AnomalyDINO: Boosting Patch-based Few-shot Anomaly Detection with DINOv2

Simon Damm[1], **Mike Laszkiewicz**[1], **Johannes Lederer**[2], **Asja Fischer**[1]
[1]Department of Computer Science, Ruhr University Bochum, Germany
[2]Department of Mathematics, Computer Science, and Natural Sciences, University of Hamburg, Germany
{simon.damm, mike.laszkiewicz, asja.fischer}@rub.de
johannes.lederer@uni-hamburg.de

Abstract

Recent advances in multimodal foundation models have set new standards in few-shot anomaly detection. This paper explores whether high-quality visual features alone are sufficient to rival existing state-of-the-art vision-language models. We affirm this by adapting DINOv2 for one-shot and few-shot anomaly detection, with a focus on industrial applications. We show that this approach does not only rival existing techniques but can even outmatch them in many settings. Our proposed vision-only approach, AnomalyDINO, follows the well-established patch-level deep nearest neighbor paradigm, and enables both image-level anomaly prediction and pixel-level anomaly segmentation. The approach is methodologically simple and training-free and, thus, does not require any additional data for fine-tuning or meta-learning. Despite its simplicity, AnomalyDINO achieves state-of-the-art results in one- and few-shot anomaly detection (e.g., pushing the one-shot performance on MVTec-AD from an AUROC of 93.1% *to* 96.6%*). The reduced overhead, coupled with its outstanding few-shot performance, makes AnomalyDINO a strong candidate for fast deployment, e.g., in industrial contexts.*

1. Introduction

Anomaly detection (AD) in machine learning attempts to identify instances that deviate substantially from the nominal data distribution $p_{\text{norm}}(x)$. Anomalies, therefore, raise suspicion of being 'generated by a different mechanism' [16]—often indicating critical, rare, or unforeseen events. The ability to reliably distinguish anomalies from normal samples is highly valuable across various domains, including security [40], healthcare [14, 41], and industrial inspection. In this work, we focus on the latter, where fully automated systems necessitate the ability to detect defective or missing parts to prevent malfunctions in downstream products, raise alerts for potential hazards, or analyze these to optimize production lines. See the right-hand side of Figure 1 for anomalous samples in this context.

AD for industrial images has gained tremendous interest over the last couple of years. The close-to-optimal results on benchmark data make it seem that the problem of anomaly detection is essentially solved. For instance, Mousakhan et al. [28] report 99.8% and 98.9% AUROC on the popular benchmarks MVTec-AD [2] and VisA [51], respectively. The most popular AD techniques use the training data to train an anomaly classifier [38], or a generative model coupled with reconstruction-based [26, 28, 42], or likelihood-based [9, 35] anomaly scoring. However, these approaches operate within the full-shot setting, meaning they rely on access to a sufficiently large amount of training data. Given the challenges associated with dataset acquisition, the attractivity of a fast and easy-to-deploy methodology, and the requirement to rapidly adapt to covariate shifts in the nominal data distribution [22], there is an increasing interest in few-shot and zero-shot anomaly detection.

Few-shot techniques, however, heavily rely on meaningful features, or as [33] frame it: 'anomaly detection requires better representations'. Such better representations are now available with the increasing availability and capabilities of foundation models, i.e., large-scale models trained on massive datasets in unsupervised/self-supervised fashion [5, 30, 31]. The performance of few-shot anomaly detection techniques has already been boosted by the use of foundation models, mostly by multimodal approaches incorporating language and vision [4, 19, 21, 50].

Here, we propose to focus on a vision-only approach, in contrast to such multimodal techniques. This perspective is

This paper was accepted at the Winter Conference on Applications of Computer Vision (WACV), 2025. The final version is available here (Open Access) and on IEEE Xplore.

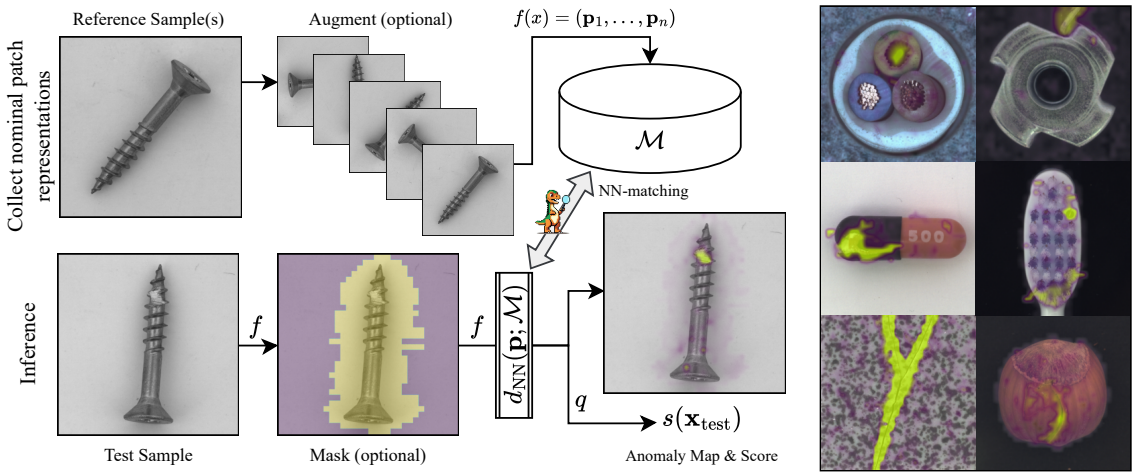


Figure 1. **Anomaly detection with AnomalyDINO** based on a single immaculate reference sample (here category 'Screw' from MVTec-AD). We collect the nominal patch representations from the (potentially augmented) reference sample(s) in the memory bank $\mathcal{M}$. At test time, we select the relevant patch representation via masking (if applicable). The distances of those to the nominal representations in $\mathcal{M}$ give rise to an anomaly map and the corresponding anomaly score $s(\mathbf{x}_{\text{test}})$ using the aggregation statistic q. For both, masking and feature extraction, we utilize DINOv2 (f). Further examples for other categories are depicted on the right (and in Figures 4 to 7 in Appendix A).

motivated by the observation that few-shot anomaly detection is feasible for human annotators based on visual features only, and does not require additional textual description of the given object or the expected types of anomalies (which are typically not known a priori).

Our approach, termed AnomalyDINO, follows the well-established AD framework of *patch-level deep nearest neighbor* [34, 46], and leverages DINOv2 [30] as a backbone. We carefully design a suitable preprocessing pipeline for the few-shot scenario, using the zero-shot segmentation abilities of DINOv2 (which alleviates the additional overhead of another segmentation model). At test time, anomalous samples are detected based on the high distances between their patch representations and the closest counterparts in the nominal memory bank $\mathcal{M}$.

Due to its simplicity, AnomalyDINO can be deployed in industrial contexts very easily—in strong contrast to more complex approaches such as [7] or [21]. Yet, the proposed method achieves new state-of-the-art performance on anomaly detection in the few-shot regime on MVTec-AD [2] and outperforms all but one competing method on VisA [51].

The structure of the paper is as follows: Section 2 reviews relevant prior studies and clarifies the distinctions between the settings addressed by zero- and few-shot, and batched zero-shot techniques. Section 3 introduces our proposed method, AnomalyDINO. An extension of this method to the batched zero-shot scenario is detailed in Appendix D. Section 4 presents the experimental outcomes. Additional results and an ablation study are provided in Appendices A and C, respectively. We address identified failure cases of AnomalyDINO in Appendix B. The code to reproduce the experiments is available at https://github.com/dammsi/AnomalyDINO.

Contributions

- We propose AnomalyDINO, a simple and training-free yet highly effective patch-based technique for visual anomaly detection. Our method builds upon the high-quality feature representation extracted by DINOv2.
- An extensive analysis demonstrates the efficiency and effectiveness of the proposed approach, outperforming other multimodal few-shot techniques in terms of performance *and* inference speed. Specifically, AnomalyDINO achieves state-of-the-art results for few-shot anomaly detection on MVTec-AD, e.g., pushing the one-shot detection from an AUROC of 93.1% to 96.6% (thereby halving the gap between the few- and full-shot setting). Moreover, our results on VisA are not only competitive with other few-shot methods but also establish a new state-of-the-art for all training-free few-shot anomaly detectors, achieving the best localization performance across all methods.

2. Related Work

Foundation Models for Vision Multimodal foundation models have emerged as powerful tools for a wide range of tasks, see e.g., [3, 6, 17, 23, 25, 29, 31]. Most relevant to visual AD are multimodal approaches based on CLIP [31] or recent LLMs [29], but also vision-only approaches like

DINO [5, 30]. CLIP [31] learns visual concepts from natural language descriptions by training on a dataset of images paired with textual annotations. The model uses a contrastive learning objective that aligns the embeddings from image and text encoders, optimizing the similarity between corresponding image-text pairs. This common feature space for vision and language can be utilized for several downstream tasks, such as zero-shot image classification by assessing similarities to a set of class-specific prompts. DINO [5, 30] leverages a self-supervised student-teacher framework based on vision transformers [12]. It employs a multi-view strategy to train Vision Transformers (ViT) [11] to predict softened teacher outputs, thereby learning robust and high-quality features for downstream tasks. DINOv2 [30] combines ideas from DINO with patch-level reconstruction techniques [49] and scales to larger architectures and datasets. The features extracted by DINO are well-suited for anomaly detection as they incorporate both local and global information, are robust to multiple views and crops, and benefit from large-scale pre-training. GroundingDINO [25], builds upon the DINO framework and focuses on improving the alignment of textual and visual information, enhancing the model's performance in tasks requiring detailed object localization and multimodal understanding.

Anomaly Detection Given a predefined notion of normality, the anomaly detection task is to detect test samples that deviate from this concept [36, 45]. In this work, we focus on *low-level sensory anomalies of industrial image data*, i.e., we do not target the detection of semantic anomalies but of low-level features such as scratches of images of industrial products (see e.g., Figure 1). Several works tackled this task by either training an anomaly classifier [38] or a generative model, which allows for reconstruction-based or likelihood-based AD [9, 26, 28, 42, 48].

A well-established approach is that of *deep nearest neighbor* AD, where test instances are scored according to the distance to their nearest neighbors (in feature space) from the memory bank $\mathcal{M}$, containing features of nominal instances. This approach dates back to (at least) 2002 [13], but got later popularized first on image-level [1] and then by the seminal works on patch-level: Patch support vector data description (SVDD) [46] trains a patch-based feature extractor (with a mixture of DeepSVDD [37] and a self-supervised loss), and PatchCore [34] utilizes pre-trained classification models on Imagenet for patch-level feature extraction. Various further approaches build on this simple, but effective AD approach [8, 24, 43]. Besides kNN matching, other 'classical' AD methods like the Mahalanobis distance are employed [10]. Evidently, the performance of these approaches crucially depends on the feature extractor f. Classical choices are ResNets [18], Wide ResNets [47], and Vision Transformers [11], mostly pre-trained on a supervised task. Another line of work builds upon the success of pre-trained language-vision models in zero-shot classification. The underlying idea consists of two steps. First, these approaches define sets of prompts describing nominal samples and anomalies. Second, the corresponding textual embeddings are compared against the image embeddings [7, 19, 21, 50]. Images whose visual embedding is close to the textual embedding of a prompt associated with an anomaly are classified as anomalous. However, these methods either require significant prompt engineering (e.g., [7] use a total of 35×7 different prompts for describing normal samples) or fine-tuning of the prompt(-embeddings). Lastly, another type of few-shot anomaly detection builds upon the success of multimodal chatbots. These methods require more elaborate prompting and techniques for interpreting textual outputs [44]. Since these methods do not require a memory bank, they are capable of performing zero-shot anomaly detection.

Categorization of Few-/Zero-Shot Anomaly Detectors Previous works consider different AD setups, which complicates their evaluation and comparison. To remedy this, we provide a taxonomy of recent few- and zero-shot AD based on the particular 'shot'-setting, the training requirements, and the modes covered by the underlying models. We categorize three 'shot'-settings: zero-shot, few-shot, and *batched zero-shot*. Zero- and few-shot settings are characterized by the number of nominal training samples a method can process, before making predictions on the test samples. In batched zero-shot, inference is not performed sample-wise but based on a whole batch of test samples, usually the full test set. For instance, the method proposed in [24] benefits from the fact that a significant majority of pixels correspond to normal pixels, which motivates the strategy of matching patches across a batch of images. Another work that considers this setting [22], deploys

Table 1. **Taxonomy of recent few- and zero-shot anomaly detection methods**. The † indicates approaches that were introduced as full-shot detectors but then considered as few-shot detectors in later works, see e.g., [34].

Method	Setting	Training Type	Modes
DN2 [1]	Few-Shot	Training-Free	Vision
SPADE [8]	Few-Shot†	Training-Free	Vision
PaDiM [10]	Few-Shot†	Fine-Tuning	Vision
PatchCore [34]	Few-Shot	Training-Free	Vision
PatchCore–opt [39]	Few-Shot	Training-Free	Vision
MuSc [24]	Batched Zero-Shot	Training-Free	Vision
GraphCore [43]	Few-Shot	Training (GNN)	Vision
WinCLIP [19]	Zero-/Few-Shot	Training-Free	Vision + Language
APRIL-GAN [7]	Zero-/Few-Shot	Meta-Training	Vision + Language
ADP [21]	Few-Shot	Fine-Tuning	Vision + Language
AnomalyCLIP [50]	Zero-Shot	Meta-Training	Vision + Language
GPT4-V [44]	Zero-Shot	Training-Free	Vision + Language
ACR [22]	Batched Zero-Shot	Meta-Training	/
AnomalyDINO (Ours)	Few-Shot	Training-Free	Vision

a parameter-free anomaly detector based on the effect of batch normalization. We split the training requirements into the categories 'Training-Free', 'Fine-Tuning', and 'Meta-Training'. 'Training-Free' approaches do not require any training, while 'Fine-Tuning' methods use the few accessible samples to modify the underlying model. In contrast, 'Meta-Training' is associated with training the model on a dataset related to the test data. For example, [22] train their model on MVTec-AD containing all classes except the class they test against. [50] and [7] train their model on VisA when evaluating the test performance on MVTec-AD and vice versa. Finally, we differentiate the leveraged models, which are either vision models (such as pre-trained ViT) or language-vision models (such as CLIP). We provide a detailed summary in Table 1.

3. Matching Patch Representations for Visual Anomaly Detection

This section introduces AnomalyDINO, which leverages DINOv2 to extract meaningful patch-level features. We build upon the well-established deep nearest neighbor approaches [1, 8, 13, 24, 34, 43, 46], i.e., we first gather relevant patch representations of nominal patches in a memory bank $\mathcal{M}$. Then, for each test patch, we compute its distance to the nearest nominal patch in $\mathcal{M}$. A suitable aggregation of the patch-based distances gives anomaly scores on image-level.

Our work differs from previous deep nearest neighbor approaches [34, 46] by tailoring the memory bank concept to the few-shot regime, specifically utilizing the strong features from DINOv2: We design a pipeline that incorporates *zero-shot masking* and *augmentations*, and we propose a *more robust aggregation statistic*. Importantly, we identify DINOv2 as the ideal backbone for our scenario due to its strong patch-level features and masking ability. In addition, this simplifies the deep nearest neighbor framework by reducing the complexity of the feature engineering stage (e.g., by making it unnecessary to think of which representations/layers to use and how to aggregate them), and *increasing the flexibility* w.r.t. input resolution (that can be chosen to be any multiple of 14). The proposed method is described in detail in the following subsections.

3.1. Anomaly Detection via Patch (Dis-) Similarities

Let us briefly review the idea of patch-level deep nearest neighbor AD. We assume to have access to a suitable feature extractor $f : \mathcal{X} \to \mathcal{F}^n$ that maps each image $\mathbf{x} \in \mathcal{X}$ to a tuple of patch-features $f(\mathbf{x}) = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$, where $\mathcal{X}$ denotes some space of images and $\mathcal{F}$ the feature space. Note that n depends on the image resolution and the patch size (in the case of DINOv2, a patch is 14×14 pixels). Given $k \geq 1$ nominal reference sample(s) $X_{\text{ref}} := \{\mathbf{x}^{(i)} \mid i \in [k]\}$ (with shorthand $[k] := \{1, \ldots, k\}$), we collect the nominal patch features and store them in a memory bank

$$\mathcal{M} := \bigcup_{\mathbf{x}^{(i)} \in X_{\text{ref}}} \left\{ \mathbf{p}_j^{(i)} \mid f(\mathbf{x}^{(i)}) = (\mathbf{p}_1^{(i)}, \ldots, \mathbf{p}_n^{(i)}),\ j \in [n] \right\} . \tag{1}$$

To score a test sample $\mathbf{x}_{\text{test}}$, we collect the extracted patch representations $f(\mathbf{x}_{\text{test}}) = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ and then check how well they comply with $\mathcal{M}$. To do so, we leverage a nearest neighbor approach to find the distance to the closest reference patch for a given test patch $\mathbf{p} \in \mathcal{F}$

$$d_{\text{NN}}(\mathbf{p}; \mathcal{M}) := \min_{\mathbf{p}_{\text{ref}} \in \mathcal{M}} d(\mathbf{p},\ \mathbf{p}_{\text{ref}}) \tag{2}$$

for some distance metric d. In our experiments, we set d as the cosine distance, that is,

$$d(\mathbf{x}, \mathbf{y}) := 1 - \frac{\langle \mathbf{x},\ \mathbf{y} \rangle}{\|\mathbf{x}\|\ \|\mathbf{y}\|} . \tag{3}$$

The image-level score $s(\mathbf{x}_{\text{test}})$ is given by aggregating the patch distances via a suitable statistic q

$$s(\mathbf{x}_{\text{test}}) := q\big(\{d_{\text{NN}}(\mathbf{p}_1; \mathcal{M}), \ldots, d_{\text{NN}}(\mathbf{p}_n; \mathcal{M})\}\big) . \tag{4}$$

Throughout this paper, we define q as the average distance of the 1% most anomalous patches, i.e, $q(\mathcal{D}) := \text{mean}(H_{0.01}(\mathcal{D}))$ with $H_{0.01}(\mathcal{D})$ containing the 1% highest values in the set $\mathcal{D}$. The statistic q can be understood as an empirical estimate of the tail value at risk for the 99% quantile [27] and turns out to be suitable for a wide range of settings because it balances two desirable properties: First, we want $s(\mathbf{x}_{\text{test}})$ to depend on the highest patch distances, as they may provide the strongest anomaly signal. Similarly, we want a certain degree of robustness against a singular high patch distance (in particular in few-shot scenarios where $\mathcal{M}$ is sparsely populated). However, one could also replace q with a different statistic to cater to special cases. If anomalies are expected to cover larger parts of the image, for example, a high percentile of the patch distances might be a suitable choice. If, on the contrary, anomalies may occur only locally such that only very few patches/or a single patch might be affected, the score q should be sensitive to the highest patch distances. Frequently, the maximum pixel-wise anomaly score is considered. Such an anomaly score on pixel-level is usually obtained by upsampling to full image resolution and applying some smoothing operation.

Following [34], we utilize bilinear upsampling and Gaussian smoothing ($\sigma = 4.0$) to turn the patch distances into pixel-level anomaly scores for localization of potential defects. Examples of the resulting anomaly maps are visualized in Figure 1 and Appendix A (Figures 4 to 7).

We extend AnomalyDINO also to the batched zero-shot scenario, see Appendix D (Figure 18).

3.2. Enriching the Memory Bank & Filtering Relevant Patches

In the few-shot anomaly detection setting, the primary challenge is to effectively capture the concept of normality from the limited set of nominal samples. A useful strategy involves applying simple augmentations, like rotations (following the insights in [43]), to enhance the variability of nominal patch features. This is essential because the variability at test time is likely to be significantly greater than in the reference data, X_{ref}. In contrast, full-shot methods aim to reduce the size of $\mathcal{M}$ to reduce inference time, e.g., [34].

To avoid irrelevant areas of the test image from leading to falsely high anomaly scores, we propose masking the object of interest. This approach mitigates the risk of false positives, particularly in the few-shot regime where the limited reference samples may not adequately capture the natural variations in the background, as exemplified in Figure 10, Appendix C. It is important to note that the appropriate preprocessing technique—whether to apply masking and/or augmentations—should depend on the specific characteristics of the object(s) of interest. For a more detailed discussion on the challenges and considerations in designing an effective preprocessing pipeline, see Appendix C.1.

Masking Masking, i.e., delineating the primary object(s) in an image from its background, can help to reduce false-positive predictions, thereby improving the robustness in the low-data regime. To minimize the overhead of the proposed pipeline, we utilize DINOv2 also for masking. This is achieved by thresholding the first PCA component of the patch features [30]. We observe empirically that this sometimes produces erroneous masks. Frequently, such failure cases occur for close-up shots, where the objects of interest account for $\gtrsim 50\%$ of patches. To address this issue, we check if the PCA-based mask accurately captures the object in the first reference sample and apply the mask accordingly, which gives rise to the 'masking test' in Figure 2. This test is performed *only once* per object as the procedure yields very consistent outputs. In addition, we utilize dilation and morphological closing to eliminate small holes and gaps within the predicted masks. See Figures 14 and 15 for examples of this masking procedure, Table 8 for the outcomes of the masking test per object, and Figure 10 for a visualization of the benefits of masking in the presence of background noise (all in Appendix C). In general, we do not mask textures (e.g., 'Wood' or 'Tile' in MVTec-AD).

Rotation Rotating the reference sample may improve the detection performance by better resembling the variations within the concept of normality captured in $\mathcal{M}$. Consider, e.g., the 'Screw' in MVTec-AD as depicted in Figure 1 for which rotation-invariant features are desirable. On the other hand, rotation can also be detrimental in cases where rotations of (parts of) the object of interest can be considered anomalies themselves (see Figure 13 for an example).

Figure 2. **Masking test on MVTec-AD**. For 'Capsule' and 'Hazelnut' the masking works successfully (top row), while for 'Cable' and 'Transistor' (bottom row) some areas are incorrectly predicted as background that should belong to the object of interest (highlighted in red). See App. C.1 for the outcomes per object.

We consider two different settings. In the 'agnostic' case, which we focus on in the main paper, we always augment the reference sample with rotations. We also consider the case where we know about the samples' potential rotations ('informed'). The 'informed' case is sensible as the data collection process can usually be controlled in an industrial/medical setting (or test images can be aligned), and may lead to lower inference time. See Appendix C.1 for the effect of masking and rotation, and a comparison of 'informed' vs. 'agnostic'.

4. Experiments

AnomalyDINO – Defaults Using DINOv2[1] as backbone allows us to choose from different distillation sizes, which range from small (ViT-S, 21×10^6 parameters) to giant (ViT-G, 1.1×10^9 parameters). To prioritize low latency, we take the smallest model as our default (and denote our default pipeline AnomalyDINO-S, accordingly) and evaluate two input resolutions, 448 and 672 pixels (smaller edge). As discussed above, we utilize the 'agnostic' preprocessing by default (see Appendix C).

Datasets For the experiments, we consider MVTec-AD[2] and VisA,[3] two datasets with high-resolution images for industrial anomaly detection. MVTec-AD consists of fifteen categories, depicting single objects or textures, and up to eight anomaly types per category. VisA has twelve categories and comes with diverse types of anomalies condensed into one anomaly class 'bad'. Some categories in

[1] Code/weights of DINOv2 [30] are available under Apache 2.0 license.
[2] The MVTec-AD dataset [2] is available under CC BY-NC-SA 4.0.
[3] The VisA dataset [51] is released under the CC BY 4.0 license.

VisA can be considered more challenging as multiple objects and complex structures are present.

Evaluation Metrics We assess the image-level detection and the pixel-level segmentation performance with three metrics each, following [7, 19, 24]. For evaluating the detection performance, we measure the area under the receiver-operator curve (AUROC), the F1-score at the optimal threshold (F1-max), and the average precision (AP) using the respective image-level anomaly scores. We quantify the anomaly segmentation performance using the AUROC, F1-max, and the per-region overlap (PRO, [2]) of the segmentation using the pixel-wise anomaly scores. Note that due to a high imbalance of nominal and anomalous pixels—for MVTec-AD we have 97.26% nominal pixel, for VisA even 99.45% [24]—it is not recommended to assess performance solely on segmentation AUROC [32]. We repeat each experiment three times and report mean and standard deviation.[4]

Baselines We compare AnomalyDINO with a range of modern zero- and few-shot AD models, e.g., SPADE [8], PaDiM [10], PatchCore [34], WinCLIP+ [19], and APRIL-GAN [7]. It is important to note that ACR [22] and MuSc [24] consider the batched zero-shot setting (see Table 1), thus, covering a different setting than AnomalyDINO. We adapt AnomalyDINO to this setting in Appendix D. Moreover, APRIL-GAN and AnomalyCLIP, require training on a related dataset, which is in contrast to our proposed training-free method. In Tables 2 and 3, results reported from [19] are indicated by †, the result of the WinCLIP re-implementation [21] (where J. Jeong is also coauthor) by *. All other results are taken from the original publication. For ADP [21] we report the (usually slightly better) variant ADP$_\ell$ (with access to the class label). For GraphCore no standard deviations are reported [43].

4.1. Few-Shot Anomaly Detection and Anomaly Segmentation

Results We summarize the results for few-shot anomaly detection on MVTec-AD and VisA in Table 2 and Table 3, respectively. Regarding MVTec-AD, our method achieves state-of-the-art k-shot detection performance across all $k \in \{1, 2, 4, 8, 16\}$ for every reported metric, outperforming approaches that require additional training data sets (such as ADP and APRIL-GAN). The method also demonstrates superior anomaly localization, with results showing that while detection performance remains comparable across different resolutions (448 vs. 672), a higher resolution enhances localization performance. Furthermore, we observe clear improvements as the number of samples increases.

Table 2. **Anomaly detection on MVTec-AD**. Quantitative results for detection (image-level) and segmentation (pixel-level). For each shot, we highlight the **best result** in bold, the results from the second best method as underlined, and the best training-free result by a gray box (see also Table 1). All results in %.

Method	Classification			Segmentation		
	AUROC	F1-max	AP	AUROC	F1-max	PRO
1-shot						
SPADE†	81.0$_{\pm 2.0}$	90.3$_{\pm 0.8}$	90.6$_{\pm 0.8}$	91.2$_{\pm 0.4}$	42.4$_{\pm 1.0}$	83.9$_{\pm 0.7}$
PatchCore†	83.4$_{\pm 3.0}$	90.5$_{\pm 1.5}$	92.2$_{\pm 1.5}$	92.0$_{\pm 1.0}$	50.4$_{\pm 2.1}$	79.7$_{\pm 2.0}$
GraphCore	89.9$_{\pm /}$	/	/	95.6$_{\pm /}$	/	/
WinCLIP+	93.1$_{\pm 2.0}$	93.7$_{\pm 1.1}$	96.5$_{\pm 0.9}$	95.2$_{\pm 0.5}$	55.9$_{\pm 2.7}$	87.1$_{\pm 1.2}$
APRIL-GAN	92.0$_{\pm 0.3}$	92.4$_{\pm 0.2}$	95.8$_{\pm 0.2}$	95.1$_{\pm 0.1}$	54.2$_{\pm 0.0}$	90.6$_{\pm 0.2}$
AnomalyDINO-S (448)	96.5$_{\pm 0.4}$	**96.0**$_{\pm 0.2}$	98.1$_{\pm 0.3}$	96.3$_{\pm 0.1}$	57.9$_{\pm 0.8}$	91.7$_{\pm 0.1}$
AnomalyDINO-S (672)	**96.6**$_{\pm 0.4}$	95.8$_{\pm 0.5}$	**98.2**$_{\pm 0.2}$	**96.8**$_{\pm 0.1}$	**60.2**$_{\pm 1.1}$	**92.7**$_{\pm 0.1}$
2-shot						
SPADE†	82.9$_{\pm 2.6}$	91.1$_{\pm 1.0}$	91.7$_{\pm 1.2}$	92.0$_{\pm 0.3}$	44.5$_{\pm 1.0}$	85.7$_{\pm 0.7}$
PatchCore†	86.3$_{\pm 3.3}$	92.0$_{\pm 1.5}$	93.8$_{\pm 1.7}$	93.3$_{\pm 0.6}$	53.0$_{\pm 1.7}$	82.3$_{\pm 1.3}$
GraphCore	91.9$_{\pm /}$	/	/	96.9$_{\pm /}$	/	/
WinCLIP+	94.4$_{\pm 1.3}$	94.4$_{\pm 0.8}$	97.0$_{\pm 0.7}$	96.0$_{\pm 0.3}$	58.4$_{\pm 1.7}$	88.4$_{\pm 0.9}$
ADP$_\ell$	95.4$_{\pm 0.9}$	/	/	/	/	/
APRIL-GAN	92.4$_{\pm 0.3}$	92.6$_{\pm 0.1}$	96.0$_{\pm 0.2}$	95.5$_{\pm 0.0}$	55.9$_{\pm 0.5}$	91.3$_{\pm 0.1}$
AnomalyDINO-S (448)	96.7$_{\pm 0.8}$	**96.5**$_{\pm 0.4}$	98.1$_{\pm 0.7}$	96.5$_{\pm 0.2}$	58.5$_{\pm 0.5}$	92.0$_{\pm 0.2}$
AnomalyDINO-S (672)	**96.9**$_{\pm 0.7}$	96.1$_{\pm 0.3}$	**98.2**$_{\pm 0.5}$	**97.0**$_{\pm 0.2}$	**61.0**$_{\pm 0.5}$	**93.1**$_{\pm 0.2}$
4-shot						
SPADE†	84.8$_{\pm 2.5}$	91.5$_{\pm 0.9}$	92.5$_{\pm 1.2}$	92.7$_{\pm 0.3}$	46.2$_{\pm 1.3}$	87.0$_{\pm 0.5}$
PatchCore†	88.8$_{\pm 2.6}$	92.6$_{\pm 1.6}$	94.5$_{\pm 1.5}$	94.3$_{\pm 0.5}$	55.0$_{\pm 1.9}$	84.3$_{\pm 1.6}$
GraphCore	92.9$_{\pm /}$	/	/	**97.4**$_{\pm /}$	/	/
WinCLIP+	95.2$_{\pm 1.3}$	94.7$_{\pm 0.8}$	97.3$_{\pm 0.6}$	96.2$_{\pm 0.3}$	59.5$_{\pm 1.8}$	89.0$_{\pm 0.8}$
ADP$_\ell$	96.2$_{\pm 0.8}$	/	/	/	/	/
APRIL-GAN	92.8$_{\pm 0.2}$	92.8$_{\pm 0.1}$	96.3$_{\pm 0.1}$	95.9$_{\pm 0.0}$	56.9$_{\pm 0.1}$	91.8$_{\pm 0.1}$
AnomalyDINO-S (448)	97.6$_{\pm 0.1}$	**97.0**$_{\pm 0.3}$	98.4$_{\pm 0.3}$	96.7$_{\pm 0.1}$	59.2$_{\pm 0.4}$	92.4$_{\pm 0.1}$
AnomalyDINO-S (672)	**97.7**$_{\pm 0.2}$	96.6$_{\pm 0.0}$	**98.7**$_{\pm 0.1}$	97.2$_{\pm 0.1}$	**61.8**$_{\pm 0.1}$	**93.4**$_{\pm 0.1}$
8-shot						
GraphCore	95.9$_{\pm /}$	/	/	**97.8**$_{\pm /}$	/	/
WinCLIP+	94.6$_{\pm 0.1}$*	/	/	/	/	/
ADP$_\ell$	97.0$_{\pm 0.2}$	/	/	/	/	/
APRIL-GAN	93.1$_{\pm 0.2}$	93.1$_{\pm 0.2}$	96.4$_{\pm 0.2}$	96.2$_{\pm 0.1}$	57.7$_{\pm 0.2}$	92.4$_{\pm 0.2}$
AnomalyDINO-S (448)	98.0$_{\pm 0.1}$	**97.4**$_{\pm 0.1}$	99.0$_{\pm 0.2}$	97.0$_{\pm 0.1}$	59.6$_{\pm 0.2}$	92.7$_{\pm 0.0}$
AnomalyDINO-S (672)	**98.2**$_{\pm 0.2}$	**97.4**$_{\pm 0.2}$	**99.1**$_{\pm 0.1}$	97.4$_{\pm 0.1}$	**62.3**$_{\pm 0.1}$	**93.8**$_{\pm 0.1}$
16-shot						
WinCLIP+	94.8$_{\pm 0.1}$*	/	/	/	/	/
ADP$_\ell$	97.0$_{\pm 0.3}$	/	/	/	/	/
APRIL-GAN	93.2$_{\pm 0.1}$	93.0$_{\pm 0.1}$	96.5$_{\pm 0.1}$	96.4$_{\pm 0.0}$	58.5$_{\pm 0.1}$	92.6$_{\pm 0.1}$
AnomalyDINO-S (448)	98.3$_{\pm 0.1}$	**97.7**$_{\pm 0.2}$	99.3$_{\pm 0.0}$	97.1$_{\pm 0.1}$	60.0$_{\pm 0.1}$	92.9$_{\pm 0.1}$
AnomalyDINO-S (672)	**98.4**$_{\pm 0.1}$	97.6$_{\pm 0.1}$	**99.3**$_{\pm 0.0}$	**97.5**$_{\pm 0.0}$	**62.7**$_{\pm 0.1}$	**94.0**$_{\pm 0.1}$

In terms of anomaly detection in the VisA benchmark (Table 3), APRIL-GAN demonstrates the strongest performance for $k \in \{1, 2\}$. Nonetheless, AnomalyDINO consistently achieves second-best results for $k \in \{1, 2\}$, comparable results in the 4-shot setting, and sets new state-of-the-art for $k \in \{8, 16\}$. This can be attributed to AnomalyDINO's ability to benefit more from a richer memory bank $\mathcal{M}$ than APRIL-GAN. We hypothesize that meta-learning exerts a greater influence on APRIL-GAN (i.e., training on MVTec-AD, when testing on VisA, and vice-versa) compared to learning from the nominal features of the given reference samples. Note also that AnomalyDINO outperforms all other training-free approaches.

Comparing the segmentation performance on VisA, Table 3 reveals a clear picture: AnomalyDINO consistently shows the strongest localization performance in all metrics

[4] The randomness stems from the choice of reference samples X_{ref}. For straightforward reproducibility, we simply set X_{ref} to the first k, second k and third k train samples for the three different runs, respectively.

Table 3. **Anomaly detection on VisA**. Quantitative results for detection (image-level) and segmentation (pixel-level). For each shot, we highlight the **best result** in bold, the results from the second best method as underlined, and the best training-free result by a gray box (see also Table 1). All results in %.

Method	Classification			Segmentation		
	AUROC	F1-max	AP	AUROC	F1-max	PRO
1-shot						
SPADE†	79.5±4.0	80.7±1.9	82.0±3.3	95.6±0.4	35.5±2.2	84.1±1.6
PaDiM†	62.8±5.4	75.3±1.2	68.3±4.0	89.9±0.8	17.4±1.7	64.3±2.4
PatchCore†	79.9±2.9	81.7±1.6	82.8±2.3	95.4±0.6	38.0±1.9	80.5±2.5
WinCLIP+	83.8±4.0	83.1±1.7	85.1±4.0	96.4±0.4	41.3±2.3	85.1±2.1
APRIL-GAN	**91.2**±0.8	**86.9**±0.6	**93.3**±0.8	96.0±0.0	38.5±0.3	90.0±0.1
AnomalyDINO-S (448)	85.6±1.5	83.1±1.1	86.6±1.3	97.5±0.1	41.9±0.5	90.7±0.5
AnomalyDINO-S (672)	87.4±1.2	84.3±0.5	89.0±1.0	**97.8**±0.1	**45.1**±0.9	**92.5**±0.5
2-shot						
SPADE†	80.7±5.0	81.7±2.5	82.3±4.3	96.2±0.4	40.5±3.7	85.7±1.1
PaDiM†	67.4±5.1	75.7±1.8	71.6±3.8	92.0±0.7	21.1±2.4	70.1±2.6
PatchCore†	81.6±4.0	82.5±1.8	84.8±3.2	96.1±0.5	41.0±3.9	82.6±2.3
WinCLIP+	84.6±2.4	83.0±1.4	85.8±2.7	96.8±0.3	43.5±3.3	86.2±1.4
ADP$_\ell$	86.9±0.9	/	/	/	/	/
APRIL-GAN	**92.2**±0.3	**87.7**±0.3	**94.2**±0.3	96.2±0.0	39.3±0.2	90.1±0.1
AnomalyDINO-S (448)	88.3±1.8	84.8±1.2	89.2±1.3	97.8±0.1	44.2±0.3	91.7±0.5
AnomalyDINO-S (672)	89.7±1.3	86.3±1.2	90.7±0.8	**98.0**±0.1	**47.6**±0.5	**93.4**±0.6
4-shot						
SPADE†	81.7±3.4	82.1±2.1	83.4±2.7	96.6±0.3	43.6±3.6	87.3±0.8
PaDiM†	72.8±2.9	78.0±1.2	75.6±2.2	93.2±0.5	24.6±1.8	72.6±1.9
PatchCore†	85.3±2.1	84.3±1.3	87.5±2.1	96.8±0.3	43.9±3.1	84.9±1.4
WinCLIP+	87.3±1.8	84.2±1.6	88.8±1.8	97.2±0.2	47.0±3.0	87.6±0.9
ADP$_\ell$	88.4±0.4	/	/	/	/	/
APRIL-GAN	**92.6**±0.4	88.4±0.5	**94.5**±0.3	96.2±0.0	40.0±0.1	90.2±0.1
AnomalyDINO-S (448)	91.3±0.8	87.5±1.0	91.8±0.7	98.0±0.0	46.1±0.3	92.5±0.2
AnomalyDINO-S (672)	**92.6**±0.9	**88.8**±0.9	92.9±0.7	**98.2**±0.0	**49.4**±0.3	**94.1**±0.1
8-shot						
WinCLIP+	85.0±0.0*	/	/	/	/	/
ADP$_\ell$	89.2±0.1	/	/	/	/	/
APRIL-GAN	93.0±0.2	88.8±0.2	**94.9**±0.3	96.3±0.0	40.2±0.1	90.2±0.0
AnomalyDINO-S (448)	92.6±0.1	88.6±0.2	92.9±0.2	98.2±0.0	47.6±0.5	93.3±0.2
AnomalyDINO-S (672)	**93.8**±0.3	**90.0**±0.1	94.3±0.4	**98.4**±0.0	**51.1**±0.4	**94.8**±0.2
16-shot						
WinCLIP+	85.0±0.1*	/	/	/	/	/
ADP$_\ell$	90.1±0.5	/	/	/	/	/
APRIL-GAN	93.2±0.2	89.0±0.1	95.2±0.1	96.3±0.0	40.6±0.1	90.2±0.1
AnomalyDINO-S (448)	93.8±0.1	89.9±0.3	94.2±0.3	98.3±0.0	48.6±0.3	93.8±0.2
AnomalyDINO-S (672)	**94.8**±0.2	**90.9**±0.2	**95.3**±0.3	**98.5**±0.0	**52.5**±0.5	**95.3**±0.2

considered. While AnomalyDINO-S (448) already demonstrates strong performance, the advantages of using a higher resolution (672) become more evident on the VisA dataset. We attribute this fact to smaller anomalous regions (for which smaller effective patch sizes are beneficial) and more complex scenes (compared to MVTec-AD).

We have further adapted AnomalyDINO to the batched zero-shot setting (see Appendix D). This adaptation is straightforward and relatively simple, especially compared to MuSc. Notably, we did not employ additional techniques such as 'Re-scoring with Constrained Image-level Neighborhood' or 'Local Neighborhood Aggregation with Multiple Degrees' [24]. The results obtained, as detailed in Table 4, are therefore quite satisfactory.

We conclude that AnomalyDINO—despite its simplicity—rivals other methods in all settings and even comes out ahead in most of the settings (e.g., significantly reducing the gap between few-shot and full-shot methods for MVTec-AD). Within the class of training-free models, it is the clear winner essentially across the board. These results do not only demonstrate the virtues of AnomalyDINO itself but, more generally, highlight the merits of strong visual features as compared to highly engineered architectures. We provide further qualitative results in Appendix A, and discuss the limitations (such as detecting semantic anomalies) and specific failure cases of our approach in Appendix B.

Table 4. **Detection results for other settings**. All results are AUROC values (in %).

Setting	Method	MVTec-AD	VisA
0-shot	WinCLIP	91.8	78.1
	AnomalyCLIP	91.5	82.1
	APRIL-GAN	86.1	78.0
Batched 0-shot	ACR	85.8	/
	MuSc	97.8	92.8
	AnomalyDINO-S (448)	93.0	89.7
	AnomalyDINO-S (672)	94.2	90.7

4.2. Ablation Study

We conduct additional experiments to assess the effect of specific design choices in our pipeline. The full ablation study is provided in Appendix C, here we briefly summarize the main insights. Figure 3 supports the ablation study by highlighting the two key aspects: performance and runtime.

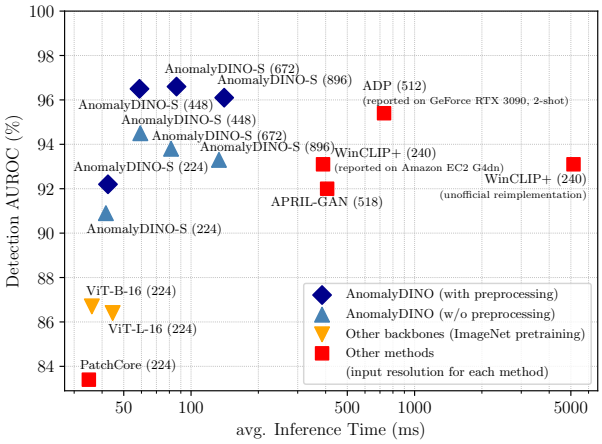


Figure 3. **Detection AUROC vs. inference time** per sample on MVTec-AD in the 1-shot setting. The input resolution is given in parentheses after the method name. All runtimes are measured on a single NVIDIA A40 if not stated otherwise. (Note that for ADP and WinCLIP+ no official code is available.)

Inference time Comparing the inference times of AnomalyDINO with the other approaches in the 1-shot setting (see Figure 3), we observe that AnomalyDINO achieves significantly faster inference times than SOTA

few-shot competitors (note the logarithmic scale). For example, AnomalyDINO-S takes roughly 60ms to process an image at a resolution of 448. The only approaches with lower inference time are PatchCore and AnomalyDINO with ViT-backbones trained on ImageNet, which however sacrifice performance. Notably, while augmentations increase the memory bank, and thus, the potential runtime of the nearest neighborhood search, we find that the effect is negligible in practice.[5] A more detailed runtime analysis is included in Appendix C, Table 9.

Preprocessing To study the impact of preprocessing, we compare the resulting detection AUROCs of AnomalyDINO with and without preprocessing in Figure 3. Across four different configurations of AnomalyDINO, we observe that by incorporating the proposed preprocessing, the resulting AUROCs increase by approximately 2% without significantly affecting the inference time. The effect of proper preprocessing seems to increase with higher resolutions. Additionally, we compare the two different preprocessing settings, 'agnostic' (our default) and 'informed', and find that the agnostic preprocessing slightly outperforms the informed counterpart, see Figures 11 and 12. Depending on the product category, suitable preprocessing steps lead to substantial improvements. And while in principle augmenting the reference sample with rotations may negatively impact the detection performance, we observe this in only very few examples. For the full discussion, see Appendix C.1.

Aggregation statistic We compare the default scoring method, the empirical tail value at risk, to other suitable aggregations in Appendix C.2. The mean of the 1% highest distances from test patches to $\mathcal{M}$ improves over the standard choice (maximum of the upsampled and smoothed patch distances).

Architecture size/choice We also evaluated the proposed framework to Vision Transformers pre-trained on ImageNet, see Figure 3 and Appendix C.4. While these backbones slightly outperform PatchCore, they give weaker features compared to DINOv2 and are incompatible with the proposed masking procedure. This underlines that DINOv2 is indeed excellently suited for visual AD. In addition, ViT-based architectures like DINOv2 easily allow handling images with varying resolutions. In this context, we find that operating at a resolution of 448 gives the best trade-off between performance and inference time, but even higher detection AUROCs are possible at higher resolutions. We also evaluated our pipeline with DINOv2 at different distillation sizes (ViT-S, ViT-B, ViT-L). The full results are given in Appendix C.3, Figure 16. We observe no considerable differences, in particular, also larger backbones give state-of-the-art results on MVTec-AD (all k) and VisA ($k \geq 8$). Interestingly, larger architectures do not necessarily translate to higher performance. The smallest model demonstrates the best performance on MVTec-AD, which primarily consists of single objects and less complex scenes. In contrast, the larger architecture sizes perform better on VisA, which often involves multiple and more complex objects. This suggests the importance of achieving the 'just right' level of abstraction for the specific task, particularly in the few-shot regime.

[5] Our implementation leverages GPU-accelerated neighborhood search [20], leading to only slightly increased inference time. With increasing size of $\mathcal{M}$ reduction techniques such as coreset subsampling [34] are advisable.

5. Conclusion

This paper proposes a vision-only approach for one- and few-shot anomaly detection. Our method is based on similarities between patch representations extracted by DINOv2 [30]. We carefully design means to populate the nominal memory bank with diverse and relevant features while minimizing false positives by utilizing zero-shot segmentation and simple data augmentation. Industrial settings require fast deployment, easy debugging and error correction, and rapid adaptation for covariance shifts in the normal data distribution. Our pipeline caters to these requirements through its simplicity and computational efficiency. The proposed method, AnomalyDINO, achieves state-of-the-art results on MVTec-AD and rivals all competing methods on VisA while outperforming all other training-free approaches. Thus, our approach achieves the best of both worlds: improved performance *and* reduced inference time, especially compared to more complex vision-language methods. Its simplicity and strong performance render AnomalyDINO an excellent candidate for practitioners for industrial anomaly detection and an effective baseline for assessing few-shot and even full-shot anomaly detection in ongoing research.

Follow-up research directions The specific pipeline exemplified in the paper focuses on simplicity and high throughput. However, the individual parts of our pipeline can easily be exchanged for more sophisticated alternatives. For example, our simple masking approach could be replaced by more specialized and adaptive masking techniques (which may also be relevant to other methods building on DINOv2), and the simple upsampling and smoothing approach could be substituted by more sophisticated methods like [15]. It would be interesting to see if that leads to further improvements in anomaly detection and localization, thereby further reducing the gap between few- and full-shot anomaly detectors. We also plan to improve the batched zero-shot performance of AnomalyDINO.

Acknowledgement The authors acknowledge funding from TRR 391 *Spatio-temporal Statistics for the Transition of Energy and Transport* by the German Research Foundation (DFG).

References

[1] Liron Bergman, Niv Cohen, and Yedid Hoshen. Deep nearest neighbor anomaly detection. *CoRR*, abs/2002.10445, 2020. 3, 4

[2] 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. 1, 2, 5, 6

[3] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2022. 2

[4] Yunkang Cao, Xiaohao Xu, Chen Sun, Yuqi Cheng, Zongwei Du, Liang Gao, and Weiming Shen. Segment any anomaly without training via hybrid prompt regularization. *arXiv preprint arXiv:2305.10724*, 2023. 1

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 1, 3

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. 2

[7] Xuhai Chen, Yue Han, and Jiangning Zhang. A zero-/few-shot anomaly classification and segmentation method for cvpr 2023 vand workshop challenge tracks 1&2: 1st place on zero-shot ad and 4th place on few-shot ad. *arXiv preprint arXiv:2305.17382*, 2023. 2, 3, 4, 6

[8] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *CoRR*, abs/2005.02357, 2020. 3, 4, 6

[9] Tae Hyun Kim Daehyun Kim, Sungyong Baik. Sanflow: Semantic-aware normalizing flow for anomaly detection and localization. In *In Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1, 3

[10] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: A patch distribution modeling framework for anomaly detection and localization. In Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani, editors, *Pattern Recognition. ICPR International Workshops and Challenges - Virtual Event, January 10-15, 2021, Proceedings, Part IV*, volume 12664 of *Lecture Notes in Computer Science*, pages 475–489. Springer, 2020. 3, 6

[11] 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. *CoRR*, abs/2010.11929, 2020. 3

[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. In *International Conference on Learning Representations*, 2021. 3

[13] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of data mining in computer security*, pages 77–101, 2002. 3, 4

[14] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep learning for medical anomaly detection – a survey. *ACM Comput. Surv.*, 54(7), jul 2021. 1

[15] Stephanie Fu, Mark Hamilton, Laura E. Brandt, Axel Feldmann, Zhoutong Zhang, and William T. Freeman. Featup: A model-agnostic framework for features at any resolution. In *The Twelfth International Conference on Learning Representations*, 2024. 8

[16] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980. 1

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[19] Jongheon Jeong, Yang Zou, Taewan Kim, Dongqing Zhang, Avinash Ravichandran, and Onkar Dabeer. Winclip: Zero-/few-shot anomaly classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19606–19616, 2023. 1, 3, 6

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 8

[21] Sangkyung Kwak, Jongheon Jeong, Hankook Lee, Woohyuck Kim, Dongho Seo, Woojin Yun, Wonjin Lee, and Jinwoo Shin. Few-shot anomaly detection via personalization. *IEEE Access*, 2024. 1, 2, 3, 6, 20

[22] Aodong Li, Chen Qiu, Marius Kloft, Padhraic Smyth, Maja Rudolph, and Stephan Mandt. Zero-shot anomaly detection via batch normalization. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 3, 4, 6, 21

[23] Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. Multimodal foundation models: From specialists to general-purpose assistants. *arXiv preprint arXiv:2309.10020*, 2023. 2

[24] Xurui Li, Ziming Huang, Feng Xue, and Yu Zhou. Musc: Zero-shot industrial anomaly classification and segmentation with mutual scoring of the unlabeled images. In *International Conference on Learning Representations*, 2024. 3, 4, 6, 7, 21

[25] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 2, 3

[26] Victor Livernoche, Vineet Jain, Yashar Hezaveh, and Siamak Ravanbakhsh. On diffusion modeling for anomaly detection. *CoRR*, abs/2305.18593, 2023. 1, 3

[27] Alexander J McNeil. Extreme value theory for risk managers. *Departement Mathematik ETH Zentrum*, 12(5):217–37, 1999. 4

[28] Arian Mousakhan, Thomas Brox, and Jawad Tayyub. Anomaly detection with conditioned denoising diffusion models. *CoRR*, abs/2305.15956, 2023. 1, 3

[29] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024. 2

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. 1, 2, 3, 5, 8, 18

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 1, 2, 3

[32] Mehdi Rafiei, Toby P Breckon, and Alexandros Iosifidis. On pixel-level performance assessment in anomaly detection. *arXiv preprint arXiv:2310.16435*, 2023. 6

[33] Tal Reiss, Niv Cohen, Eliahu Horwitz, Ron Abutbul, and Yedid Hoshen. Anomaly detection requires better representations. In *European Conference on Computer Vision*, pages 56–68. Springer, 2022. 1

[34] 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, 3, 4, 5, 6, 8, 19

[35] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Fully convolutional cross-scale-flows for image-based defect detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022, Waikoloa, HI, USA, January 3-8, 2022*, pages 1829–1838. IEEE, 2022. 1

[36] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. 3

[37] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018. 3

[38] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 1, 3

[39] João Santos, Triet Tran, and Oliver Rippel. Optimizing patchcore for few/many-shot anomaly detection. *arXiv preprint arXiv:2307.10792*, 2023. 3

[40] Md Amran Siddiqui, Jack W Stokes, Christian Seifert, Evan Argyle, Robert McCann, Joshua Neil, and Justin Carroll. Detecting cyber attacks using anomaly detection with explanations and expert feedback. In *ICASSP 2019-2019 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2872–2876. IEEE, 2019. 1

[41] Yu Tian, Fengbei Liu, Guansong Pang, Yuanhong Chen, Yuyuan Liu, Johan W. Verjans, Rajvinder Singh, and Gustavo Carneiro. Self-supervised pseudo multi-class pretraining for unsupervised anomaly detection and segmentation in medical images. *Medical Image Anal.*, 90:102930, 2023. 1

[42] Julia Wolleb, Florentin Bieder, Robin Sandkühler, and Philippe C Cattin. Diffusion models for medical anomaly detection. In *International Conference on Medical image computing and computer-assisted intervention*, pages 35–45. Springer, 2022. 1, 3

[43] Guoyang Xie, Jinbao Wang, Jiaqi Liu, Feng Zheng, and Yaochu Jin. Pushing the limits of fewshot anomaly detection in industry vision: Graphcore. *arXiv preprint arXiv:2301.12082*, 2023. 3, 4, 5, 6

[44] Xiaohao Xu, Yunkang Cao, Yongqi Chen, Weiming Shen, and Xiaonan Huang. Customizing visual-language foundation models for multi-modal anomaly detection and reasoning. *arXiv preprint arXiv:2403.11083*, 2024. 3

[45] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021. 3

[46] 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, 3, 4

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. 3

[48] Hui Zhang, Zheng Wang, Zuxuan Wu, and Yu-Gang Jiang. Diffusionad: Denoising diffusion for anomaly detection. *arXiv preprint arXiv:2303.08730*, 2023. 3

[49] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations*, 2022. 3

[50] Qihang Zhou, Guansong Pang, Yu Tian, Shibo He, and Jiming Chen. AnomalyCLIP: Object-agnostic prompt learning for zero-shot anomaly detection. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 3, 4

[51] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pretraining for anomaly detection and segmentation. In *European Conference on Computer Vision*, pages 392–408. Springer, 2022. 1, 2, 5

Supplementary Material

A. Detailed experimental results

Further anomaly maps, predicted by AnomalyDINO, are presented in Figures 4 to 7. The full results per category for MVTec-AD and VisA are given in Tables 6 and 7, respectively.

The results presented in Tables 6 and 7 show that a) anomalies in some categories are more difficult to than others, b), that the performance of AnomalyDINO increases across the board with more available reference samples. In particular, more complex objects like 'PCB3' and 'PCB4' in VisA or 'Transistor' in MVTec-AD seem to benefit the most from more available reference samples. In addition, we see that also the variance in the reported metrics decreases with the number of nominal samples.

In this context, we observe a peculiarity of the few-shot regime (which does not occur for sufficiently populated $\mathcal{M}$). The results crucially depend on the chosen reference image(s).[6] The high variances, predominantly in the 1- and 2-shot setting, for category 'Capsule' for MVTec-AD, or 'Cashew' or 'PCB4' for VisA demonstrate this. We discuss the potential failure cases of choosing a sub-optimal reference sample in the following section (Appendix B.2).

Finally, we also report the full-shot performance, see Table 5. As the diversity within $\mathcal{M}$ is sufficiently high given the large number of reference samples, we only apply masing here (no augmentations). The results demonstrate that the performance further improves for all considered metrics. Notably, AnomalyDINO-S (672) achieves new state-of-the-art segmentation performance measured in (AU)PRO in the full-shot setting (see here, accessed 11/26/2024).

Table 5. **Full-shot results** on MVTec-AD and VisA with AnomalyDINO-S in the default setting (no std reported as results are deterministic when all samples are considered).

Dataset	Resolution	Detection			Segmentation		
		AUROC	F1-max	AP	AUROC	F1-max	PRO
MVTec-AD	448	99.3	98.8	99.7	97.9	61.8	93.9
	672	99.5	99.0	99.8	98.2	64.3	95.0
VisA	448	97.2	93.7	97.6	98.7	50.5	95.0
	672	97.6	94.5	98.0	98.8	53.8	96.1

B. Limitations and failure cases

The proposed method relies on similarities to patch representations captured in $\mathcal{M}$. Therefore, we can only expect the model to detect those anomalies caused by regions in the test images that are particularly different from patches

[6]Note that this holds for all one- and few-shot methods, not only for AnomalyDINO.

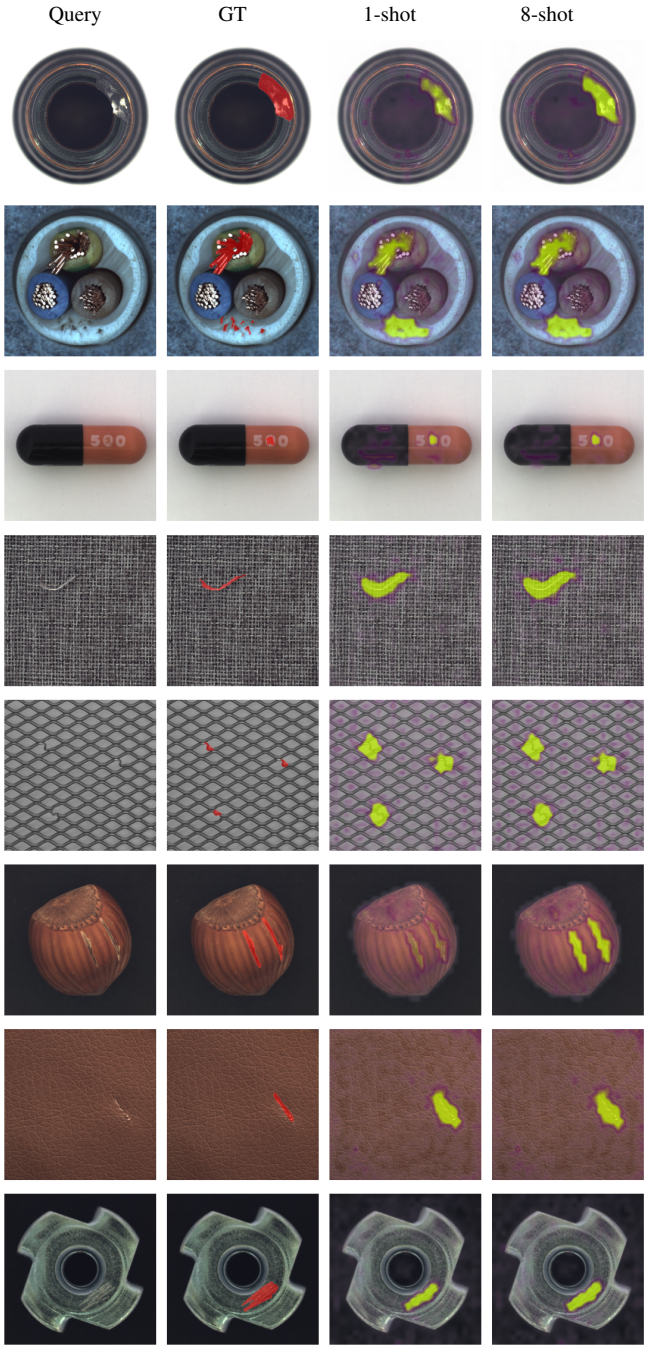


Figure 4. **Examples – MVTec-AD (1/2)**. Depicted are, from left to right, a test sample per category (Query), the ground truth anomaly annotation (GT), and the predicted anomaly map from AnomalyDINO-S (448) in the 1- and 8-shot settings. The color coding is normalized by the max. score over 'good' test samples.

Table 6. **Detailed results on MVTec-AD**. Reported are results on anomaly detection (image-level AUROC) and segmentation (PRO) of AnomalyDINO-S (672) with default preprocessing (mean and standard deviation over three independent runs, all results in %).

Shots	1-shot		2-shot		4-shot		8-shot		16-shot	
	AUROC	PRO	AUROC	PRO	AUROC	PRO	AUROC	PRO	AUROC	PRO
Bottle	99.7±0.2	95.9±0.3	99.9±0.1	96.3±0.0	99.9±0.1	96.7±0.1	100.0±0.0	96.5±0.4	99.9±0.1	96.4±0.4
Cable	92.7±0.8	89.4±0.4	92.4±1.1	89.5±0.4	93.8±0.9	90.4±0.3	95.2±0.3	90.5±0.2	95.1±0.6	90.5±0.6
Capsule	90.2±5.5	97.1±0.3	89.2±7.9	97.3±0.6	95.8±0.5	97.9±0.1	95.6±0.5	97.9±0.1	95.5±0.6	98.1±0.1
Carpet	100.0±0.0	97.8±0.0	100.0±0.0	97.9±0.0	100.0±0.0	97.8±0.0	100.0±0.0	97.8±0.0	100.0±0.0	97.8±0.0
Grid	99.1±0.2	97.2±0.1	99.2±0.4	97.2±0.1	99.5±0.3	97.2±0.1	99.5±0.1	97.2±0.0	99.7±0.3	97.0±0.2
Hazelnut	97.5±2.6	97.4±0.4	99.6±0.5	98.0±0.3	99.8±0.1	98.0±0.1	100.0±0.0	98.1±0.1	100.0±0.0	98.1±0.1
Leather	100.0±0.0	97.9±0.1	100.0±0.0	97.8±0.0	100.0±0.0	97.6±0.1	100.0±0.0	97.6±0.1	100.0±0.0	97.3±0.2
Metal nut	99.9±0.1	94.2±0.0	100.0±0.0	94.6±0.2	100.0±0.0	95.3±0.1	100.0±0.0	95.4±0.3	100.0±0.0	95.7±0.1
Pill	93.7±0.9	97.3±0.1	95.4±0.7	97.5±0.1	96.0±0.2	97.6±0.1	97.2±0.2	97.7±0.1	97.9±0.1	97.8±0.1
Screw	93.2±0.3	93.4±0.4	93.5±0.8	94.3±0.4	92.7±2.3	94.5±0.9	93.5±1.1	95.2±0.5	94.7±0.9	95.9±0.3
Tile	100.0±0.0	88.0±0.2	100.0±0.0	87.6±0.4	100.0±0.0	87.1±0.4	100.0±0.0	86.7±0.2	100.0±0.0	86.0±0.5
Toothbrush	97.4±0.5	94.0±0.8	98.1±1.0	94.7±0.3	97.5±0.6	94.7±0.3	97.7±1.6	95.1±0.8	98.1±1.8	95.8±1.0
Transistor	90.9±1.2	67.3±2.1	89.4±4.6	68.4±3.1	93.2±2.2	70.6±1.4	96.2±1.3	75.3±1.9	97.6±0.3	78.2±1.4
Wood	98.0±0.2	94.7±0.1	98.0±0.1	94.6±0.0	97.9±0.2	94.6±0.1	98.3±0.4	94.4±0.3	98.3±0.6	94.2±0.4
Zipper	97.4±0.9	89.2±1.2	98.9±0.4	90.2±0.4	99.0±0.4	91.2±0.3	99.6±0.1	91.1±0.4	99.6±0.3	91.7±0.5
Mean	96.6±0.4	92.7±0.1	96.9±0.7	93.1±0.2	97.7±0.2	93.4±0.1	98.2±0.2	93.8±0.1	98.4±0.1	94.0±0.1

Table 7. **Detailed results on VisA**. Reported are results on anomaly detection (image-level AUROC) and segmentation (PRO) of AnomalyDINO-S (672) with default preprocessing (mean and standard deviation over three independent runs, all results in %).

Shots	1-shot		2-shot		4-shot		8-shot		16-shot	
	AUROC	PRO	AUROC	PRO	AUROC	PRO	AUROC	PRO	AUROC	PRO
Candle	87.9±0.3	96.8±0.4	89.4±3.0	97.0±0.2	91.3±2.9	97.2±0.1	93.5±1.2	97.3±0.2	94.5±0.5	97.6±0.2
Capsules	98.4±0.5	95.1±0.7	98.9±0.1	95.5±0.2	99.2±0.1	96.3±0.4	99.2±0.1	96.7±0.2	99.2±0.2	97.2±0.3
Cashew	86.1±3.6	96.1±0.9	89.4±3.8	96.7±0.7	94.5±0.7	97.4±0.5	95.3±0.6	97.3±0.2	96.0±0.3	97.3±0.1
Chewinggum	98.0±0.4	92.0±1.0	98.6±0.4	92.9±0.3	98.8±0.2	93.0±0.1	98.8±0.2	93.1±0.3	98.8±0.2	93.0±0.3
Fryum	94.8±0.5	93.2±0.2	96.5±0.2	93.9±0.3	97.0±0.1	94.5±0.4	97.6±0.4	94.9±0.3	97.9±0.2	95.1±0.0
Macaroni1	87.5±1.1	97.5±0.3	87.5±0.9	97.9±0.3	89.5±1.4	98.3±0.2	90.1±1.7	98.6±0.2	90.4±1.1	98.7±0.1
Macaroni2	62.2±4.3	92.0±0.7	66.9±1.9	93.0±0.4	70.0±1.7	93.9±0.8	74.9±0.4	95.0±0.6	77.6±0.8	95.7±0.3
PCB1	91.5±2.0	92.6±0.2	91.2±2.7	92.5±0.5	94.0±2.1	93.3±0.5	95.5±0.5	93.9±0.2	96.8±0.7	94.2±0.2
PCB2	84.8±1.2	89.9±0.2	88.1±2.5	90.7±0.3	91.1±1.7	91.4±0.2	92.6±0.3	92.0±0.1	93.2±0.1	92.5±0.2
PCB3	84.9±3.3	88.5±1.3	89.4±3.8	90.8±0.5	94.3±0.4	91.7±0.4	95.6±0.2	93.1±0.3	96.5±0.3	93.9±0.3
PCB4	79.9±13.7	78.5±6.8	87.4±11.3	82.0±6.1	96.2±2.6	84.1±1.5	98.0±0.3	87.9±2.3	99.0±0.4	90.4±1.8
Pipe fryum	92.7±2.7	98.0±0.0	93.3±1.2	97.9±0.2	94.6±1.9	97.8±0.1	95.0±1.6	97.6±0.1	97.2±0.9	97.7±0.1
Mean	87.4±1.2	92.5±0.5	89.7±1.3	93.4±0.6	92.6±0.9	94.1±0.1	93.8±0.3	94.8±0.2	94.8±0.2	95.3±0.2

of the given reference sample(s). Our experiments reveal that this may lead to some specific failure cases.

B.1. Semantic Anomalies

The first relates to the distinction between low-level sensory anomalies and high-level semantic anomalies. Semantic anomalies might be present because a logical rule or a specific semantic constraint is violated. In contrast, AnomalyDINO is developed with low-level sensory anomalies in mind. Consider, for instance, the anomalies shown in Figure 8.

While the cable with anomaly type 'Bent Wire' (Fig-

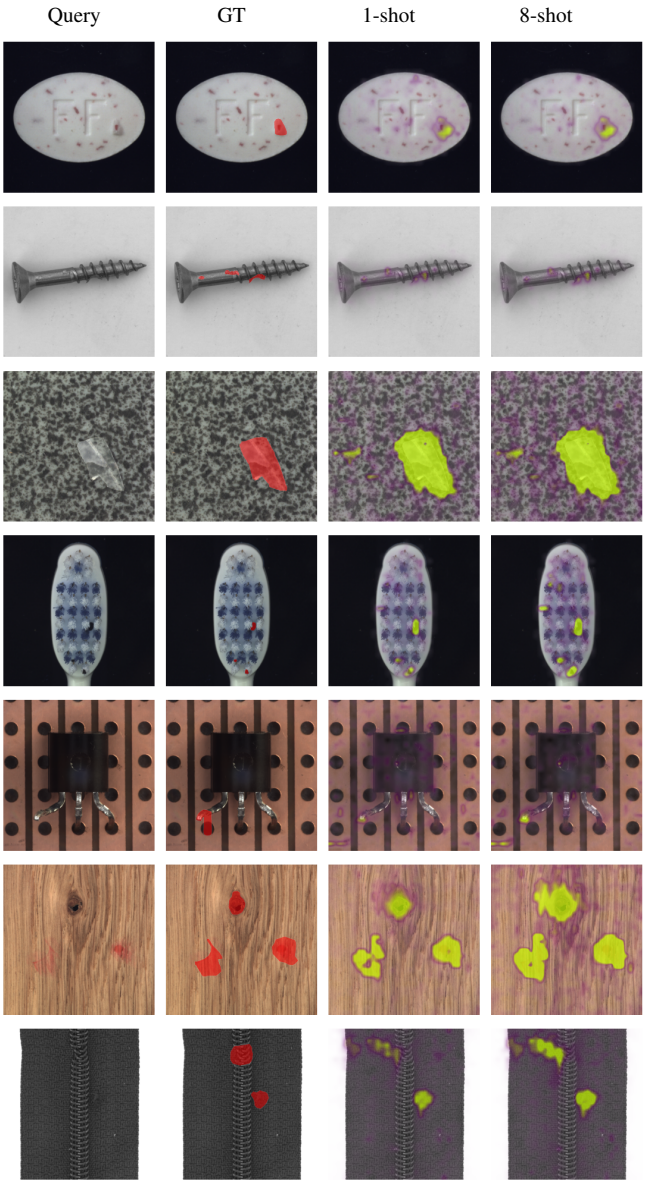


Figure 5. **Examples – MVTec-AD (2/2)**. See Fig. 4 for a description.

ure 8b) contains patches that cannot be well matched to patches of the reference image, this does not apply to the anomaly 'Cable Swap' (Figure 8c). The latter shows a *semantic anomaly* with two blue wires, while a nominal image of a 'Cable' should depict all three different cable types. As a result, all test patches of Figure 8b can be matched well with reference patches in $\mathcal{M}$, and thus, AnomalyDINO does not detect this anomaly type. Evaluating the detection performance for this specific failure case, 'Cable Swap', our proposed method essentially performs on chance level, giving a detection AUROC of 50.2% ($\pm$ 4.9%).

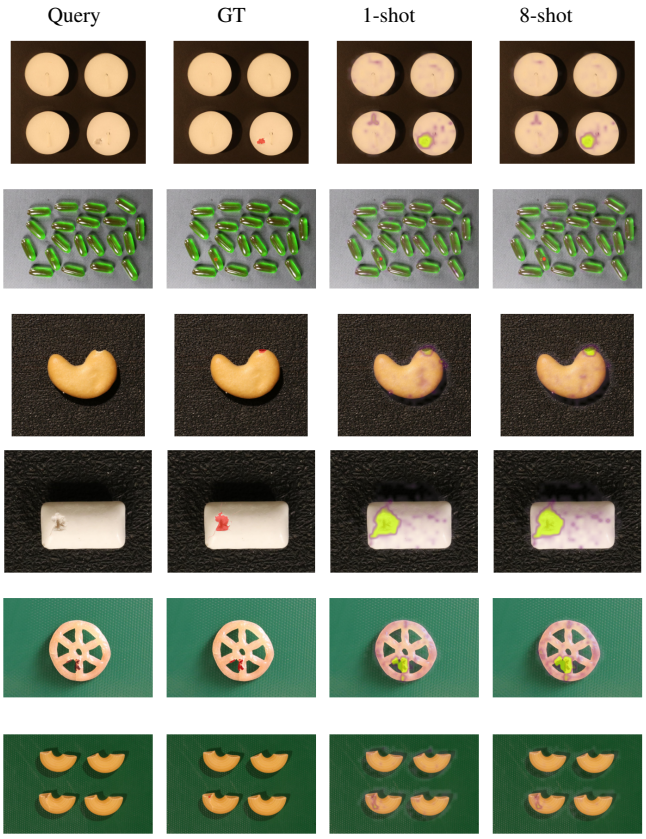


Figure 6. **Examples – VisA (1/2)**. Depicted are, from left to right, a test sample per category (Query), the ground truth anomaly annotation (GT), and the predicted anomaly map from AnomalyDINO-S (448) in the 1- and 8-shot settings. The color bar is normalized by the maximum score on the 'good' test samples (per category). For the category 'Capsules' (left column, second from top) we changed the color map for better visibility. Best viewed at a higher zoom level as some anomalies are quite small.

B.2. The importance of informative reference samples

The second failure case occurs if the reference sample(s) does not resemble all concepts of normality, therefore $\mathcal{M}$ does not capture all variations of the nominal distribution p_{norm}. As an illustration, consider the nominal samples of 'Capsule' in MVTec-AD, which may be rotated in such a way that the imprinted text is hidden (see Figure 9b). Therefore, parts of the text of a nominal test sample may be falsely recognized as anomalies. Due to the strong dependency on a suitable reference sample, we observe higher AD variances for some products in the one-shot setting, as shown in Tables 6 and 7. We like to remark, that this is only relevant to the few-shot setting, and with an increasing number of reference samples (and higher diversity of nominal patches in $\mathcal{M}$), the variance decreases notably.

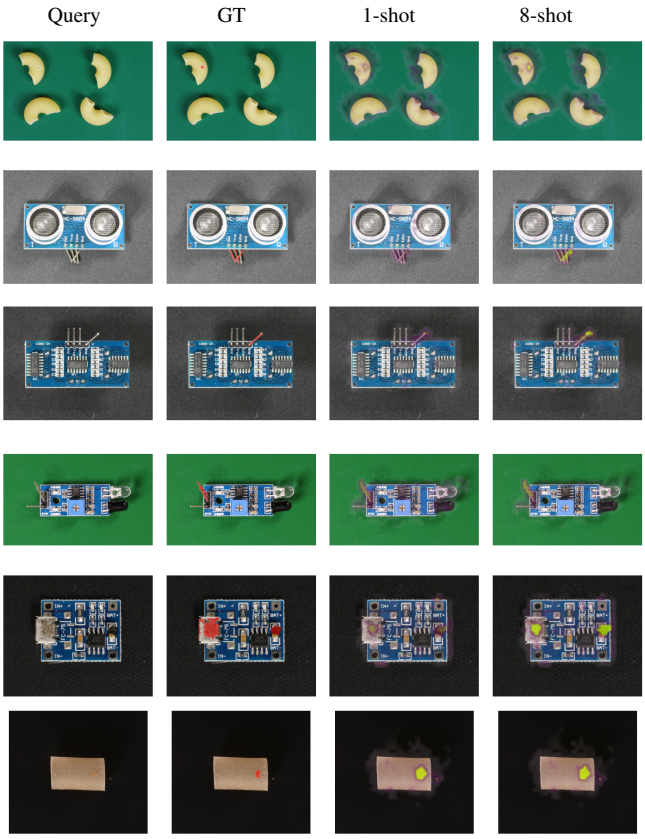
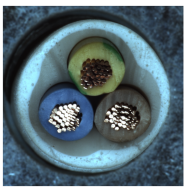
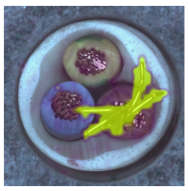


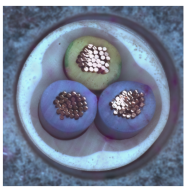
Figure 7. **Examples – VisA (2/2)**. See Fig. 6 for the description.



(a) Reference sample (No anomalies)



(b) Anomaly type 'Bent Wire'



(c) Anomaly type 'Cable Swap'

Figure 8. **Example of a semantic anomaly** in category 'Cable' (MVTec-AD). Depicted are a nominal reference sample, a sensory anomaly (Figure 8b), and a semantic anomaly (Figure 8c) with anomaly maps predicted by AnomalyDINO (1-shot).

(a) Reference sample (left) and estimated anomaly map of 'good' test sample (right).

(b) Reference sample (left) and estimated anomaly map of 'good' test sample (right).

Figure 9. **Example of an uninformative reference sample** in category 'Capsule' (MVTec-AD). Some reference samples do not resemble the full concept of normality (here, the sample in Figure 9b does not show the text on the capsule, i.e., any nominal sample with text be considered anomalous). Anomaly maps predicted by AnomalyDINO based on the depicted reference sample.

C. Ablation Study

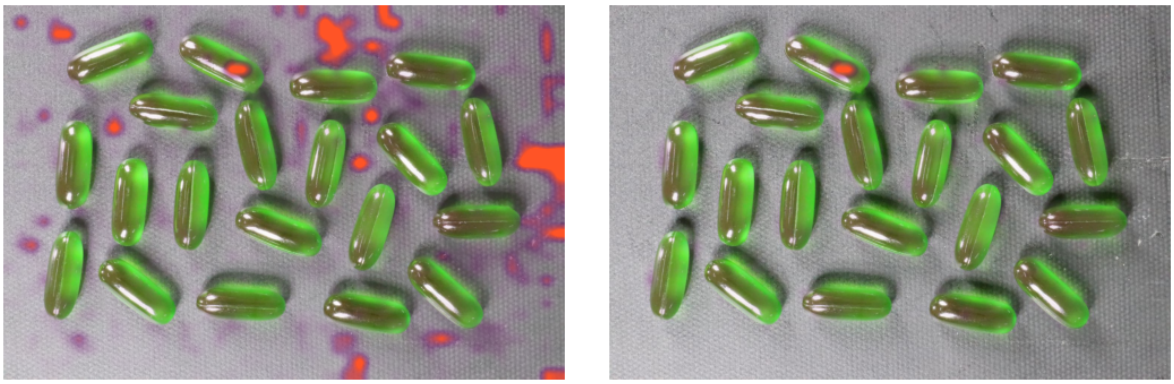
C.1. Preprocessing

As discussed in Section 3, we consider two potential preprocessing steps in our pipeline: masking and rotations. We mask out irrelevant background patches, whenever the zero-shot segmentation of DINOv2 captures the object correctly (see Figure 2). Discarding background patches helps to mitigate the problem of potential background noise, thereby reducing the number of false positives. A representative example, here from the category 'Capsules' from the VisA dataset, is depicted in Figure 10. Without masking, the method would correctly predict the depicted sample to show an anomaly, but for the wrong reason (high anomaly scores caused by background noise/contamination). In contrast, the irrelevant background areas are discarded by our masking approach, such that the anomalous region is correctly identified. We conclude that masking is essential to faithfully detect anomalies whenever higher variations in the background are expected—and as the issue of low variations is particularly pressing in the few-shot regime, caused by the minimal amount of nominal reference samples, suitable masking can significantly boost the performance in these cases. This is showcased by the superior anomaly localization performance A quantitative analysis is provided in the next paragraphs.

In addition to masking (which decreases the size of $\mathcal{M}$), we consider augmenting the reference sample with rotations (which increases the size and diversity of $\mathcal{M}$). Here we distinguish the 'agnostic' scenario, where we do not know a priori about potential rotations, and augment by default. Whenever we know about potential rotations of reference or test samples, we can deactivate the augmentation to reduce the size of the memory bank $\mathcal{M}$, the time to construct $\mathcal{M}$, as well as the test time itself.

The preprocessing decision per category for MVTec-AD

(a) Test sample (left), the same test sample with ground-truth anomaly annotation (center), and a magnified view of a region with strong background artifacts (right).



(b) Anomaly map predicted by AnomalyDINO (1-shot) without masking (left) and with masking (right).

Figure 10. **Visualization of the effect of masking** in the presence of high background noise for the category 'Capsules' in VisA (1-shot). As in Figure 6 we depict the anomaly map for 'Capsules' using a different colormap (red instead of yellow) for better visibility. Best viewed on a higher zoom level.

and VisA, inferred (solely) from the first nominal reference sample in X_{ref} (to comply with the one-shot setting), are given in Table 8.

Table 8. **Default preprocessing steps for MVTec-AD and VisA**. We do not mask textures, as indicated by (T). In addition, we do not apply masking when the masking test on the first train sample failed, as indicated by (MT). See Section 3 and Figure 2 for further discussion and visualization.

MVTec-AD	Mask?	Rotation?	
Object		informed	agnostic
Bottle	✗(MT)	✗	✓
Cable	✗(MT)	✗	✓
Capsule	✓	✗	✓
Carpet	✗(T)	✗	✓
Grid	✗(T)	✗	✓
Hazelnut	✓	✓	✓
Leather	✗(T)	✗	✓
Metal nut	✗(MT)	✗	✓
Pill	✓	✗	✓
Screw	✓	✓	✓
Tile	✗(T)	✗	✓
Toothbrush	✓	✗	✓
Transistor	✗(MT)	✗	✓
Wood	✗(T)	✗	✓
Zipper	✗(MT)	✗	✓

VisA	Mask?	Rotation?	
Object		informed	agnostic
Candle	✓	✗	✓
Capsules	✓	✗	✓
Cashew	✓	✗	✓
Chewinggum	✓	✗	✓
Fryum	✓	✗	✓
Macaroni1	✓	✗	✓
Macaroni2	✓	✗	✓
PCB1	✓	✗	✓
PCB2	✓	✗	✓
PCB3	✓	✗	✓
PCB4	✓	✗	✓
Pipe fryum	✓	✗	✓

Effect of preprocessing on detection performance As discussed in Section 3 (and in the previous paragraph), suitable means to fill $\mathcal{M}$ and preprocess test samples, influence the detection performance. The results per object for MVTec-AD and VisA are given in Figures 11 and 12, respectively.

Rotation Increasing the diversity of nominal patch representations in $\mathcal{M}$ by rotating the reference sample can significantly improve the detection performance. Consider, e.g., the category 'Screw' in MVTec-AD, where the detection AUROC can be boosted from 65.6% to 89.2% in the 1-shot setting. This is intuitive, as the test samples of 'Screw' are taken from various angles. With sufficiently many reference samples, such data augmentation is not necessary anymore, but for the few-shot setting, we see major improvements.

The same holds—although to a lesser extent—for the categories 'Hazelnut', 'Cable', and 'Wood' (all MVTec-AD). However, we also observe that rotations of the reference sample can also *decrease* the detection performance in some categories, namely 'PCB1/2/3' and 'Macaroni1' (all VisA), and to a small extent also 'Transistor' (MVTec-AD).

We attribute this to the fact the specific anomalies for

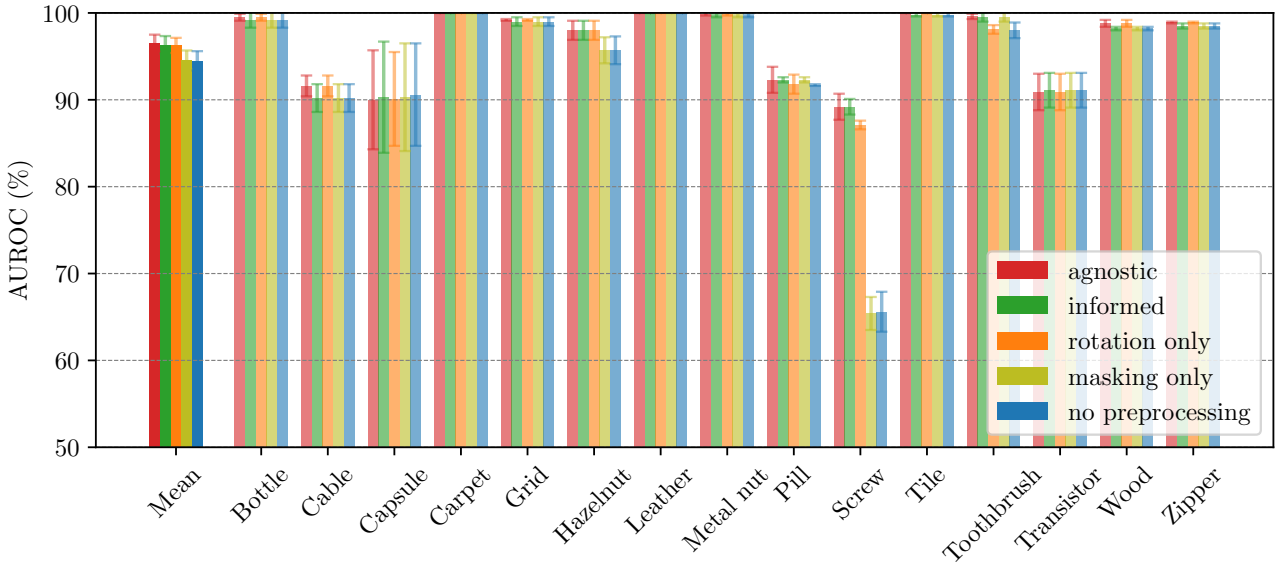


Figure 11. **Effect of preprocessing on MVTec-AD**. Anomaly detection of AnomalyDINO-S (448) in the 1-shot setting for different choices of the preprocessing pipeline (detection AUROC on image-level in %, mean and standard deviation over three independent runs).

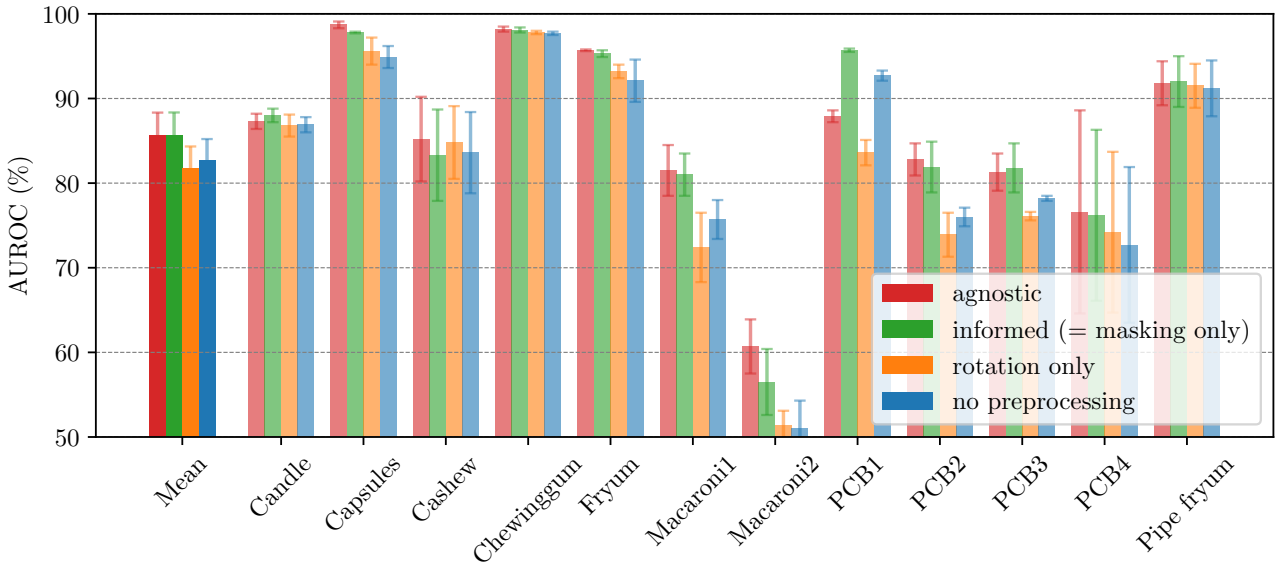


Figure 12. **Effect of preprocessing on VisA.** Anomaly detection with AnomalyDINO-S (448) in the 1-shot setting for different choices of the preprocessing pipeline (detection AUROC on image-level in %, mean and standard deviation over three independent runs). For VisA, the 'informed' scenario is equivalent to only applying masking (all categories), while 'agnostic' is equivalent to masking and augmentations (all categories), see Table 8.

these printed circuit boards contain rotated or bent connectors (see the PCB examples in Figure 6, right column). And rotations of nominal samples may falsely reduce the distance between those patches depicting bent connectors and the nominal memory bank $\mathcal{M}$.

Such a failure case based on a suboptimal preprocessing decision is depicted in Figure 13. The anomaly refers to a rotated transistor (anomaly type 'misplaced'), and when (falsely) rotating the reference sample, such anomalies will not be detected—in contrast to the 'informed' preprocessing (right side of Figure 13). This highlights the importance of carefully designing the pre- and postprocessing pipeline for each object/product considered.

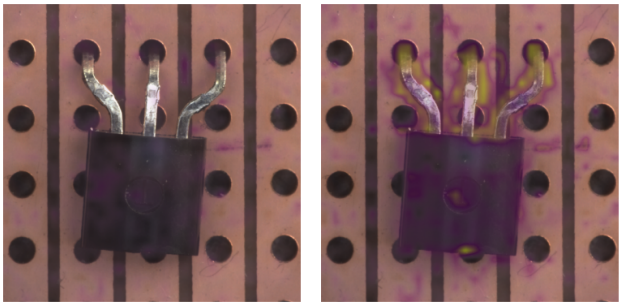


Figure 13. **Rotations as anomalies** ('misplaced' transistor from MVTec-AD). The left anomaly map is estimated from a reference sample *with* rotations ('agnostic'), and the anomaly is not detected. In contrast, the right anomaly map is based on a reference sample *without* rotations ('informed'), and the anomaly is successfully detected. This example highlights the importance of a carefully designed preprocessing pipeline for each object.

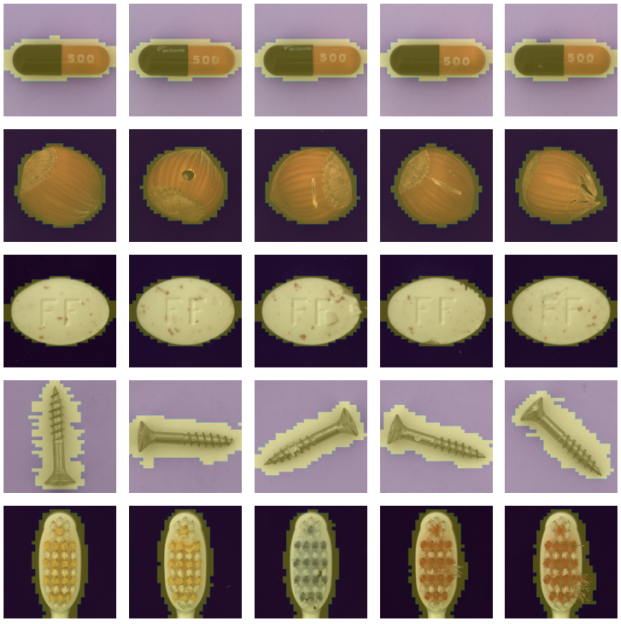


Figure 14. **Masking examples from MVTec-AD** for all categories that passed the masking test (on a single train sample, see Table 8).

Masking We utilize the zero-shot masking capabilities of DINOv2 to keep the overhead for this preprocessing step minimal. By applying a threshold to the first principal component [30], we can typically distinguish between the background and the foreground.

We also employ a straightforward rule-based strategy to enhance the robustness and generalizability of the PAC-based masking in industrial settings: ensuring that the center crop of the image is predominantly occupied by the object of interest. This minor adjustment is necessary because

Figure 15. **Masking examples from VisA**.

industrial images often differ significantly from the data on which DINOv2 was originally trained. In addition, we use dilation and morphological closing to improve the quality of the mask. In our experiments, we applied masking only to the test samples as the size of $\mathcal{M}$ did not matter in the few-shot regime. Across the board, we see that the proposed masking technique improves the detection performance—in many categories even significantly, e.g., for 'Capsules', 'Macaroni1/2', or 'PCB1/2/3/4' (all VisA). We leave further improvements and the exploration of more advanced masking techniques for future work.

Runtime analysis The design choices in our pipeline influence the run time of the proposed method. To see the potential effects we measure the inference time per sample on MVTec-AD in various scenarios together with the time to set up the memory bank $\mathcal{M}$. The runtime was assessed utilizing a single NVIDIA A40 GPU, consistently employed throughout all experiments detailed in this paper (each ex-

Table 9. **Runtime analysis** on MVTec-AD (mean and std of inference time over all 1725 test samples, and mean and std of time to populate the memory bank for each object) for different shots (Tab. 9a), preprocessing choices (Tab. 9b), sample resolutions (Tab. 9c) and model sizes (Tab. 9d). All times are reported in seconds, measured on a single NVIDIA A40 with GPU warmup and CUDA kernel synchronization. The default setting is 1-shot, agnostic preprocessing, model size S, and resolution 448 (underlined for reference).

(a) Runtime in dependence of shots.

Shots	Inference	Memory Bank
1	$0.060_{\pm 0.012}$	$0.52_{\pm 0.04}$
2	$0.059_{\pm 0.012}$	$0.85_{\pm 0.07}$
4	$0.059_{\pm 0.012}$	$1.58_{\pm 0.08}$
8	$0.060_{\pm 0.011}$	$3.05_{\pm 0.17}$
16	$0.063_{\pm 0.012}$	$6.02_{\pm 0.39}$
full (masking only)	$0.067_{\pm 0.010}$	$16.62_{\pm 4.43}$
full (agnostic)	$0.130_{\pm 0.044}$	$93.72_{\pm 20.56}$

(b) Runtime in dependence of preprocessing choices.

Mask?	Rotate?	Inference	Memory Bank
no	no	$0.055_{\pm 0.011}$	$0.17_{\pm 0.03}$
no	yes	$0.055_{\pm 0.012}$	$0.51_{\pm 0.05}$
yes	no	$0.068_{\pm 0.012}$	$0.18_{\pm 0.03}$
yes	yes	$0.067_{\pm 0.012}$	$0.51_{\pm 0.04}$
informed		$0.059_{\pm 0.016}$	$0.22_{\pm 0.11}$
agnostic		$0.060_{\pm 0.012}$	$0.52_{\pm 0.04}$

(c) Runtime in dependence of image resolution.

Resolution	Inference	Memory Bank
224	$0.043_{\pm 0.010}$	$0.31_{\pm 0.04}$
448	$0.060_{\pm 0.012}$	$0.52_{\pm 0.04}$
672	$0.086_{\pm 0.014}$	$0.75_{\pm 0.06}$
896	$0.141_{\pm 0.021}$	$1.26_{\pm 0.05}$

(d) Runtime in dependence of model size.

Model Size	Inference	Memory Bank
S (21 M)	$0.060_{\pm 0.012}$	$0.52_{\pm 0.04}$
B (86 M)	$0.084_{\pm 0.021}$	$0.77_{\pm 0.05}$
L (300 M)	$0.141_{\pm 0.029}$	$1.24_{\pm 0.06}$
G (1,100 M)	$0.306_{\pm 0.034}$	$2.67_{\pm 0.14}$

periment can be executed on a single NVIDIA A40 GPU). The runtime is measured with GPU warmup and CUDA kernel synchronization for a fair comparison.

The results are given in Table 9. The average inference time per sample with AnomalyDINO-S (448) amounts to approximately 60ms in the few-shot regime ($\approx$ 16.7 samples/s), and only moderately increases with a larger memory bank (to 67ms ($+11\%$) for the full-shot scenario without augmentations). Compared to SOTA competitors in the one-shot regime, this is at least one order of magnitude faster (see Figure 3). Without any preprocessing steps, the 1-shot inference time amounts to approximately 55ms per sample ($\approx$ 18 samples/s). When applying both masking and rotations, the runtime increases from 55ms to 67ms, a moderate increase of roughly 23% (compared to the scenario without any preprocessing steps). The informed scenario can reduce the time to build the memory bank, but inference time is only affected for larger sample sizes. As expected, higher resolutions and larger architectures lead to increased runtimes.

C.2. Scoring

In Section 3 we investigate different ways of aggregating the anomaly scores $\mathcal{D}$ on patch-level (in our case, distances to the nominal memory bank $\mathcal{M}$) to an image-level anomaly score via a statistic q. Note that the segmentation results are therefore not affected by the choice of q. A standard choice is upsampling the patch distances of lower resolution to the full resolution of the test image using bilinear interpolation, then applying Gaussian smoothing operation (we follow [34] and set $\sigma = 4.0$), to obtain an anomaly map $\mathcal{A}$, and set $q = \max(\mathcal{A})$. We also evaluate two potential alternatives of q, our default choice $q = \text{mean}(H_{0.01}(\mathcal{D}))$ (mean of the 1% highest entries in $\mathcal{D}$) and $q = \max(\mathcal{D})$.

The results in Table 10 show that just the maximum patch distance leads to already good results while upsampling and smoothing the patch distances giving slightly weaker results (maximum of $\mathcal{A}$).

Taking the mean of all patch distances above the 99% quantile ($q = \text{mean}(H_{0.01}(\mathcal{D}))$) improves above the standard choice. We did not optimize over the percentile and instead fixed it to 99%. Typically, the number of patches per image ranges between 200 and 1000 (depending on resolution and masking), such that 2 between 10 patches are considered. For specific products/objects and expected anomaly types, other statistics q might be (more) suitable.

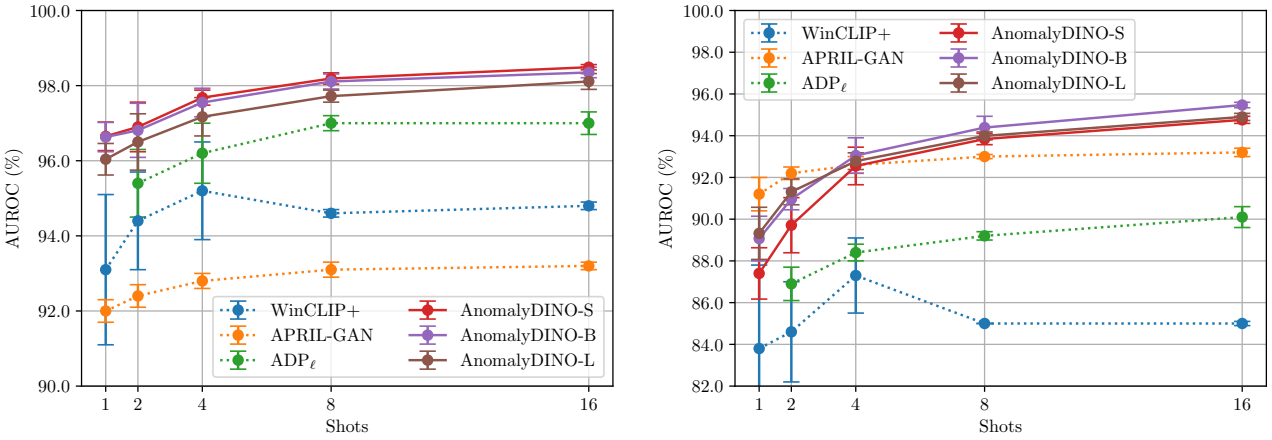
C.3. Architecture Size

DINOv2 is available in different distillation sizes (S, B, L, and G). This section analyzes the implications of choosing different backbone sizes for AnomalyDINO. The comparison including the best competing methods is depicted in Figure 16. We see that different architecture sizes have indeed an effect on the image-level AUROC.

On MVTec-AD, we see that AnomalyDINO-S performs best, followed by the next larger model AnomalyDINO-B

Table 10. **Effect of aggregation statistics** q on the detection performance on MVTec-AD and VisA, evaluated for AnomalyDINO-S (448). $\mathcal{D}$ denotes the set of patch distances to the nominal memory bank, $H_{0.01}(\mathcal{D})$ the 1% highest entries thereof, and $\mathcal{A}$ denotes the anomaly map derived from $\mathcal{D}$ (anomaly scores for each pixel, after upsampling and smoothing, see Section 3). All results in %.

	Scoring	$q = \text{mean}(H_{0.01}(\mathcal{D}))$			$q = \max(\mathcal{D})$			$q = \max(\mathcal{A})$		
	Shots	AUROC	F1-max	AP	AUROC	F1-max	AP	AUROC	F1-max	AP
MVTec-AD	1	$96.5_{\pm 0.4}$	$96.0_{\pm 0.2}$	$98.1_{\pm 0.3}$	$95.0_{\pm 0.5}$	$94.7_{\pm 0.2}$	$97.4_{\pm 0.4}$	$94.9_{\pm 0.7}$	$94.6_{\pm 0.6}$	$97.5_{\pm 0.4}$
	2	$96.7_{\pm 0.8}$	$96.5_{\pm 0.4}$	$98.1_{\pm 0.7}$	$95.5_{\pm 1.0}$	$95.3_{\pm 0.5}$	$97.4_{\pm 0.6}$	$95.0_{\pm 1.3}$	$94.8_{\pm 0.8}$	$97.4_{\pm 0.9}$
	4	$97.6_{\pm 0.1}$	$97.0_{\pm 0.3}$	$98.4_{\pm 0.3}$	$96.6_{\pm 0.2}$	$96.0_{\pm 0.1}$	$97.8_{\pm 0.4}$	$96.3_{\pm 0.4}$	$95.7_{\pm 0.2}$	$98.1_{\pm 0.3}$
	8	$98.0_{\pm 0.1}$	$97.4_{\pm 0.1}$	$99.0_{\pm 0.2}$	$97.2_{\pm 0.1}$	$96.5_{\pm 0.4}$	$98.6_{\pm 0.1}$	$97.0_{\pm 0.0}$	$96.3_{\pm 0.1}$	$98.6_{\pm 0.1}$
	16	$98.3_{\pm 0.1}$	$97.7_{\pm 0.2}$	$99.3_{\pm 0.0}$	$97.6_{\pm 0.2}$	$96.9_{\pm 0.3}$	$98.9_{\pm 0.1}$	$97.4_{\pm 0.1}$	$96.6_{\pm 0.2}$	$98.8_{\pm 0.1}$
VisA	1	$85.6_{\pm 1.5}$	$83.1_{\pm 1.1}$	$86.6_{\pm 1.3}$	$82.4_{\pm 1.9}$	$81.4_{\pm 1.0}$	$84.0_{\pm 1.7}$	$80.5_{\pm 1.3}$	$80.6_{\pm 0.8}$	$82.1_{\pm 0.9}$
	2	$88.3_{\pm 1.8}$	$84.8_{\pm 1.2}$	$89.2_{\pm 1.3}$	$85.1_{\pm 1.8}$	$82.5_{\pm 1.3}$	$86.4_{\pm 1.1}$	$83.3_{\pm 1.8}$	$82.2_{\pm 1.1}$	$84.5_{\pm 1.5}$
	4	$91.3_{\pm 0.8}$	$87.5_{\pm 1.0}$	$91.8_{\pm 0.7}$	$88.4_{\pm 0.3}$	$84.9_{\pm 0.6}$	$89.0_{\pm 0.4}$	$86.8_{\pm 1.4}$	$84.3_{\pm 0.9}$	$87.6_{\pm 1.3}$
	8	$92.6_{\pm 0.1}$	$88.6_{\pm 0.2}$	$92.9_{\pm 0.2}$	$90.0_{\pm 0.3}$	$86.3_{\pm 0.2}$	$90.6_{\pm 0.5}$	$88.8_{\pm 0.5}$	$85.3_{\pm 0.3}$	$89.7_{\pm 0.3}$
	16	$93.8_{\pm 0.1}$	$89.9_{\pm 0.3}$	$94.2_{\pm 0.3}$	$91.6_{\pm 0.5}$	$87.2_{\pm 0.7}$	$92.2_{\pm 0.4}$	$90.5_{\pm 0.3}$	$86.8_{\pm 0.5}$	$91.5_{\pm 0.4}$



(a) Detection AUROC for MVTec-AD.

(b) Detection AUROC for VisA.

Figure 16. **Effect of model size** on detection AUROC for MVTec-AD and VisA (mean and std over three seeds). The image resolution of AnomalyDINO is set to 672. Note, that the results for WinCLIP+ for 8 and 16 shots are those of the WinCLIP re-implementation [21].

(which might contrast the common belief that larger models always perform better). In particular, all architecture sizes outperform the closest competitor (ADP$_\ell$).

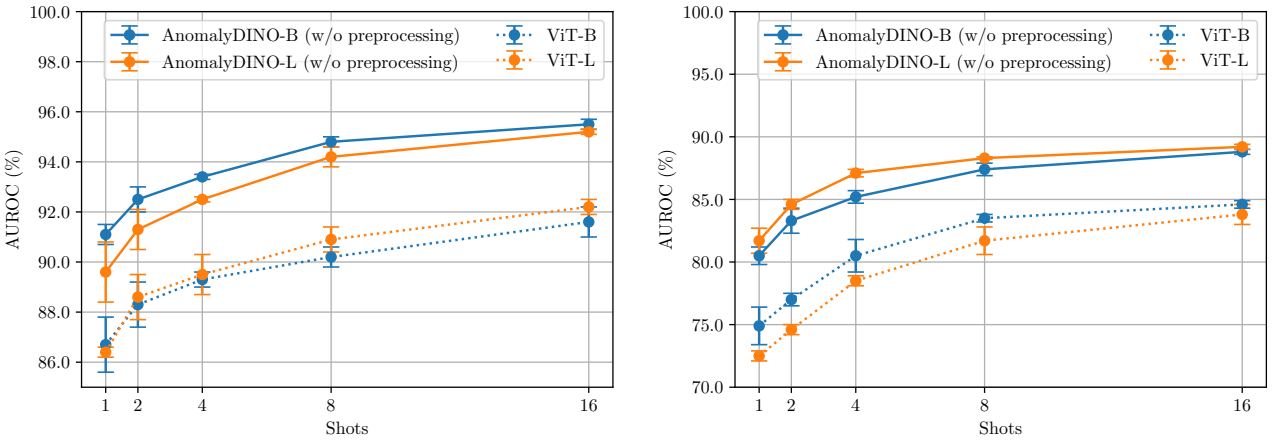
Regarding VisA, larger architectures slightly outperform our default setting, AnomalyDINO-S. All architecture sizes are on par with APRIL-GAN at $k = 4$, but outperform all competitors for $k > 4$. Figure 16 also showcases that the performance of the proposed method scales preferably with the number of reference samples.

C.4. Backbone Choice

In our experiments, we find that DINOv2 provides excellently suited features for few-shot AD. This is already evident in Fig. 3, here we investigate the effect on the detection performance on MVTec-AD and VisA in more detail.

We compare the performance in few-shot AD of ViT-B and ViT-L pre-trained on ImageNet to that of AnomalyDINO-B and AnomalyDINO-L (utilizing DINOv2 in distillation sizes B and L). To ensure comparability, we deactivate any pre- and postprocessing for AnomalyDINO (no augmentations, no zero-shot masking) and set the image resolution to 224 for all models.

The results are depicted in Figure 17. We observe that DINOv2 significantly improves the detection performance compared to those of the ViT pre-trained on the image classification tasks, giving a performance gain of at least $+4\%$ AUROC on MVTec-AD and VisA. This demonstrates that the features extracted by DINOv2 are better suited com-



(a) Detection AUROC for MVTec-AD.

(b) Detection AUROC for VisA.

Figure 17. **Effect of backbone choice** DINOv2 (trained in a self-supervised fashion) compared to ViT (supervised training on ImageNet) on detection AUROC for MVTec-AD and VisA (mean and std over three seeds). For better comparability, image resolution of AnomalyDINO is 224 for all models.

pared to those of ImageNet-pretrained ViT-{B/L}.

In Sections 3 and 4 we demonstrate that further (substantial) improvements are possible with suitable pre- and postprocessing like augmentations and zero-shot masking (which is not possible based on the features from supervised features) and that the image resolution (or the effective patch size) can also greatly boost performance. See also Fig. 16 for the performance in dependence of the model size.

D. Extending AnomalyDINO to the Batched Zero-Shot Setting

We extend the proposed method to the *batched* zero-shot setting. Recall, that in this setting all test samples X_{test} are provided (or at least a sufficiently large batch), but *no* (labeled) training or reference samples. The underlying (and necessary) assumption to meaningfully predict anomalies solely based on test samples, is that the majority of samples (or in our case, patches) at test time are from the nominal data distribution (see e.g., [22]).

We need to alter our method, outlined in Section 3, only slightly to adapt it to the batched setting. We score a test sample $\mathbf{x}^{(j)} \in X_{\text{test}}$ in comparison to all remaining test sample $X_{\text{test}} \setminus \{\mathbf{x}^{(j)}\}$, following the idea of mutual scoring [24]. For each test sample $\mathbf{x}^{(j)} \in X_{\text{test}}$ we therefore collect all patch representations not belonging to $\mathbf{x}^{(j)}$ in a memory bank, again utilizing DINOv2 as patch-level feature extractor f,

$$\mathcal{M}_j := \bigcup_{\mathbf{x}^{(i)} \in X_{\text{test}} \setminus \{\mathbf{x}^{(j)}\}} \left\{ \mathbf{p}_m \mid f(\mathbf{x}^{(i)}) = (\mathbf{p}_1, \ldots, \mathbf{p}_n),\, m \in [n] \right\} . \tag{5}$$

We need to infer anomaly scores for each patch representation $\mathbf{p}_{\text{test}}$ of $\mathbf{x}^{(j)}$. We could again assess the distances between $\mathbf{p}_{\text{test}}$ and $\mathcal{M}_j$ based on the distance to the nearest neighbor, as done in Equation (2). Note, however, that $\mathcal{M}_j$ may now contain nominal *and* anomalous patches. Thus, the nearest neighbor approach is not suitable anymore: the nearest neighbor in $\mathcal{M}_j$ for (the representation of) an anomalous patch might also be abnormal, and the resulting distance therefore not informative.

A simple solution, based on the assumption that the majority of patches are nominal, is to replace the nearest neighbor with a suitable aggregation statistic over the distribution of patches distances in $\mathcal{M}_j$

$$\mathcal{D}(\mathbf{p}_{\text{test}}, \mathcal{M}_j) := \left\{ d(\mathbf{p}_{\text{test}}, \mathbf{p}) \mid \mathbf{p} \in \mathcal{M}_j \right\} , \tag{6}$$

where we again use the cosine distance d, defined in Equation (3). Specifically, we can again make use of the tail value at risk—now for the lowest quantile as we are interested in the tail behavior of the distribution of distances to the nearest neighbors—to derive a patch-level anomaly score

$$s(\mathbf{p}_{\text{test}}) := \text{mean}\big(L_\alpha\big(\mathcal{D}(\mathbf{p}_{\text{test}}, \mathcal{M}_j)\big)\big) , \tag{7}$$

where $L_\alpha(\mathcal{D})$ contains the values below the α quantile of $\mathcal{D}$. We set $\alpha = 0.1\%$ as anomalous patches are rare by assumption, and $\mathcal{D}(\mathbf{p}_{\text{test}}, \mathcal{M}_j)$ large enough to accurately estimate the tail of the distribution.[7] The image-level score $s(\mathbf{x}_{\text{test}})$ is again given by aggregating the patch-level anomaly score following Equation (4). Computation of the

[7] For MVTec-AD the total number of test patches extracted by DINOv2 (ViT-S) at a resolution of 448 range between 43.008 and 171.008 per category, and for VisA between 163.200 and 334.464.

cosine distances and the proposed aggregation statistics can be effectively implemented as matrix operations on GPU such that the batched zero-shot inference time for AnomalyDINO amounts to roughly 60 ms/sample for MVTec-AD at a resolution of 448 (again measured on an NVIDIA A40 GPU). Some resulting anomaly maps in the batched zero-shot setting are depicted in Figure 18.

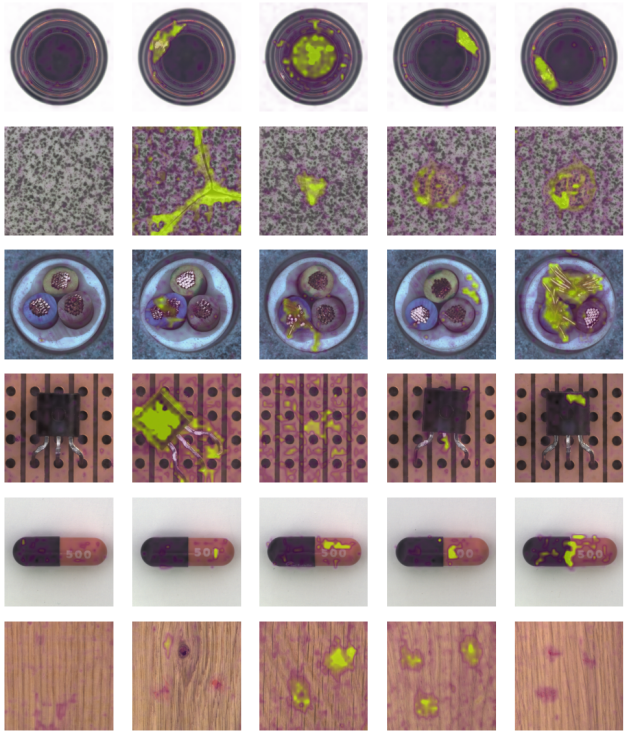


Figure 18. **Anomaly maps for the batched zero-shot setting** on MVTec-AD. The left-most sample in each category is a 'good' test sample for reference, followed by four randomly picked samples with anomalies.

E. Broader Impacts

Advancing few-shot visual anomaly detection methods can offer various benefits by enhancing manufacturing quality control through the rapid identification of defects with minimal nominal examples, which improves efficiency, reduces waste, and improves overall safety in the product lifecycle. Similar positive benefits can be expected outside the industrial domain, e.g., for healthcare diagnostics or environmental monitoring. It is, however, essential to recognize the shortcomings of automated anomaly detection systems. We believe that simpler methods can be adapted more quickly, monitored more effectively, and are therefore more reliable. In this context, awareness of the risks of over-reliance must be heightened (see Appendix B for identified failure cases of the proposed method). In addition, strong visual anomaly detectors could also lead to potentially malicious or unintended uses. To address these concerns, including potential privacy infringements and possible socioeconomic impacts of automation, strategies such as establishing robust data governance, and implementing strict privacy protections are essential. Additionally, investing in workforce development can help manage the socioeconomic effects of automation and leverage the full potential of strong visual anomaly detection.