

Domain-independent detection of known anomalies

Jonas Bühler

Karlsruhe Institute of Technology (KIT)

& preML GmbH

<https://preml.io/>

jonas.buehler@preml.io

Jonas Fehrenbach

preML GmbH

<https://preml.io/>

jonas.fehrenbach@preml.io

Lucas Steinmann

preML GmbH

<https://preml.io/>

lucas.steinmann@preml.io

Christian Nauck

preML GmbH

<https://preml.io/>

christian.nauck@preml.io

Marios Koulakis

Karlsruhe Institute of Technology (KIT)

<https://www.kit.edu/>

Abstract

One persistent obstacle in industrial quality inspection is the detection of anomalies. In real-world use cases, two problems must be addressed: anomalous data is sparse and the same types of anomalies need to be detected on previously unseen objects. Current anomaly detection approaches can be trained with sparse nominal data, whereas domain generalization approaches enable detecting objects in previously unseen domains. Utilizing those two observations, we introduce the hybrid task of domain generalization on sparse classes. To introduce an accompanying dataset for this task, we present a modification of the well-established MVTec AD dataset by generating three new datasets. In addition to applying existing methods for benchmark, we design two embedding-based approaches, Spatial Embedding MLP (SEMLP) and Labeled PatchCore. Overall, SEMLP achieves the best performance with an average image-level AUROC of 87.2 % vs. 80.4 % by MIRO. The new and openly available datasets allow for further research to improve industrial anomaly detection.

1. Introduction

In industrial environments, machine-made components are susceptible to flaws and necessitate inspection before shipping. Typically, this task is performed manually by human operators who visually inspect each component. However, as with any repetitive task, human fatigue can lead to errors and the potential oversight of faulty components. Additionally, humans tend to judge anomalies inconsistently, with some being more stringent than others. Consequently, there is a growing interest in applying computer vision models to

automate this process in a cost-effective manner while also reducing errors.

However, applying conventional classification approaches to this task introduces new challenges. First, anomalies may be rare, making it difficult to collect a sufficient number of images for training. Second, not all possible types of anomalies may be known a priori, requiring models to be able to detect previously unseen anomaly types akin to humans capabilities. To address these challenges, anomaly detection methods were developed that can be trained effectively with limited training data. The advancement of robust anomaly detection models remains a subject of active research interest. For a comprehensive overview of industrial image anomaly detection, readers are directed to the reviews: [10, 20, 24].

Existing approaches have demonstrated the capacity to achieve near-perfect results on existing benchmark datasets [16, 22, 29]. However, they are limited to detect anomalies, but cannot predict the type of the anomaly. In certain applications, there may be a necessity to differentiate between various types of anomalies in order to manage them accordingly. For instance, while some anomalies may be severe, others may be acceptable for a particular use case, such as color stains or flipped objects.

Another issue is that in certain industrial contexts, a company may produce a multitude of different objects [20]. Collecting sufficient data for all potential objects, even if it is merely normal images, would require a lot of time and resources. It would be beneficial if one could reuse a trained anomaly detection model for new objects. In addition to from the work presented in Section 2, which have their own disadvantages, it is unclear whether existing anomaly detection models can be applied to previously unseen domains without retraining them. For future applications, models

should work reliably on objects that are not known a priori. Therefore, the first problem to solve is the lack of industrial anomaly detection datasets that cover multiple domains [20].

Our main contributions are:

- We propose the hybrid task of detecting known anomalies across different, previously unseen objects.
- We create three new datasets for this task, which are based on the popular MVTec Anomaly Detection dataset [3, 4].
- We present two new methods which provide benchmarks on the task on the modified MVTec datasets: Labeled PatchCore and Spatial Embedding MLP (SEMLP).

The paper is organized as follows: We commence with a review of related work (Section 2). Subsequently, we detail the process of dataset generation and introduce the new methods, Labeled PatchCore and SEMLP (Section 3). Following this, Section 4 specifies the conducted experiments and presents the results. Finally, we draw conclusions (Section 5).

2. Related Work

Recently, many approaches were presented to solve the problem of anomaly detection. Furthermore, several domain generalization approaches tackle the problem of classifying previously unseen objects. We will give a quick overview over the recent approaches. Furthermore, due to the lack of appropriate datasets for our tasks, we will also briefly describe existing datasets.

2.1. Anomaly Detection

There are many different approaches to solve visual anomaly detection tasks on a single object using only normal images. We will present the two most common types of anomaly detection approaches in the first two paragraphs, representation-based and reconstruction-based ones, followed by some more exotic approaches.

Representation-based approaches use a backbone model like ResNet [14] or one of its many extensions (e.g. [27, 30]) to create embeddings for single patches of the input image by extracting the outputs of intermediate layers. These patches can be used to learn a representation of normal patches. For example PaDiM [11] estimates their distribution and reports out-of-distribution test embeddings as anomalous. PatchCore [22] creates a coresnet which stores the embeddings of normal patches and uses the distance between test embeddings and the coresnet as anomaly score.

Reconstruction-based approaches train a model to reconstruct normal images, e.g. a (variational) autoencoder or generative adversarial network [5, 23]. Since anomalies are not present in the training data, it can be assumed, that these models fail to reconstruct anomalous areas. Hence, a pixel-based reconstruction error measure can be used as an anomaly score.

Another approach, that does not fit into the categories above, is [17], which creates artificial anomalies to train a CNN as binary classifier, or few-shot anomaly detection approaches.

Due to their simplicity, we will focus on representation-based approaches, especially PatchCore, which reports great results in the literature.

2.2. Domain Generalization

Inspired by the fact that the domain of the dataset does not necessarily match the domains during the real-world usage, many papers have explored the problem of domain generalization [31], where the domains of the training and test dataset differ. One common scenario is an autonomous car which must be able to detect and classify objects under all possible combinations of weather and environment conditions like illumination, fog, rain and snow [9, 26].

Cha et al. [7] proposes an approach called MIRO which applies a regularization loss while training a feature extractor. It uses a pre-trained model that has already seen many different types of images and applies an advanced fine-tuning. We will use this approach as one of the baselines.

A first step towards domain generalization on the MVTec dataset is shown by Chen et al. [8]. It uses normal and anomalous images for training. But it has some limitations such as requiring normal images from the target domain during test time and only working on texture categories.

Huang et al. [15] also explores anomaly detection in a domain generalization setting: While many approaches train their models for only a single category at a time, this approach trains a category-agnostic model with a few images from multiple categories. It can detect anomalies on new categories without requiring any retraining. However, it still needs some normal images from the new category during testing.

2.3. Dataset

The MVTec Anomaly Detection dataset [3, 4] introduced in 2019 is widely used in anomaly detection papers [11, 17, 19, 22, 28, 29] and quickly became a common benchmark dataset. It contains 15 different categories of which 10 are single objects and 5 textured surfaces. It covers dozens of different anomaly types for which it provides pixel-wise segmentation maps. Newer approaches [22, 29] already report nearly perfect results for classification and segmentation on it.

A dataset used for domain generalization is PACS [18], whose name is based on its 4 domains: photos, art paintings, cartoons, and sketches. It contains 7 different categories and a total of 9991 images.

Cao et al. [6] recently tried to adapt different datasets for anomaly detection with out-of-distribution test data. The

PACS dataset is adapted for the anomaly detection part by defining one class as normal and all others as anomalous. The same is done for the MNIST dataset, where the images from the MNIST-M dataset are used as an additional domain. As for the MVTec dataset, new domains are created by augmenting the images.

The new adapted datasets are a good first step towards domain generalization in anomaly detection, but as for the PACS and MNIST adaption, they use completely different objects as anomalies whereas anomalies are usually only small flaws on the normal object. The anomalous images of the MVTec adaption are out-of-distribution as advertised, but consist of only slight augmentations of the same object. None of these datasets allow to test the detection of small flaws on completely different objects.

3. Methods

First, we create our new datasets by modifying the MVTec AD dataset. Then, we introduce two new approaches as baselines: i) we extend PatchCore by introducing multiple Corests, which we refer to as Labeled PatchCore, and ii) we apply a regular MLP to the embeddings used by PatchCore (Spatial Embedding MLP). The method is called spatial embedding, because the embeddings are extracted from vision ML models such as Transformers and CNNs where the embeddings contain spatially related information in contrast to MLPs.

3.1. Generation of new datasets

For our task, we want to propose a dataset which contains multiple objects with similar anomalies. This would allow us to treat each object type (category) as another domain, so that we can train the detection of a specific anomaly on some of the categories and test it on previously unseen categories. As shown in Section 2.3, none of the existing datasets satisfies our requirements. Therefore, three new datasets for domain generalized anomaly detection are created based on the MVTec AD dataset [3, 4].

For the first step, we look at all anomalies from all categories and determine which anomalies look similar. For example, the five categories (*cable*, *carpet*, *hazelnut*, *leather* & *wood*) have anomaly types called either *hole* or *poke*, which we deem similar looking.

In the next step, we create a dataset based on the categories containing these similar anomaly types. We use their good images and the anomalous images from said anomaly type. In our example, we would use the normal images from the previously mentioned categories (*cable*, *carpet*, *hazelnut*, *leather* & *wood*) as well as the *hole* or *poke* anomaly images from these objects. We use all those images to create a new dataset called *hole*.

The same procedure is applied to create two more datasets, that we refer to as *cut* and *color*. Hence, we gener-

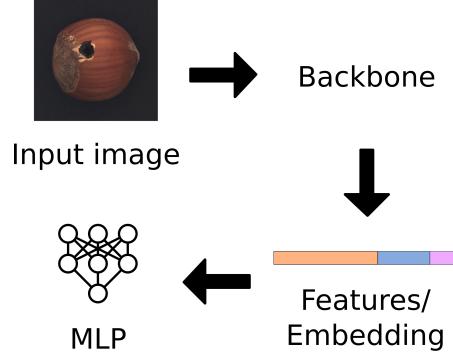


Figure 1. SEMLP passes the input image to a backbone. The intermediate features are extracted and concatenated to create embeddings. Each embedding is passed to an MLP which will classify single embeddings as normal or anomalous. The image is considered anomalous if at least one embedding is classified as anomalous.

ate the three datasets: *hole*, *cut* and *color*. Details regarding the exact object and corresponding anomaly types are listed for all three datasets in Table 1. To illustrate the objects and anomalies, example images are provided in the appendix (Figure 3).

3.2. Extension of PatchCore: Labeled PatchCore

Since we now have anomalous images available from the training domains, we aim to extend the idea of PatchCore [22] to improve classification by correctly identifying the type of the anomaly. PatchCore collects all good embeddings in a coresnet. We leverage the additional information from anomalous images by creating a second coresnet for the embeddings from anomalous image parts (i.e. an anomaly coresnet). The second coresnet can be used to improve the results. More details can be found in Section A.1.

3.3. Simplified Approach: Spatial Embedding MLP

Instead of explicitly providing structure to the classification process by measuring the distances between the embeddings, we now analyze the capabilities of MLPs to directly classify the embeddings. For this approach, we extract embeddings, just like described above for the Labeled PatchCore. The embeddings are used as inputs to train small MLPs (about 49k trainable parameters), which are supposed to classify all single embeddings as good or anomalous. We call this approach Spatial Embedding MLP (SEMLP). Figure 1 visualizes the classification flow.

Training on patches gives us the advantage of having a lot more training data: While some anomaly classes contain less than a dozen images, we get a multitude of input data by training on patch embeddings. Even classes with only a few images have now about one hundred embeddings, some even over a thousand.

Table 1. This table shows how anomaly types across different categories are merged into one dataset. Each category is treated as its own domain. For example the categories *cable*, *carpet*, *hazelnut*, *leather* & *wood* have similar looking anomaly types called *hole* or *poke* which are used to create a new dataset called *hole* (left column).

<i>hole</i> (Figure 3a)		<i>cut</i> (Figure 3b)		<i>color</i> (Figure 3c)	
category	anomaly type	category	anomaly type	category	anomaly type
<i>cable</i>	<i>poke_insulation</i>	<i>cable</i>	<i>cut_inner_insulation</i>	<i>carpet</i>	<i>color</i>
<i>carpet</i>	<i>hole</i>	<i>cable</i>	<i>cut_outer_insulation</i>	<i>hazelnut</i>	<i>print</i>
<i>hazelnut</i>	<i>hole</i>	<i>carpet</i>	<i>cut</i>	<i>leather</i>	<i>color</i>
<i>leather</i>	<i>poke</i>	<i>hazelnut</i>	<i>cut</i>	<i>metal_nut</i>	<i>color</i>
<i>wood</i>	<i>hole</i>	<i>leather</i>	<i>cut</i>	<i>pill</i>	<i>color</i>
		<i>tile</i>	<i>crack</i>	<i>tile</i>	<i>gray_stroke</i>
				<i>tile</i>	<i>oil</i>
				<i>wood</i>	<i>color</i>

4. Experiments

4.1. Setup

To evaluate our proposed models as well as the baselines at detecting anomalies across different categories, we always choose one category from the dataset as target domain and use the other ones as source domains for training. The anomalous images of the source domains are split evenly into training and validation set, whereas we use all anomalous images from the target domain for testing.

As backbone models for SEMLP, we use Wide-ResNet50-2 and Vision Transformers (ViT). Details can be found in Section A.2.1. We compare the methods to PatchCore (cf. Section A.2.2 and MIRO [7]). For MIRO, we continue training on the pre-trained Wide-ResNet50-2 and ViT respectively. We report those results as plain "MIRO". Furthermore, we use the adjusted models as backbones to extract embeddings for SEMLP and PatchCore (which we report as "SEMLP (MIRO)" and "PatchCore (MIRO)").

4.2. Results

To evaluate the performance of the tested models and allow for comparison runs, we provide the resulting image-level AUROC for Wide-Resnet50-2 and Vision Transformer as backbones in Table 2. More detailed results for the image-level AUROC and F1-Score can be found in the appendix in Tables 3 to 5. As we can see, SEMLP achieves an image-level AUROC of 87.2 % with the WideResNet backbone and is already nearly perfect on many categories. Using the trained models from MIRO as backbones for SEMLP does not work better than regular backbones pretrained on ImageNet.

Even though SEMLP achieves good average performances, SEMLP fails pretty badly on some categories (like the *pill* in the *color* dataset or the *cable* in the *hole* dataset), even though other methods perform well. Future work can

identify the reasons, potential limitations and provide improvement strategies to increase the generalization performance. Plain MIRO seems to be the second best approach, followed by plain PatchCore, which is better than our Labeled PatchCore approach. Providing the additional structure in our way is not helpful to improve the performance of PatchCore. It would be interesting to study if other modifications can further increase the performance of PatchCore.

Furthermore, the anomaly scores of some target domains are significantly higher than for the source domains. But since the anomalous images usually have an even higher score, there still exists a good threshold in most cases.

Even though our approaches require no data from the target domain to train the models, one limitation of our method is the need for normal and anomalous images from the target domain to determine a good threshold.

5. Conclusion

We presented a new task on domain generalization of sparse anomalous classes. To conduct this task, we introduced three new datasets based on the MVTec AD dataset to compare different anomaly detection models on domain generalization tasks.

We presented first baselines on these datasets, covering some existing models and introducing two new approaches. One of this is SEMLP which classifies single embeddings and requires only a few labeled training images. The new model can distinguish normal and anomalous data in most cases and achieves an image AUROC of 87.2 %, which is better than MIRO. Notably SEMLP outperforms PatchCore and also Labeled PatchCore. Apparently providing additional structure by introducing the labeled coresets hinders prediction and it is beneficial to purely rely on the predictive performance of MLPs.

Even though our approach does not require large amounts of data, we still need some labeled images of the

Table 2. Image-level AUROC in % for all datasets with different approaches using Wide-Resnet50-2 and a Vision Transformer. (Note: the average is calculated across all target domains, as shown in 4 and 5.)

	Wide-ResNet50-2				ViT-B/8			
	Cut	Color	Hole	Avg.	Cut	Color	Hole	Avg.
SEMLP	92.8 ± 2.5	82.9 ± 2.1	87.6 ± 2.5	87.2	85.7 ± 3.9	85.7 ± 1.4	89.0 ± 3.2	86.7
PatchCore	71.1 ± 3.0	66.4 ± 10.8	79.8 ± 12.1	71.7	67.6 ± 4.8	57.7 ± 8.9	64.0 ± 4.6	62.5
Labeled PatchCore	47.3 ± 2.3	48.4 ± 3.9	46.2 ± 2.2	47.4	47.7 ± 2.1	56.8 ± 2.0	48.7 ± 1.5	51.7
MIRO	87.0 ± 2.8	71.5 ± 4.2	86.9 ± 2.3	80.4	75.9 ± 8.2	75.4 ± 5.1	79.1 ± 4.3	76.6
SEMLP (MIRO)	92.0 ± 3.1	81.1 ± 0.8	87.6 ± 1.8	86.2	77.4 ± 5.8	80.0 ± 5.1	79.3 ± 7.4	79.0
PatchCore (MIRO)	67.6 ± 0.5	69.5 ± 3.0	73.9 ± 4.4	70.2	68.1 ± 1.9	80.9 ± 7.5	71.7 ± 4.0	74.4

new domain to determine a good threshold. Although a threshold can be easily set by the user, future work may focus on eliminating this need by finding a good threshold. Nevertheless, SEMLP shows potential for real-world applications. Especially the opportunity of detecting known anomalies can be advantageous for industrial applications.

References

- [1] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training, 2018. arXiv:1805.06725 [cs]. 8
- [2] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomolib: A Deep Learning Library for Anomaly Detection, 2022. arXiv:2202.08341 [cs]. 8
- [3] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592, 2019. ISSN: 2575-7075. 2, 3, 9
- [4] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. *International Journal of Computer Vision*, 129(4): 1038–1059, 2021. 2, 3, 9
- [5] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders. pages 372–380, 2024. 2
- [6] Tri Cao, Jiawen Zhu, and Guansong Pang. Anomaly Detection under Distribution Shift, 2023. arXiv:2303.13845 [cs]. 2
- [7] Junbum Cha, Kyungjae Lee, Sungrae Park, and Sanghyuk Chun. Domain Generalization by Mutual-Information Regularization with Pre-trained Models, 2022. arXiv:2203.10789 [cs]. 2, 4
- [8] Shang-Fu Chen, Yu-Min Liu, Chia-Ching Lin, Trista Pei-Chun Chen, and Yu-Chiang Frank Wang.
- Domain-Generalized Textured Surface Anomaly Detection, 2022. arXiv:2203.12304 [cs]. 2
- [9] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain Adaptive Faster R-CNN for Object Detection in the Wild, 2018. arXiv:1803.03243 [cs]. 2
- [10] Yajie Cui, Zhaoxiang Liu, and Shiguo Lian. A Survey on Unsupervised Anomaly Detection Algorithms for Industrial Images. *IEEE Access*, 11:55297–55315, 2023. arXiv:2204.11161 [cs]. 1
- [11] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization, 2020. arXiv:2011.08785 [cs]. 2, 8
- [12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021. arXiv:2010.11929 [cs]. 7
- [13] Falcon, William. pytorch-lightning <https://github.com/Lightning-AI/pytorch-lightning/>. 8
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. ISSN: 1063-6919. 2
- [15] Chaoqin Huang, Haoyan Guan, Aofan Jiang, Ya Zhang, Michael Spratling, and Yan-Feng Wang. Registration based Few-Shot Anomaly Detection, 2022. arXiv:2207.07361 [cs]. 2
- [16] Jeeho Hyun, Sangyun Kim, Giyoung Jeon, Seung Hwan Kim, Kyunghoon Bae, and Byung Jun Kang. ReConPatch : Contrastive Patch Representation Learning for Industrial Anomaly Detection, 2024. arXiv:2305.16713 [cs]. 1
- [17] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. CutPaste: Self-Supervised Learn-

- ing for Anomaly Detection and Localization, 2021. arXiv:2104.04015 [cs]. 2
- [18] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, Broader and Artier Domain Generalization, 2017. arXiv:1710.03077 [cs]. 2
- [19] Ning Li, Kaitao Jiang, Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Yihong Gong. Anomaly Detection via Self-organizing Map, 2021. arXiv:2107.09903 [cs]. 2
- [20] Jiaqi Liu, Guoyang Xie, Jinbao Wang, Shangnian Li, Chengjie Wang, Feng Zheng, and Yaochu Jin. Deep Industrial Image Anomaly Detection: A Survey. *Machine Intelligence Research*, 21(1):104–135, 2024. arXiv:2301.11514 [cs]. 1, 2
- [21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary 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. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 8
- [22] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards Total Recall in Industrial Anomaly Detection, 2022. arXiv:2106.08265 [cs]. 1, 2, 3, 7, 8
- [23] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery, 2017. arXiv:1703.05921 [cs]. 2
- [24] Xian Tao, Xinyi Gong, Xin Zhang, Shaohua Yan, and Chandranath Adak. Deep Learning for Unsupervised Anomaly Localization in Industrial Images: A Survey. *IEEE Transactions on Instrumentation and Measurement*, 71:1–21, 2022. Conference Name: IEEE Transactions on Instrumentation and Measurement. 1
- [25] Ross Wightman. PyTorch image models, <https://github.com/rwightman/pytorch-image-models>, 2019. 7, 8
- [26] Aming Wu, Rui Liu, Yahong Han, Linchao Zhu, and Yi Yang. Vector-Decomposed Disentanglement for Domain-Invariant Object Detection, 2021. arXiv:2108.06685 [cs]. 2
- [27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks, 2017. arXiv:1611.05431 [cs]. 2
- [28] Jihun Yi and Sungroh Yoon. Patch SVDD: Patch-level SVDD for Anomaly Detection and Segmentation, 2020. arXiv:2006.16067 [cs]. 2
- [29] 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, 2021. arXiv:2111.07677 [cs]. 1, 2, 8
- [30] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks, 2017. arXiv:1605.07146 [cs]. 2
- [31] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain Generalization: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2022. arXiv:2103.02503 [cs]. 2

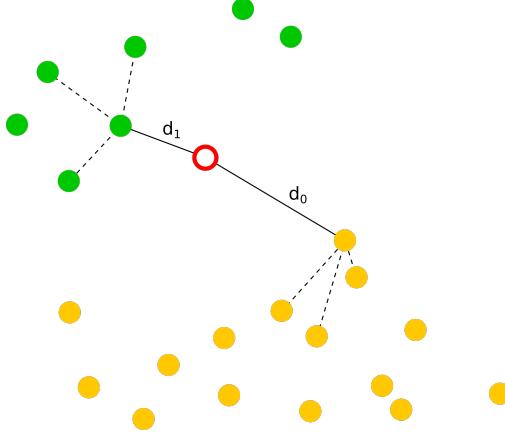


Figure 2. Classification using multiple coresets in a two-dimensional scenario: yellow is the coreset containing embeddings of good images and green the coreset with anomalous embeddings. The red circle will be classified based on its distance to the next embedding of each coreset (d_0 and d_1), after weighting the distances to the next neighbors in their respective coreset like in Roth et al. [22] (dotted lines).

A. Appendix

A.1. Labeled PatchCore

This section provides a more in-depth description of our Labeled PatchCore approach.

Since anomalous images also contain normal areas, we will only consider patches of the anomaly itself for our anomaly coreset. Depending on the original resolution and chosen backbone, a single patch contains roughly 1024 (32x32) pixels of the original image. Therefore, we consider patches anomalous, if at least a tenth of the pixels are anomalous. This way, we obtain a sufficient share of anomalous patches for our anomaly coreset, without considering patches anomalous that just have one or two anomalous pixels. The good patches from anomalous images are discarded because there are already enough *good* patches from the normal images.

Then, we add the embeddings of these patches (also called locally aware patch features in [22]) to the memory bank of the anomaly coreset. Because of the limited number of anomalous images and patches, we always keep at least 1,000 embeddings for the anomaly coreset when subsampling or use all embeddings if necessary, regardless of the given coreset subsampling ratio.

To classify an image, we extract the embeddings for each patch. Then, we classify each embedding based on whether they are closer to the normal or anomalous coreset. We hope that the second coreset provides more information and a better structure to differentiate the anomalies from normal patches.

To do so, we calculate the weighted anomaly score for

each embedding like the original implementation (see section 3.3 in Roth et al. [22]) but in relation to each of the two coresets (as shown in Figure 2 for a two-dimensional example). In other words, each embedding is classified by calculating the distance d_i to the next embedding m_i^* for each coreset M_i . The distances are weighted with the nearest neighbors of m_i^* in its coreset to get score s_i (see Equation 7 in [22]). Afterwards, we classify each patch as belonging to the coreset M_i with the smallest score s_i , i.e. the coreset to which the embedding is closest after weighting.

The process is repeated for all patches of the images. If all patches are classified as *good*, the image is deemed as normal. Otherwise it is considered anomalous.

A.2. Details for experimental setup

As in Roth et al. [22], we also resize the images to 256x256 and center crop them to 226x226 pixels and we use a neighbourhood size of 3 (i. e. apply an 3x3-average pooling across the patch features before creating the embeddings).

A.2.1 Backbone models

Similar to Roth et al. [22], we use Wide-ResNet50-2 as backbone for all our tasks and use the features of the second and third block to create our embeddings. Since Vision Transformers (ViT) have recently gained popularity in computer vision, we also test a ViT as backbone. In this case, we use the base variant as presented in Dosovitskiy et al. [12] with a patch size of 8. Here, we extract the features after the layers 5 and 9. We use pre-trained weights provided by Wightman [25] for all backbones.

A.2.2 Details on the application of the specific approaches

Unless otherwise mentioned, the SEMLP has two fully-connected layers with a hidden size of 32 and a leaky ReLU layer (with $\alpha = 0.01$) in-between and predicts the scalar anomaly score, which leads to relatively small MLPs with about 49k trainable parameters.

For comparison, we use the default PatchCore implementation, which only uses normal images to create a single coreset. Because we now have a multitude of training data, we reduce the coreset subsampling rate for default PatchCore and Labeled PatchCore to 0.1 divided by the number of different categories in the current dataset (i.e. $\frac{0.1}{5}$ for *hole* and *cut*, and $\frac{0.1}{7}$ for *color*) due to memory constraints. For this reason, we also implement an online subsampling algorithm for most categories, which returned nearly identical results on the default MVTec AD dataset (see Section A.3 for details).

A.3. Online subsampling

The default coresset subsampling algorithm of PatchCore collects all embeddings before creating the coresset. This will result in huge memory usage if using a larger amount of images or higher resolutions. It also requires that all training data is available before calculating the coresset and running any tests.

We therefore define a simple algorithm to learn online, which allows us to calculate the coresset with minimal memory usage and to continue training at any time. This is done by making two small changes to the original subsampling algorithm in Roth et al. [22]: First, we do not collect the embeddings beforehand but sample them as we create the coresset. Second, while PatchCore subsamples a fix percentage r of all embeddings for the coresset, we add $r\%$ of the current image’s embeddings to the coresset. Apart from that, the algorithm remains the same.

The new algorithm is shown in Algorithm 1 whereas all inputs are defined exactly as in Roth et al. [22].

We compared the default and the online subsampling algorithm on the default MVTec AD dataset and achieved nearly identical results (both having an average image-level AUROC of 98.7 %).

Algorithm 1 online coresset subsampling

Input:

ϕ : pre-trained feature extractor

$j \in \mathbb{N}_0^n$: one or multiple indices of the layers from which the features are extracted

$\mathcal{X}_N^{(train)} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^{m \times n \times c}\}_{i \leq N}$: normal training data

P : locally aware patch-feature collector

$s \in \mathbb{N}$: stride (on patch-level after extracting the features, usually 1)

$p \in \mathbb{N}$: neighbourhood size (kernel size of the average pooling, default: 3)

$r \in (0, 1]$: coresset ratio (ratio of embeddings that are added to the coreset)

ψ : random linear projection

Output:

M_C memory bank

```

1:  $M_C \leftarrow \{\}$ 
2: for  $x_i \in \mathcal{X}_N^{(train)}$  do
3:    $M \leftarrow P_{s,p}(\phi_j(x_i))$ 
4:   for  $i \in [0, \dots, \lceil |M| * r \rceil - 1]$  do
5:      $m_i \leftarrow \operatorname{argmax}_{m \in M - M_C} \min_{n \in M_C} \|\psi(m) - \psi(n)\|_2$ 
6:      $M_C \leftarrow M_C \cup \{m_i\}$ 
7:   end for
8: end for
```

It is also possible to learn batch-wise by iterating over batches instead of images and simply accumulating all

Algorithm 2 batch-wise online coresset subsampling

Input:

ϕ : pre-trained feature extractor

$j \in \mathbb{N}_0^n$: one or multiple indices of the layers from which the features are extracted

$\mathcal{X}_{N_b}^{(train)} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^{m \times n \times c}\}_{i \leq N}$: normal training data

P : locally aware patch-feature collector

$s \in \mathbb{N}$: stride (on patch-level after extracting the features, usually 1)

$p \in \mathbb{N}$: neighbourhood size (kernel size of the average pooling, default: 3)

$r \in (0, 1]$: coresset ratio (ratio of embeddings that are added to the coreset)

ψ : random linear projection

$b \in \mathbb{N}$: batchsize

Output:

M_C memory bank

```

1:  $M_C \leftarrow \{\}$ 
2: for  $(B := (x_k, x_{k+1}, \dots, x_{k+b})) \in \mathcal{X}_{N_b}^{(train)}$  do
3:    $M \leftarrow \{\}$ 
4:   for  $x_i \in B$  do
5:      $M \leftarrow M \cup P_{s,p}(\phi_j(x_i))$ 
6:   end for
7:   for  $i \in [0, \dots, \lceil |M| * r \rceil - 1]$  do
8:      $m_i \leftarrow \operatorname{argmax}_{m \in M - M_C} \min_{n \in M_C} \|\psi(m) - \psi(n)\|_2$ 
9:      $M_C \leftarrow M_C \cup \{m_i\}$ 
10:  end for
11: end for
```

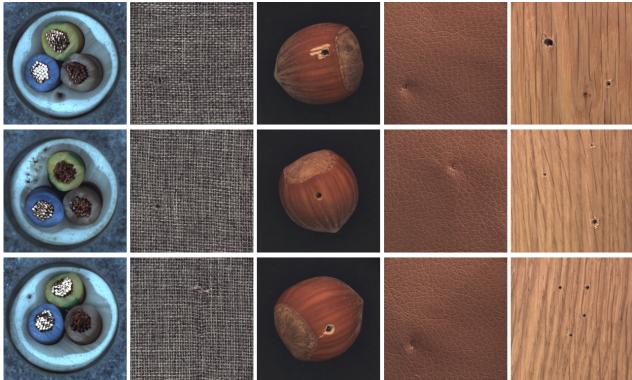
patches from all images of the batch instead of just one image in line 3.

One could also think of more elaborate online learning algorithms, for example changing the number of added embeddings depending on how different the embeddings of the new image/batch are. But our tests show that our simple algorithm already achieves good results compared to the original one.

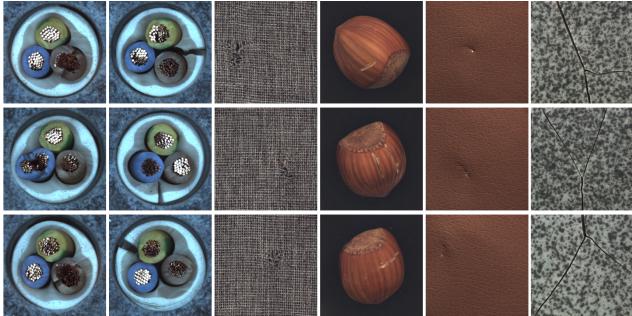
A.4. Software details

To perform the experiments in this paper, we use anomalib [2], a library that implements many state of the art anomaly detection models like PaDiM [11] and PatchCore [22], but also Fastflow [29], GANomaly [1] and others. It is based on PyTorch [21] and PyTorchLightning [13]. It contains dataloaders for the MVTec AD dataset and allows an easy validation and visualization of the results. It also provides ready-to-use scripts for training and testing as well as configuration files.

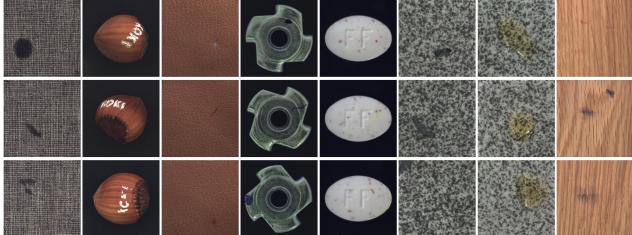
Anomalib uses PyTorch Image Models (timm) [25] which provides many ImageNet pre-trained models among



(a) *hole*



(b) *cut*



(c) *color*

Figure 3. Example images of the custom defined anomaly types
(source of individual images: [3, 4])

other things. For example, it provides many different ResNet and Vision Transformer models.

We will extend and adjust quite a few parts of the library to implement our experiments.

A.5. Data and source code availability

The extended anomalib code can be found at <https://doi.org/10.5281/zenodo.11924708>. The dataset is available at <https://doi.org/10.5281/zenodo.11920346>.

A.6. Dataset example images

Figure 3 shows some example images for each dataset.

A.7. Detailed results

The following Tables 3 to 5 will show more detailed results from our experiments.

Table 3. Image-level F1-Scores in % for all datasets with different approaches using Wide-Resnet50-2 and a Vision Transformer

	Wide-ResNet50-2				ViT-B/8			
	Cut	Color	Hole	Avg.	Cut	Color	Hole	Avg.
SEMLP	86.3 \pm 2.6	83.7 \pm 1.0	83.2 \pm 1.7	84.3	82.3 \pm 3.8	87.1 \pm 1.3	84.9 \pm 1.9	85.0
PatchCore	66.4 \pm 3.4	70.9 \pm 7.4	72.4 \pm 16.2	70.0	66.6 \pm 5.6	64.9 \pm 4.7	58.0 \pm 3.0	63.4
Labeled PatchCore	52.9 \pm 1.0	59.6 \pm 0.6	48.9 \pm 0.8	54.5	52.2 \pm 0.5	67.4 \pm 1.0	54.6 \pm 1.4	59.2
MIRO	85.2 \pm 2.0	75.2 \pm 2.2	82.8 \pm 2.2	80.4	71.9 \pm 6.8	76.1 \pm 4.8	71.5 \pm 4.1	73.5
SEMLP (MIRO)	87.7 \pm 4.4	80.2 \pm 0.8	84.3 \pm 0.7	83.6	72.0 \pm 6.3	78.7 \pm 3.7	73.0 \pm 4.8	75.0
PatchCore (MIRO)	65.6 \pm 1.1	71.0 \pm 1.8	65.7 \pm 3.7	67.9	65.1 \pm 2.9	80.0 \pm 6.5	66.2 \pm 3.9	71.6

Table 4. Image-level AUROC in % for different target domains classified by different approaches using Wide-Resnet50-2

Dataset	Target Domain	SEMLP	PatchCore	Labeled PatchCore	MIRO	SEMLP (MIRO)	PatchCore (MIRO)
Cut	Cable	76.0 \pm 11.0	63.5 \pm 1.5	59.6 \pm 13.6	47.9 \pm 16.8	73.6 \pm 10.6	67.1 \pm 2.6
	Carpet	99.7 \pm 0.4	74.2 \pm 7.0	51.5 \pm 6.7	95.2 \pm 4.2	98.8 \pm 2.1	74.5 \pm 8.8
	Hazelnut	91.0 \pm 6.2	68.6 \pm 3.1	46.0 \pm 4.4	99.1 \pm 1.6	98.0 \pm 2.1	60.9 \pm 3.2
	Leather	98.6 \pm 3.0	49.0 \pm 5.7	55.0 \pm 7.1	93.1 \pm 6.9	90.4 \pm 9.4	36.2 \pm 11.9
	Tile	98.5 \pm 1.9	100 \pm 0.0	24.6 \pm 11.8	99.4 \pm 0.3	99.0 \pm 1.2	99.4 \pm 0.5
Color	Carpet	100 \pm 0.0	67.0 \pm 8.5	55.0 \pm 12.7	85.5 \pm 20.2	98.7 \pm 2.9	74.7 \pm 8.4
	Hazelnut	53.4 \pm 9.3	86.9 \pm 1.5	34.5 \pm 5.2	52.1 \pm 11.8	60.0 \pm 7.4	80.4 \pm 16.2
	Leather	99.2 \pm 1.5	35.4 \pm 1.8	59.0 \pm 8.0	82.6 \pm 17.0	90.6 \pm 13.1	55.0 \pm 17.1
	Metal Nut	70.2 \pm 3.6	48.6 \pm 2.6	56.2 \pm 2.6	53.6 \pm 7.8	71.1 \pm 10.1	53.9 \pm 5.9
	Pill	60.7 \pm 10.5	56.9 \pm 5.1	57.8 \pm 13.0	48.7 \pm 7.8	63.1 \pm 11.6	59.7 \pm 11.8
	Tile	96.8 \pm 0.9	70.0 \pm 4.6	42.7 \pm 5.0	78.1 \pm 5.3	84.6 \pm 7.3	64.8 \pm 11.1
Hole	Wood	100 \pm 0.0	100 \pm 0.0	33.4 \pm 7.0	100 \pm 0.0	100 \pm 0.0	97.9 \pm 2.4
	Cable	45.9 \pm 9.6	50.8 \pm 1.7	51.1 \pm 3.6	52.1 \pm 11.9	44.4 \pm 10.1	51.6 \pm 2.2
	Carpet	97.6 \pm 1.8	87.9 \pm 3.3	49.0 \pm 3.3	98.0 \pm 2.5	98.6 \pm 1.5	80.4 \pm 6.1
	Hazelnut	98.6 \pm 2.0	85.5 \pm 2.0	51.1 \pm 7.5	96.0 \pm 4.2	100.0 \pm 0.0	59.3 \pm 16.8
	Leather	100 \pm 0.0	74.8 \pm 14.5	47.4 \pm 4.8	94.7 \pm 7.7	99.6 \pm 0.9	79.7 \pm 12.3
	Wood	95.9 \pm 2.1	100 \pm 0.0	32.5 \pm 5.1	93.8 \pm 3.8	95.5 \pm 3.7	98.3 \pm 2.1
	Average	87.2	71.7	47.4	80.6	86.2	70.2

Table 5. Image-level AUROC in % for different target domains classified by different approaches using ViT-B/8

Dataset	Target Domain	SEMLP	PatchCore	Labeled PatchCore	MIRO	SEMLP (MIRO)	PatchCore (MIRO)
Cut	Cable	52.3 \pm 8.9	69.7 \pm 2.8	57.8 \pm 2.1	56.0 \pm 4.3	56.5 \pm 4.0	65.0 \pm 3.3
	Carpet	100 \pm 0.0	52.3 \pm 3.1	39.7 \pm 1.2	74.8 \pm 27.2	78.0 \pm 18.9	63.9 \pm 14.0
	Hazelnut	77.8 \pm 18.0	35.9 \pm 1.8	32.0 \pm 2.8	73.3 \pm 15.0	73.0 \pm 13.2	58.1 \pm 22.5
	Leather	100 \pm 0.0	90.2 \pm 1.5	43.7 \pm 5.2	85.5 \pm 8.8	93.2 \pm 4.6	74.4 \pm 17.8
	Tile	98.6 \pm 1.4	90.2 \pm 1.8	65.0 \pm 1.9	89.9 \pm 11.9	86.4 \pm 15.2	79.1 \pm 25.4
Color	Carpet	100 \pm 0.0	67.5 \pm 0.8	48.8 \pm 4.2	63.8 \pm 13.5	70.4 \pm 6.4	80.0 \pm 4.2
	Hazelnut	84.8 \pm 18.3	36.0 \pm 1.6	98.4 \pm 0.7	69.1 \pm 16.9	67.4 \pm 20.3	76.7 \pm 13.1
	Leather	100 \pm 0.0	28.9 \pm 4.7	49.4 \pm 8.9	92.4 \pm 17.0	90.5 \pm 21.3	85.0 \pm 31.1
	Metal Nut	70.2 \pm 3.6	54.8 \pm 1.0	33.5 \pm 8.1	65.8 \pm 17.5	67.9 \pm 22.1	59.0 \pm 14.2
	Pill	59.8 \pm 13.5	45.0 \pm 2.2	51.1 \pm 3.5	71.2 \pm 15.9	73.2 \pm 11.9	75.8 \pm 9.7
	Tile	98.3 \pm 0.7	88.7 \pm 3.5	54.3 \pm 2.6	68.0 \pm 21.3	91.8 \pm 11.7	94.3 \pm 2.4
Hole	Wood	100 \pm 0.0	82.6 \pm 8.0	62.2 \pm 8.7	97.4 \pm 5.2	98.7 \pm 2.9	95.1 \pm 10.9
	Cable	47.1 \pm 16.3	44.3 \pm 2.1	80.4 \pm 2.2	54.2 \pm 10.0	55.2 \pm 15.6	51.8 \pm 11.0
	Carpet	100 \pm 0.0	45.1 \pm 3.3	40.3 \pm 1.1	71.4 \pm 10.4	64.4 \pm 14.4	56.1 \pm 15.5
	Hazelnut	100 \pm 0.0	71.0 \pm 5.2	7.1 \pm 1.5	92.3 \pm 15.5	98.2 \pm 2.4	82.8 \pm 15.5
	Leather	100 \pm 0.0	83.0 \pm 3.2	54.2 \pm 1.2	81.0 \pm 9.5	82.7 \pm 24.0	84.1 \pm 16.8
	Wood	97.8 \pm 1.0	76.5 \pm 6.1	61.6 \pm 5.1	96.7 \pm 5.3	96.1 \pm 3.6	83.5 \pm 22.7
	Average	86.7	62.5	51.7	76.6	79.0	74.4