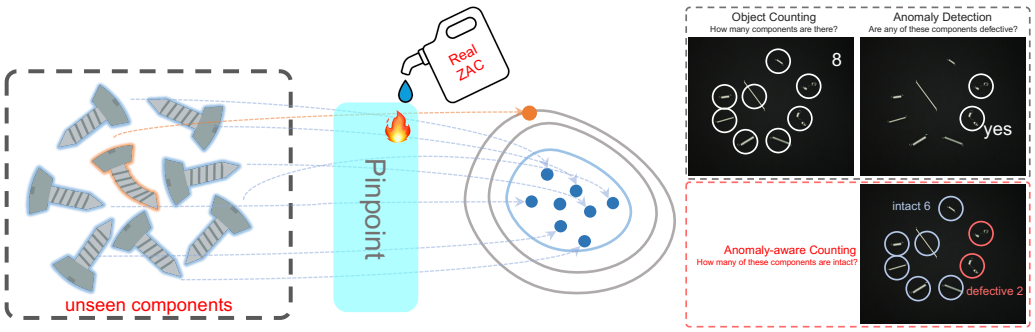
ZAC : ZERO-SHOT ANOMALY-AWARE COUNTING FOR INDUSTRIAL COMPONENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Counting industrial components is essential in real-world manufacturing systems, and what matters is not only how many parts there are but how many are usable, since production lines carry defective items alongside intact ones. As manufacturing lines continually introduce new components with distinct defect types, no single model can be trained on every class it will encounter. To address this, we introduce **Zero-shot Anomaly-Aware Counting**, a task that counts objects while distinguishing their defect states. Traditional object counting models focus on quantity and ignore quality, while industrial anomaly detection models evaluate quality but struggle with dense multi-object scenes. Multimodal large language models can be queried for either, but their performance drops substantially once a query asks which of the objects they have found are defective. These failures share a cause, since existing benchmarks do not combine instance-level defect-state supervision with class-disjoint evaluation on unseen component classes. Therefore, we construct **Real-ZAC**, a benchmark of real industrial components in dense multi-instance settings. Every instance is annotated with an intact or defective label, and test classes are strictly withheld from training to evaluate zero-shot capabilities. Furthermore, we propose a baseline method that utilizes the multi-instance structure of Real-ZAC. The model suppresses visual patterns shared across instances and emphasizes subtle defect-related deviations, reducing reliance on class-specific appearance and improving generalization to unseen component classes. The code and dataset will be available soon.

1 INTRODUCTION



(a) (b)

Figure 1: **Zero-shot anomaly-aware counting. (a)** Given multiple instances of an unseen component class, Pinpoint suppresses dominant appearance shared across instances and emphasizes fine-grained defect cues to identify intact and defective instances. **(b)** Unlike conventional object counting, which ignores defect state, or anomaly detection, which only determines whether a defect is present, anomaly-aware counting requires instance-level state recognition together with counting.

Industrial manufacturing lines handle components in bulk, where the practical quantity of interest is often how many parts remain intact and usable. This becomes difficult when new component classes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

are introduced, since collecting defect examples and retraining a model for every class is impractical. We therefore formulate *Zero-shot Anomaly-Aware Counting*: given an image and a defect-state query, the task is to identify all instances satisfying that defect-state and report their count, while the component classes at test time are unseen during training. As illustrated in Figure 1b, anomaly-aware counting differs from conventional anomaly detection and object counting by requiring instance-level defect-state recognition and counting within the same scene.

Object counting has evolved from few-shot exemplar-based settings (Ranjan et al., 2021) to zero-shot (Xu et al., 2023) and open-world counting (Amini-Naieni et al., 2024), with recent referring-expression methods (Dai et al., 2024) further allowing targets to be specified by text. These advances broaden what can be counted, but existing counting benchmarks still evaluate object identity and quantity rather than the defect state of each object. Consequently, no existing counting benchmark evaluates anomaly-aware counting on component classes entirely unseen during training.

Zero-shot anomaly detection methods leverage pretrained vision-language models to distinguish normal and abnormal patterns in unseen categories, typically by adapting textual concepts of normality and abnormality and aligning visual representations with them (Zhou et al., 2024; Cao et al., 2024; Ma et al., 2025). Although these methods improve category-independent defect recognition, existing benchmarks (Bergmann et al., 2019; Heckler-Kram et al., 2026; Zou et al., 2022) remain focused on image or pixel-level anomalies and do not provide instance-level object locations, making them unsuitable for zero-shot anomaly-aware counting.

Zero-shot anomaly-aware counting requires scenes that hold many instances of the same category, where intact and defective parts coexist. A benchmark for it must therefore vary the instance count, the component category, the background, and the mix of intact and defective instances within each scene, and must withhold entire categories from training. Existing datasets meet only part of this. FIOC (Cho et al., 2026) provides instance-level defect attributes and multi-instance scenes but draws its training and test splits from the same object categories, while conventional anomaly-detection datasets are not built for dense multi-instance counting. We therefore construct **Real-ZAC**, which contains 5,485 real industrial images across 25 component classes with diverse backgrounds and defect types, five of which are withheld entirely from training.

We further develop **Pinpoint**, a baseline designed specifically for zero-shot anomaly-aware counting. Figure 1a provides an overview of this design and its role in the proposed benchmark. The key challenge is to determine the state of each component without relying on class-specific defect patterns, since the component categories encountered at test time are unseen during training. Pinpoint is therefore designed to focus on subtle visual differences that distinguish individual instances, rather than on the dominant appearance shared by components of the same class. Pinpoint provides a practical baseline for studying anomaly-aware counting under class-disjoint zero-shot evaluation.

We evaluate Pinpoint against multimodal large language models, counting models, and zero-shot anomaly detection methods, all under the same class-disjoint protocol. It reaches 0.872 F_1 with 2.09 MAE on the unseen classes of Real-ZAC, improving F_1 by 7.0 points over the strongest baseline, and attains the highest defect separability on three external benchmarks it was never trained on.

Our contributions are as follows.

- We introduce **Zero-shot Anomaly-Aware Counting**, a task that requires deciding for every instance in a frame whether it is intact or defective and reporting how many of each, on component classes unseen during training.
- We construct **Real-ZAC**, a benchmark of real industrial components in multi-instance scenes, with instance-level defect-state annotations and class-disjoint training and evaluation splits for measuring generalization to unseen component classes.
- We propose **Pinpoint**, which enhances region representations with within-image comparative descriptors that suppress shared component appearance and emphasize defect-related deviations. Pinpoint consistently outperforms existing approaches on unseen components.

2 Related Works

Zero-shot Anomaly Detection and Benchmarks Zero-shot anomaly detection recognizes defects in categories unseen during training, and recent methods largely build on pretrained vision-language

models. WinCLIP (Jeong et al., 2023) relies on hand-crafted normal and abnormal text prompts, while AnomalyCLIP (Zhou et al., 2024), AdaCLIP (Cao et al., 2024), and AA-CLIP (Ma et al., 2025) learn or adapt anomaly-related textual representations and align visual features with them. A parallel line extends the setting to multimodal language models for anomaly reasoning and localization (Gu et al., 2024; Chao et al., 2025). The benchmarks that support this work, including MVTec AD (Bergmann et al., 2019), VisA (Zou et al., 2022), MVTec LOCO (Bergmann et al., 2022), Real-IAD (Wang et al., 2024), and MVTec AD 2 (Heckler-Kram et al., 2026), cover diverse industrial defects and acquisition conditions, but they annotate anomalies at the image or pixel level and therefore cannot assign a defect state to each instance in a multi-object scene.

Object Counting and Benchmarks Object counting has progressed from few-shot exemplar-based methods (Ranjan et al., 2021) to zero-shot (Xu et al., 2023) and open-world approaches (Amini-Naieni & Zisserman, 2026) that specify targets by class name, text prompt, or visual exemplar. Referring-expression counting (Dai et al., 2024) extends this further, allowing a target to be defined by attributes and relations. The benchmarks evaluate increasingly flexible target specification, and FIOC (Cho et al., 2026) brings counting into industrial scenes by annotating each instance with a defect attribute. Its training and test splits are drawn from the same object categories, however, so neither FIOC nor any other counting benchmark evaluates defect-state-aware counting on component classes unseen during training.

Table 1: **Comparison with industrial anomaly detection benchmarks.** *Multi-inst.* indicates whether an image contains several instances, where *partial* means only some categories do. *Zero-shot* describes how training and test data are divided, and *AC* whether the annotations support anomaly-aware counting, which requires a label on every instance rather than on the image or its pixels. Only Real-ZAC withholds entire part classes from training, so zero-shot generalization to unseen components can be measured.

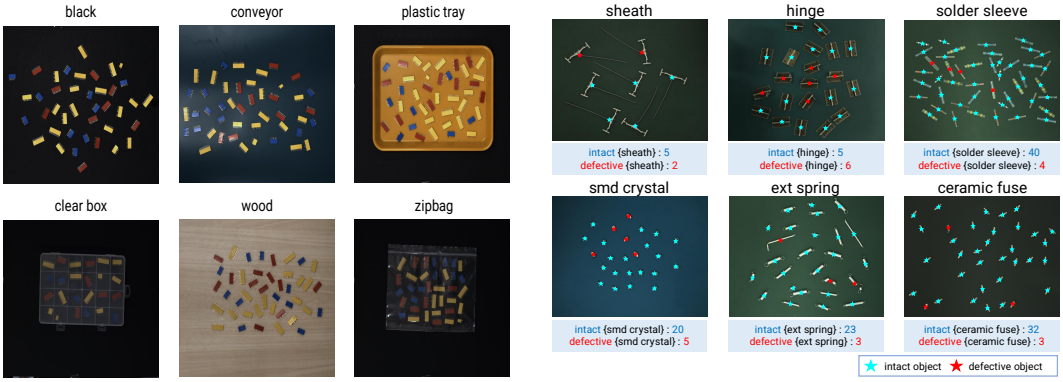
Dataset	Year	Cls.	Img.	Multi-inst.	Backgrounds	Zero-shot	AC
MVTec AD	2019	15	5,354	✗	1	✗	✗
BTAD	2021	3	2,830	✗	1	✗	✗
MVTec LOCO	2022	5	3,644	partial	1	✗	✗
VisA	2022	12	10,821	partial	1	✗	✗
Real-IAD	2024	30	151,050	✗	1	✗	✗
MVTec AD 2	2026	8	8,004	partial	1	✗	✗
FIOC	2026	10	1,000	✓	1	✗	✓
Real-ZAC	**2026**	**25**	**5,485**	✓	**6**	✓	✓

3 Anomaly Counting Dataset

Existing industrial anomaly detection benchmarks assign a single ground-truth mask per image, and thus cannot measure how many instances within it are intact. Real inspection lines, however, handle multiple components in a single frame and must judge each one independently. Therefore, we construct **Real-ZAC** that labels every instance as intact or defective, posing anomaly detection at the instance level. Table 1 compares Real-ZAC with existing benchmarks.

3.1 Dataset Collection

Real-ZAC contains 5,485 real images at a resolution of 2448×2048 across 25 classes of industrial components, each handled in bulk during inspection, with defects taken from failure modes that occur in manufacturing. Images are captured against six background conditions to reflect the real manufacturing environment (Figure 2a): black, conveyor, plastic tray, clear box, wood, and zip bag. Each class contains roughly 100 images on the black background and another 100 spread across the remaining five, giving 218 to 222 images per class. Classes whose parts cannot be placed reliably on every surface, because of their size or shape, are captured on four or five backgrounds rather than all six. Each image contains multiple instances of a single class, 21.4 on average, arranged without occlusion, since a defect hidden behind another part cannot be judged from the image.



(a) Background conditions (b) Point-level annotations

Figure 2: **Overview of the dataset conditions and annotations. (a)** The six background conditions used to evaluate robustness: black, conveyor, plastic tray, clear box, wood, and zip bag. **(b)** Annotated examples across six classes. Each object instance carries a single point labeled intact (cyan) or defective (red), with the total counts shown below.

3.2 Annotation and Statistics

Point-level instance annotation. Each instance is annotated with a single point labeled as either intact or defective, yielding a total of 117,124 point annotations. Points are primarily positioned at the geometric center of each object. For thin or elongated components whose geometric center falls outside the object boundary, we instead select a central point strictly within the visible region to ensure every point lies squarely on its target instance. Figure 2b shows annotated examples across several classes.

Defect statistics. The proportion of defective instances ranges from 10% to 31% across classes, averaging 17.8%. We additionally include images in which every instance is intact, or every instance is defective, together accounting for 8% to 12% of the images in each class. A model that judges an instance by comparing it against its neighbors has no reference to rely on in these images, so they reveal whether a model recognizes defects directly or only finds the instance that differs from the rest. Each class has its own set of defects, and a class may exhibit several of them, so a model cannot rely on a single canonical failure mode per part.

Dataset split. We split the 25 classes into 20 for training and 5 for evaluation, so that no class seen during training reappears at test time. The five evaluation classes include all four defect categories, deformation, fracture, missing part, and surface damage, and they come from four different material and shape families: metal wire forms (paperclip, ext spring), axial electronic components (resistor), molded plastic blocks (keyboard switch), and hybrid plastic-metal connectors (jst connector). Evaluation therefore tests whether defect-state recognition transfers to unseen component classes. Both intact and defective instances are available during training, so the difficulty comes from the gap between training and evaluation classes, not from withholding defect labels. Detailed information about the dataset is in Appendix A.

4 Method

4.1 Problem Definition

Given an image I containing multiple component instances and a query q, zero-shot anomaly-aware counting identifies all instances satisfying the query and returns their count. Each instance is associated with a component class c and a defect state $s \in \{\text{intact}, \text{defective}\}$. For a query $q = (c, s)$,

$$\mathcal{R}_q = \{r_i \mid c_i = c,\ s_i = s\}, \qquad \hat{N}_q = |\mathcal{R}_q|, \tag{1}$$

where all evaluation classes are unseen during training.

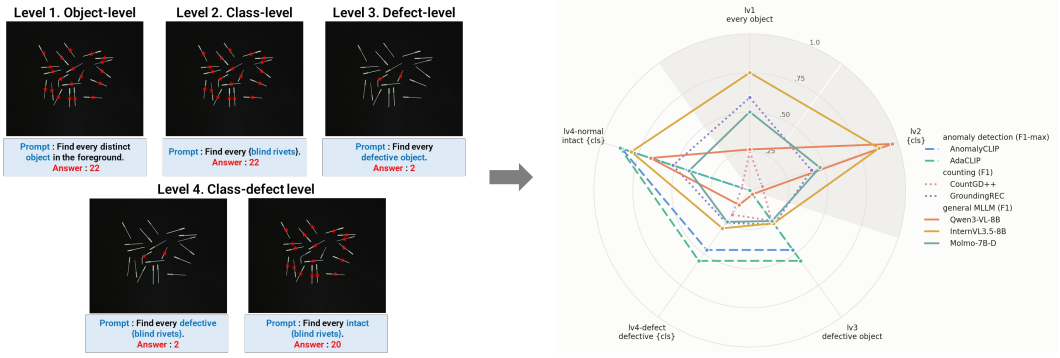


Figure 3: **Why do existing models fail at anomaly-aware counting?** Four queries are posed over the same images, each adding one condition: localization (lv1), class naming (lv2), defect state (lv3), and both class and state (lv4). Performance drops sharply when the query requires distinguishing defective from intact instances.

To disentangle these requirements, we organize queries into four levels of increasing specificity. **lv1** asks for every object in the scene, requiring only localization; **lv2** additionally specifies the component class; **lv3** specifies only the defect state; and **lv4** jointly specifies both class and state. Figure 3 shows that no existing model handles all four levels reliably. Anomaly detection models are specialized for distinguishing normal from abnormal images, but do not naturally perform class-conditioned instance counting. Counting models localize and enumerate objects effectively, but lack an explicit mechanism for distinguishing intact from defective instances. MLLMs can accept all query types, but their performance drops substantially when defect-state reasoning is required. These results identify instance-level state discrimination, together with localization and class grounding, as the key challenge of anomaly-aware counting.

4.2 Baseline Selection

The analysis suggests that zero-shot anomaly-aware counting requires strong region-level localization and explicit defect-state discrimination. We therefore build on ObjEmbed (Fu et al., 2026a), which represents all proposed regions as object tokens within a single multimodal forward pass. Its RoIAlign and spatial features provide strong localization and region-text alignment, but primarily encode object appearance and identity. We therefore incorporate a comparative anomaly descriptor into each region token to capture how an instance differs from other instances in the same image. Figure 4 illustrates the overall Pinpoint pipeline, from region encoding and anomaly descriptor construction to defect-state conditioned query matching.

4.3 Comparative Anomaly Descriptor

Let $\mathcal{R}$ denote the proposed regions in an image. Each region is encoded by a frozen vision encoder (Siméoni et al., 2025) into ℓ_2-normalized patch features $f_{r,p}$.

Mutual scoring. For each patch, we measure its distance to the most similar patch from any other region:

$$s_{r,p} = \min_{\substack{r' \in \mathcal{R} \setminus \{r\} \\ q \in \{1,\dots,P\}}} \left(1 - f_{r,p}^{\top} f_{r',q}\right). \tag{2}$$

Patches shared across instances receive low scores, whereas instance-specific deviations receive high scores. Unlike MuSc (Li et al., 2024), which compares patches across test images, we compare patches within a single image.

Anomaly-weighted pooling. Because defects may occupy only a few patches, we emphasize high-scoring patches:

$$\tilde{v}_r = \sum_p \frac{\exp(s_{r,p}/\tau)}{\sum_{p'} \exp(s_{r,p'}/\tau)} f_{r,p}. \tag{3}$$

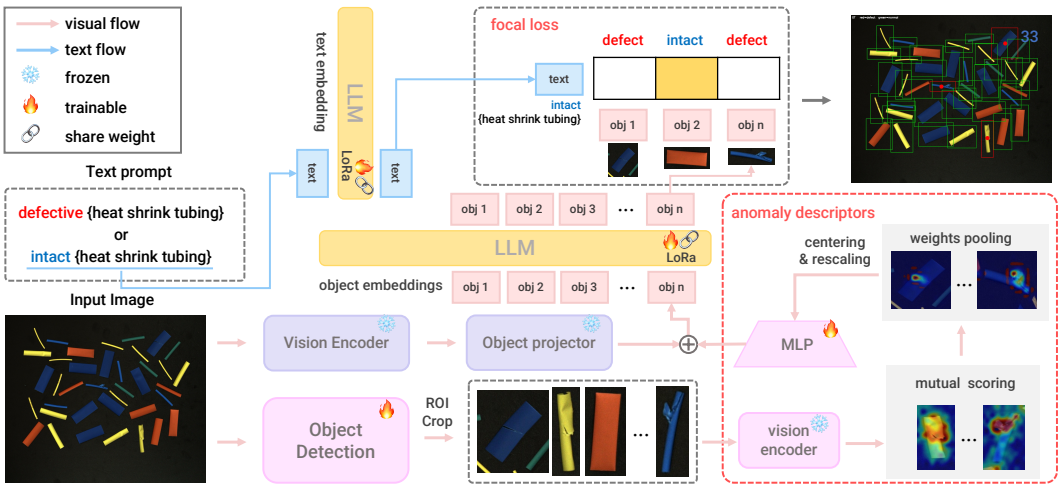


Figure 4: **Method Overview.** Pinpoint augments region tokens with within-image comparative anomaly descriptors derived from frozen DINOv3 features. The enhanced object representations are aligned with intact/defective text queries to perform instance-level state recognition and counting.

Within-image centering. To suppress appearance shared across instances, we subtract the median normalized descriptor:

$$\hat{v}_r = \frac{\tilde{v}_r}{\|\tilde{v}_r\|} - \underset{r' \in \mathcal{R}}{\text{median}} \frac{\tilde{v}_{r'}}{\|\tilde{v}_{r'}\|}. \tag{4}$$

Magnitude restoration. We summarize anomaly strength using the mean of the top-k patch scores,

$$\bar{s}_r = \frac{1}{k} \sum_{p \in \text{top}_k(s_{r,\cdot})} s_{r,p}, \tag{5}$$

and obtain the final descriptor

$$v_r^{\text{dino}} = \bar{s}_r \hat{v}_r. \tag{6}$$

Thus, v_r^{dino} encodes both the direction and magnitude of each instance's deviation from the others.

4.4 Descriptor Injection

ObjEmbed forms each region token from its RoIAlign feature and spatial encoding. We inject the comparative descriptor as an additional gated term:

$$\texttt{<object>} = \text{roi_feats} + \text{box_coor} + g \cdot \text{LN}\big(\text{MLP}(v_r^{\text{dino}})\big). \tag{7}$$

The gate g is initialized to zero, preserving the original ObjEmbed representation at the start of training and allowing the model to gradually incorporate anomaly information.

4.5 Training and Inference

Training. Regions are matched one-to-one with ground-truth points using Hungarian assignment. Matched regions inherit the corresponding state label, while unmatched regions are treated as negatives. We use focal loss (Lin et al., 2017) to account for the imbalance between intact and defective instances and sample state queries from a synonym pool. The vision encoder, RoIAlign path, and ObjEmbed projectors remain frozen, while the descriptor projection and gate are trained directly and the language-model attention projections are adapted with LoRA (Hu et al., 2021).

Inference. Each region is scored by its similarity $s_q(r)$ to the query embedding. For object queries, including class-agnostic and class-conditional queries, we apply a quantile-based threshold, with the quantile selected on held-out validation categories. For defect-state conditioned queries, we directly compare the defective and intact scores using

$$m(r) = s_{\text{defective}}(r) - s_{\text{intact}}(r). \tag{8}$$

Table 2: **Anomaly-aware counting on Real-ZAC.** All methods are evaluated on Real-ZAC test set. We compare models without Real-ZAC training and after training on the class-disjoint Real-ZAC training split, with the change from training shown in parentheses. AAC denotes anomaly-aware counting. Bold marks the best result.

		No Real-ZAC training			Trained on Real-ZAC		
Method	Type	MAE↓	RMSE↓	F1↑	MAE↓	RMSE↓	F1↑
Molmo-7B-D	MLLM	16.36	19.91	0.322	4.43 -11.93	7.46 -12.45	0.743 +0.421
Llama-3.2-11B-Vision	MLLM	12.17	17.20	0.011	3.74 -8.43	6.66 -10.54	0.731 +0.720
Qwen3-VL-8B	MLLM	6.09	9.95	0.594	6.76 +0.67	11.46 +1.51	0.666 +0.072
InternVL3.5-8B	MLLM	9.01	13.60	0.608	3.26 -5.75	6.46 -7.14	0.723 +0.115
ObjEmbed-2B	MLLM	3.60	4.77	0.728	2.73 -0.87	3.70 -1.07	0.802 +0.074
SAM3	Seg.	12.60	17.08	0.082	12.96 +0.36	17.37 +0.29	0.148 +0.066
FALCON	AAC	6.54	10.93	0.553	8.80 +2.26	13.44 +2.51	0.408 -0.145
LocateAnything-3B	Grounding	19.70	28.70	0.347	17.80 -1.90	26.14 -2.56	0.434 +0.087
CountGD++	Counting	12.16	16.51	0.112	3.78 -8.38	5.69 -10.82	0.752 +0.640
GroundingREC	Counting	7.36	10.76	0.438	7.34 -0.02	10.85 +0.09	0.477 +0.039
Pinpoint	AAC	—	—	—	**2.09**	**3.29**	**0.872**

A region is classified as defective when $m(r) > 0$ and intact otherwise, which is equivalent to selecting the state with the higher similarity score.

5 Experiments

5.1 Experimental Setup

Evaluation Dataset As additional test data, we collect anomaly detection benchmark datasets that contain multi-instance images. From VisA (Zou et al., 2022), MVTec AD 2 (Heckler-Kram et al., 2026), and MVTec LOCO (Bergmann et al., 2022), we take the test sets of their multi-instance categories and re-annotate them for anomaly-aware counting. For MVTec LOCO (Bergmann et al., 2022), we use only the structural anomalies, since logical anomalies are defined as violations of scene-level constraints rather than defects of individual objects and therefore cannot be assigned to a specific instance. In total, we annotate 3,624 images across 8 categories, yielding 32,267 instances of which 822 are defective.

Evaluation Metrics For an image i and a defect-state query q,

$$\text{MAE} = \frac{1}{|\mathcal{D}|} \sum_{(i,q)\in\mathcal{D}} |\hat{c}_{i,q} - c_{i,q}|, \qquad \text{RMSE} = \sqrt{\frac{1}{|\mathcal{D}|} \sum_{(i,q)\in\mathcal{D}} (\hat{c}_{i,q} - c_{i,q})^2}, \tag{9}$$

where $\mathcal{D}$ denotes the set of image-query pairs and $\hat{c}_{i,q}$ and $c_{i,q}$ are the predicted and ground-truth counts. Since counting metrics alone do not verify whether the correct instances were identified, we additionally report instance-level F_1. Predicted points are matched one-to-one with ground-truth points using Hungarian matching and are considered correct within $0.05\times$ the image diagonal, requiring both correct localization and defect state.

For zero-shot anomaly detection models, which output continuous anomaly scores, we report instance-level AUROC and $\text{F1}_{\max}$. AUROC measures ranking quality, while $\text{F1}_{\max}$ sweeps the decision threshold and measures the best achievable separation between intact and defective instances. Since these scores are uncalibrated and shift in scale across unseen categories, sweeping removes calibration from the comparison and grants every baseline its best operating point. For Pinpoint, the swept score is the margin $m(r)$ between the defective and intact query scores, so its inference rule at $t = 0$ is a single point on the same curve.

Implementation Details. Region proposals are generated by CountGD++ (Amini-Naieni & Zisserman, 2026), fine-tuned on the Real-ZAC training split with the class-agnostic query `"object ."`.

Table 3: **Anomaly-Aware counting on AD benchmarks.** AUROC measures ROI-level anomaly ranking, and $F1_{max}$ is the maximum F1 obtained by sweeping the anomaly-score threshold on the evaluation set. * indicates released weights trained on the corresponding dataset.

	MVTec AD 2		MVTec LOCO		VisA		Avg	
Method	AUROC	$F1_{max}$	AUROC	$F1_{max}$	AUROC	$F1_{max}$	AUROC	$F1_{max}$
AnomalyCLIP	.766	.341	.454	.089	.809*	.417*	.709	.316
AdaCLIP	.730	.238	.627	.173	.796	.410	.737	.308
AnomalyR1	.738	.376	.599*	.116*	.670*	.169*	.669	.208
DAPO	**.803**	.384	.565	.093	.715	.265	.700	.252
MuSc	.569	.189	.626	.192	.834	.514	.716	.352
Pinpoint	.783	**.462**	**.792**	**.454**	**.903**	**.560**	**.845**	**.509**

For the zero-shot anomaly-detection comparison, all methods instead use identical WeDetect (Fu et al., 2026b) proposals to isolate defect-state recognition from proposal quality. Each proposal is expanded by an 8% margin, resized to 224×224, and encoded with a frozen DINOv3 ViT-L/16 (Siméoni et al., 2025). Remaining hyperparameters are given in Appendix B.1.

5.2 Comparison with Existing Models

Generic Vision-Language Models Table 2 compares Pinpoint with MLLMs, grounding and counting baselines, and FALCON (Cho et al., 2026), the closest prior anomaly-aware counting method, under the same class-disjoint zero-shot evaluation. Models are evaluated either without Real-ZAC training or after training on the Real-ZAC training classes, while all evaluation classes remain unseen. CountGD++ (Amini-Naieni & Zisserman, 2026), GroundingREC (Dai et al., 2024), and SAM3 (Carion et al., 2026) are trained in full with their official recipes, while MLLMs (Bai et al., 2025; AI@Meta, 2024; Zhu et al., 2025) are adapted with LoRA on the attention projections. Without Real-ZAC training, most baselines struggle with defect-conditioned counting. FALCON performs comparatively well because its FIOC training set contains component classes and defect types that overlap with those in Real-ZAC, giving it prior exposure to part of the evaluation distribution. However, FALCON was not designed for class-disjoint zero-shot generalization, and its performance decreases after training on the Real-ZAC training classes, suggesting that additional adaptation to seen classes can hurt transfer to unseen ones. In contrast, most MLLMs and CountGD++ benefit substantially from Real-ZAC training, while Pinpoint achieves the best MAE, RMSE, and F1 among all compared methods.

Zero-shot Anomaly Detection Models Table 3 evaluates anomaly detection models on identical WeDetect ROIs, isolating defect-state recognition from localization. Zero-shot AD models generally produce uncalibrated anomaly scores whose scale can vary across unseen categories and datasets. This is particularly evident for test-time comparative methods such as MuSc (Li et al., 2024), which derives anomaly scores by comparing unlabeled test samples with one another, making the score distribution dependent on the target set itself. Consequently, applying a single fixed threshold across unseen datasets can confound defect separability with score calibration. We therefore report $F1_{max}$, obtained by sweeping the decision threshold, to measure how well each method's anomaly score can separate intact from defective instances without assuming target-domain calibration. Pinpoint achieves the highest average instance-level AUROC and $F1_{max}$, and obtains the best $F1_{max}$ on all three external benchmarks, indicating that its scores preserve strong defect-state separability even on component categories and datasets not seen during training.

5.3 Cross-dataset transfer.

Table 4 examines whether Real-ZAC provides transferable supervision beyond its own acquisition domain. Training on Real-ZAC yields competitive results across VisA, MVTec AD 2, and MVTec LOCO, with the effect varying across models and metrics. The gain is most consistent on MVTec LOCO, where four of the five models improve on every metric, even though its scenes differ from Real-ZAC in that several part classes share a frame. Absolute scores on MVTec LOCO remain low for all models, since a query naming one class also retrieves the others, and the models trained on Real-ZAC narrow this gap without closing it. These results suggest that Real-ZAC contains

Table 4: **Cross-dataset transfer for zero-shot anomaly-aware counting.** For each held-out benchmark, we train models either on the remaining anomaly-detection datasets or on Real-ZAC, and evaluate them using identical WeDetect proposals. Bold indicates the best result in each metric column.

		VisA			MVTec AD 2			MVTec LOCO		
Model	Training corpus	MAE ↓	RMSE ↓	$F1_{max}$ ↑	MAE ↓	RMSE ↓	$F1_{max}$ ↑	MAE ↓	RMSE ↓	$F1_{max}$ ↑
AnomalyCLIP	Other AD datasets	0.13	0.37	.277	0.96	1.49	.399	0.18	0.43	.071
	Real-ZAC	0.36	0.74	.212	2.61	3.44	.412	**0.13**	**0.36**	.071
AdaCLIP	Other AD datasets	1.49	3.12	.077	2.48	3.57	.383	0.23	0.49	.088
	Real-ZAC	1.30	2.54	.067	2.25	2.83	.330	0.17	0.43	.136
AnomalyR1	Other AD datasets	0.55	0.93	.120	2.32	3.33	.427	0.23	0.50	.076
	Real-ZAC	1.49	2.34	.102	1.86	2.59	.496	0.15	0.39	.103
ObjEmbed-2B	Other AD datasets	0.18	0.43	.104	0.99	1.59	.162	0.15	0.40	.164
	Real-ZAC	0.15	0.40	.167	1.08	1.47	.207	0.52	0.82	.083
Pinpoint (ours)	Other AD datasets	0.27	0.62	.509	0.61	**0.87**	**.527**	1.09	1.39	.162
	Real-ZAC	**0.09**	**0.30**	**.643**	**0.58**	0.94	**.527**	0.35	0.62	**.263**

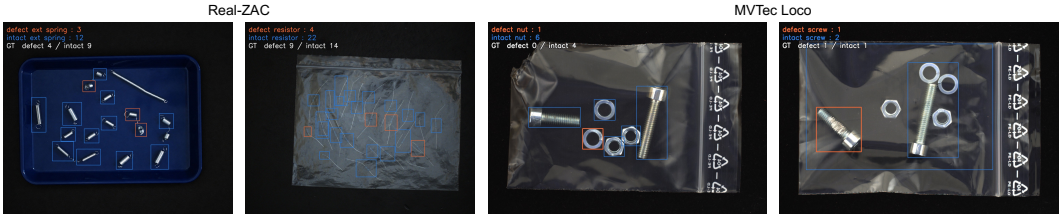


Figure 5: **Failure cases.** Blue boxes mark instances predicted as intact and red as defective, with predicted and ground-truth counts above each image.

defect-state supervision that generalizes to external industrial datasets rather than being limited to its own component classes and imaging conditions.

5.4 Failure Cases and Discussion

Figure 5 shows representative failure cases. On Real-ZAC, thin components leave few patches inside the region, so the comparison has little evidence to work with, especially when the defect lies in a narrow section such as a bent lead. A second difficulty arises when a frame contains only defective instances, since the reference drifts toward the defective appearance and the deviation that should mark a defect is subtracted away with the class identity, leading the model to report most parts as intact. On MVTec LOCO, frames holding several part classes violate the assumption that instances share a class, so a part of a different class is genuinely unlike its neighbors and receives a high anomaly score for the wrong reason, and the proposal stage compounds this by occasionally covering a cluster of mixed parts with one box. These cases point to finer defect representations, to a reference that does not depend entirely on the frame, and to modeling scenes that mix classes.

6 Conclusion

We introduced Zero-shot Anomaly-Aware Counting (ZAC), a task for counting intact or defective instances of component classes unseen during training. We also constructed Real-ZAC, a class-disjoint benchmark with instance-level annotations for this setting. Finally, we developed Pinpoint, a baseline that suppresses dominant class-specific appearance and emphasizes subtle defect-related variations between instances. This design improves anomaly-aware counting across unseen component classes by focusing less on object identity and more on fine-grained state differences. We hope our work provides a foundation for developing zero-shot anomaly-aware counting methods that generalize beyond object identity and explicitly reason about the condition of each instance.

AI USE STATEMENT

We used generative AI tools (large language models) as a general-purpose writing and coding assistant. Specifically, AI assistance was used for (i) polishing the grammar and clarity of text that was first drafted by the authors, and (ii) writing boilerplate code for data loading, evaluation scripts, and plotting. We did not use generative AI tools for research ideation, for deriving the method, for designing the experiments. All technical contributions, claims, and analyses are the authors' own.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or sensitive data. The newly introduced Real-ZAC dataset consists exclusively of images of industrial components. The authors declare no conflicts of interest and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The external benchmarks used in our evaluation are publicly available, and Real-ZAC together with the complete training and evaluation code will be released upon publication. Implementation details, including the training schedule, learning rates, and the parameters adapted in each baseline, are given in Appendix B.

REFERENCES

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3/MODEL_CARD.md.

Niki Amini-Naieni and Andrew Zisserman. CountGD++: Generalized prompting for open-world counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Niki Amini-Naieni, Tengda Han, and Andrew Zisserman. Countgd: Multi-modal open-world counting. *Advances in Neural Information Processing Systems*, 37:48810–48837, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. MVTec AD – a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9592–9600, 2019. doi: 10.1109/CVPR.2019.00982.

Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *International Journal of Computer Vision*, 130(4):947–969, 2022. doi: 10.1007/s11263-022-01578-9.

Yunkang Cao, Jiangning Zhang, Luca Frittoli, Yuqi Cheng, Weiming Shen, and Giacomo Boracchi. AdaCLIP: Adapting clip with hybrid learnable prompts for zero-shot anomaly detection. In *European Conference on Computer Vision (ECCV)*, pp. 55–72, 2024. doi: 10.1007/978-3-031-72761-0_4.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al.

Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Yuhao Chao, Jie Liu, Jie Tang, and Gangshan Wu. AnomalyR1: A GRPO-based end-to-end MLLM for industrial anomaly detection. *arXiv preprint arXiv:2504.11914*, 2025.

Suhye Cho, Younghun Byeon, Sang Jun Park, Jun-Seok Yun, Ye Jin Kim, Hong-In Won, Min Su Kim, and Jong Pil Yun. FALCON: Fine-grained alignment for counting objects in industry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 7230–7239, 2026.

Siyang Dai, Jun Liu, and Ngai-Man Cheung. Referring expression counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16985–16995, 2024.

Shenghao Fu, Yukun Su, Fengyun Rao, Jing Lyu, Xiaohua Xie, and Wei-Shi Zheng. ObjEmbed: Towards universal multimodal object embeddings. In *International Conference on Machine Learning (ICML)*, 2026a.

Shenghao Fu, Yukun Su, Fengyun Rao, Jing Lyu, Xiaohua Xie, and Wei-Shi Zheng. WeDetect: Fast open-vocabulary object detection as retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20377–20387, 2026b.

Zhaopeng Gu, Bingke Zhu, Guibo Zhu, Yingying Chen, Ming Tang, and Jinqiao Wang. AnomalyGPT: Detecting industrial anomalies using large vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 1932–1940, 2024. doi: 10.1609/aaai.v38i3.27963.

Lars Heckler-Kram, Jan-Hendrik Neudeck, Ulla Scheler, Rebecca König, and Carsten Steger. The MVTec AD 2 dataset: Advanced scenarios for unsupervised anomaly detection. *International Journal of Computer Vision*, 134(4), 2026. doi: 10.1007/s11263-026-02743-0. Article 175.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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 (CVPR)*, pp. 19606–19616, 2023. doi: 10.1109/CVPR52729.2023.01878.

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*, volume 2024, pp. 2172–2197, 2024.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Wenxin Ma, Xu Zhang, Qingsong Yao, Fenghe Tang, Chenxu Wu, Yingtai Li, Rui Yan, Zihang Jiang, and S. Kevin Zhou. AA-CLIP: Enhancing zero-shot anomaly detection via anomaly-aware clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4744–4754, 2025. doi: 10.1109/CVPR52734.2025.00447.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3393–3402. IEEE, 2021.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025.

Chengjie Wang, Wenbing Zhu, Bin-Bin Gao, Zhenye Gan, Jiangning Zhang, Zhihao Gu, Shuguang Qian, Mingang Chen, and Lizhuang Ma. Real-IAD: A real-world multi-view dataset for benchmarking versatile industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22883–22892, 2024.

Jingyi Xu, Hieu Le, Vu Nguyen, Viresh Ranjan, and Dimitris Samaras. Zero-shot object counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15548–15557, 2023.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

Qihang Zhou, Guansong Pang, Yu Tian, Shibo He, and Jiming Chen. AnomalyCLIP: Object-agnostic prompt learning for zero-shot anomaly detection. In *International Conference on Learning Representations (ICLR)*, 2024.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Yue Cao, Yangzhou Liu, Weiye Xu, Hao Li, Jiahao Wang, Han Lv, Dengnian Chen, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. SPot-the-Difference self-supervised pre-training for anomaly detection and segmentation. In *European Conference on Computer Vision (ECCV)*, 2022. doi: 10.1007/978-3-031-20056-4_23.

ZAC: Zero-shot Anomaly-Aware Counting for Industrial Components: Appendix

Table of Contents

A Real-ZAC Dataset Details 14
A.1 Per-class Composition 14
A.2 Defect Examples 14
A.3 Defect Ratio and Instance Density 15
B Implementation and Evaluation Details 15
B.1 Training Details 15
B.2 External Benchmark Evaluation 15
C Additional Experimental Results 17
C.1 Full Results Across Query Levels 17
C.2 Per-level and Per-class Results 17
C.3 Robustness Across Class Splits 17
C.4 Efficiency Comparison 18
C.5 Qualitative Results 18
D Ablation Studies 19
D.1 Components and Language-Model Backbone 19
D.2 Representation Analysis 20
E Additional Anomaly Detection Results 20
E.1 Fine-tuning Anomaly Detection Models on Real-ZAC 20
E.2 Per-category Anomaly Detection Results 20

A REAL-ZAC DATASET DETAILS

A.1 PER-CLASS COMPOSITION

Table 5: **Real-ZAC per class.** *Intact only* and *Defective only* count images whose objects are all intact or all defective and the remaining images mix both. Defect categories group the annotated defect types: fracture (broken, cut, torn, ruptured), deformation (bent, crushed, dented, extended, shrunken), missing part (missing or lost pins, omitted or truncated parts, missing terminals) and surface damage (damaged, worn, peeled).

		Images				Objects		
Class	Split	Total	Intact only	Defective only	Mixed	Total	Defective (%)	Defect category
sheath	Train	219	9	10	200	1,692	538 (31.8)	Deformation
fuse	Train	220	9	9	202	5,306	945 (17.8)	Fracture, Deformation, Surface damage
cradle	Train	220	10	10	200	5,778	1,250 (21.6)	Fracture, Deformation, Missing part
gear	Train	220	10	10	200	4,805	789 (16.4)	Fracture, Surface damage
aluminum capacitor	Train	220	10	10	200	5,614	964 (17.2)	Deformation
switch	Train	220	10	10	200	5,503	939 (17.1)	Fracture, Deformation
heat shrink tubing	Train	218	9	8	201	4,837	917 (19.0)	Fracture, Deformation
potentiometer	Train	220	7	10	203	5,086	941 (18.5)	Fracture, Deformation
capacitor	Train	220	10	10	200	5,725	722 (12.6)	Fracture, Missing part, Surface damage
blind rivet	Train	220	10	10	200	4,229	855 (20.2)	Fracture, Deformation
dip 8pin	Train	218	15	3	200	4,895	831 (17.0)	Fracture
ceramic capacitor	Train	221	26	1	194	5,419	575 (10.6)	Fracture
tie mount	Train	218	9	9	200	4,836	746 (15.4)	Fracture
smart card interface ic	Train	220	10	10	200	4,610	991 (21.5)	Fracture
tact switch	Train	219	9	10	200	4,594	762 (16.6)	Fracture
hinge	Train	218	9	9	200	2,964	647 (21.8)	Deformation
ceramic fuse	Train	222	11	11	200	4,660	911 (19.5)	Fracture
led indicator	Train	218	9	9	200	4,644	709 (15.3)	Missing part, Surface damage
smd crystal	Train	220	10	10	200	4,921	820 (16.7)	Surface damage
solder sleeve	Train	218	9	9	200	5,048	829 (16.4)	Missing part, Surface damage
resistor	Test	218	9	9	200	4,686	1,022 (21.8)	Fracture
ext spring	Test	218	9	9	200	3,966	746 (18.8)	Deformation
paperclip	Test	220	14	9	197	4,254	725 (17.0)	Fracture, Deformation
keyboard switch	Test	220	10	10	200	4,213	836 (19.8)	Fracture, Deformation
jst connector	Test	220	11	10	199	4,839	883 (18.2)	Fracture, Missing part
Total (20 classes)	Train	4,389	211	178	4,000	95,166	16,681 (17.5)	
Total (5 classes)	Test	1,096	53	47	996	21,958	4,212 (19.2)	
Total (25 classes)	All	5,485	264	225	4,996	117,124	20,893 (17.8)	

Table 5 summarizes the composition of all 25 component classes. Images are approximately balanced across classes, while individual classes contain multiple defect types. The evaluation classes are entirely excluded from training, providing a class-disjoint setting for measuring generalization to unseen components.

A.2 DEFECT EXAMPLES

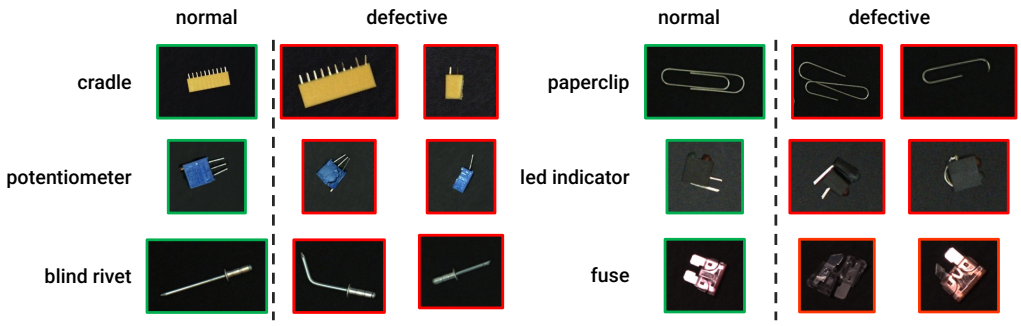
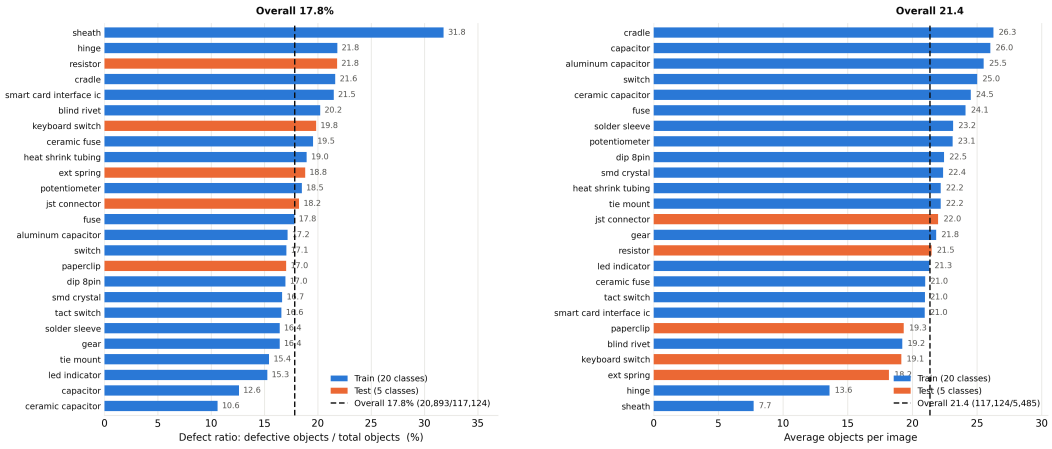


Figure 6: **Examples of defective instances.** For each component class, one intact and two defective instances are shown.

Figure 6 illustrates the diversity of defect appearances within individual component classes. Defects are not represented by a single canonical pattern, and even intact instances exhibit moderate

appearance variation, requiring models to distinguish subtle state-related cues from normal intra-class variation.

A.3 Defect Ratio and Instance Density



(a) Proportion of defective instances

(b) Average instances per image

Figure 7: **Per-class statistics. (a)** Fraction of defective instances in each component class; evaluation classes are marked with an orange box. **(b)** Average number of annotated instances per image.

Figure 7 shows substantial variation in both defect frequency and instance density across classes. Intact instances remain dominant in most scenes, while defective instances appear among multiple visually similar components. Instance density also varies considerably with component size, producing scenes ranging from sparse arrangements to densely packed small parts.

B Implementation and Evaluation Details

B.1 Training Details

The comparative descriptor uses $P = 196$ patch tokens from layer 17 of the frozen DINOv3 encoder, with softmax temperature $\tau = 0.05$, and restores anomaly magnitude from the top five mutual scores.

Region proposals are matched one-to-one with ground-truth points using Hungarian assignment. Matched regions inherit the corresponding intact or defective state, while unmatched proposals are treated as negatives. We use focal loss (Lin et al., 2017) to account for the imbalance between intact and defective instances.

We train for 3.5k steps with AdamW using a weight decay of 10^{-4}. The learning rate is 10^{-4} for the LoRA parameters and descriptor projection MLP, and 10^{-2} for the gate and scalar parameters. LoRA with rank 16 is applied to the language-model attention projections. The vision encoder, RoIAlign pathway, DINOv3 encoder, and pretrained ObjEmbed projectors remain frozen. In total, Pinpoint has 12.7M trainable parameters, corresponding to 0.47% of the full model.

B.2 External Benchmark Evaluation

For external anomaly-detection benchmarks, all methods are evaluated using the same region proposals, NMS, and point-matching protocol to separate defect-state recognition from proposal quality. Multi-instance scenes are evaluated at the instance level using intact/defective annotations. For anomaly-detection methods that output continuous anomaly scores, we report instance-level AUROC and $\mathrm{F1}_{\max}$ in addition to counting metrics.

Table 6: Query-levels counting on the Real-ZAC test split. Performance is compared between zero-shot (released weights) and fine-tuned on the 20 training classes. Best per level in bold.

	Zero-shot					Fine-tuned				
Method	MAE ↓	RMSE ↓	R ↑	P ↑	F1 ↑	MAE ↓	RMSE ↓	R ↑	P ↑	F1 ↑
lv1 — *every object*										
Molmo-7B-D	13.40	19.01	0.386	0.718	0.502	4.38	7.91	0.976	0.815	0.888
Llama-3.2-11B-V	15.01	19.37	0.079	0.103	0.089	1.51	2.56	0.918	0.937	0.927
Qwen3-VL-8B	16.95	22.53	0.151	0.971	0.262	4.55	10.62	0.861	0.824	0.842
InternVL3.5-8B	7.03	12.10	0.636	0.919	0.752	4.26	8.91	0.936	0.786	0.855
CountGD++	17.16	21.53	0.143	**0.977**	0.249	2.12	5.37	**0.995**	0.902	0.946
GroundingREC	11.34	15.45	0.427	0.969	0.593	2.69	4.13	0.850	0.962	0.903
ObjEmbed-2B	0.88	1.74	0.984	0.962	0.973	**0.82**	**1.66**	0.986	0.964	**0.975**
Ours	–	–	–	–	–	**0.82**	1.68	0.986	0.964	**0.975**
lv2 — {*class*}										
Molmo-7B-D	14.34	19.54	0.339	0.779	0.473	4.62	8.66	0.978	0.807	0.884
Llama-3.2-11B-V	11.51	16.17	0.153	0.149	0.151	1.55	2.67	0.917	0.937	0.927
Qwen3-VL-8B	1.05	2.99	0.959	0.963	0.961	3.07	8.79	0.898	0.887	0.892
InternVL3.5-8B	3.24	7.80	0.838	0.909	0.872	4.57	9.61	0.936	0.782	0.852
CountGD++	19.15	22.94	0.044	**0.992**	0.084	1.78	4.55	**0.995**	0.917	0.954
GroundingREC	14.42	18.91	0.268	0.934	0.416	14.63	19.90	0.266	0.984	0.419
ObjEmbed-2B	1.30	2.15	0.950	0.961	0.955	**0.76**	**1.59**	0.988	0.966	**0.977**
Ours	–	–	–	–	–	0.92	1.93	0.984	0.962	0.973
lv3 — *defective object*										
Molmo-7B-D	21.02	23.62	**0.895**	0.140	0.242	4.32	7.44	0.547	0.369	0.441
Llama-3.2-11B-V	5.59	9.80	0.008	0.010	0.009	3.43	6.08	0.495	0.480	0.488
Qwen3-VL-8B	3.79	4.60	0.015	0.580	0.030	6.35	10.87	0.338	0.224	0.269
InternVL3.5-8B	14.70	19.09	0.717	0.159	0.260	3.26	7.13	0.444	0.411	0.427
CountGD++	10.23	14.67	0.435	0.165	0.239	2.42	3.22	0.318	0.288	0.302
GroundingREC	8.85	12.26	0.528	0.179	0.268	9.36	12.25	**0.656**	0.216	0.325
ObjEmbed-2B	4.16	5.30	0.065	0.167	0.093	2.80	3.68	0.358	0.479	0.410
Ours	–	–	–	–	–	**1.96**	**2.94**	0.476	**0.813**	**0.600**
lv4-D — *defective* {*class*}										
Molmo-7B-D	20.40	22.89	0.899	0.144	0.248	4.61	8.01	0.566	0.357	0.438
Llama-3.2-11B-V	8.35	14.24	0.027	0.015	0.019	3.55	6.39	0.488	0.462	0.475
Qwen3-VL-8B	3.82	4.76	0.065	0.506	0.115	7.68	12.87	0.494	0.230	0.314
InternVL3.5-8B	14.12	18.19	0.836	0.182	0.299	3.10	6.70	0.477	0.425	0.449
SAM3	9.01	13.65	0.293	0.150	0.198	9.65	14.21	0.382	0.168	0.233
LocateAnything-3B	25.40	34.87	0.964	0.128	0.225	23.48	31.50	**0.973**	0.137	0.241
CountGD++	8.61	13.03	0.298	0.141	0.191	2.37	3.17	0.313	0.289	0.300
GroundingREC	5.40	7.11	0.361	0.200	0.257	5.41	7.62	0.560	0.279	0.372
ObjEmbed-2B	3.28	4.29	0.082	0.173	0.112	2.90	3.79	0.392	0.468	0.427
Ours	–	–	–	–	–	**1.87**	**2.86**	0.513	**0.788**	**0.621**
lv4-N — *intact* {*class*}										
Molmo-7B-D	12.32	16.39	0.296	0.662	0.409	4.25	6.87	0.867	0.802	0.833
Llama-3.2-11B-V	15.98	19.71	0.003	0.035	0.006	3.93	6.93	0.759	0.838	0.796
Qwen3-VL-8B	8.35	13.24	0.570	0.810	0.669	5.83	9.85	0.870	0.714	0.785
InternVL3.5-8B	3.91	6.24	0.856	0.749	0.799	3.42	6.22	0.817	0.764	0.790
SAM3	16.19	19.93	0.000	0.000	0.000	16.28	20.05	0.048	0.470	0.088
LocateAnything-3B	14.01	20.77	0.421	0.595	0.493	12.11	19.35	0.615	0.628	0.621
CountGD++	15.71	19.38	0.028	**0.948**	0.054	5.20	7.40	**0.981**	0.748	0.849
GroundingREC	9.32	13.45	0.391	0.766	0.518	9.28	13.31	0.394	0.795	0.527
ObjEmbed-2B	3.91	5.20	0.887	0.779	0.829	2.56	**3.62**	0.922	0.842	0.880
Ours	–	–	–	–	–	**2.32**	3.67	0.975	**0.867**	**0.918**

C Additional Experimental Results

C.1 Full Results Across Query Levels

Table 6 provides the full results across the four query levels, with L4 reported separately for defective (L4-D) and intact (L4-N) class-conditioned queries. We compare released models without Real-ZAC training against models fine-tuned on the 20 training classes, while all evaluation classes remain unseen during training. Fine-tuning substantially improves most methods on object and class-conditioned queries (lv1–lv2), whereas defect-conditioned queries (lv1–lv2) remain considerably more challenging. The persistent gap between lv1–lv2 and lv3–lv4 shows that recognizing the state of individual instances remains the main challenge even when localization and class grounding are reliable.

C.2 Per-level and Per-class Results

Table 7: Per-level performance of our method on the Real-ZAC test split.

Level	Query	GT	MAE↓	RMSE↓	F1↑
lv1	`every object`	21,958	0.82	1.68	0.975
lv2	`{cls}`	21,958	0.92	1.93	0.973
lv3	`defective object`	4,212	1.96	2.94	0.600
lv4-defect	`defective {cls}`	4,212	1.87	2.86	0.621
lv4-normal	`intact {cls}`	17,746	2.32	3.67	0.918
lv4 pooled (both queries)			**2.09**	**3.29**	**0.872**

Table 8: Level-4 performance test category.

	`defective {cls}`			`intact {cls}`		
Category	GT	MAE↓	F1↑	GT	MAE↓	F1↑
resistor	1,022	3.98	0.244	3,664	4.44	0.881
ext spring	746	1.47	0.603	3,220	3.49	0.875
paperclip	725	1.27	0.692	3,529	1.10	0.953
keyboard switch	836	1.05	0.764	3,377	1.05	0.935
jst connector	883	1.57	0.719	3,956	1.56	0.949

Table 7 evaluates Pinpoint across the four query levels. Localization and class-conditioned queries remain comparatively reliable, whereas defect-aware queries are more challenging, confirming that instance-level state discrimination is a major bottleneck beyond localization and class grounding.

Table 8 further reports anomaly-aware counting performance by evaluation class. Performance varies with component geometry and defect appearance, with thin structures being particularly challenging because they provide fewer visual patches from which subtle defects can be identified.

Table 9: **Five-fold class-disjoint evaluation on Real-ZAC.** Each fold holds out five component classes that are entirely absent from training. We report anomaly-aware counting performance on the held-out classes. The final row reports the mean and standard deviation across the five folds.

Fold	Test classes	MAE ↓	RMSE ↓	P ↑	R ↑	F_1 ↑	AUROC ↑
A	resistor, ext spring, paperclip, keyboard switch, jst connector	2.09	3.29	0.858	0.886	0.872	0.858
B	fuse, heat shrink tubing, dip 8pin, ceramic capacitor, hinge	1.27	2.34	0.916	0.917	0.916	0.931
C	potentiometer, capacitor, blind rivet, tact switch, solder sleeve	1.83	3.58	0.882	0.892	0.887	0.932
D	gear, smart card interface ic, ceramic fuse, led indicator, smd crystal	1.43	3.05	0.911	0.942	0.926	0.949
E	sheath, cradle, aluminum capacitor, switch, tie mount	1.99	3.09	0.855	0.885	0.870	0.878
Mean ± std		1.72 ±0.32	3.07 ±0.41	0.884 ±0.026	0.904 ±0.022	0.894 ±0.023	0.910 ±0.035

C.3 Robustness Across Class Splits

The main evaluation holds out five component classes entirely from training. To examine whether the results depend on this particular split, we additionally perform five-fold class-disjoint evaluation, where every component class serves as an unseen evaluation category once. As shown in Table 9, performance remains consistent across folds, with relatively small variation in F_1 and AUROC despite changing the entire set of evaluation classes. This indicates that the benchmark conclusions are not specific to a particular choice of held-out classes.

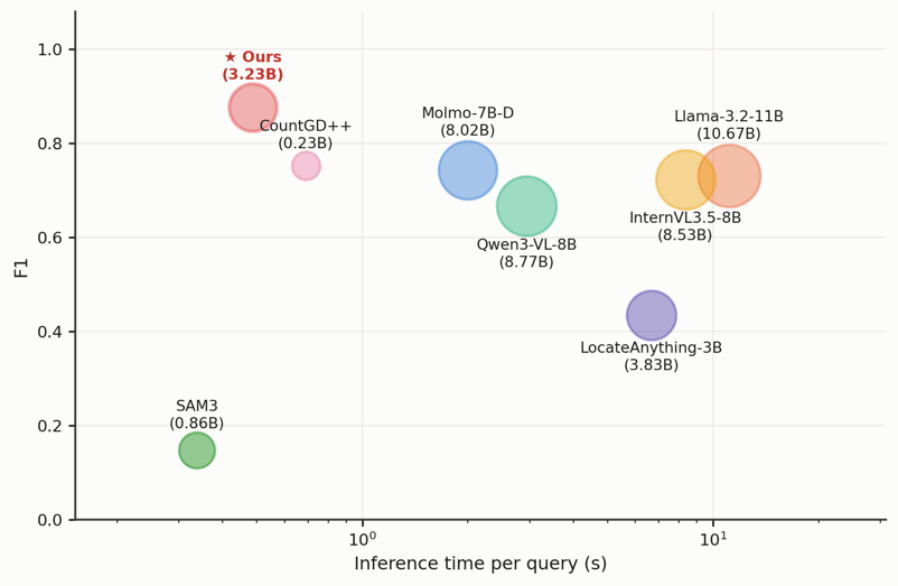


Figure 8: **Efficiency comparison.** F_1 versus inference time per query, with marker size proportional to model size.

C.4 Efficiency Comparison

Figure 8 compares anomaly-aware counting performance with inference cost. Pinpoint processes all region representations jointly within a shared multimodal forward pass and avoids autoregressive decoding of counts or coordinates. This provides a favorable balance between counting accuracy and inference efficiency.

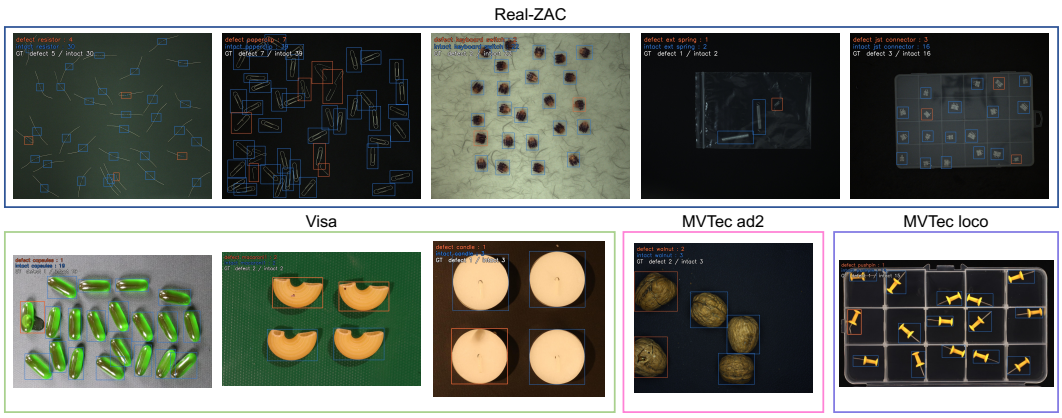


Figure 9: **Qualitative results on unseen component classes.** Blue boxes denote instances predicted as intact and red boxes denote defective predictions, with predicted and ground-truth counts shown for each image.

C.5 Qualitative Results

Figure 9 shows predictions on unseen Real-ZAC classes and external multi-instance benchmarks. The examples illustrate successful defect-state recognition across diverse component appearances and backgrounds, as well as challenging cases involving very small defects, various backgrounds, and mixed-class scenes.

D Ablation Studies

D.1 Components and Language-Model Backbone

Table 10: **Component and inference-rule ablation.** We evaluate the cumulative contribution of Pinpoint's representation components and compare validation-based thresholding with the calibration-free argmax rule.

Setting	MAE↓	RMSE↓	F_1↑
Representation components			
ObjEmbed fine-tuned (original recipe)	2.73	3.70	0.802
+ LoRA with focal loss	2.40	3.49	0.834
+ DINOv3 comparative descriptor	2.62	3.86	0.843
+ within-image feature centering	1.97	3.04	0.872
+ magnitude restoration	**1.93**	**3.02**	**0.876**
Inference rule (full representation)			
Validation-selected threshold	**1.93**	**3.02**	**0.876**
Argmax ($t = 0$, no calibration)	2.09	3.29	0.872

Component ablation. Table 10 separates the contribution of Pinpoint's representation components from its inference rule. LoRA fine-tuning with focal loss improves the ObjEmbed baseline, and the comparative descriptor further enhances defect-state discrimination. The largest gain comes from within-image centering, which suppresses shared class-specific appearance and substantially improves generalization to unseen component classes. Magnitude restoration provides an additional refinement by preserving anomaly strength. Using the threshold-free argmax rule ($t = 0$) slightly reduces counting performance compared with a validation-selected threshold, while avoiding state-threshold calibration on unseen classes.

Table 11: **ROI feature backbone**

Feature backbone	MAE↓	RMSE↓	F_1↑
WideResNet-50-2	2.26	3.43	0.861
DINOv2 ViT-L/14	2.28	3.52	0.858
DINOv3 ViT-L/16	**2.09**	**3.29**	**0.872**

Table 12: **LLM backbone**.

Backbone	Params	MAE↓	RMSE↓	F_1↑
Llama-3.2-11B-Vision	10.7B	2.90	3.98	0.687
InternVL3.5-8B	8.5B	2.31	**3.13**	0.720
Qwen3-VL-2B(ObjEmbed-2B)	**2.7B**	**2.09**	3.29	**0.872**

Feature backbone for the comparative descriptor. Table 11 replaces the vision encoder (Zagoruyko & Komodakis, 2016; Oquab et al., 2023; Siméoni et al., 2025) used to construct the comparative anomaly descriptor while recomputing mutual scoring, pooling, centering, and magnitude restoration from its features. Similar performance across different feature backbones indicates that the benefit mainly arises from comparative reasoning across instances rather than from a specific visual pretraining objective.

Language-model backbone. Table 12 replaces the language-model backbone while keeping the remaining components fixed. Performance does not directly follow model size, suggesting that region-text alignment learned before anomaly-aware training is more important than simply increasing language-model capacity.

Proposal source. Table 13 compares different proposal generators while keeping the defect-state recognition stage fixed. CountGD++ trained on Real-ZAC provides the most accurate proposals and leads to the best overall counting performance. In contrast, fine-tuning WeDetect on the Real-ZAC training classes substantially degrades performance on unseen classes compared with its pretrained version, showing that proposal adaptation can trade off domain fit against class-disjoint generalization.

Table 13: **Proposal source.**

Proposal source	MAE↓	RMSE↓	F_1↑
WeDetect (pretrained)	2.47	4.85	0.815
WeDetect (Real-ZAC-trained)	7.20	10.84	0.447
CountGD++ (Real-ZAC-trained)	**2.09**	**3.29**	**0.872**

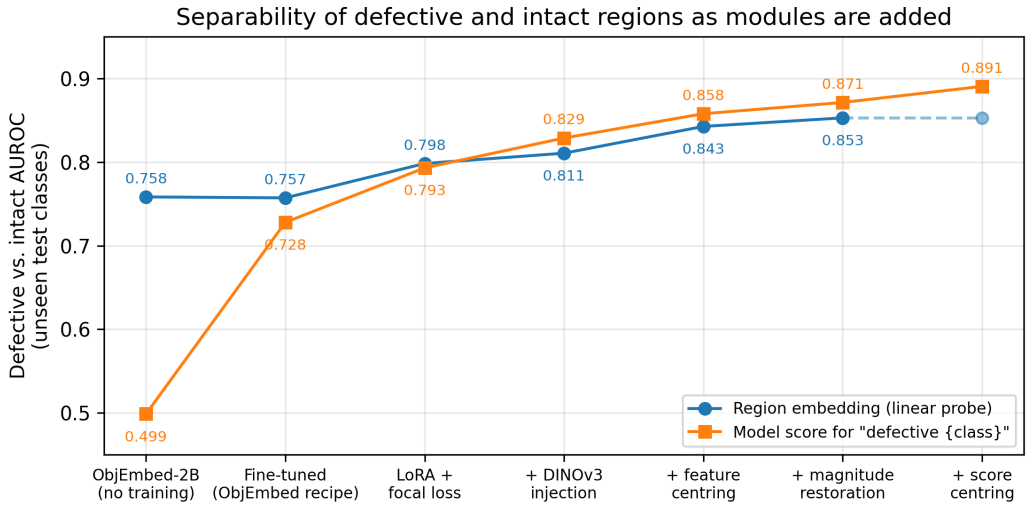


Figure 10: **Defect separability as model components are added.** For each ablation setting, we compare the AUROC of a linear probe trained on frozen region embeddings with the AUROC of the model's defect-state query score.

D.2 Representation Analysis

Figure 10 analyzes how each module changes defect separability in the learned representation. For each ablation setting, we freeze the resulting region embeddings and train a linear probe to classify intact and defective instances, reporting its AUROC as a measure of representation-level separability. We additionally compute the AUROC of the model's own score for the corresponding defect-state query to measure how well this information is aligned with the text query. Comparing the two curves distinguishes whether a module improves the underlying representation itself or mainly improves query alignment. The comparative descriptor and within-image centering improve both measures, indicating that they make defect-related differences more explicit in the region representation rather than only changing the final decision rule.

E Additional Anomaly Detection Results

E.1 Fine-tuning Anomaly Detection Models on Real-ZAC

Table 14 evaluates anomaly-detection models after fine-tuning on ROI crops from Real-ZAC training classes and testing on unseen component classes. Fine-tuning improves instance-level anomaly separation for most methods, suggesting that domain mismatch contributes to their weaker zero-shot performance. The varying behavior across models also indicates that the adaptation strategy itself affects transfer to unseen component categories.

E.2 Per-category Anomaly Detection Results

Table 15 provides the per-category results underlying the dataset-level averages reported in the main paper. Performance varies considerably across component geometry and defect appearance, while the

Table 14: Fine-tuning zero-shot anomaly detectors on Real-ZAC ROI crops. Each method adapts whatever parameters its authors expose; the threshold and epoch are selected on held-out validation categories and applied unchanged to the five test categories. AUROC is instance-level; the remaining columns follow our level-4 counting protocol with the defective and intact counts pooled. Since Real-ZAC provides object points rather than pixel masks, each method is trained on the image-level term of its objective alone.

Method	Trained parameters	#Params	Setting	$\text{AUROC}_{\text{inst}}$	MAE↓	RMSE↓	P	R	F1
AnomalyCLIP	text prompt	0.4M	zero-shot	.616	9.84	13.54	.427	.364	.393
			fine-tuned	.706 (+.090)	5.51	8.90	.705	.601	.649
AnomalyGPT	LoRA + proj + prompt	145M	zero-shot	.596	11.79	15.67	.303	.258	.279
			fine-tuned	.750 (+.154)	5.67	9.59	.690	.587	.634
AnomalyR1	LoRA (text tower)	7.4M	zero-shot	.610	12.20	16.01	.307	.261	.282
			fine-tuned	.733 (+.123)	4.67	7.20	.751	.639	.690
DAPO	prompt + adapter + **VPT**	3.4M	zero-shot	.719	4.35	7.28	.712	.607	.655
			fine-tuned	.689 (-.030)	5.45	9.09	.695	.592	.639
MuSc	none (training-free)	–	zero-shot, cross-image	.800	4.67	7.73	.719	.642	.678
			zero-shot, within-image	.857	3.83	6.35	.765	.684	.722
Ours	LoRA + DINOv3 injection	12.7M	fine-tuned	**.858**	**2.09**	**3.29**	**.858**	**.886**	**.872**

Table 15: **Per-category anomaly detection results on external multi-instance benchmarks.** Detailed results corresponding to Table 3 are reported for MVTec AD 2, MVTec LOCO, and VisA using identical WeDetect ROIs. We report image-level AUROC (img), instance-level AUROC (inst), and instance-level F1-max (F1). For MVTec LOCO, nut, screw, and washer instances sharing the same images are grouped as *screw bag*. Avg denotes the mean over all eight categories. * indicates that the released model weights were trained on the corresponding dataset. Bold marks the best result for each row and metric.

Dataset	Category	AnomalyCLIP			AdaCLIP			AnomalyR1			DAPO			MuSc			**Ours**		
		img	inst	F1	img	inst	F1	img	inst	F1	img	inst	F1	img	inst	F1	img	inst	F1
MVTec AD 2	wallplug	.364	**.793**	.233	.467	.714	.110	.427	.669	.215	.451	.777	.265	**.705**	.600	.110	.465	.705	**.290**
	walnut	.419	.738	.448	.391	.746	.366	.746	.806	.537	.682	.829	.502	.772	.537	.268	**.882**	**.861**	**.635**
MVTec LOCO	pushpin	.634	.379	.053	.578	.571	.182	.560*	.774	.143	.564	.548	.097	.503	.688	.229	**.808**	**.950**	**.627**
	screw bag	.489	.529	.124	.569	**.684**	.164	.518*	.423	.088	.517	.582	.090	.480	.563	.155	**.573**	.635	**.280**
VisA	candle	.880*	.908	.511	.946	.903	.649	.834*	.791	.186	.882	.766	.270	**.970**	**.973**	.713	.947	.933	**.758**
	capsules	.894*	.499	.025	.784	.497	.016	.690*	.464	.022	.847	.464	.017	.526	.451	.015	**.936**	**.986**	**.691**
	macaroni1	.893*	.931	.670	.808	.937	.571	.665*	.764	.268	.776	.890	.550	**.956**	**.984**	**.745**	.852	.940	.591
	macaroni2	.794*	.895	.462	.589	.846	.404	.588*	.661	.202	.659	.739	.224	**.842**	**.928**	**.584**	.549	.754	.201
Avg		.671	.709	.316	.641	.737	.308	.629	.669	.208	.672	.700	.252	.719	.716	.352	**.751**	**.845**	**.509**

overall trends remain consistent across multiple categories and datasets. These results show that the observed differences are not driven by a single category or benchmark.