

# MMR-AD: A Large-Scale Multimodal Dataset for Benchmarking General Anomaly Detection with Multimodal Large Language Models

## Supplementary Material

### A. More Discussions

**Chain-of-Thought Data.** One concern about our work is that our MMR-AD dataset relies on Chain-of-Thought (CoT) reasoning, which is primarily designed for language-centric reasoning tasks, while anomaly detection fundamentally depends on fine-grained visual perception. Excessive textual reasoning may distract the model’s attention from visual cues. We think that CoT and fine-grained visual perception are not conflicting. Our CoT reasoning is to teach the model to carefully analyze the main areas in two images and compare them to identify discrepancies. The CoT texts are closely around image contents (for fine-grained image-text alignment) rather than just language-centric reasoning (see Fig.3). In Tab.5, we have also verified that the reasoning data is critical. Without reasoning data, the model performs significantly worse (88.8 to 81.3 and 74.9 to 66.1). Our Anomaly-R1-7B is finetuned from Qwen2.5-VL-7B and can significantly outperform Qwen2.5-VL-7B (see Tab.4), demonstrating the effectiveness of our dataset.

**Domain Shifts in MMR-AD** To ensure data scale and diversity, our MMR-AD dataset is extensively collected and sampled from 14 publicly available AD datasets. Using public datasets is because they are collected from the real world and already have remarkable domain shifts (*e.g.*, MANTA is from five domains: mechanics, electronics, medicine, agriculture, and groceries, 3CAD is from 3C electronics, GoodsAD is from groceries). The sensors and lighting used during collection are also different. Among these datasets, the product categories are completely different. Thus, we think that cross-dataset experiments (Tab.4) include strong domain shift challenges. The expected scope of MMR-AD is still industrial AD and is not suitable for direct application in medical or video anomaly detection.

**Pixel-level Localization.** Unlike conventional anomaly detection methods that generate anomaly score maps, we employ bounding boxes (bbox) to predict the location of anomalies. Predicting bbox is because that current MLLMs cannot output images, hard to generate segmentation maps directly. However, in industrial applications, bbox is also a commonly used way (*e.g.*, many YOLO-based defect detectors output bboxes). A feasible extension method is to combine with SAM, where MLLM generates several special <SEG> tokens after bbox tokens, and then utilizes these tokens as conditional control to SAM, which predicts segmentation maps based on the input image and conditional tokens.

### B. Spatially-aligned Normal Reference Sample

Normal samples are usually diverse, especially the foreground objects may not be highly aligned in different normal samples. This may lead to a reduction in the normal reference effect. To this end, we can select a spatially-aligned nearest sample as the normal reference sample for each input sample. In this paper, we follow [20] to employ a spatial alignment retrieval approach to retrieve spatially aligned samples from test normal images for the input image  $I$ . Given total  $N$  normal images, we extract first layer features with a pre-trained light-weight network (*e.g.*, ResNet-18), the corresponding features are denoted as  $\{F_i \in \mathbb{R}^{H_p \times W_p \times C_p} | i = 1, 2, \dots, N\}$ . Then we collect all raw patch features of different normal images together into the feature set  $\mathcal{P} = \{\text{Flatten}(F_i)^k \in \mathbb{R}^{C_p} | k = 1, 2, \dots, N H_p W_p\}$ . The K-means clustering algorithm is performed on  $\mathcal{P}$  to obtain  $N_c$  clustering centers  $\mathcal{C} = \{c_k \in \mathbb{R}^{C_p} | k = 1, 2, \dots, N_c\}$ . The feature map  $F_i$  and clustering centers  $\mathcal{C}$  are further utilized to calculate the block-wise statistics for each normal image  $I_i$ . Specifically, the feature map  $F_i$  is evenly divided into  $S \times S$  grids as:

$$\hat{F}_i^{u,v} \in \mathbb{R}^{\frac{H_p}{S} \times \frac{W_p}{S} \times C_p}, u, v = 1, 2, \dots, S \quad (3)$$

For each grid, we can calculate its block-wise statistics in the “Bag of Words” style as:

$$\begin{aligned} \hat{F}_i^{u,v} &\xrightarrow{\text{Flatten}} \{f_k \in \hat{F}_i^{u,v} | k = 1, 2, \dots, H_p W_p / S^2\} \\ &\xrightarrow{\text{BoW}} h_{u,v} \in \mathbb{R}^{N_c} \end{aligned} \quad (4)$$

where  $h_{u,v}$  denotes a BoW histogram of  $\hat{F}_i^{u,v}$  with respect to the codebook  $\mathcal{C}$ . Specifically, we first construct  $N_c$  bins and then calculate the distances from  $f_k$  to all centers in  $\mathcal{C}$ . The value in the bin that is related to the minimum distance will be added by one. The histogram is then normalized so that  $\|h_{u,v}\|_1 = 1, \forall u, v$ . We can obtain all statistical histograms for an image  $I_i$  and denote them as  $H^i \in \mathbb{R}^{S^2 \times N_c}$ . Then, for the input image  $I$  and  $i$ -th normal image, we can calculate the KL divergence between their histograms to measure the spatial alignment degree between them. Specifically, we calculate the block-wise KL divergence  $D_{u,v}^i = KL(H_{u,v}^i, H_{u,v})$  for each block  $(u, v)$ , where  $H_{u,v}$  means the histogram in  $(u, v)$  position of the input image. By sorting the block divergences in increasing order, we can get the following sorted block divergences

$$\{D_1^i \leq D_2^i \leq D_3^i \dots \leq D_{S^2}^i\} \quad (5)$$

The spatial alignment degree between  $I$  and  $I_i$  is estimated as:

$$D_{align}^i \triangleq \frac{1}{S^2 - \tau} \sum_{j=1}^{S^2 - \tau} D_j^i, i = 1, 2, \dots, N \quad (6)$$

where  $\tau$  is a small number for ignoring large block divergences.

Based on the spatial alignment degree, we can retrieve the top-K neighbor normal images (we denote the set of indices as  $\iota$ ) as few-shot normal reference images  $\mathcal{N}_{ref} = \{I_{\iota_1}, I_{\iota_2}, \dots, I_{\iota_K}\}$ . Then, the nearest normal image  $I_{\iota_1}$  is used as the normal reference image in our method (see Sec.3.2 and Tab.2). In implementation, the hyperparameters  $S$ ,  $N_c$ ,  $\tau$ , and  $K$  are set as 5, 12, 5, and 10, respectively.

### C. Text Verification Process

After obtaining the MLLM-generated texts, we should further verify whether the generated texts are correct and filter out the incorrect texts. To this end, we propose an automatic verification approach. Specifically, we extract the predicted anomaly regions from “<answer></answer>” and verify whether they are consistent with the true anomaly regions. We have checked all the generated texts and found that the texts in all answers followed our data generation instruction very well. When the answer contains “No”, there will be no anomaly location outputs; only when the answer contains “Yes”, there will be corresponding anomaly location outputs. All the generated anomaly regions are in {‘bbox\_2d’: [x1, y1, x2, y2], ‘label’: ‘<anomaly type>’} format.

For normal samples, their generated answers should include “No”. Otherwise, the corresponding texts are considered incorrect. For abnormal samples, the answers should contain “Yes”, and the predicted anomaly regions should be consistent with the true anomaly regions. For each predicted anomaly instance (one “{...}”, Python dict), we calculate the IoU between its bounding box and the bounding box of one true anomaly region, and we calculate the semantic similarity (SS) between their labels (we utilize the [SentenceTransformers](#) library to measure text semantic similarity). Only when the IoU > 0.9 and the SS > 0.6, we consider the predicted anomaly instance can match one true anomaly region. For one abnormal sample, the generated answer is considered correct only if the number of predicted anomaly instances is equal to the number of true anomaly regions, and each predicted anomaly instance matches a true anomaly region. Otherwise, the corresponding generated text of this sample is considered incorrect. Moreover, we also hired multiple experienced annotators to further check whether the texts in <think> match the final answer and remove the inconsistent texts.

### D. Implementation Details

For the MMR-AD dataset preparation, we call the Qwen2.5-VL-72B API to generate reasoning-based text data. For the cold-start initialization of Anomaly-R1, we adopt Qwen2.5-VL-7B as the base model and post-train it via supervised fine-tuning for 3 epochs. We employ the LoRA technique [22] to fine-tune the base model. The low rank  $r$  and  $\alpha$  hyperparameter in LoRA are both set to 16. During the SFT phase, the learning rate is set to 2e-5, and the batch size is set to 4. The learning rate warmup ratio and the weight decay hyperparameter are both set to 0.1.

During the RL phase, for rollout, the prompt batch size is 32, and we sample 8 responses for each prompt. When generating responses, the key sampling hyperparameters, such as temperature, top\_p and top\_k, are set to 1.0, 1.0, and -1, respectively. The mini-batch size for model update is set to 8, which means we employ off-policy optimization and the model will be updated 4 times for each rollout (*i.e.*, 32 // 8). We limit the maximum number of input prompt tokens (including image tokens) to 2048, and the maximum number of tokens for generation is set to 512 tokens. The  $\epsilon$  and  $\beta$  in the GRPO algorithm are set to 0.2 and 0.001, respectively. The learning rate is set to 2e-6. The learning rate warmup ratio and the weight decay hyperparameter are set as 0.1 and 0.01, respectively. We utilize the VeRL framework [43] for RL training.

During inference, we use the min-p sampling [33] strategy with the temperature and mip-p hyperparameters set to 1.5 and 0.1. The maximum number of tokens for generation is set to 512.

### E. Qualitative Results

In Fig.4, 5, and 6, we provide the qualitative comparison of our Anomaly-R1-7B with GPT-4o and Qwen2.5-VL-72B. For commercial models, GPT-5 and o4-mini cannot be qualitatively compared as the APIs don’t provide users with <think> texts. The <think> texts generated by Gemini-2.5-pro are too long to effectively present in a limited space, so we also don’t include it in the qualitative comparison. For open-source models, as our model is based on Qwen2.5-VL-7B, it’s very suitable for comparison with Qwen2.5-VL-72B. Although both GPT-4o and Qwen2.5-VL-72B can successfully identify the three examples as abnormal, their decisions are based on hallucination (see Fig.4, 5), and their anomaly localization accuracy is also poor (Fig.6). By comparison, our Anomaly-R1-7B has correct reasoning and shows more accurate anomaly localization results.

In Fig.7 and 8, we further show some failure cases generated by our Anomaly-R1-7B.

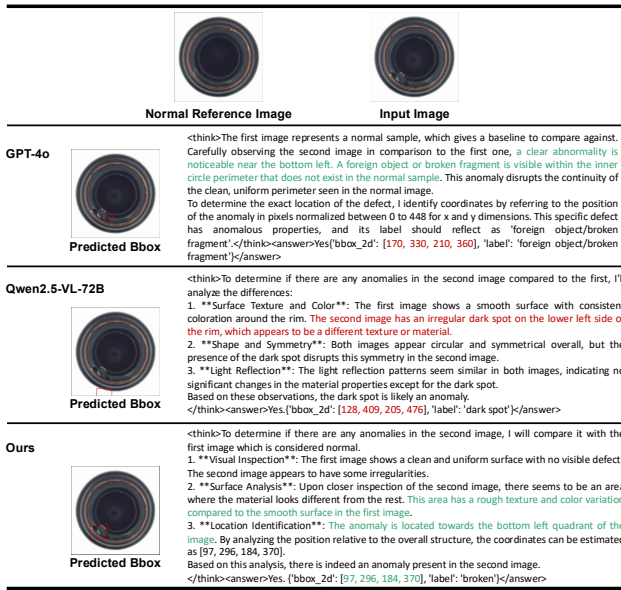


Figure 4. Green marks correct reasoning and correct bbox coordinates, red marks wrong reasoning and imprecise bbox coordinates. Qwen2.5-VL-72B shows severe hallucination, thinking that there is an abnormal dark spot, and locating to the white area at the bottom of the image.

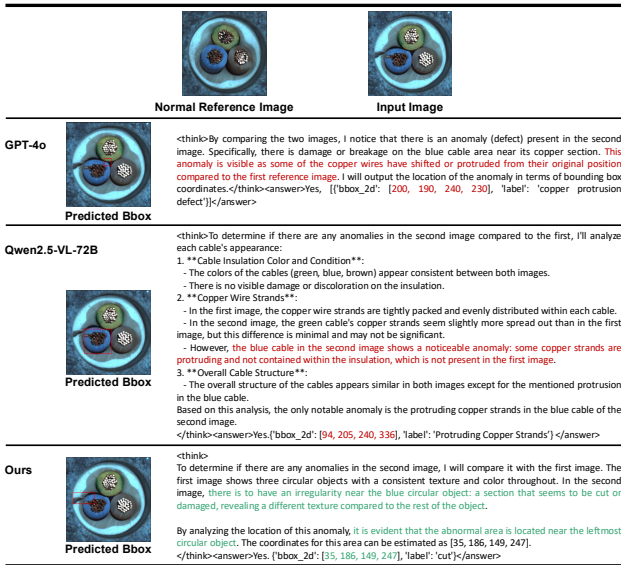


Figure 5. Green marks correct reasoning and correct bbox coordinates, red marks wrong reasoning and imprecise bbox coordinates. Both GPT-4o and Qwen2.5-VL-72B show hallucination, thinking that there is the copper protrusion defect without observing the severe cut defect.

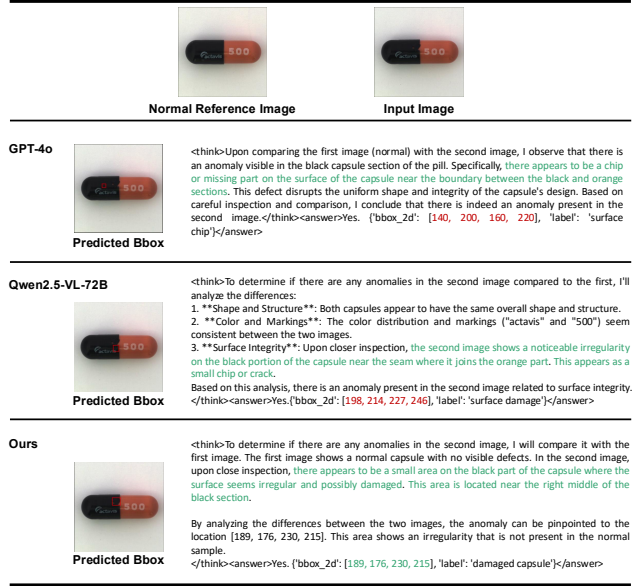


Figure 6. Green marks correct reasoning and correct bbox coordinates, red marks wrong reasoning and imprecise bbox coordinates. Although both GPT-4o and Qwen2.5-VL-72B generate correct reasoning, the anomaly localization results are still not precise enough.

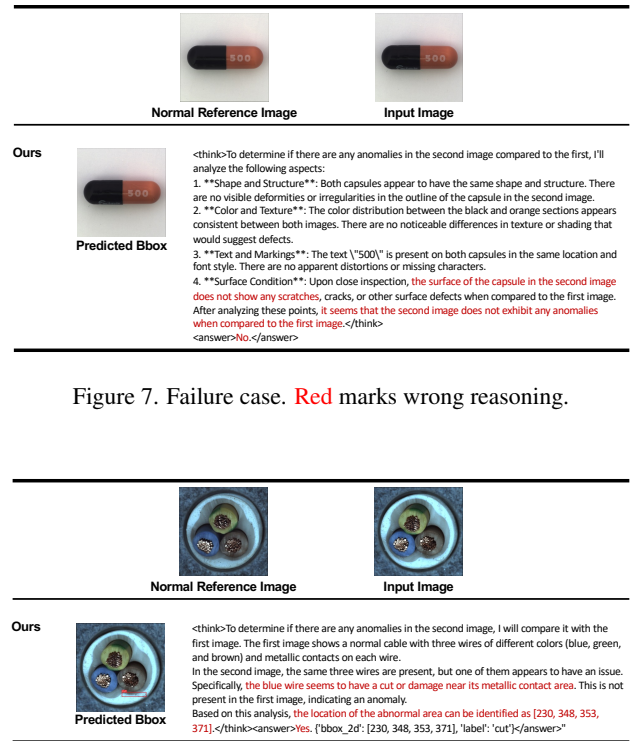


Figure 7. Failure case. Red marks wrong reasoning.

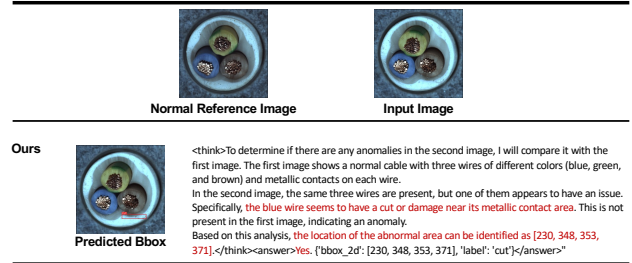


Figure 8. Failure case. Red marks wrong reasoning and incorrect bbox coordinates.