Long-Tailed Anomaly Detection with Learnable Class Names

Ho, Chih-Hui; Peng, Kuan-Chuan; Vasconcelos, Nuno
TR2024-040 April 25, 2024

Abstract
Anomaly detection (AD) aims to identify defective images and localize their defects (if any). Ideally, AD models should be able to detect defects over many image classes; without relying on hard-coded class names that can be uninformative or inconsistent across datasets; learn without anomaly supervision; and be robust to the long-tailed distributions of real-world applications. To address these challenges, we formulate the problem of long-tailed AD by introducing several datasets with different levels of class imbalance and metrics for performance evaluation. We then propose a novel method, LTAD, to detect defects from multiple and long-tailed classes, without relying on dataset class names. LTAD combines AD by reconstruction and semantic AD modules. AD by reconstruction is implemented with a transformer-based reconstruction module. Semantic AD is implemented with a binary classifier, which relies on learned pseudo class names and a pretrained foundation model. These modules are learned over two phases. Phase 1 learns the pseudo-class names and a variational autoencoder (VAE) for feature synthesis that augments the training data to combat long-tails. Phase 2 then learns the parameters of the reconstruction and classification modules of LTAD. Extensive experiments using the proposed long-tailed datasets show that LTAD substantially outperforms the state-of-the-art methods for most forms of dataset imbalance. The long-tailed dataset split is available at https://zenodo.org/records/10854201.

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2024
Long-Tailed Anomaly Detection with Learnable Class Names

Chih-Hui Ho\(^1\) Kuan-Chuan Peng\(^2\) Nuno Vasconcelos\(^1\)

\(^1\)University of California, San Diego \(^2\)Mitsubishi Electric Research Laboratories (MERL)

chh279@ucsd.edu kpeng@merl.com nvasconcelos@ucsd.edu

Abstract

Anomaly detection (AD) aims to identify defective images and localize their defects (if any). Ideally, AD models should be able to detect defects over many image classes; without relying on hard-coded class names that can be uninformative or inconsistent across datasets; learn without anomaly supervision; and be robust to the long-tailed distributions of real-world applications. To address these challenges, we formulate the problem of long-tailed AD by introducing several datasets with different levels of class imbalance and metrics for performance evaluation. We then propose a novel method, LTAD, to detect defects from multiple and long-tailed classes, without relying on dataset class names. LTAD combines AD by reconstruction and semantic AD modules. AD by reconstruction is implemented with a transformer-based reconstruction module. Semantic AD is implemented with a binary classifier, which relies on learned pseudo-class names and a pretrained foundation model. These modules are learned over two phases. Phase 1 learns the pseudo-class names and a variational autoencoder (VAE) for feature synthesis that augments the training data to combat long-tails. Phase 2 then learns the parameters of the reconstruction and classification modules of LTAD. Extensive experiments using the proposed long-tailed datasets show that LTAD substantially outperforms the state-of-the-art methods for most forms of dataset imbalance. The long-tailed dataset split is available at https://zenodo.org/records/10854201.

1. Introduction

Anomaly detection (AD) is an important problem for many manufacturing settings [23, 50, 60, 63]. To reflect practical manufacturing constraints, most datasets [5, 83, 89] are curated under the unsupervised AD setting, where no defect images are available for training. Various methods [19, 20, 43, 60, 69] have shown that this problem can be solved with high accuracy; e.g., [3, 20, 31, 36, 41, 44, 69, 77, 79, 84] have success rates >95% for anomaly detection and localization on the MVTec dataset [5]. However, as illustrated in Fig. 1, these methods require a different model per image category, which compromises scalability to many classes. Recently, there has been interest in more efficient methods that use a single model to detect anomalies in all object classes [9, 14, 26, 30, 32, 76, 87]. These methods can be grouped according to the level of image semantics where they operate. On one hand, AD by reconstruction methods [76, 87] use a reconstruction module to project the input image into the manifold of normal images. The difference between the image and its projection is then used to detect possible defects. On the other hand, semantic AD methods [9, 26, 32] build explicit models of normal/abnormal images. Given the absence of abnormal training data, this is done by leveraging the knowledge of visual-language foundation models [25, 39, 57, 67]. Abnormal regions are detected using the predefined text prompts for normal and abnormal plus an image class name, e.g., “a normal photo of a [CLASS]” and “an abnormal photo of a [CLASS],” where [CLASS] is a class name in the dataset, e.g., “bottle” in the MVTec dataset.

The two types of methods have limitations. Reconstruction methods require modeling the complex manifold, especially for problems requiring many classes. Even when trained on large datasets, the distance between the input image and this manifold can be smaller for certain anomalies. The foundation model used by semantic AD methods can provide additional clarity, because it enables framing AD as a binary classification problem. However, this is difficult when the dataset class names are ambiguous or unknown to the foundation model. Fig. 1 shows an example of ambiguity due to the fact that the class name “bottle” refers to visually different concepts in MVTec [5] (where it means “bottle bottom”) and the GoodsAD dataset [83] (where it means “bottle side”). Hence, in MVTec, the “bottle” label may not be the most
informative for the foundation model, which may associate the images with alternative labels, e.g. “black sphere.” Sometimes, class names can be simply unknown to the foundation model, e.g. the classes “PCB1” and “PCB2” also shown in Fig. 1. This suggests that the foundation model should learn what are the class names that best align with these images. When this is difficult, semantic AD might benefit from the flexibility of the AD by reconstruction methods, which are not constrained by class names. Hence, this work investigates the design of AD methods that combine both AD by reconstruction and semantic AD.

Beyond this, it remains unclear whether the resulting models generalize to the long-tailed setting [2, 18, 29, 34, 88] where, as illustrated in Fig. 1, the sample distribution is skewed. This is particularly important because long-tailed distributions are natural in manufacturing, where different objects can have very different popularity. We formulate the problem of long-tailed AD by introducing long-tailed datasets, which are obtained by resampling current AD benchmarks with different imbalance factors and types of imbalance. We also propose a set of performance metrics for the long-tailed setting.

To address the challenges above, we propose a new method, LTAD, which combines AD by reconstruction and semantic AD. AD by reconstruction is implemented by combining the ALIGN [33] image encoder and a transformer-based reconstruction module (RM), trained to project image patches into the manifold of normal images. An anomaly score is then obtained by computing the difference between the input image and the result of this projection. The latent patch representation produced by the ALIGN image encoder is also mapped to the feature space of the ALIGN text encoder, to enable the implementation of semantic AD. For this, a binary normal/abnormal classifier is implemented in ALIGN text space, by using as classifier weight vectors the ALIGN representation of text prompts for “normal” and “abnormal.” The posterior probability of the abnormal class, under this classifier is then used as a second anomaly score. Anomalies are detected with a combination of the two AD scores.

To address the long-tailed setting, we propose a preliminary training phase for data augmentation. This consists of learning a VAE [37, 38], which is then used to synthesize features. To make these class sensitive, the VAE is conditioned by the text encoding of the [CLASS] name, according to the ALIGN model. However, to address the ambiguity of class names, a set of learnable [CLASS] prompts are learned by backpropagation, during VAE training. In a second training phase, a mix of real and synthetic examples is used to train the LTAD model. Since there are no training anomalies, these are simulated by adding noise to the synthesized features during this stage.

Overall, we make the following contributions:
1. We show that prior methods do not perform well on long-tailed setting and formulate long-tailed AD based on 3 datasets with 9 imbalance settings and performance metrics.
2. We propose LTAD, which combines AD by reconstruction and semantic AD, performs multi-class AD, and overcomes dataset [CLASS] name ambiguity by learning names consistent with the semantic space of the ALIGN model.
3. We propose a new training strategy for LTAD which uses a novel data-augmentation procedure to address the data scarcity of long-tailed data, and learn [CLASS] names.
4. We show that LTAD outperforms the SOTA methods on the long-tailed AD. Extensive ablations confirm the efficacy of the various LTAD modules, showing that LTAD generalizes across various datasets and imbalance configurations.

2. Related works

Unsupervised anomaly detection (AD): Unsupervised AD aims to identify defective images and localize the defects without observing any defect images during training. Tab. 1 groups recent AD methods into 3 categories. Category $C_0$ contains earlier works [16, 19, 20, 31, 43, 60, 69, 84] that use a different model per image category. Student-teacher methods [20, 69, 84] use a pretrained teacher encoder and a student encoder optimized to match its predictions. AD is based on the difference of their predictions. Flow based methods [59, 61, 77] fit a Gaussian distribution to the feature vectors of normal images and use out-of-distribution criteria to perform AD. Reconstruction based methods [54, 79–81] train models to reconstruct normal samples and use reconstruction error for AD. Some other methods in category $C_1$ are discussed in the AD survey papers [11, 12, 47, 65]. More parameter efficient than these methods, LTAD only needs a single model for all the classes instead of one model per class.

This is the setting adopted by the more recent methods of Category $C_1$ [26, 30, 49, 76, 87]. For example, UniAD [76] improves prior reconstruction-based methods by using a neighborhood attention mask to avoid information leak, while AnomalyGPT [26] uses a large vision-language model to provide explanations for defective regions. While typically leveraging foundation models, these methods do not use the class name to detect anomalies. This akin to LTAD but unlike methods in category $C_2$ [9, 10, 14, 32, 82], which also use a single model for all classes but require class names. For example, WinCLIP [32] uses the CLIP [57] model to compute the anomaly score, by measuring similarity between image and text feature vectors for several predefined normal/abnormal text prompts. Some works in categories $C_1$ and $C_2$ [14, 26] also leverage auxiliary training data (e.g., train on VisA [89] and test on MV Tec [5]), a setting that is not considered in this work.

<table>
<thead>
<tr>
<th>Unsupervised AD Method Conditions</th>
<th>Unsupervised AD Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_0$</td>
<td>$C_1$</td>
</tr>
<tr>
<td>Single model for all classes</td>
<td>✓</td>
</tr>
<tr>
<td>No class name prior</td>
<td>✓</td>
</tr>
<tr>
<td>Designed for the long-tailed setting</td>
<td>✗</td>
</tr>
<tr>
<td>Learnable class names</td>
<td>✗</td>
</tr>
</tbody>
</table>

Table 1. LTAD addresses some important challenges for real-world AD, previously not considered in (e.g., $C_0$: [16, 19, 20, 31, 43, 54, 59–61, 64, 69, 77, 79–81, 84], $C_1$: [26, 30, 49, 76, 87], $C_2$: [9, 10, 14, 32, 82]).
3. Long-tailed anomaly detection

Motivation: Previous AD works assume that different image classes are equally populated. However, in most industrial applications, different objects have different costs, production schedules, etc. This creates long-tailed distributions where certain classes have much higher example cardinality than others. Extensive research in areas like classification, indicates that systems not trained to account for this class imbalance tend to overfit on popular classes and ignore the less popular ones [18, 34, 40, 45, 58, 71]. To test whether this also holds for AD, we perform some preliminary experiments, using the MVTec dataset and a long-tailed version, obtained by image

Lesser-populated long-tailed Data

Table 2. The statistics of the long-tailed splits we use across the 3 datasets. For all datasets, we consider both the $\exp$ and $\step$ imbalance.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Max Class Sample</th>
<th>Imbalanced Factor $\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVTec [5]</td>
<td>391</td>
<td>${100, 200}$</td>
</tr>
<tr>
<td>VisA [89]</td>
<td>905</td>
<td>${100, 200, 500}$</td>
</tr>
<tr>
<td>DAGM [70]</td>
<td>1000</td>
<td>${50, 100, 200, 500}$</td>
</tr>
</tbody>
</table>

resampling. The sample distributions of the two datasets are shown in Fig. 2(a). Fig. 2(b) compares the performance on the two datasets of UniAD [76], one of the best open-source AD methods based on a single model that detects multi-class anomalies. The figure confirms that UniAD performs well on MVTec but degrades considerably for the long-tailed version, where it overfits to majority classes and severely underperforms on minority ones. This experiment highlights the need for the methods that explicitly address the long-tailed AD problem. This requires datasets and performance metrics, which we discuss next.

Dataset collection: Following [8], given a balanced dataset, we create a long-tailed version by sampling the training set, while the test set remains unchanged. The distribution of the sampled training set depends on two factors: the imbalance factor $\beta$ and the imbalance type. $\beta$ is the ratio between the cardinalities of the most and least populated classes, i.e., $\beta = \max\{N_c\} / \min\{N_c\}$ where $N_c$ is the number of samples of class $c$. To prevent overfitting on a specific imbalance distribution, we consider two types of imbalance: exponential ($\exp$) and step ($\step$). While the former indicates that $N_c$ decays exponentially across classes $c$, the latter indicates a binary split into majority classes of size $\max\{N_c\}$ and minority classes of size $\min\{N_c\}$. Individual long-tailed datasets are denoted by type and imbalance factor. For example, $\exp 100$ indicates a training set exponentially imbalanced with $\beta = 100$. We define the half most/least populated classes as majority/minority classes and construct the long-tailed dataset by randomly sampling from the original dataset, without repetition. When the number of samples of a class is less than the desired number of samples, all samples are kept.

Tasks and metrics: Both anomaly detection (AD) and anomaly segmentation (AS) are considered. Following [26, 30, 76, 79], we use the Area Under the Receiver Operating Curve (AUROC) at image and pixel level for AD and AS, respectively. We report two types of results. The first is average performance (Avg), which is the performance averaged across all classes. The second is the pair of average performance for majority classes (High) and average performance for minority classes (Low).

Datasets: We consider 3 datasets: MVTec [5], VisA [89] and DAGM [70]. Various long-tailed datasets are built from each. Tab. 2 shows the maximum number of samples across classes and the imbalance factors considered per dataset. For all datasets, we consider both the $\exp$ and $\step$ imbalance types. Our proposed dataset splits will be released upon publication.

4. The LTAD anomaly score

In this section, we propose a novel method, LTAD, for the unsupervised long-tailed anomaly detection task. As shown in

<table>
<thead>
<tr>
<th>Category</th>
<th>Full Data</th>
<th>Long-tailed Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazelnut</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leather</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bottle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wood</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carpet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nut</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toothbrush</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zipper</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transistor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pill</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capsule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Screw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 3. The LTAD architecture combines AD by reconstruction and semantic AD scores (Srec and Ssem, respectively), implemented by the RM and SAD modules. We use an image E and text encoder T from a pretrained foundation model to extract features \( f^{\text{real}} \) and text features \( t_{a,c}, t_{n,c} \), derived from text-prompts that include a static component to discriminate between normal (\( \epsilon^n \)) and abnormal (\( \epsilon^a \)) and a learned component \( s_c \) to make this discrimination class-sensitive.

**AD by reconstruction.** The reconstruction module (RM) \( \Pi(\cdot) \) is a transformer [42, 53, 56, 76] trained to reconstruct the features extracted from the image \( I \) by a pretrained encoder \( E \) of \( L \) layers. Given an image \( I \in \mathbb{R}^{W \times H \times 3} \) class \( c \in \mathcal{C} \), \( E \) extracts feature tensor \( f^{\text{real}} \in \mathbb{R}^{W_l \times H_l \times C_l} \) from layer \( l \in \{1, L\} \). Since the feature tensor from the last layer \( f^{\text{real}}_L \) represents the global semantics of \( I \), it tends to degrade the AD performance [30, 60, 76], which requires local semantics. Hence, to perform AD, this tensor is dropped and the first \( L-1 \) tensors \( \{f^{\text{real}}_1, \ldots, f^{\text{real}}_{L-1}\} \) are remapped to the dimensions \( f^{\text{real}}_1 (i.e., W_1 \times H_1) \) by bi-linear interpolation along the spatial dimension. In the following, we use the notation \( f^{\text{real}}_l \) to represent this interpolated version and define \( f^{\text{real}} = [f^{\text{real}}_1, \ldots, f^{\text{real}}_{L-1}] \) as the feature tensor extracted across the \( L-1 \) layers. This tensor is then split into \( W_1 \times H_1 \) patch feature vectors \( \{p_i\}_{i=1}^{W_1 \times H_1} \), which are fed as tokens to the RM transformer \( \Pi(\cdot) \). Given patch \( i \), the anomaly score of the AD by reconstruction module is the squared error.

\[
S_{\text{rec}}(p_i) = ||\Pi(p_i) - p_i||^2. \tag{1}
\]

**Semantic AD (SAD).** The goal of semantic AD is two-fold: 1) to give the anomaly detector sensitivity to normal/abnormal classes, and 2) to leverage the prior knowledge about normality/abnormality available in a large foundation model. This allows the AD to discriminate between the two conditions without requiring abnormal images for training. As shown in Fig. 3, the semantic AD module is a binary classifier of a projection \( \hat{p}_i \) of patch \( p_i \) into normal/abnormal classes. The layer-wise components \( p_{il} \) of the patch feature vector \( p_i \) are first projected into vectors \( \Phi_l(p_{il}) \) with the dimension \( d \) of the text embedding of the ALIGN model. These projections are implemented by projection modules \( \Phi_l : \mathbb{R}^d \rightarrow \mathbb{R}^d \), \( l = 1, \ldots, L-1 \), where each \( \Phi_l \) is implemented with a linear layer. The layer-wise features are then aggregated into a single patch feature vector \( \hat{p}_i \).

\[
\hat{p}_i = \max_l \{ \Phi_l(p_{il}) \}_l \tag{2}
\]

by max pooling over layers. The resulting vector \( \hat{p}_i \) is then fed to a binary classifier of parameters \( t_{n,c}, t_{a,c} \) (normal) and \( t_{a,c} \) (abnormal), where \( c \) is the class of image \( I \), which computes the posterior probability of an anomaly using a softmax layer with temperature scaling \( \tau \).

\[
S_{\text{sem}}(p_i,c) = \frac{\exp(t_{n,c} \cdot \hat{p}_i/\tau) + \exp(t_{a,c} \cdot \hat{p}_i/\tau)}{\exp(t_{n,c} \cdot \hat{p}_i/\tau) + \exp(t_{a,c} \cdot \hat{p}_i/\tau)}, \tag{3}
\]

where “\( \cdot \)” denotes the dot product. This is used as the semantic AD score for the images of class \( c \).

The main difficulty of this process is to learn the classifier parameters \( t_{n,c}, t_{a,c} \) without explicit supervision, since there are no training images of anomalies. To overcome this problem we leverage the prior for normal/abnormal classification provided by the ALIGN model. This is implemented by feeding to ALIGN a normal text prompt \( v_n \) and an abnormal text prompt \( v_a \) that apply to all classes. While we have considered several possibilities (see Tab. 9), the best AD performance was achieved by setting \( v_n = "a" \) and \( v_a = "a broken." \) Unless otherwise noted, we use these prompts in what follows. To further make the anomaly score sensitive to the image semantics, this is complemented by an image class prompt \( s_c \). Unlike prior works [9, 14, 32] that assume the access to the class names (e.g., “bottle,” or “hazelnut”) in MVTec [5], we assume that the class name is unknown. This is important to support the classes that are unknown to the ALIGN model or even to most humans, such as “PCB1” vs. “PCB2” in Fig. 1.

Instead, inspired by [24], we use a pseudo class name \( s_c \) learned per class \( c \). This is implemented by prompting the text encoder \( T \) with a prompt \( s_c \) per class \( c \), and learning prompts \( s_c \) as discussed below. The resulting set of semantic sensitive AD prompts \( \mathcal{P} = \{[v^n; s_c], [v^a; s_c]\}_{c} \) is mapped to a set of classifier parameters \( \{t_{n,c}, t_{a,c}\}_{c} \) by the text encoder \( T \) of the ALIGN model, according to

\[
t_{n,c} = T([v^n; s_c]) \quad t_{a,c} = T([v^a; s_c]). \tag{4}
\]

**LTAD score:** The overall anomaly score of a patch feature \( p_i \) of class \( c \) is defined as the linear combination

\[
S(p_i,c) = S_{\text{rec}}(p_i) + \lambda S_{\text{sem}}(p_i,c), \tag{5}
\]
where $\lambda$ is the hyperparameter to balance $S_{rec}(p_i)$ and $S_{sem}(p_i, c)$ such that both scores have comparable ranges.

5. Training

The training of LTAD is divided into two phases.

5.1. Phase 1: Class sensitive data augmentation

This training phase seeks two goals: 1) overcome the data scarcity of long-tailed AD, by augmenting the training set with normal examples of minority classes and abnormal examples of all classes, 2) learn the class sensitive prompts $s_c$ required by the semantic AD score of (3)-(4).

Fig. 4 summarizes this training procedure. Given an image $I \in \mathbb{R}^W \times H \times 3$ of class $c \in C$, the pretrained encoder $E$ extracts the feature tensor $f^{\text{real}} = [f_1^{\text{real}}, ..., f_{L-1}^{\text{real}}]$ plus a latent code $z = f_{L}^{\text{real}}$, which is the feature vector from the last encoder layer ($L$). An image decoder $D$, whose architecture is the mirror copy of $E$, is then trained to sample corresponding feature vectors, using a procedure inspired by the variational autoencoder (VAE) [37, 38]. A latent feature $\tilde{z}$ is sampled from a normal distribution $\mathcal{N}(\mu, \sigma)$ of parameters $\mu = F_\mu(z)$ and $\sigma = F_\sigma(z)$, where $F_\mu$ and $F_\sigma$ are learned linear transformations. The decoder $D$ then synthesizes a feature tensor from $\tilde{z}$.

In the long-tailed setting, the performance of $D$ degrades for classes $c$ with few training images. To ameliorate this problem, $D$ is conditioned by the prior knowledge about the class, in the form of a text-derived prototype feature vector $t_c$ that represents class $c$ for feature synthesis. This is obtained by prompting the text encoder of ALIGN with the pseudo-class name $s_c$, i.e., $t_c = T(s_c)$. The feature prototype is then concatenated with the image-dependent latent feature $\tilde{z}$ to create the input to $D$, which finally synthesizes a feature tensor $\{f_{l}^{\text{syn}}\}_{l=1}^{L-1} = D(\tilde{z}, t_c)$ of dimensions equal to those of $f_{l}^{\text{real}}$.

Following standard practices for VAE training, $D$ and $s_c$ are learned by optimizing a loss function

$$
\mathcal{L}_{D_1} = \frac{1}{L-1} \sum_{l=1}^{L-1} ||f_{l}^{\text{syn}} - f_{l}^{\text{real}}||^2 - KL(\mathcal{N}(\tilde{z} - \mu, \sigma)||\mathcal{N}(0, I)),
$$

5.2. Data augmentation

When this training phase is completed, the decoder $D$ works as a data augmentation device, producing synthetic feature tensors $f^{\text{syn}}$ in the semantic neighborhood of a feature tensor $f^{\text{real}}$ extracted from a real image. This is used to augment the training data in an online fashion during the second phase of training. Two types of data augmentation are considered.

Long-tailed classes. To counteract the imbalanced nature of long-tailed datasets, data augmentation is implemented by selecting the real $f^{\text{real}}$ or synthetic $f^{\text{syn}}$ feature vectors with probabilities $(p_c, 1 - p_c)$ respectively. The selected feature vector $f$ is split into $W_1 \times H_1$ patch feature vectors $\{p_i\}_{i=1}^{W_1 \times H_1}$, where the $n$ superscript denotes that these are normal feature patches.

Anomalies. To counteract the lack of anomalies during training, random noise (sampled from normal distribution) is added to normal patch features $p_i^n$ to produce pseudo-anomaly patch features $p_i^n$, as in [76]. This process is repeated for all normal patches during training. No random noise is added during inference.

5.3. Phase 2: Anomaly detection

Data augmentation is used in the second phase of training to learn the parameters of 1) the RM transformer $\Pi(.)$ used to reconstruct features and 2) the modules $\Phi(.)$ used in (2) to project patch features into the semantic space of ALIGN.
Reconstruction Module (RM): As shown in the bottom right of Fig. 5, the RM \( \Pi(\cdot) \) is trained to project the pseudo anomaly patch features \( p_i^a \) into the reconstructed patch features \( \Pi(p_i^a) \) in the manifold of the normal patch features \( p_i^n \). This is implemented with the RM transformer to minimize the loss

\[
\mathcal{L}_{rec} = \frac{1}{W_1 H_1} \sum_{i=1}^{W_1 H_1} ||\Pi(p_i^a) - p_i^n||^2.
\]  

(7)

Semantic patch projections: As shown in the top right of Fig. 5, the functions \( \Phi(\cdot) \) compute projections of the patches \( p_i \) into the semantic space of ALIGN, where the classifier parameters of (4) are defined. The functions \( \Phi(\cdot) \) are trained to encourage the alignment between the projected patch features and the text features by minimizing the binary cross entropy loss

\[
\mathcal{L}_{sem}(c) = -\frac{1}{W_1 H_1} \sum_{i=1}^{W_1 H_1} y_i \log(S_{sem}(p_i, c)), y_i = \begin{cases} 1, & \text{if } p_i = p_i^a \\ 0, & \text{if } p_i \neq p_i^a, \end{cases}
\]  

(8)

where \( c \) is the image class and \( S_{sem}(\cdot) \) is the semantic score of (3). Note that both phase 1 and phase 2 share the same text encoder \( T \), which is fixed in both phases. The total loss function for phase 2 is \( \mathcal{L}_2 = \mathcal{L}_{rec} + \mathcal{L}_{sem}(c) \).

6. Experiments

In this section, we report on various experiments designed to evaluate LTAD. While we compute average performance across majority (High), minority (Low), and all (All) classes in all experiments, we omit the High and Low values in some cases, for brevity. A complete listing is provided in the supplement.

Baselines: For a given training configuration (e.g., exp100), the same training set is used for all the baselines \([26, 43, 64, 76, 79]\) and LTAD. For baselines other than RegAD \([30]\), their official code and training/testing setting are used. RegAD is trained on all classes, instead of its leave-one-class-out setting, for fair comparison. During testing, we use 2 training examples per class as the support set to estimate the normal distribution. Since RegAD requires a support set of 2 images per class, it is not applicable to some configurations (e.g., exp200 and step200 in MVTec) where only 1 image is available for some minority classes. Cut & Paste \([43]\) supports anomaly detection but not localization.

Training details: To train LTAD, we use the pretrained visual language foundation model ALIGN \([33]\). Each input image is scaled to 224 x 224 and the features \( f^{real} \) are extracted from layers \( \{3, 10, 17, 37\} \) of the ALIGN image encoder. By default, the length of the pseudo class name is set to 2 and initialized with the text “object object.” In phase 2, the probability \( p_i \) of selecting real features is 0.5. The hyperparameter \( \lambda \) in (5) is set to 500, 400, and 300 for MVTec, VisA, and DAGM, respectively. Unless otherwise noted, we use \( v^n = "a" \) and \( v^a = "a broken" \) in (4) and \( \tau = 1 \) in (3). Phase 1 is trained for 100 epochs and phase 2 for 500 epochs, both using the AdamW optimizer with the learning rate 1e-4. Pytorch \([55]\) is used for implementation.

6.1. Comparisons to the state-of-the-art

Tab. 3-5 summarize the AD and AS performance of all the methods on MVTec, VisA, and DAGM, respectively. The best and second best performances for each dataset configuration are highlighted in bold and underline, respectively. On MVTec, LTAD is compared to 6 baselines under 4 different configurations. As shown in Tab. 3, early methods, such as Cut & Paste \([43]\), MKD \([64]\), and DRAEM \([79]\), are not suitable to detect and localize defects across classes with a single model, leading to inferior performance. While more recent models \([26, 30, 76]\) can detect defects across classes, they do not perform well across levels of dataset imbalance. LTAD is less affected by skewed distributions and outperforms most baselines in all tasks. The only exceptions are RegAD, which outperforms LTAD in 2 of the 8 tasks, and UniAD in 1 of the 8 tasks. Tab. 4-5 show that the gains of LTAD over the baselines are even larger for VisA and DAGM, where it achieves the best performance in all tasks. Overall, LTAD outperforms the baselines on 29 out of the 32 (90.6%) tasks defined by the 3 datasets.

Fig. 7 shows a comparative visualization of anomaly detections by UniAD and LTAD. Note how the detections of LTAD are much more localized and selective of the anomaly. Additional visualizations are provided in the supplement.

Beyond LTAD, the performance of LTAD without the semantic AD module is shown in the penultimate columns of Tab. 3-5. Even without this module, LTAD outperforms on 28 out of 32 (87.5%) tasks. However, by adding the semantic AD module, performance improves on 31 out of 32 (96.9%) tasks, highlighting the efficacy of the semantic AD module.

6.2. Ablation study

We ablate different designs of LTAD in Tab. 6-8, where the experiments are denoted by the experiment ID (expID).

Ablation study of AD by reconstruction: Fig. 6 and Tab. 6 compare different RM designs (i.e., LTAD without the semantic AD module) on the MVTec step100 dataset. Fig. 6 studies the effect of the length of the pseudo class name prompt on AD performance, showing that the performance peaks for length 2. Tab. 6 summarizes the detection and segmentation performance of several variants of LTAD for majority (High), minority (Low) and all (All) classes. expID 6.2 first shows that replacing the EfficientNet \([68]\) features of UniAD by those of ALIGN increases both AD and AS performance by 2.38% and 1.44%, respectively. In expID 6.3, we replace the text conditioned VAE of LTAD by an autoencoder trained on ALIGN features, to generate features for phase 1 of LTAD training. This underperforms LTAD (expID 6.8) by 2.57% (0.6%) for AD (AS), highlighting the efficacy of the VAE of LTAD. In expID 6.4, we replace the probability \( p_i = 0.5 \), used for synthetic feature selection in phase 2 of LTAD, by a more sophisticated design, where \( p_i \) is inversely proportional to the class cardinality (i.e., the classes with less samples have higher probability of selecting synthesized features). When compared to LTAD (expID 6.8),
 Altogether, these results support the claim that, while the class names are available in the dataset. In expID 6.5, the name of class \( c \) is simply used as \( s_c \). This underperforms all other experiments, where \( s_c \) is a learned pseudo class name. These experiments vary on the procedure used to initialize \( s_c \) for learning. ExpID 6.6, which uses the class name in the dataset as initialization, outperforms expID 6.7, which randomly initializes \( s_c \). Altogether, these results support the claim that, while the class specific prompts are important, not all the class names are informative and the prompts should be learned. Finally, all methods underperform LTAD (expID 6.8), which initializes all \( s_c \) with the word "object." This initialization also outperformed various other words that we have tried. The configuration of expID 6.8 is used as the default for all other experiments in the paper.

**Ablation study of Semantic AD:** Tab. 7 ablates different SAD module designs (i.e., LTAD without the RM module) on the MVTec step100 dataset. ExpID 7.1 replaces (2) by a direct projection of the concatenated vectors \( p_s \) into the \( d \)-dimensional text space, using a linear transformation \( \Phi : \mathbb{R}^{2(c-1)} \rightarrow \mathbb{R}^d \). When compared to LTAD (ExpID 7.4), this degrades AS performance significantly, most likely due to a greater difficulty in accounting for the different resolutions of features from different layers. The remaining experiments compare the implementation of (2) with different pooling operations, namely max vs. mean. Experiments expID 7.2 and 7.4 show that mean pooling is better than no layer aware projection, but inferior to the max pooling of LTAD. Finally, expID 7.3 uses max pooling and investigates the use of multiple word-prompts \( \nu^a \) and \( \nu^b \) in (4). This is inspired by the gains reported for ensembling.
multiple prompts when visual language foundation models are used for open-set classification [57]. Comparing experiments expID 7.3 and 7.4 shows that there is no similar advantage for semantic AD. Note that jointly assigning 2 anomaly scores on MVTEC step100 (See Tab. 3 step100) outperforms using either the reconstruction by AD of expID 6.8 or the semantic AD of expID 7.4 solely, indicating both modules contribute to the gain.

Tab. 8 ablates the importance of the learned pseudo class name $s_c$ for the semantic AD module, showing that it does not solely rely on the normal/abnormal prompt for anomaly detection. ExpID 8.1 shuffles the pseudo class names across classes by assigning $s_{c,i,j}$ the pseudo class name of class $i$, to class $j$, where $i \neq j$. Compared to LTAD (expID 8.3), this hurts performance significantly. ExpID 8.2 tests the alternative of eliminating the text encoder of ALIGN completely, simply learning a binary classifier of weight vectors $t_n,c,t_a,c$ per image class $c$. This approach is even less effective, showing the importance of the prior knowledge encoded in the foundation model about both classes and normal/abnormal images. The small sizes of the AD datasets are not sufficient to overcome the use of this prior.

Tab. 9 further ablates different combinations for normal $v^r$ and abnormal $v^a$ text prompts. The combinations of (“a”, “a broken”) and (“a”, “a defective”) outperform all others. Note that a poor choice of these prompts can degrade performance, although the latter seems to be more sensitive to the choice of normal than abnormal prompt. This shows the importance of leveraging the prior knowledge of foundation model about the two conditions.

**Distribution imbalance:** A set of experiments were performed on DAGM [70] to evaluate the robustness of LTAD to the class imbalance of the training dataset. To avoid the possibility of overfitting on a specific class order, we repeated these experiment with reverse class order (the least populated classes become the most populated ones). We considered both exponential and step dataset imbalance. Fig. 8 (a,b) show the class cardinalities before (blue) and after (orange) reversing the class order. Fig. 8 (c-d) compare the AD and AS performance of the different approaches. For all dataset configurations, LTAD outperforms the baselines by at least 9.1% (5.9%) for AD (AS). Part of these gains are due to SAD module, which improves performance by 1.62% (1.01%) for AD (AS) on average. These results show that LTAD generalizes across imbalance factors, that SAD module consistently improves performance, and that its gains are insensitive to class order.

**7. Conclusion**

In this work, we have introduced the task of long-tailed AD, by proposing datasets and performance metrics and a novel AD method, LTAD, tailored for the long-tailed setting. LTAD detect defects from multiple and long-tailed classes, without relying on dataset class names. It combines AD by reconstruction and semantic AD modules. AD by reconstruction is implemented with a transformer-based reconstruction module. Semantic AD is implemented with a binary classifier, which relies on learned pseudo class names and a pretrained foundation model. These modules are learned over two phases. Phase 1 learns the pseudo-class names and a VAE for feature synthesis that augments the training data to combat long-tails. Phase 2 then learns the parameters of the reconstruction and classification modules of LTAD. Experiments show that LTAD outperforms the SOTA AD methods on most long-tailed datasets considered and all the components of LTAD contribute to its superior performance.

**Acknowledgement** CH and KCP were supported by Mitsubishi Electric Research Laboratories. CH and NV were partially funded by NSF awards IIS-2303153 and gift from Qualcomm.
References


[10] Yunkang Cao, Xiaohao Xu, Chen Sun, Xiaoman Huang, and Weiming Shen. Towards generic anomaly detection and understanding: Large-scale visual-linguistic model (GPT-IV) takes the lead. ArXiv, abs/2311.02782, 2023. 2


[14] Xuhai Chen, Yue Han, and Jiangning Zhang. A zero-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, abs/2305.17382, 2023. 1, 2, 4


[28] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21:1263–1284, 2009. 3


