2- or 3-σ of calculated $\mathbb{N}(\mu_{h,w}, \sum_{h,w})$ and erroneously classified as outliers. After analyzing the above two methods, we need a method that can: 1. treat all image characteristics equally; 2. treat large and small clusters equally; 3. high dimension calculation applicable.

3.2.3 Local Outlier Factor (LOF)

LOF[24] is a local-density-based outlier detector used mainly on E-commerce for criminal activity detection. Inspired by LOF, we can solve above mentioned three questions in 3.2.2: 1. Calculating the relative density of each cluster can normalize different density clusters; 2. Using local k-distance[24] as a metric to alleviate the overwhelming effect of large clusters; 3. Modeling distance as normalized feature distance $d(\phi_i(A), \phi_i(B)) = \phi_i(h,w) * \phi_i(h,w)^T$ can be used on high dimensional patch features. Therefore, the k-distance-based absolute local reachability density $lrd_i(h,w)$ is first calculated as:

$$lrd_i(A) = 1/(\frac{\sum_{B \in \mathcal{N}_k(A)} dist_k^{reach}(\phi_i(A), \phi_i(B))}{|\mathcal{N}_k(A)|}), \tag{4}$$

$$dist_k^{reach}(\phi_i(A), \phi_i(B)) = max(dist_k(\phi_i(B)), d(\phi_i(A), \phi_i(B))), \tag{5}$$

where A/B are two locations (h,w) on feature map, $dist_k(\phi_i(B))$ is the k-distance of B and B is A's top-k nearest neighbor. With local rechability density of each patch, the overwhelming effect of large clusters is largely reduced. To normalize local density to relative density for treating all clusters equally, the relative density $\mathcal{W}_i^{LOF}$ of image i is defined below:

$$\mathcal{W}_i^{LOF}(A) = \frac{\sum_{B \in \mathcal{N}_k(A)} Ird_i(B)}{|\mathcal{N}_k(A)| \cdot Ird_i(A))}, \tag{6}$$

where $A = (h, w) : h \in h^*, w \in w^*$. $\mathcal{W}_i^{LOF}(A)$ is the relative density of the neighbors over patch's own[24], and represents as a patch's the confidence of inlier. We also keep top τ patches sorted by $\mathcal{W}_i^{LOF}$. Our experiments found that all three noise reduction methods above are helpful in data pre-selection before coreset construction, while LOF provides the best performance. However, after visualization of our cleaned training set, we found that hard noisy samples, which are similar to nominal samples, are still hidden in the dataset. To further alleviate the effect of noisy data, we propose a soft re-weighting method that can down-weight noisy samples by anomalous level.

3.3 Anomaly Detection based on SoftCore

Besides the construction of the Coreset, outlier factors of all the selected patches are stored as soft weights in the memory bank. With the denoised patch-level memory bank $\mathcal{M}$ as shown in figure 2, the image-level anomaly score $s \in \mathbb{R}$ can be calculated for a test sample $x_i \in \mathcal{X}^{test}$ by nearest neighbor searching at patch level. Denoting the collection of patch features of a test sample as $\mathcal{P}(x_i) = \mathcal{P}_{i,j}(\phi_j(x_i))$, for each patch $p_{i,j} \in \mathcal{P}_{x_i}$ the nearest neighbour searching can be formulated as the following equation:

$$m^* = \arg\min_{m \in \mathcal{M}} \|p - m\|_2 \tag{7}$$

After nearest searching, pairs of test patch and its corresponding nearest neighbor in $\mathcal{M}$ can be achieved as (p, m^*). For each patch $p_{i,j} \in \mathcal{P}_{x_i}$, the patch-level anomaly score is calculated by $s_{ij} = \mathcal{W}_{m^*_{i,j}} \|p_{i,j} - m^*_{i,j}\|_2$. The image-level anomaly score is attained by finding the largest soft weights re-weighted patch-level anomaly score:

$$s^* = \arg\max_{(p,m^*)} s_{i,j} \tag{8}$$

Different from PatchCore [3] which directly considers patches equally, SoftCore softens anomaly scores by noisy level from noise discriminater. The soft weights, i.e., local outlier factors, have considered the local relationship around the nearest node. Thus, a similar effect can be achieved as PatchCore but with more noise robustness and fewer searches. According to the image-level anomaly score, a sample is classified into a normal sample or abnormal sample.

4 Experiments

4.1 Experimental Details

Dataset. Following [3], our experiments are mainly conducted on the MVTec Anomaly Detection benchmark[5]. MVTecAD contains 15 categories with 3629 training images and 1725 test images in total. Since each category of MVTecAD is divided into nominal-only images and a test set with both nominal and anomalous samples, to create a noisy training set, we sample anomalous images randomly from the test set and mix them with the existing training images. The sampled anomalous images are removed from the test set in test time while the other images remain untouched. In this setting(*No overlap*), the injected anomalous samples will not be evaluated, which is more likely the case in the real application. However, we still construct a different setting(*Overlap*) where the injected anomalous samples are also in the test set to demonstrate the risk that defects with similar appearance will severely exacerbate the performance of an anomaly detector trained with noisy data. By controlling the proportion of negative samples being injected into the train set, we obtain several new datasets with different noise ratios dubbed MVTecAD-noise-n, where n refers to the ratio of noise.

Evaluation Metrics. We report both image-level and pixel-level AUROC for each category in MVTecAD and average them to get average image/pixel level AUROC. In order to represent noise robustness, the performance gaps between noise-free data and noisy data are also displayed.

Implementation Details. We test three SOTA AD algorithms, PatchCore [3], PaDim [2] and CFLOW [20] in noise scene and follow their main settings. In the absence of specific instructions,

Table 1: Anomaly detection performance on MVTecAD with noise. The results are evaluated on MVTecAD-noise-0.1. *Overlap* means the injected anomalous images are included in the test set. PaDiM* uses ResNet18 as the backbone. PatchCore1%-Random uses a random subsampler instead of the greedy subsampler and decreases the sampling rate from 10% to 1%. *Gap* row shows the performance gap between a noisy scene and a normal scene.

Noise=0.1	No overlap						Overlap				
Category	PaDiM	CFLOW	PatchCore	SoftCore-nearest	SoftCore-gaussian	SoftCore-lof	PaDiM*	CFLOW	PatchCore	PatchCore 1%-Random	SoftCore-lof
bottle	0.994	0.998	**1.000**	**1.000**	0.997	**1.000**	0.937	**1.000**	0.692	0.998	**1.000**
cable	0.873	0.925	0.982	0.935	0.952	**0.995**	0.680	0.916	0.756	0.920	**0.994**
capsule	0.920	0.947	**0.976**	0.916	0.662	0.963	0.796	0.945	0.783	0.779	**0.955**
carpet	**0.999**	0.961	0.996	0.995	**0.999**	0.991	0.890	0.960	0.681	0.973	**0.993**
grid	0.966	0.891	0.971	0.972	**0.997**	0.968	0.674	0.799	0.526	0.793	**0.969**
hazelnut	0.956	**1.000**	0.998	**1.000**	**1.000**	**1.000**	0.543	0.999	0.441	0.998	**1.000**
leather	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.964	0.996	0.739	**1.000**	**1.000**
metal_nut	0.987	0.959	**0.999**	0.994	0.997	**0.999**	0.820	0.957	0.765	0.969	**1.000**
pill	0.918	0.929	**0.975**	0.921	0.873	0.963	0.722	0.897	0.770	0.874	**0.955**
screw	0.838	0.784	**0.966**	0.862	0.475	0.960	0.567	0.570	0.710	0.462	**0.923**
tile	0.977	0.991	0.985	0.996	**0.997**	0.993	0.830	0.980	0.716	**1.000**	0.981
toothbrush	0.927	0.906	0.997	**1.000**	0.997	0.997	0.700	0.878	0.800	0.797	**0.994**
transistor	0.953	0.896	0.953	**1.000**	0.992	0.990	0.471	0.872	0.491	0.943	0.999
wood	0.991	0.972	0.984	0.984	**0.997**	0.987	0.831	0.954	0.579	0.980	**0.986**
zipper	0.852	0.928	**0.981**	0.976	0.979	0.978	0.679	0.931	0.792	0.950	**0.974**
Average	0.943	0.939	0.984	0.970	0.927	**0.986**	0.740	0.910	0.683	0.896	**0.982**
Gap	-0.007	-0.03	-0.008	**+0.002**	-0.001	0.0	-0.151	-0.059	-0.309	-0.015	**-0.004**

Table 2: Anomaly localization performance on MVTecAD with noise. The results are evaluated on MVTecAD-noise-0.1.

Noise=0.1	No overlap						Overlap				
Category	PaDiM	CFLOW	PatchCore	SoftCore-nearest	SoftCore-gaussian	SoftCore-lof	PaDiM*	CFLOW	PatchCore	PatchCore 1%-Random	SoftCore-lof
Average	0.972	0.969	0.956	0.971	0.977	**0.979**	0.955	0.962	0.654	0.951	**0.969**
Gap	-0.007	-0.006	-0.025	-0.008	**-0.001**	-0.002	-0.013	-0.013	-0.327	-0.021	**-0.012**

the backbone of feature extractor is *Wide-ResNet50* and the coreset sampling ratio of PatchCore and SoftCore is 0.1. For MVTecAD images, we only use 256×256 resolution and center crops them into 224×224 along with a normalization. No other data augmentation is applied since it requires prior knowledge of the class, such as whether there is rotation in this class. We train a separate model for each MVTec class. The threshold τ in SoftCore and the *LOF-K* are set to 0.15 and 6 for all noisy scenarios and classes. The effects of hyperparameters are studied in the ablation study. All our experiments are run on Nvidia V100 GPU, costing roughly one minute for each category in MVTecAD.

4.2 Anomaly Detection Performance with Noise

As indicated in Table 1 and Table 2, when 10% of anomalous samples are added to corrupt the train set, all existing methods have different extend of performance decrease, although not disastrously under *No overlap* setting. Compared to other methods, the proposed SoftCore exhibits much stronger robustness against noisy data both in terms of anomaly detection and localization, no matter which noise discriminator is used. Among three variants of SoftCore, SoftCore-lof achieves the best overall performance with the highest accuracy and strongest robustness. Interestingly, under the *Overlap* setting, PaDiM[2], CFLOW[20] and SoftCore-gaussian show significantly less performance drop than PatchCore, which indicates that modeling feature as Gaussian distribution does help denoising. While modeling feature distribution at each spatial location as a single Gaussian distribution can't handle misaligned images, such as *screw* class in MVTecAD, which explains the poor performance on these classes(see screw row). On the other hand, PatchCore's greedy-sampling strategy is a double-edged sword with higher feature space coverage and higher sensitivity to noise. That's why using random sampling in PatchCore is more robust with compromised performance(see PatchCore 1%-Random

column). SoftCore-nearest does a slightly better job in the misaligned cases. However, it doesn't take feature distribution into account, which leads to inferior performance.

In order to explore how different methods behave with the increasing noise level, experiments are further performed on MVTecAD-noise-$\{0 \sim 0.15\}$. The results are shown in Figure 3. Besides the default noise setting(*No overlap*), we also demonstrate the *Overlap* setting where the anomalous training images are included in the test set. Under the *No overlap* setting, as the noise ratio increases, PatchCore shows a pixel-level AUROC drop up to 3.7%. The performance decreases as the noise ratio rises. On the contrary, although the default performance is slightly poor than PatchCore(about 0.006 and 0 decrease in image-level and pixel-level AUROC), the proposed SoftCore-lof deteriorates much slower, which demonstrates better denoising ability. As for SoftCore-nearest and SoftCore-gaussian, they are also more robust, however, with worse base performance(see Figure 3 at noise ratio=0). The visualization of the coreset in Figure 5 also shows that random sampling avoids sampling the outlier but can not model normal adequately. Being consistent with the discussion above, under the *Overlap* setting, PatchCore's performance is getting worse and worse catastrophically(up to 40% AUROC drop in both image and pixel level) as more noises are added. This is expectable since PatchCore uses a greedy strategy for coreset sampling, which favors outlier in feature space. SoftCore-lof consistently outperforms other methods with no significant performance drop as the noise level goes up. The experimental results indicate that the risk is hidden by the fact that defects in MVTecAD have very different appearances. In this case, even if some anomalous features are added mistakenly to the coreset, they are unlikely to be retrieved during test time. However, the risk still exists and will be triggered when similar defects show up at test time. More experiment details can be found in Appendix.

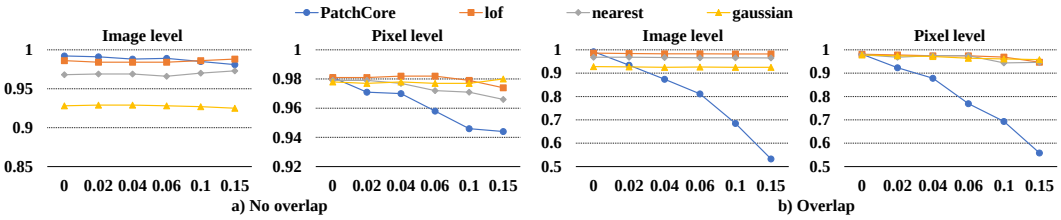

Figure 3: The comparison of anomaly detection performance under noisy training. *no overlap* means the injected anomalous images are removed from test set while *overlap* are not.

Table 3: The ablation study of soft weight. The performance scores are *Image/pixel-level AUROC* on MVTecAD.

		No overlap		Overlap	
Noise discriminator	Soft weight	Image level	Pixel level	Image level	Pixel level
None		0.985	0.946	0.685	0.693
Gaussian		0.927	0.977	0.925	0.961
Gaussian	✓	0.922	0.974	0.924	0.965
Nearest		0.970	0.971	0.966	0.944
Nearest	✓	0.972	0.978	0.968	0.958
LOF		0.985	**0.984**	**0.984**	0.963
LOF	✓	**0.986**	0.979	0.982	**0.969**

4.3 Ablation Study

4.3.1 Effectiveness of the Proposed Modules

We validated the effectiveness of two proposed modules **noise discriminator** and **soft weight** by removing them from the pipeline. As shown in Table 3, the noise discriminator significantly improves the noise robustness in terms of pixel-level AUROC. Among three decision choices of noise discriminator, LOF achieved the best balance between robustness and capacity, resulting in the most

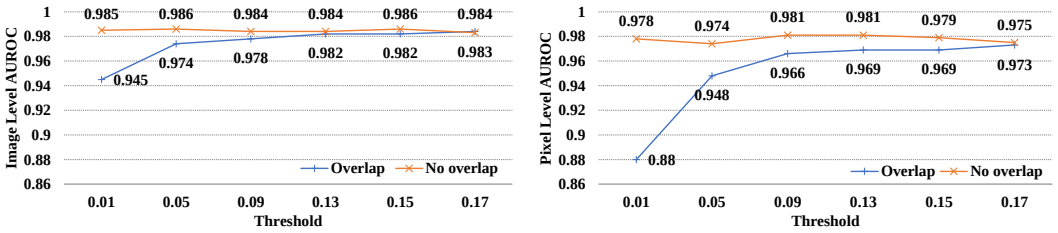

Figure 4: Performance trend with the threshold τ in Softcore-LOF. The results are evaluated on MVTecAD-noise-0.1.

performance boost under all settings. We further analyzed the intermediate results by visualizing the sampled coreset of different methods, which shows that SoftCore-LOF sampled much fewer anomalous features than the baseline(see Figure 5). Soft weight is used alongside noise discriminator to further improve the final results. We only observed minor improvement for using Soft weight in SoftCore-Nearest. We suspect that the other two kinds of noise discriminators are already robust against noise data.

Table 4: *Image/pixel-level AUROC* result for different *LOF-K* on two settings.

K	3	4	5	6	7	8	9
Overlap	**0.983**/0.955	0.982/0.951	**0.983**/0.959	0.982/**0.975**	0.981/0.973	0.982/0.968	0.980/0.968
No overlap	**0.985**/0.972	**0.985**/0.975	0.984/0.977	0.984/0.982	**0.985**/0.980	0.984/**0.983**	0.981/0.982

4.3.2 Parameter Selection

To explore the impact of two parameters (*LOF-k* and threshold τ) on the final performance, we perform parameters searching on our method. As in Table 4, our method achieves better performance when *LOF-k* is greater than 5, which suggests that our method is not sensitive to *LOF-k*, as long as it is not too small or too large. If *LOF-k* is too small, it fails to estimate the local density accurately because too few neighbors are considered. On the contrary, a large *LOF-k* may lead to undesirable cross-clusters connection that can not capture real data distribution.

Threshold τ refers to the ratio of eliminated patch features when building coreset. Figure 4 indicates an increasing trend of AUROC as threshold τ increases under *Overlap* setting, which is expected since a higher threshold means a more aggressive denoising strategy. Under *Overlap* setting, the mistakenly sampled features are the direct reason for the drastic performance drop. Therefore more aggressive denoising improves the result significantly. However, under *No Overlap* setting, the effect of the noisy feature is less prominent. Although the best *LOF-k* and threshold τ are changed according to the class and noise level, we simply use fixed values, 6 and 0.15, in all situations.

5 Conclusions

This paper emphasizes the practical value of investigating noisy data problems in unsupervised AD. Introducing a novel noisy setting on the previous task, we test the performance of existing methods and SoftCore. For existing methods, despite no adaptation to noisy settings, some of them have a slight performance decrease in some scenes. However, the performance decrease could be more significant and catastrophic for other methods or in other scenes. For the proposed SoftCore, although performance degrades slightly compared with the SOTA result in the no-noise situation. It shows consistent performance in all noise settings, which outperforms other methods.

Industrial inspection systems are an important computer vision application that requires good robustness. The noise injected into the training set break with the naive assumption that the training samples were normal. Noise also gives the model an early exposure to the distribution of anomalies. The unsupervised AD with noisy data needs more research in the future.

References

[1] Jinlei Hou, Yingying Zhang, Qiaoyong Zhong, Di Xie, Shiliang Pu, and Hong Zhou. Divide-and-assemble: Learning block-wise memory for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8791–8800, 2021.

[2] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*, pages 475–489. Springer, 2021.

[3] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. *arXiv preprint arXiv:2106.08265*, 2021.

[4] Xudong Yan, Huaidong Zhang, Xuemiao Xu, Xiaowei Hu, and Pheng-Ann Heng. Learning semantic context from normal samples for unsupervised anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3110–3118, 2021.

[5] 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.

[6] Zijian Hu, Zhengyu Yang, Xuefeng Hu, and Ram Nevatia. Simple: Similar pseudo label exploitation for semi-supervised classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15099–15108, June 2021.

[7] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608, 2020.

[8] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

[9] Shuming Kong, Yanyan Shen, and Linpeng Huang. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*, 2021.

[10] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3060–3069, 2021.

[11] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased teacher for semi-supervised object detection. *arXiv preprint arXiv:2102.09480*, 2021.

[12] Fan Yang, Kai Wu, Shuyi Zhang, Guannan Jiang, Yong Liu, Feng Zheng, Wei Zhang, Chengjie Wang, and Long Zeng. Class-aware contrastive semi-supervised learning. *arXiv preprint arXiv:2203.02261*, 2022.

[13] Shelly Sheynin, Sagie Benaim, and Lior Wolf. A hierarchical transformation-discriminating generative model for few shot anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8495–8504, October 2021.

[14] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9664–9674, 2021.

[15] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Draem-a discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8330–8339, 2021.

[16] 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.

[17] Mohammadreza Salehi, Niousha Sadjadi, Soroosh Baselizadeh, Mohammad H Rohban, and Hamid R Rabiee. Multiresolution knowledge distillation for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14902–14912, 2021.

[18] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. *arXiv preprint arXiv:2201.10703*, 2022.

[19] 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.

[20] 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.

[21] Tal Reiss, Niv Cohen, Liron Bergman, and Yedid Hoshen. Panda: Adapting pretrained features for anomaly detection and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2806–2814, 2021.

[22] Kang Zhou, Yuting Xiao, Jianlong Yang, Jun Cheng, Wen Liu, Weixin Luo, Zaiwang Gu, Jiang Liu, and Shenghua Gao. Encoding structure-texture relation with p-net for anomaly detection in retinal images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 360–377. Springer, 2020.

[23] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[24] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In the Experiments and Conclusions sections, we state that our method improve robustness at cost of slight performance decrease for some categories.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See 4.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]