# Supervised Anomaly Detection for Complex Industrial Images

Aimira Baitieva[1], David Hurych[2], Victor Besnier[2], Olivier Bernard[1]

[1]valeo, [2]valeo.ai

{aimira.baitieva, david.hurych, victor.besnier, olivier.bernard}@valeo.com

## Abstract

*Automating visual inspection in industrial production lines is essential for increasing product quality across various industries. Anomaly detection (AD) methods serve as robust tools for this purpose. However, existing public datasets primarily consist of images without anomalies, limiting the practical application of AD methods in production settings. To address this challenge, we present (1) the Valeo Anomaly Dataset (VAD), a novel real-world industrial dataset comprising 5000 images, including 2000 instances of challenging real defects across more than 20 subclasses. Acknowledging that traditional AD methods struggle with this dataset, we introduce (2) Segmentation-based Anomaly Detector (SegAD). First, SegAD leverages anomaly maps as well as segmentation maps to compute local statistics. Next, SegAD uses these statistics and an optional supervised classifier score as input features for a Boosted Random Forest (BRF) classifier, yielding the final anomaly score. Our SegAD achieves state-of-the-art performance on both VAD (+2.1% AUROC) and the VisA dataset (+0.4% AUROC). The code and the models are publicly available*[1].

## 1. Introduction

Within the manufacturing process, industrial visual inspection plays a crucial role in identifying defects in produced components. This operation holds significant importance in minimizing costs by identifying and removing faulty parts early in the production stages and, more importantly, in preventing the dispatch of defective components to customers. Traditionally, this task has relied on human operators; however, the likelihood of overlooking certain defects can be as high as 25% for specific defect types [9].

When the inspected product comprises numerous components, its examination may create a bottleneck in the production process, causing delays across the entire line. While conventional computer vision methods applied to this task demonstrate superior speed and lower error rates compared to human operators [16], their inflexibility and lack of satisfactory accuracy [15] limit their effectiveness.

Industrial deep learning anomaly detection has been an active research field in recent years. Most of anomaly detection methods use only good images for training and try to detect deviations from the training data [1, 7, 23, 29]. Development of the new methods is restrained by publicly available datasets. Recent industrial anomaly detection datasets typically contain approximately one hundred (or even fewer) abnormal images, showcasing defects in the testing set only [2, 32]. This poses a challenge for supervised anomaly detection methods aiming to utilize both normal and defective parts during training. Supervised models often undergo training with just ten abnormal images from the testing set, resulting in overfitting and reduced sensitivity to previously unseen defects [28].

In response to the limitations of current datasets, we introduce and publicly release **VAD** (Valeo Anomaly Dataset), which contains 1000 bad and 2000 good parts in the training set and 1000 bad and 1000 good parts in the test set, see Fig. 1(a). Among the defective parts used for testing, 165 contain specific defect types not present in the training data; these parts are explicitly labeled in the released dataset. All images in VAD are captured from an actual production line, showcasing a diverse range of defects, from highly obvious to extremely subtle. This dataset bridges the gap between the academic community and the industry, offering researchers the chance to advance the performance of methods in tackling more intricate real-world challenges.

Current approaches to supervised anomaly detection either yield unsatisfactory results; see Tab. 3, or demand pixel-level labels for defective parts, as seen in [28, 31]. Consequently, we introduce a novel method named **SegAD** (Segmentation-based Anomaly Detector), which is described in Fig. 1(b). The employed approach eliminates the need for pixel-level labels, requiring only a flag for each image. Anomaly map scores from each segment are used

[1] https://github.com/abc-125/segad

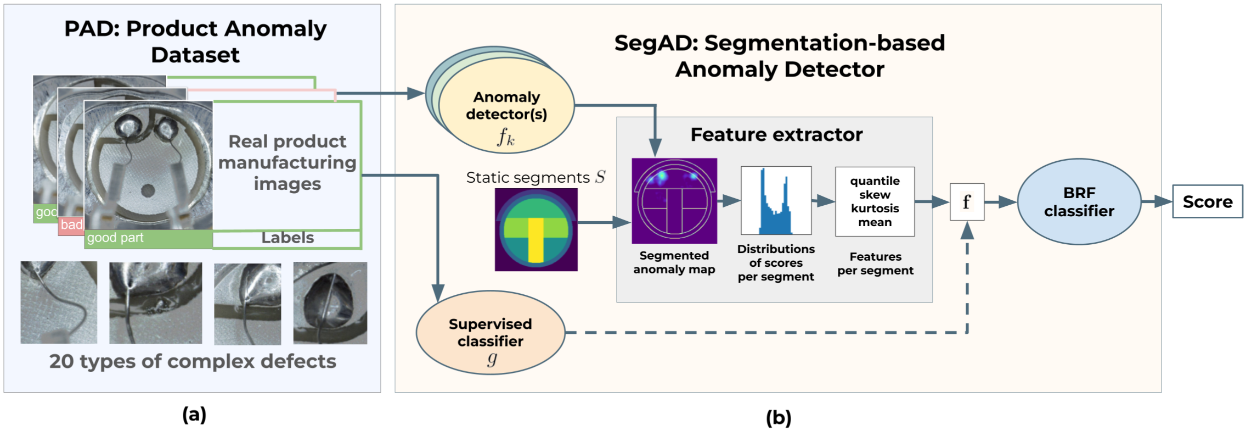


Figure 1. Overview of our contributions. **(a)** VAD, a real-world industrial dataset designed for supervised anomaly detection with complex defects. **(b)** SegAD, our method that leverages anomaly maps extracted from segmented outputs of one or more anomaly detectors. Higher-level statistical features are computed on these maps, such as skewness, kurtosis, or mean, to generate the local anomaly features. Additionally, our SegAD provides the flexibility to use the output of a supervised classifier score with the local anomaly features, creating input for a final Boosted Random Forest (BRF) classifier that yields the final score.

to delineate the parameters of the distribution, emphasizing relatively low scores in important areas while disregarding higher scores in noisy regions. In this context, the precise position and appearance of the anomaly diminishes in importance as long as it resides within a designated segment.

Our main contributions can be summarized as follows:

- We propose a supervised anomaly detection dataset with complex objects and a large variety of defects. As an addition, we establish a one-class anomaly detection benchmark, which is more challenging than most of the current ones, a supervised benchmark with a high number of bad images for training (1000 images) and a supervised benchmark with a low number of bad images for training (100 images). We share the dataset freely with the research community.
- We create a novel supervised anomaly detection method called SegAD. This approach performs noticeably better than recent anomaly detectors alone while retaining the ability to detect unknown defects. SegAD reaches SOTA results on VAD as well as on the established VisA dataset [32].

## 2. Related Work

In this section, we first discuss existing datasets and explain how VAD is different. Next, we give a quick overview of existing anomaly detection methods. We show in Table 1 a global comparison between our VAD and existing datasets.

### 2.1. Datasets

In 2019, the MVTec AD dataset [2] was introduced. This dataset significantly advanced anomaly detection methods by providing images with authentic defects and realistic objects, in contrast to earlier datasets predominantly comprising textures and simple defects [4, 26, 27]. MVTec LOCO AD [3] highlighted the challenge of logical defects, such as missing or misplaced parts, necessitating anomaly detection methods capable of capturing the global context of the image. The more recent **VisA** dataset [32] expands on this by incorporating diverse objects with complex structures, multiple instances, and variations in scale.

Existing datasets often contain simulated defects [2, 3, 32], creating a domain gap between research and practical applications. This complicates the deployment of anomaly detection methods in real-world settings. To promote supervised anomaly detection, we propose the Valeo Anomaly Dataset (VAD), a real-world industrial dataset with various defects, including logical ones.

Another notable issue stems from the saturation of solved datasets, where other methods are influenced more by dataset-specific design choices than general applicability. Compared to this, our dataset introduces a broader range of challenging and diverse anomalies and high intra-class variability of good images.

### 2.2. Methods

To define the terms we are working with, "one-class" methods are commonly called unsupervised in the context of anomaly detection [10, 15, 29]. We use "one-class" as a more accurate name since these methods use labeled images for training. It is important to distinguish such methods from ones that work with unlabelled data, as in [17]. Models that use bad and good images for training are usually called "supervised", which is the name we also adopt.

Table 1. Comparison of different anomaly detection datasets. Tr. good and Tr. bad show the average number of images per class for training. tex = *texture*, obj = *object*, Log. def. = *Logical defects*, Pix. labels = *pixel-level labels* and wl = *weakly labelled* (ellipse around the defect).

| Dataset | Year | Classes | Images | Types | Defects | Tr. good | Tr. bad | Log. def. | Pix. labels |
|---|---|---|---|---|---|---|---|---|---|
| DAGM [27] | 2007 | 10 | 11500 | tex | synthetic | 500 | 75 | no | wl |
| KSSD [26] | 2019 | 1 | 399 | tex | real-world | 115 | 17 | no | yes |
| MVTec AD [2] | 2019 | 15 | 5354 | tex, obj | simulated | 242 | 0 | no | yes |
| KSSD2 [4] | 2021 | 1 | 3335 | tex | real-world | 2085 | 246 | no | yes |
| BTAD [19] | 2021 | 3 | 2830 | tex, obj | real-world | 600 | 0 | no | yes |
| LOCO AD [3] | 2022 | 5 | 3644 | obj | simulated | 354 | 0 | yes | yes |
| VisA [32] | 2022 | 12 | 10821 | obj | simulated | 720 | 0 | no | yes |
| VAD (ours) | 2023 | 1 | 5000 | obj | real-world | 2000 | 1000 | yes | no |

**One-class Anomaly Detection.** **PatchCore** [23] relies on a pretrained feature extractor model to extract features from the training set into a memory bank and reduce the size of a memory bank using coreset subsampling. Features extracted from the new input are compared to their nearest neighbors in the memory bank. **FastFlow** [29] uses a similar feature extractor model and maps extracted features to the Gaussian distribution using normalizing flow [22]. **RD4AD** [7] utilizes a teacher-student architecture, which combines features from several layers of feature extractor to eliminate redundant ones. **EfficientAD** [1] introduces many innovations, including using an autoencoder to detect logical defects as an addition to a classic teacher-student architecture and their own pretrained feature extractor, which imitates the behavior of a bigger model with a noticeably lesser inference time.

**Supervised Anomaly Detection.** In theory, leveraging defective parts for training is advantageous for refining class boundaries [24]. However, practical challenges arise, such as the small size of anomalies and the impossibility of collecting all potential defects [31]. Several recent anomaly detection methods use both good and bad images [21, 28, 31], but some of them overfit to seen defects, and others require pixel masks for defects to calculate loss or to generate new defects, which can be problematic in a real-world case. Real-world datasets (as VAD) might contain logical defects that cannot be generated by existing defects generation strategies, which usually cut and paste existing defects with various modifications [28, 31], which is not going to improve results for a wrong wire shape as or other similar types of defects.

Very few supervised anomaly detection benchmarks are available; the most popular is Supervised Anomaly Detection on MVTec AD. **DevNet** [21] is one of the first supervised anomaly detection models that tried to solve industrial MVTec AD dataset, compared to earlier methods which were mostly addressing Out-of-Distribution problem [11, 24] using non-industrial datasets such as MNIST [14] or CIFAR-10 [13]. **DRA** [8] uses several heads to learn both seen and pseudo anomalies, as well as normal examples, which should reduce overfitting on seen anomalies. These methods perform well on MVTec AD but might fail on more complex problems, as shown in Tab. 3 and Tab. 5. Compared to modern supervised anomaly detection methods, SegAD (ours) performs better than current SOTA methods, even on complex problems without pixel-level labels, extensive augmentation, and long training.

## 3. Valeo Anomaly Dataset (VAD)

VAD consists of one class with predefined training and testing sets. The training set contains 1000 bad and 2000 good images, and the testing set contains 1000 bad, 165 of them are unseen bad, and 1000 good images. Unseen bad images in the test dataset refer to several rare defect types that are not present in the training data, several examples of which are shown in Fig. 4. Having such images is important to avoid turning this anomaly detection problem into a supervised classification problem. Some images might have a thin black border at the bottom due to how they were filmed, an example can be seen in Fig. 2. Defects might occur in the whole area of the image. Image-level annotation is provided, but there is no pixel-level annotation due to the complexity of defects and the fact there is no exact position for missing or misplaced components. All images are 512 × 512 pixels in PNG format. Examples of all types of defects can be found in Appendix C.

Good parts consist of two wires connected to two pins on one side and two soldering dots on the other, placed on a piezo. Piezo means piezoelectric element, a big round area under the other components. Wires should have the right shape, not too straight, too long, or bent too much. Wires should connect the soldering dots as close to their centers as possible. Some amount of deviation is allowed, but if the wire is too close to the side of a solder dot, it is a defect. Soldering should be well-rounded, not too small, and placed in the correct location, not overlapping with the

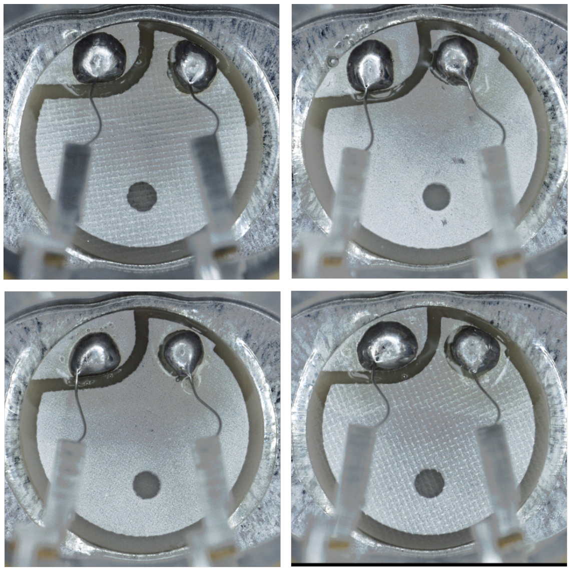


Figure 2. Good parts. Small scratches on the piezo are allowed, as well as wire being closer to the side of the soldering. (VAD)

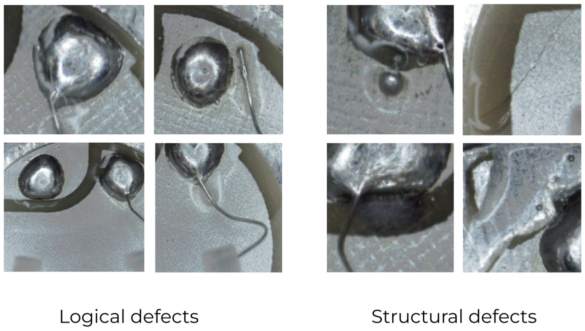


Figure 3. Variety of defects: logical, from the top left: wire on the side, wire out of solder dot, missing wire, bad wire shape (bent too much). Structural: soldering paste pollution, crack on a piezo, burned area under the solder dot, broken piezo. (VAD)

piezo contours. The wire should be properly connected to the pin without any pieces of it visible on the side of the pin. The piezo can have different textures. Any kind of pollution is considered to be an anomaly, which makes the part defective.

Logical defects result from components being misplaced or misshapen rather than being damaged, e.g. bad wire shape, bad soldering shape, wrong wire position, wrong soldering position, and missing parts. Structural defects include various cracks and broken parts of the piezo, as well as a variety of burns and pollution. For both types of defects, see Fig. 3. The importance of logical defects is often overlooked in the existing datasets (excluding MVTec LOCO AD) because they are hard to imitate if the dataset is filmed in the lab, but also hard to detect. VAD provides the research community the opportunity to work with real logical constraints of a complex object.

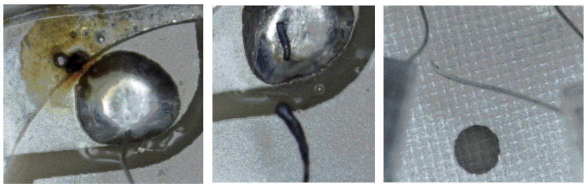


Figure 4. Unseen defects for left to right: burned solder dot, burned wire, extra unnecessary wire. (VAD)

### 3.1. Benchmarks

We establish three benchmarks for the VAD dataset. The first is **one-class anomaly detection**. As shown in Tab 2, even SOTA methods struggle to perform well on our dataset compared to other popular datasets, which leaves an opening for further research. For this benchmark, no bad images are used for training. The second benchmark is **high-shot supervised anomaly detection**, which uses all 1000 bad images from the training set. The last benchmark is **low-shot supervised anomaly detection**. Only 100 bad images from the training set are used. These images are randomly selected using 5 seeds, with the average result over all seeds being expected. The same 2000 good parts are used for the training in all benchmarks.

## 4. Method

SegAD works with image data at its input, and after inference, it is expected to output a score that describes a probability of anomaly appearance. Let $\mathcal{I} = [\![0, \dots, 255]\!]^{W,H,3}$ be a space of all images of width $W$, height $H$ and with 3 color channels. The $i-$th image from a dataset is marked as $I^{(i)} \in \mathcal{I}$. We denote $\mathbf{x}$ to be a 2D pixel coordinate vector. SegAD inference consists of three consecutive stages in a pipeline. First is anomaly map(s) calculation with optional supervised classifier score calculation. Next is the calculation of simple statistics from anomaly maps over individual segments. Lastly, the obtained statistics, and optionally also the classifier score, are used as input features for a BRF classifier [18] that delivers the final result.

Let us denote $K$ as the number of used pixel-wise anomaly detectors and mark them as functions $f_k(I) : \mathcal{I} \rightarrow \mathbb{R}^{W,H}, k \in \{1, \dots, K\}$. A supervised classifier is a function $g(I) : \mathbb{R}^{W,H} \rightarrow \mathbb{R}$. Next, we introduce a set of $L$ mutually exclusive segments $S = \{s_1, \dots, s_L\}$, where $s_l \in [\![0, 1]\!]^{W,H}$.

After the calculation of anomaly maps for all $f_k(I)$, the masks $s_l$ are used for segment-wise extraction of results, followed by the calculation of several statistics. The final feature vector $\mathbf{f} = \left[g(I), [\mathbf{q}^L]_{k=1}^K, [\mathbf{z}^L]_{k=1}^K, [\mathbf{c}^L]_{k=1}^K, [\mathbf{m}^L]_{k=1}^K\right]$, extracted from one image, is the concatenation of 4 vectors containing different statistics. $\mathbf{q}_k^L$ stands for a vector of $L$ numbers where each stands for a $99.5\%$ quantile from anomaly map $f_k(I)$ over pixels where $s_l(\mathbf{x}) = 1$. Similarly, for every anomaly map $f_k(I)$ we calculate a vector for skew $\mathbf{z}_k^L$, kurtosis $\mathbf{c}_k^L$ and mean $\mathbf{m}_k^L$. The length of the feature vector $\mathbf{f}$ equals to $K * L * 4 + 1$.

For notation simplicity, we keep the supervised classifier score $g(I)$ always present in $\mathbf{f}$, but it may be omitted, which is clearly defined in every experiment setup. Feature vectors $\mathbf{f}^{(i)}$ are used as inputs for training the final classifier [18], which, at inference time, delivers the final result.

### 4.1. Training

Training SegAD requires two separate subsets from the training dataset. The first subset is used to train base models $f_k$, the second subset is used to train the BRF classifier. To create these subsets from one training dataset, we split available good images into two parts. For VAD, we report average results over 5 different splits to show more reliable results.

Each model ($f_k$, $g$, BRF) is trained independently. Training can be separated into two stages. In the first stage, anomaly detector(s) $f_k$ are trained using the same data. In the same stage, a segmentation map $S$ must be defined. As will be elaborated in more detail in Sec. 5, the segmentation map can be static or produced by a segmentation model, depending on the dataset. A supervised classifier $g$ is trained separately on 80% of the full training set (1600 good and 800 bad parts for VAD) to ensure that the training sets for this model and SegAD are not fully overlapping. The next stage is to train a final BRF classifier in SegAD, which requires both good and bad images, bad images can be replaced with artificial defects in some cases, as shown in Tab. 2. BRF is trained with the output of the models from the previous stage.

An important thing to note is that SegAD is very fast to train as long as you already have other models available. Extraction of features and training the final classifier takes noticeably less time than training anomaly detectors. The same is true for the inference; SegAD adds a negligible time to processing the image while improving results significantly.

### 4.2. Setups for SegAD

SegAD can use any anomaly detector(s) $f_k$ as long as it produces an anomaly map showing pixel-level anomaly scores, although, in this paper, we use only one-class anomaly detectors because they require less data to train. Any supervised classifier $g$ that returns a classification score can be used. In experiments, we use several setups, which will be described here in detail. SegAD is denoted as Ours. Supervised classifier $g$ is not used unless stated otherwise.

**Anomaly Detector + Ours** means $K = 1$, $f_1 =$ "Anomaly Detector name". This setup allows us to show improvement for one anomaly detection model in particular and demonstrate that this improvement can be achieved for different types of anomaly detectors.

**All AD + Ours** denotes that all anomaly detectors from the list $AllAD$ of length $N$ are used to produce anomaly maps. Such a list can be found in the caption of the table. In that case $K = N$, $f_1 = AllAD_1, \ldots, f_N = AllAD_N$. Anomaly detectors might use an ensemble of feature extractors, similarly to [23], which makes them slower but can improve the result. In our case, anomaly detectors themselves can be successfully assembled in a supervised way.

**All AD + Supervised Classifier + Ours** denotes a setup in which $g =$ "Supervised Classifier name", and the rest is the same as in the previous setup. A supervised classifier allows us to detect seen defects, which can be invisible to a one-class classifier, similarly to [25]. Such a setup shows the best result for academic purposes, but the speed of inference can be unsatisfying for real-world applications. We expect that, in that case, the number and selection of anomaly detectors can be optimized to fit the problem as well as the time constraints.

**Anomaly Detector + Supervised Classifier + Ours** mean a setup in which $g =$ "Supervised Classifier name", $K = 1$, $f_1 =$ "Anomaly Detector name". It shows an example of an optimization mentioned in the previous setup. Only one anomaly detector gives a higher inference speed compared to the ensemble of anomaly detectors while giving a satisfying result.

## 5. Experiments

**Evaluation metrics.** We compare results using the following metrics: (1) Area Under the Receiver Operating Characteristic Curve (AUROC) [5], (2) False Positive Rate (FPR) at 95% True Positive Rate (TPR) denoted as FPR@95TPR. Classification AUROC, which describes the performance on the image level for multiple thresholds, denoted by Cl. AUROC. FPR@95TPR shows the percentage of misclassified good parts (false positive) at 95% of bad parts classified correctly (true positive). We report mean results ± standard deviation.

**Implementation and evaluation details.** SegAD uses the output of one-class detectors PatchCore [23], FastFlow [29], Reverse Distillation [7] (referred to as RD4AD), EfficientAD [1] either alone or several anomaly detectors at once. SegAD results are compared to the results of these models alone, as well as supervised anomaly detectors DevNet [21] and DRA [8]. We also compare our results to the supervised classifier Wide ResNet50 [30] (denoted as

WRN). The final classifier BRF was trained with the same parameters for all datasets using XGBoost library [6]. More details on the implementation can be found in Appendixes A and B.

Images are resized to W = H = 256 pixels to make the comparison less biased. The DRA model originally uses a resolution 448 × 448, which shows better results on VAD than 256 × 256, 92.8 Cl. AUROC, which is still lower than the top results. The higher resolution also improves results for some of the other models, so we do not use it. 256 × 256 is sufficient for defects to remain visible, but it also allows to have a reasonable speed of inference, especially for anomaly detection models.

For VAD, results averaged for 5 seeds and training runs are reported. These seeds are used for training and splitting the training set for SegAD into two sets, as described in Subsection 4.1. 2000 good parts from the train set were split in half: 1000 for training the base model(s) and 1000 for training SegAD. 0, 1000, or 100 bad parts (depending on the benchmark) were also used to train SegAD. Details on the VisA benchmark can be found in Subsection 5.4 No augmentations were used for training, and existing augmentations were removed from used methods. Augmentations tend to be suitable only for specific tasks (as rotation in DRA or DevNet, removing it improved results by these models on the VAD drastically), and using augmentations makes it hard to compare models themselves.

We include three benchmarks for the new **VAD** and one additional for the **VisA** dataset [32]. We compare results for VAD for one-class, high-shot, and low-shot supervised benchmarks, as described in Subsection 3.1. The benchmark for the VisA dataset is explained in 5.4. The segmentation maps for VAD is the same static image for every input image from the dataset. The segmentation map can be seen in Fig. 5. Due to the fact that the object in the VAD is always centered with the constant size and component placement, this is the simplest choice, which may be sub-optimal, but serves the purpose and does not require any manual or model-inferred annotation of the data. It was created to separate the anomaly maps into several meaningful parts, such as piezo border, wire areas, soldering area, etc., which can contain different defects and different levels of falsely high scores on the anomaly map. Another source of segmentation maps can be a specially trained segmentation model or a zero-shot segmentation model.

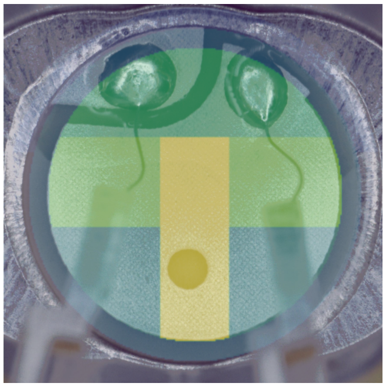


Figure 5. Static segmentation map for objects in VAD dataset, overlayed over the image. The image is separated into $L = 7$ segments: background, outer half-circle, piezo border, solder dots area, wires area, pins area, and piezo in the middle.

### 5.1. VAD, one-class benchmark

This benchmark allows only good images for training, but SegAD requires bad images as well, so we try to replace them with artificially generated defects. Because SegAD works with anomaly maps, generating realistic defects is unnecessary. For this reason, we have applied several simple augmentations to the 1000 good images available for training SegAD. These augmentations include Gaussian blur or a randomly placed, randomly sized thin rectangle of a random shade of gray. They were applied to randomly selected segments or random parts of the segments; segments are defined with the segmentation map. Code for generating defects can be found in the SegAD GitHub repository.

| Method | Cl. AUROC ↑ | FPR@95TPR ↓ |
|---|---|---|
| *One-class Anomaly Detection* | | |
| FastFlow | 79.1±1.7 | 76.2±1.9 |
| RD4AD | 84.5±0.2 | 65.8±2 |
| PatchCore | 88.3±0.3 | 60.4±2.8 |
| EfficientAD | 89.1±0.8 | **44.5**±2.2 |
| *SegAD (Ours)* | | |
| FastFlow + Ours | 83.3±1.3 | 71.5±5 |
| RD4AD + Ours | 87.5±1.2 | 61.0±7.9 |
| PatchCore + Ours | 89.7±0.8 | 58.8±3.7 |
| EfficientAD + Ours | 85.6±1.6 | 100±0 |
| All AD + Ours | **90.4**±0.5 | 52.4±4.5 |

Table 2. One-class benchmark (VAD). The best result is marked in bold. All AD means PatchCore, FastFlow, and RD4AD.

Results in Table 2 show that even such a naive approach improves results compared to anomaly detection methods alone. Even greater improvement can be achieved by using several anomaly detectors to produce anomaly maps for SegAD. This strategy fails for EfficientAD + Ours, anomaly maps produced by EfficientAD for the generated defects are too different from anomaly maps for real defects, creating a large discrepancy between training and test distributions for SegAD and causing worse results compared to anomaly detector alone.

### 5.2. VAD, high-shot supervised benchmark

| Method | Cl. AUROC ↑ | FPR@95TPR ↓ |
|---|---|---|
| *Supervised Anomaly Detection* | | |
| DevNet | 86.9±1.2 | 66.6±2.7 |
| DRA | 87.4±0.6 | 60.9±4.4 |
| *Supervised Classifier* | | |
| WRN | 95.0±0.6 | 32.8±4.3 |
| *SegAD (Ours)* | | |
| FastFlow + Ours | 86.9±0.5 | 56.9±3.9 |
| RD4AD + Ours | 90.2±0.3 | 46.9±2.6 |
| PatchCore + Ours | 91.7±0.3 | 48.0±2.9 |
| EfficientAD + Ours | 91.4±0.2 | 36.8±1 |
| AllAD+WRN+Ours | **96.5**±0.3 | **16.4**±1.9 |

Table 3. High-shot supervised benchmark (VAD). The best result is marked in bold. All AD means PatchCore, FastFlow, RD4AD, and EfficientAD. Improvement calculated compared to base method results in Tab. 2.

The high-shot benchmark allows training supervised classifier $g$, which shows competitive results as can be seen in Tab. 3 in the row "WRN", but it won't be able to detect unseen defects. Histograms in Fig. 6 visualize this problem. On the left is the one-class anomaly detector, which can detect unseen defects (in red) but shows a bad separation between good and bad parts. In the center, the supervised classifier $g$ shows a better separation of good and bad parts, but unseen defects distribution is shifted to the left because their scores are relatively low and such defects cannot be detected. On the right, SegAD shows a good separation between classes, but also unseen defects have a similar distribution to the seen defects. We still compare the supervised classifier Wide ResNet50 (WRN) with other methods, but the ability to detect unseen defects is crucial in real-world applications.

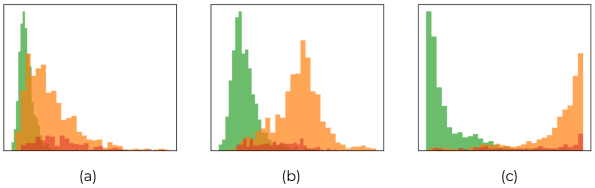


Figure 6. Distributions of scores in the VAD test set. Green color shows good parts, orange bad parts, and red bad parts with unseen defects. (a) shows one-class anomaly detector PatchCore, (b) supervised classifier Wide ResNet, (c) SegAD (PatchCore + WRN + Ours). X axis is score values, Y axis is frequency.

Combining SegAD, supervised classifier WRN, and all one-class anomaly detection models showed the best results presented in Table 3. A more practical setting can be EfficientAD + WRN + Ours, which can achieve Cl. AUROC 96 and FPR@95TPR 19.4 with a much higher inference speed.

### 5.3. VAD, low-shot supervised benchmark

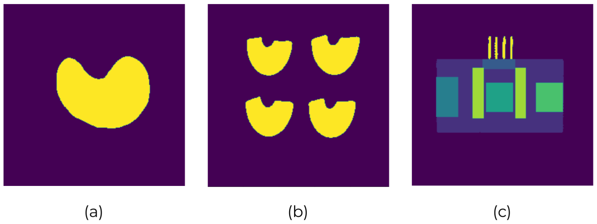
This benchmark uses just 100 bad images for training, which can be a relatively low number for a dataset with more than 20 types of defects. However, such a setting is closer to real-world applications, where a limited amount of defective images can be available. In Table 4, WRN shows much lower results due to a smaller training dataset, as well as DevNet and DRA. SegAD also performs worse, than with a high-shot benchmark, but it still can visibly improve the performance of other models. We do not use WRN in SegAD for this benchmark, because it showed low results by itself.

| Method | Cl. AUROC ↑ | FPR@95TPR ↓ |
|---|---|---|
| *Supervised Anomaly Detection* | | |
| DevNet | 73.1±2.3 | 88.3±2.3 |
| DRA | 78.9±1.8 | 74.6±1.9 |
| *Supervised Classifier* | | |
| WRN | 78.7±1.8 | 77.9±4.4 |
| *SegAD (ours)* | | |
| FastFlow + Ours | 84.3±1.4 | 64.5±2.5 |
| RD4AD + Ours | 88.6±0.2 | 51.1±0.9 |
| PatchCore + Ours | 90.8±0.5 | 52.7±4.3 |
| EfficientAD + Ours | 90.6±0.7 | 40.1±2.4 |
| All AD + Ours | **92.7**±0.5 | **36.8**±3.1 |

Table 4. Low-shot supervised benchmark (VAD). The best result is marked in bold. All AD means PatchCore, FastFlow, RD4AD, and EfficientAD.

### 5.4. Results on VisA dataset

VisA contains fewer good images per class for training than VAD (500-1000 in VisA, compared to 2000 in VAD). The training set consists of good images only. For this reason, we use a different strategy to split the training dataset to make sure we have enough images to train the base anomaly detection model with a competitive result. For RD4AD, the training set was divided into 90% of images to train the RD4AD itself and 10% to train SegAD. EfficientAD uses a validation set (10% of good images from the train set) to normalize anomaly maps. In our case, the same images were used to train SegAD. The segmentation maps for this dataset were created with the Segment Anything Model (SAM) [12]. For most classes, it is a binary mask for the object and background, and for pcb1-4, static components were mapped on top of it; see Fig. 7.



(a) (b) (c)

Figure 7. VisA. Examples of segmentation maps created using SAM for (a) cashew, (b) macaroni1, (c) pcb2.

| Method | Cl. AUROC ↑ | FPR@95TPR ↓ |
|---|---|---|
| *One-class Anomaly Detection* | | |
| RD4AD | 94.2 | 25.5 |
| EfficientAD | 98.0 | 8.1 |
| *Supervised Anomaly Detection* | | |
| DRA | 88.9 | 42.4 |
| DevNet | 89.3 | 44.0 |
| *SegAD (ours)* | | |
| RD4AD + Ours | 96.1 | 15.5 |
| EfficientAD + Ours | 98.3 | **6.2** |
| All AD + Ours | **98.4** | 7.5 |

Table 5. Results on VisA dataset, supervised benchmark. The best result is marked in bold. All AD means RD4AD and EfficientAD.

With VisA, we create a new, challenging, supervised benchmark on a public dataset with 12 different classes. In a similar way to the supervised benchmark on MVTec AD [20], we move 10 randomly selected bad images from the test set to the train set, reporting results over 10 different seeds. In Table 5, this benchmark shows how SegAD performs on a different data from VAD and creates a new possibility to compete on a difficult supervised anomaly detection problem. Detailed results per class can be found in Appendix D.

### 5.5. Ablation study

We perform an ablation study on the VAD high-shot supervised benchmark. We use the average result of the PatchCore, FastFlow, RD4AD, and EfficientAD as the baseline in the Table 6, denoted as "An.Det.". These anomaly detectors use the maximum value from the anomaly map to calculate the score. On the contrary, we compute our proposed features $\mathbf{f}$, which are described in Section 4, and use BRF to yield a final score. This shows an improvement of 1.3 Cl. AUROC in the row "One Seg.". In addition, we evaluate the segmentation map's impact by computing the anomaly map's maximum value per segment. Doing this gives an improvement in 2.1 Cl. AUROC, which can be seen in the row "Max.". This shows how both features and segmentation maps are important for SegAD. We also show the strength of BRF by comparing it against other algorithms. We compare BRF with BT and RF using a segmentation map and features. The results show that the BRF is better than these methods by 1.7 and 0.3 Cl. AUROC, respectively.

| Description | Feat. $\mathbf{f}$ | Seg. $S$ | BRF | Cl. AUROC ↑ |
|---|---|---|---|---|
| An. Det. | | | | 87.3 |
| One Seg. | ✓ | | ✓ | 88.6 |
| Max. | | ✓ | ✓ | 89.4 |
| BT | ✓ | ✓ | | 88.4 |
| RF | ✓ | ✓ | | 89.8 |
| SegAD | ✓ | ✓ | ✓ | 90.1 |

Table 6. Ablation study on the methodical differences between our method (SegAD) and anomaly detection methods. An.Det. = *Anomaly Detectors*. One Seg. = *One Segment*. Max. means using maximum value. BT = *Boosted Tree*. RF = *Random Forest*. BRF = *Boosted Random Forest*. SegAD denotes the average result from PatchCore + Ours, FastFlow + Ours, RD4AD + Ours, EfficientAD + Ours.

**Limitations:** SegAD (ours) requires to have segmentation maps. Static maps can be used for objects like the product in VAD. Obtaining segmentation maps for unaligned objects can be more difficult, yet SAM [12] or a specially trained segmentation model may be a solution. The performance improvement of SegAD depends on the complexity and structure of the objects. For less complex objects in VisA, improvement is lower than for VAD, as can be seen in Tab. 3 and Tab. 5. Another possible limitation can be the difference in the anomaly maps between training and test data. Due to the nature of anomaly detection, this problem might be impossible to mitigate, because our task is to detect such differences.

## 6. Conclusion

This study introduces VAD, a brand new supervised anomaly detection dataset derived from real production, offering challenging benchmarks for the research community to address real-world defect detection. Additionally, we propose SegAD, an innovative supervised anomaly detection method that achieves state-of-the-art performance on VAD as well as on a supervised benchmark on standard VisA dataset. Experimental results reveal the limitations of existing supervised anomaly detection methods in handling complex problems, while the integration of SegAD with top-performing one-class anomaly detection models further enhances their results.

## References

[1] Kilian Batzner, Lars Heckler, and Rebecca König. Efficientad: Accurate visual anomaly detection at millisecond-level latencies, 2023. 1, 3, 5

[2] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *CVPR*, pages 9584–9592, 2019. 1, 2, 3

[3] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *IJCV*, 130(4):947–969, 2022. 2, 3

[4] Jakob Božič, Domen Tabernik, and Danijel Skočaj. Mixed supervision for surface-defect detection: from weakly to fully supervised learning. *Computers in Industry*, 2021. 2, 3

[5] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *PR*, 30(7): 1145–1159, 1997. 5

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pages 785–794, 2016. 6

[7] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. In *CVPR*, pages 9737–9746, 2022. 1, 3, 5

[8] Choubo Ding, Guansong Pang, and Chunhua Shen. Catching both gray and black swans: Open-set supervised anomaly detection. In *CVPR*, 2022. 3, 5

[9] Colin Drury and Murray Sinclair. Human and machine performance in an inspection task. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 25:391–399, 1983. 1

[10] Lars Heckler, Rebecca König, and Paul Bergmann. Exploring the importance of pretrained feature extractors for unsupervised anomaly detection and localization. In *CVPRW*, pages 2917–2926, 2023. 2

[11] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *ICLR*, 2018. 3

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *ICCV*, pages 4015–4026, 2023. 7, 8

[13] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 3

[14] Yann LeCun, Corinna Cortes, Chris Burges, et al. Mnist handwritten digit database, 2010. 3

[15] Jiaqi Liu, Guoyang Xie, Jingbao Wang, Shangnian Li, Chengjie Wang, Feng Zheng, and Yaochu Jin. Deep industrial image anomaly detection: A survey. *arXiv e-prints*, pages arXiv–2301, 2023. 1, 2

[16] Muriel Mazzetto, Marcelo Teixeira, Érick Oliveira Rodrigues, and Dalcimar Casanova. Deep learning models for visual inspection on automotive assembling line. *arXiv preprint arXiv:2007.01857*, 2020. 1

[17] Declan McIntosh and Alexandra Branzan Albu. Inter-realization channels: Unsupervised anomaly detection in images beyond one-class classification, 2023. 2

[18] Yohei Mishina, Masamitsu Tsuchiya, and Hironobu Fujiyoshi. Boosted random forest. In *VISAPP*, pages 594–598, 2014. 4, 5

[19] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. VT-ADL: A vision transformer network for image anomaly detection and localization. In *ISIE*, 2021. 3

[20] Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks. In *SIGKDD*, pages 353–362, 2019. 8

[21] Guansong Pang, Choubo Ding, Chunhua Shen, and Anton van den Hengel. Explainable deep few-shot anomaly detection with deviation networks. *arXiv preprint arXiv:2108.00462*, 2021. 3, 5

[22] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, pages 1530–1538. PMLR, 2015. 3

[23] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *CVPR*, pages 14318–14328, 2022. 1, 3, 5

[24] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. In *ICLR*, 2019. 3

[25] Maximilian Schwab, Charles Madeline-Dérou, Steffen Klarmann, Nils Thielen, Sven Meier, Jörg Franke, Sandan Chintanippu, and Wilhelm Stork. Multi-model machine learning based industrial vision framework for assembly part quality control. In *ETFA*, pages 1–4, 2022. 5

[26] Domen Tabernik, Samo Šela, Jure Skvarč, and Danijel Skočaj. Segmentation-based deep-learning approach for surface-defect detection. *In Journal of Intelligent Manufacturing*, 31(3):759–776, 2020. 2, 3

[27] Matthias Wieler and Tobias Hahn. Weakly supervised learning for industrial optical inspection. In *DAGM symposium in*, 2007. 2, 3

[28] Xincheng Yao, Ruoqi Li, Jing Zhang, Jun Sun, and Chongyang Zhang. Explicit boundary guided semi-push-pull contrastive learning for supervised anomaly detection. In *CVPR*, pages 24490–24499, 2023. 1, 3

[29] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows, 2021. 1, 2, 3, 5

[30] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*. British Machine Vision Association, 2016. 5

[31] Hui Zhang, Zuxuan Wu, Zheng Wang, Zhineng Chen, and Yu-Gang Jiang. Prototypical residual networks for anomaly detection and localization. In *CVPR*, pages 16281–16291, 2023. 1, 3

[32] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In *ECCV*, pages 392–408. Springer, 2022. 1, 2, 3, 6