

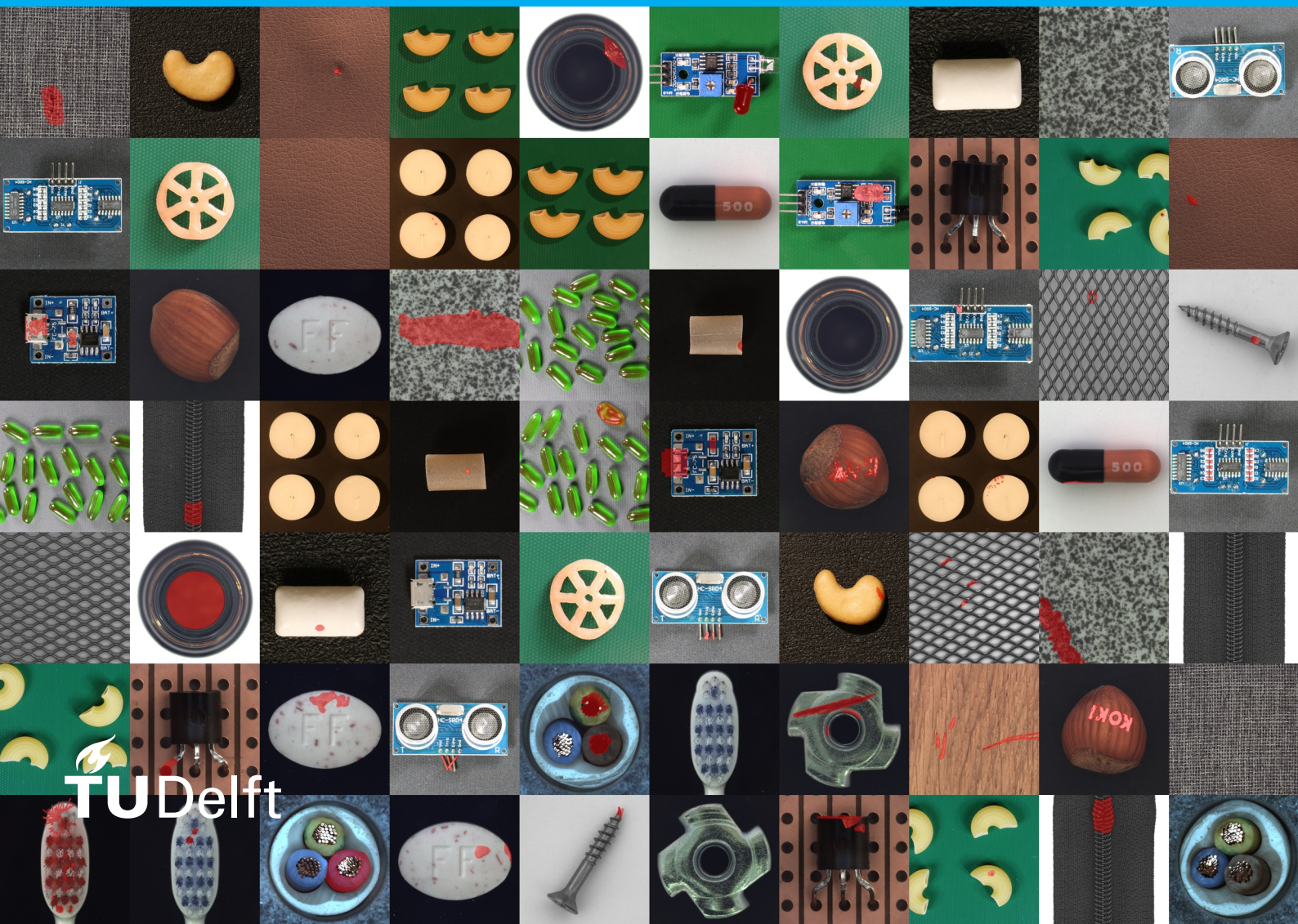
Reasoning-Enhanced Vision-Language Models

for Explainable Industrial Anomaly Detection

Adnane Acudad

MSc Thesis
Electrical Engineering
Signal Processing Systems

2026



Reasoning-Enhanced Vision-Language Models

for Explainable Industrial
Anomaly Detection

by

Adnane Acudad

to obtain the degree of *Master of Science in Electrical Engineering*
at the Delft University of Technology,
to be defended publicly on Thursday, 17 September 2026 at 14:30.

Student number:	5565499		
Track:	Signals and Systems		
Research group:	Signal Processing Systems, Department of Microelectronics, Faculty EEMCS		
Project duration:	September, 2025 – September, 2026		
Thesis committee:	Prof. dr. ir. Justin Dauwels,	TU Delft, supervisor (chair)	
	Prof. dr. Willem van Driel,	TU Delft	
	Amir Ghorbani Ghezeljehmeidan, MSc,	TU Delft, daily supervisor	
Cover:	Montage of MVTec AD and VisA samples with ground-truth defect masks in red		

An electronic version of this thesis is available at <https://repository.tudelft.nl/>.

Reasoning-Enhanced Vision-Language Models for Explainable Industrial Anomaly Detection

© 2026 Adnane Acudad

This thesis is submitted in partial fulfilment of the requirements for the degree of *Master of Science in Electrical Engineering* at Delft University of Technology. The research code, the generated reasoning traces, the training scripts and the model checkpoints described in this thesis are openly released. The links are listed in §8.5.

An electronic version of this thesis is available at <https://repository.tudelft.nl>.

Abstract

Industrial quality control needs inspection systems that are accurate, explainable, and adaptable to new product categories with few labelled data. Traditional deep-learning detectors, such as memory-bank, normalising-flow, and reconstruction-based methods, reach near-perfect AUROC on the established MVTecAD benchmark, yet they emit only a binary defect verdict and a heatmap. They give no human-readable account of why a sample was flagged or what type of defect it carries, and that lack of interpretability is a primary obstacle to deploying automated inspection in safety- and audit-critical manufacturing.

This thesis asks whether an open vision-language model (VLM), fine-tuned on structured reasoning traces and refined with reinforcement learning, can deliver explainable anomaly detection that is competitive with the state of the art, on products it was not trained on and without per-product retraining or reference images. We build an end-to-end framework that fine-tunes the open-weight Qwen2.5-VL family on structured chain-of-thought reasoning traces. Each trace follows a six-phase template (Framing, Scan, Focus, Evaluate, Alternatives, Decide) and emits an XML output with a free-text rationale, a defect-type tag, a spatial location, and a binary verdict. Using Gemini 2.5-Flash we generate 14,472 such traces from the Real-IAD multi-view dataset, released here as the AnomalyThink dataset, and partition them into product- and class-stratified splits of 6,000 for supervised fine-tuning, 4,236 for GRPO, and 4,236 held out.

We run a controlled four-factor comparison of fine-tuning configurations across model size (3B versus 7B), vision-encoder freezing, supervised-data scale (6K versus the full 14.5K union), and training epoch (up to four), with the unfrozen encoder evaluated only at the larger scale. This is followed by Group Relative Policy Optimisation (GRPO) with an unweighted two-function reward scoring four bounded signals: format, verdict, semantic type, and location. Evaluation uses the DS-MVTec (1,670 samples) and VisA (2,141 samples) subsets of MMAD, together with a held-out Real-IAD split.

Our main result comes from a three-stage pipeline in which each stage feeds the next: supervised fine-tuning, GRPO, and a second supervised stage trained on data the GRPO model produced. Supervised fine-tuning on the 6,000-trace dataset reaches 80.16 % balanced accuracy on DS-MVTec and 64.78 % on VisA. GRPO with our reward functions lifts this to 82.73 % and 70.39 %. We then roll that GRPO model out at $k=8$ over all 10,236 training images, which is the 6,000-image SFT split plus the 4,236-image GRPO split. We keep the traces it gets right and have a teacher correct or rewrite the rest. The new corpus is rebuilt on the 6,000 SFT images. Training the base Qwen2.5-VL-7B on it reaches 82.80 % and 72.07 %, the best result on that backbone, and beats the released IAD-R1 model on both subsets (81.92 % and 71.34 %) under the same evaluation harness. We repeat the same recipe end to end on a second backbone, LLaVA-OneVision-7B-SI. There the curated corpus reaches 87.32 % and 72.65 % at the selected epoch against 87.66 % and 72.38 % for SFT plus GRPO, and 74.29 % on VisA by epoch 4. On both backbones the supervised stage matches the reinforcement-learned one and exceeds it on VisA. The finding is therefore not specific to one model family. The last stage is a supervised one. What GRPO gave us was not the final model but the data we used to train it.

GRPO on its own helps when the supervised baseline is weak. Applied to the earlier supervised baseline it raises balanced accuracy to 82.73 % on DS-MVTec, 70.39 % on VisA, and 80.87 % on the held-out Real-IAD split, exceeding IAD-R1 on DS-MVTec. Applied on top of the stronger curated dataset, GRPO did not improve binary accuracy in our experiments. Reward kept rising while balanced accuracy stayed 1.6 to 4.0 points below the supervised initialisation. This held for all three advantage estimators we tried, within a single 120-step probe on one initialisation. Two further results follow. Within our data, the 6,000-trace split beats the full 14.5K union by 8.5 balanced-accuracy points on DS-MVTec at matched epochs, a data-count effect rather than a trace-length one. And freezing the vision encoder consistently beats unfreezing it, most clearly on the harder VisA benchmark.

The framework is reproducible on commodity hardware, two RTX A6000 GPUs with ZeRO-3 CPU offloading, and yields a single 7B model that detects, localises, and explains industrial defects in natural language, with no per-product retraining and no pixel-level masks at inference.

Keywords: industrial anomaly detection, vision-language models, chain-of-thought, supervised fine-tuning, GRPO, Real-IAD, MMAD, Qwen2.5-VL, explainable AI, dataset, benchmark.

Statement on the use of AI tools. I designed the research questions and the experiments, built and ran the pipeline, chose the models, data and metrics, analysed the results and wrote this thesis. Claude (Anthropic) was used as a coding assistant for evaluation scripts, figure generators and data-processing code, and as a writing aid to paraphrase sentences and check spelling and consistency. Gemini 2.5-Flash and Gemini 3-Flash are part of the method, as the trace teacher, the corrector in the Keep-Correct-Revise loop and the explanation judge, and their roles are described in Chapters 3, 5 and 6. Every number reported here was produced by the evaluation harness of Appendix J and can be recomputed from the result files in the public repository. Text produced with assistance was checked against the sources and edited by me, and no AI tool was used to generate results or references.

Acknowledgements

This thesis is the product of nine months of work at the Department of Microelectronics of the Faculty of Electrical Engineering, Mathematics and Computer Science at TU Delft.

First and foremost, I thank my supervisor, Prof. dr. ir. Justin Dauwels, and my daily supervisor, Amir Ghorbani Ghezeljehmeidan, for the trust they placed in me to pursue this topic that bridges machine learning, computer vision, and industrial inspection. I also thank them for guidance that was always sharp on the science while leaving me considerable room to explore.

The computational experiments described in Chapters 4 and 5, particularly the fine-tuning and post-training of Qwen2.5-VL-7B, required substantially more GPU-hours than a single workstation could deliver. I am grateful to the Microelectronics department for granting me reliable access to the RTX A6000 cluster on which the experimental runs were performed.

I owe a special debt to the open-source community whose code and models this work builds upon: the Alibaba Qwen team for Qwen2.5-VL [4], the DeepSeek group for the GRPO algorithm [66], the Hugging Face transformers and trl maintainers, and the authors of LLaMA-Factory and Unsloth. Reasoning-trace generation relied mainly on Google's Gemini 2.5-Flash. I also thank the authors of IAD-R1 [39] for releasing their model weights, which enabled a fair head-to-head comparison on the benchmarks.

Finally, and most importantly, I thank my mother and father, my siblings, and the rest of my family for their unwavering support throughout my studies and through every late night, failed run, and moment of doubt this thesis demanded.

*Adnane Acudad
Delft, September 2026*

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement and Scope	2
1.3 Research Questions	2
1.4 Contributions	3
1.5 Thesis Outline	5
2 Background and Related Work	7
2.1 Classical Industrial Anomaly Detection	7
2.1.1 Memory-bank and feature-embedding methods	7
2.1.2 Normalising-flow methods	7
2.1.3 Reconstruction and distillation methods	7
2.1.4 Self-supervised and synthetic-defect methods	8
2.1.5 Limitations of classical IAD methods	8
2.2 Vision-Language Models	8
2.2.1 From CLIP to instruction-tuned VLMs	8
2.2.2 The Qwen-VL family	8
2.3 Chain-of-Thought Reasoning	9
2.4 Reinforcement Learning for Language Models	9
2.4.1 From PPO to RLHF	9
2.4.2 Group Relative Policy Optimisation (GRPO)	10
2.4.3 Self-training: rejection sampling, STaR, and self-distillation	10
2.5 VLM-Based Industrial Anomaly Detection	11
2.5.1 Zero-shot CLIP-based methods	11
2.5.2 Supervised fine-tuning methods	11
2.5.3 SFT+RL methods	11
2.6 Benchmark Datasets	12
3 Reasoning Trace Generation	13
3.1 Source Dataset	13
3.2 Output Format: Structured Six-Phase Trace	14
3.3 Generation Pipeline	15
3.4 Generation-Time Quality Control	16
3.5 Dataset Assembly	17
3.6 Dataset Statistics	18
3.7 Positioning vs. Existing Reasoning-Trace Datasets	19
4 Supervised Fine-Tuning	21
4.1 Training Objective	21
4.2 Model and Vision Encoder Treatment	21
4.3 Data Format and Prompt	22
4.4 Training Infrastructure	22
4.5 Hyperparameters	23
4.6 Ablation Design	24
4.7 Evaluation Protocol	24

5	GRPO Reinforcement Learning	27
5.1	GRPO as Implemented	27
5.2	Reward Function	28
5.2.1	Consistency (format) reward R_{cons}	28
5.2.2	Accuracy reward R_{acc}	28
5.2.3	Total reward and run variants	29
5.3	Data Preparation	29
5.4	Training Loop and Hyperparameters	31
5.5	Implementation Details and Production-Run Statistics	31
5.6	Iterative SFT $\leftrightarrow$ RL: A Self-Distillation Cycle	32
5.7	Gaussian-Advantage GRPO (G ² RPO)	33
5.8	Teacher-Intervention Arms: Constructing Self-Distilled SFT Datasets.	34
6	Results and Analysis	37
6.1	Evaluation Protocol and Metrics	37
6.2	Zero-Shot Baseline Performance	38
6.3	SFT Ablation: Four-Factor Grid	40
6.4	GRPO on the SFT Baseline	42
6.5	Iterative SFT $\leftrightarrow$ RL and Alternative RL Variants.	43
6.6	Teacher Ablation: External vs. Self-Distillation	44
6.7	Post-GRPO SFT Refinement on Cleaned Data.	46
6.8	GRPO on the KCR Baseline	47
6.9	Replication on a Second Backbone	49
6.10	Model Behaviour: Operating Point, Per Product and Cases	51
6.10.1	The Operating Point	51
6.10.2	Per Product	52
6.10.3	Case by Case	55
6.10.4	Localisation	56
6.11	Explanation Quality: a Reasoning-Faithfulness Comparison	57
6.12	Comparison to the State of the Art	59
6.13	Training and Reward Dynamics	62
6.14	Chapter Summary	63
7	Discussion	65
7.1	Why Frozen > Unfrozen Vision Encoder	65
7.2	Why the Smaller Split Wins	66
7.3	Why VisA Remains Harder than DS-MVTec	66
7.4	Checkpoint Selection and Reward Over-Optimisation	67
7.5	Revisiting the Value of GRPO	67
7.6	Failure-Mode Analysis	69
7.7	Limitations	69
8	Conclusion and Future Work	73
8.1	Summary of Contributions	73
8.2	Answers to the Research Questions	74
8.3	Practical Deployment and Broader Impact	75
8.4	Future Work.	75
8.5	Closing Remarks	76
A	Full Hyperparameter Tables	87
A.1	Trace Generation Hyperparameters.	87
A.2	Real-IAD Products and Split Coverage.	87
A.3	SFT Hyperparameters (per cell)	89
A.4	GRPO Hyperparameters (SFT+GRPO run)	89
A.5	DeepSpeed Configuration.	90

B	Prompts (verbatim)	91
B.1	Gemini Generation Prompt	91
B.2	SFT Training and Evaluation Prompt (train-prompt mode)	97
B.3	GRPO Training and Evaluation Prompt	97
C	Per-Product Tables	99
C.1	Per-Product Balanced Accuracy on DS-MVTec	99
C.2	Per-Product Confusion-Matrix Counts	100
D	Per-Product Tables on VisA	101
D.1	Per-Product Balanced Accuracy on VisA	101
D.2	Post-GRPO Refinement against SFT+GRPO on VisA	102
D.3	Per-Product Confusion-Matrix Counts	102
E	Reward Function Reference	103
E.1	Consistency (format) reward	103
E.2	Accuracy reward	103
E.3	Total reward and run variants	104
E.4	Pseudocode	104
E.5	Type-Reward Similarities on the Corpus Vocabulary	107
F	Qualitative Trace Gallery	109
F.1	Correct Detections	109
F.2	Failure Cases	111
F.3	One Image through Base, SFT and SFT+GRPO	114
F.4	Post-GRPO Refinement against SFT+GRPO: Overlays.	115
F.5	SFT Against KCR on the Same Image	116
G	Teacher vs. Student Traces	127
G.1	DS-MVTec: Student Corrects the Teacher	127
G.2	DS-MVTec: Teacher and Student Agree	128
G.3	VisA: Where the Teacher's Bias Cuts Both Ways	129
H	Training Curves	131
H.1	SFT Balanced Accuracy vs. Epoch	131
H.2	GRPO Training Dynamics	132
H.3	Per-Item Loss of the 6K Pool under the GRPO Checkpoint	133
I	Datasheet for the AnomalyThink Dataset	135
I.1	Motivation	135
I.2	Composition	135
I.3	Collection Process	137
I.4	Preprocessing / Cleaning / Labeling	137
I.5	Uses.	138
I.6	Distribution.	138
I.7	Maintenance	138
J	Evaluation Harness and Prompt-Mode Robustness	141
J.1	Prompt Modes	141
J.2	Decoding and Image Settings	142
J.3	Fixed Evaluation Sets and Headline Metric	142
J.4	Prompt-Mode Robustness	143
J.5	Type-Label Transfer to MMAD	143
K	Explainability Scoring: Executed Protocol and Proposed Extension	145
K.1	The Executed Protocol	145
K.2	Proposed Extension.	146
K.3	Scope and Design Principles	146
K.4	Decomposed Multi-Axis Rubric	146
K.5	Separating Faithfulness from Plausibility	147
K.6	Aggregation	147

K.7	Judge Sampling	148
K.8	Example Judge Prompt	148
K.9	Validation Against a Human Golden Set.	149
L	Held-Out Real-IAD Evaluation	151
M	The Corrected LLaVA-OneVision Corpus	153
M.1	What Differed	153
M.2	The Corrected Build.	153
N	GRPO with a Frozen Vision Encoder	155
N.1	Setup	155
N.2	Result	155

Introduction

1.1. Motivation

Automated visual inspection is the backbone of modern manufacturing quality assurance. Cameras at the end of an assembly line acquire tens of thousands of high-resolution images per shift. Downstream systems must typically decide, within a fraction of a second, whether the item is up to standard or defective. The two kinds of mistakes are asymmetric. Missing a defect (a false negative) ships a faulty product to the customer, and in safety-critical industries this can trigger a recall that costs far more than the inspection equipment itself. Wrongly rejecting a good part (a false positive) sends it for unnecessary manual rechecking, which slows down the line. For about thirty years, the standard solution was a pipeline of classical computer-vision algorithms that used techniques such as thresholding, template matching and blob analysis, all hand-tuned for each product. In the last five years that pipeline has been replaced by deep neural networks, primarily unsupervised *anomaly detection*: the models are trained only on normal images, and they emit a scalar anomaly score and optionally a localisation heatmap.

On the canonical MVTec AD benchmark [6], PatchCore [58], EfficientAD [5] and SimpleNet [42] all exceed 99% image-level AUROC at millisecond-level latency. MVTec AD is essentially saturated on pure detection accuracy. Yet in real factories two operational frictions appear repeatedly:

1. **Opacity.** The output is a scalar score and a heatmap. It does not say *what* the defect is, relate the hot region to the object's anatomy ("left side of the bottle neck"), or articulate the comparison to a normal sample. Engineers cannot trust, audit, or write up a result they cannot read.
2. **Per-product training.** Almost every state-of-the-art method needs a per-category training run and a per-category model or memory bank. A line with 30 distinct products (SKUs) maintains 30 detectors and retrains on every tooling, lighting or packaging change.

The recent wave of *vision-language models* (VLMs), such as BLIP-2 [36], LLaVA [41], the Qwen-VL line [3, 4, 70] and InternVL3 [92], promises to address both frictions simultaneously: they ingest an image and a textual prompt, emit free-form natural language, and have been pre-trained on enough image-text pairs to perform respectably on novel categories in zero shot. A VLM can in principle describe *what*, *where* and *why* an anomaly is present in one generated answer from one model, with no separate segmentation or classification stage, and can be steered with a textual instruction rather than a category-specific weight checkpoint.

This explanatory power is not free. A 7B VLM writes about 170 tokens per answer. With plain HuggingFace generation that is 5 to 6 seconds per image at about 30 tokens per second on one A6000. A batched vLLM server on the same GPU brings the throughput to about 0.2 seconds per image (Appendix J), but the latency of one answer stays in the seconds, against the millisecond-level latency of PatchCore or EfficientAD. It therefore cannot sit in-line at the sensor on a high-throughput line. Its natural place is either as an explainable second stage that reviews only the items a fast detector flags, or as the primary inspector on lower-throughput, audit- and safety-critical lines, where a readable, traceable verdict is worth more than raw frames-per-second.

The challenge is that out-of-the-box VLMs are far from accurate enough for industrial inspection. On the

comprehensive multimodal IAD benchmark MMAD [30], GPT-4o reaches only 74.9% average accuracy, far below the reliability required for production deployment. Several recent works have begun to close this gap by fine-tuning VLMs on industrial data, either with pure supervised fine-tuning (SFT) (AnomalyGPT [19], Anomaly-OV [75]) or with reinforcement-learning post-training (AnomalyR1 [9], LR-IAD [84], EMIT [20], IAD-R1 [39]). The latter line, in particular, applies the recently popularised Group Relative Policy Optimisation (GRPO) algorithm [11, 66] on top of an SFT cold-start to align the model on multi-component rewards. The most directly comparable prior work is IAD-R1 [39], which uses a Qwen2.5-VL-7B backbone with SFT+GRPO and is the most recent strong SFT+GRPO open-weight baseline on the MMAD subsets we evaluate. Structured, tagged outputs trained with GRPO are common to this line. AnomalyR1 [9] and LAD-Reasoner [37] emit a reasoning span and a verdict tag, OmniAD [87] adds a patch-coordinate segment, and IAD-R1 adds a defect-type and a location tag, which is the format this thesis adopts.

Despite this rapid progress, three gaps remain. First, the strongest reasoning-based detectors each bundle a different backbone, dataset, reward and training budget, so it is unclear which ingredient drives the reported gains. Second, their training data is undisclosed and they are released only as checkpoints, and sometimes not at all, leaving the community unable to reproduce or extend them. Third, they are judged almost entirely on detection accuracy, with the quality of the explanation itself left unmeasured. This thesis addresses all three: a controlled, like-for-like study of the SFT→GRPO recipe on a single commodity two-GPU workstation, an openly released reasoning-trace dataset, **AnomalyThink**, built on Real-IAD [69], and an explicit, reference-guided evaluation of explanation grounding alongside detection accuracy.

1.2. Problem Statement and Scope

We address the problem of *explainable industrial anomaly detection*: given an image of a manufactured part, output (i) a binary verdict (defect/no defect), (ii) the type of defect if any, (iii) its spatial location, and (iv) a free-text rationale that grounds the verdict in visible evidence. Detection performance is measured by *balanced accuracy* on the MMAD subsets DS-MVTec and VisA, plus a held-out Real-IAD test split. We also care about the explanation itself, not just the verdict. A good explanation has to be faithful to the image. The defect type, the location, and the reasoning all have to match what is actually visible, not just sound convincing. We check this on its own, apart from whether the verdict is right. We use the per-component signals from the structured output, and a reference-guided rubric that scores how faithful the reasoning is (§6.11).

The scope is intentionally restricted in two ways. First, every controlled study fixes the backbone family to Qwen2.5-VL [4]. When this thesis started, Qwen3-VL did not yet exist and Qwen2.5-VL was the strongest open-weight family with both a 3B and a 7B model that fit on a single A6000, it is the backbone of most GRPO-based IAD work (AnomalyR1, LAD-Reasoner), and the comparable IAD-R1 baseline releases a checkpoint on it, which we re-run under our own harness. The two exceptions are in §6.9, which reruns the final recipe once on a second backbone, LLaVA-OneVision-7B-SI, and tests the corpus once on Qwen3-VL-8B, released while the work was under way. Second, we restrict the training data to Real-IAD [69] (§2.6), and we do *not* train on any sample drawn from the MMAD benchmark categories used at evaluation, so no MMAD image is ever seen by our training pipeline. This gives a clean zero-shot transfer setting with respect to our own training. As for every model we compare against, it cannot exclude that the public DS-MVTec and VisA images were seen during backbone pre-training (§7.7).

1.3. Research Questions

The experimental work of the thesis is organised around one main question, which §8.2 answers directly:

Can an open vision-language model, fine-tuned on structured reasoning traces and refined with reinforcement learning, deliver explainable industrial anomaly detection that is competitive with the state of the art?

Four research questions make it specific. Each is answered by one part of Chapter 6.

RQ1 To what extent does supervised fine-tuning on an automatically generated corpus of structured reasoning traces improve balanced accuracy on industrial anomaly detection, and how do the corpus's size, composition and author affect the gain?

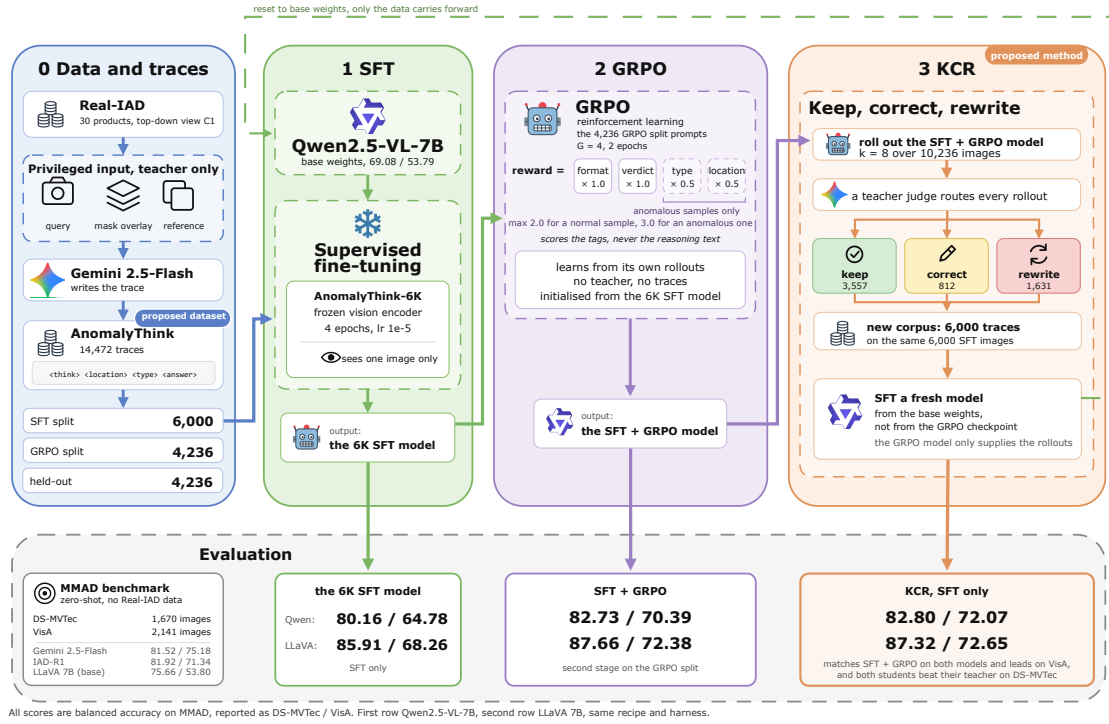


Figure 1.1: Overview of the thesis. Real-IAD images and their defect masks are turned into 14,472 structured reasoning traces by a Gemini 2.5-Flash teacher, the **AnomalyThink** corpus. A VLM is then fine-tuned on the 6,000-trace split with its vision encoder frozen. From that one initialisation two second stages are compared. Stage 2 is GRPO reinforcement learning. Stage 3 is KCR, in which the SFT+GRPO model is rolled out eight times over all 10,236 training images, the teacher keeps, corrects or rewrites each trace, and a fresh model is trained from base on the resulting corpus. KCR gives the headline result, 82.80 % / 72.07 % balanced accuracy on DS-MVTec / VisA with Qwen2.5-VL-7B and 87.32 % / 72.65 % with LLaVA-OneVision-7B-SI. Every model answers with a structured rationale whose quality is graded by a blind judge (§6.11).

RQ2 Does Group Relative Policy Optimisation with a multi-component reward improve on the supervised baseline, does that depend on the baseline’s strength, and what does it add beyond the binary verdict?

RQ3 Within the supervised design space, how do model capacity, vision-encoder treatment and training-set size trade off, and what does the per-product breakdown reveal about the remaining failure modes?

RQ4 Beyond the binary verdict, are the model’s explanations *grounded* in the visible evidence rather than merely fluent, and can explanation quality be assessed independently of detection accuracy?

1.4. Contributions

Figure 1.1 summarises the complete pipeline of this thesis in one view: from Real-IAD images and Gemini-generated reasoning traces, through supervised fine-tuning and GRPO, to the self-distilled corpus that produces the strongest detector reported here, and the common harness on which every checkpoint is evaluated. Every number below comes from that one strict harness, under which the released IAD-R1 checkpoint and six proprietary models were re-scored (§6.12, Appendix J). The contributions of this thesis are:

1. An **openly released reasoning-trace dataset for Real-IAD**, which we name **AnomalyThink**: 14,472 XML-structured, six-phase chain-of-thought traces partitioned into product- and class-stratified splits, a 6,000-trace SFT pool (**AnomalyThink-6K**), a 4,236-trace GRPO pool, and a 4,236-trace held-out test set. The union is referred to as **AnomalyThink-15K**, covering 30 Real-IAD products. To the best of our knowledge **AnomalyThink** is the first openly-released reasoning-trace dataset that is both built on Real-IAD

and purpose-designed for GRPO with structured XML output, distinct from Anomaly-Instruct-125k [75], which is conversation-format and draws its industrial subset from MVTec/VisA, and from Expert-AD [39], which is likewise Real-IAD-based and structured but is released only behind a gated entry that does not grant access. The full prompt template, vocabulary, and generation-time quality controls (auto-reject rules plus schema validation, no second model) are detailed in Chapter 3.

2. A **controlled four-factor SFT study** on Qwen2.5-VL. We fully fine-tune (no LoRA) Qwen2.5-VL at 3B and 7B for up to four epochs each, with the vision encoder frozen at both 6K and 15K, unfrozen at 15K, and one unfrozen 6K control at the frozen learning rate. The 15K results, the 6K control and prior work all show that unfreezing does not help on this task. The resulting ablation, detailed in Chapter 4, establishes two findings. (a) The 7B-frozen-6K configuration reaches **80.16 % balanced accuracy on DS-MVTec**, a strong SFT operating point that comes within ~ 1.8 pp of the released IAD-R1 SFT+GRPO checkpoint without any GRPO step. (b) The often-quoted “more data is always better” heuristic is wrong in this regime. At the 7B scale the curated 6K SFT split outperforms the full ≈ 15 K union by 8.5 pp balanced accuracy on DS-MVTec at matched training epochs (80.16 % vs. 71.66 %). This is a data-count and composition effect under the same four-epoch schedule, and it holds despite the larger set receiving $2.4\times$ as many optimiser steps. It is not a trace-length difference, since both datasets share the same ≈ 137 -word mean trace length.
3. A **GRPO post-training stage and a characterisation of its limits**. We implement GRPO with two un-weighted reward functions, a consistency (format) reward and an accuracy reward, decomposing into four bounded sub-signals (format, verdict, semantic type, location, max 3.0), the unbiased KL k3 estimator monitored during training (coefficient $\beta=0$, i.e. not added to the loss), group size $G=4$, learning rate 1×10^{-6} and a 4,236-prompt class-balanced training set, on commodity hardware ($2\times$ RTX A6000). Applied to the SFT checkpoint it reaches **82.73 % on DS-MVTec, 70.39 % on VisA and 80.87 % on a held-out Real-IAD split** (Chapter 5). Under a common harness (§6.12) this exceeds the released IAD-R1 checkpoint by +0.81 pp on DS-MVTec but trails it by -0.95 pp on VisA (70.39 vs. 71.34). The lift is conditioned on the weak SFT baseline. Applied on top of the KCR corpus of the next contribution it did not improve balanced accuracy in our experiments. Reward kept rising while balanced accuracy stayed 1.6 to 4.0 pp below the supervised initialisation on DS-MVTec, for all three advantage estimators we tried, vanilla z-score, Dr.GRPO and G^2 RPO, within a single 120-step probe on one initialisation, while one prompt-aligned run reached a checkpoint numerically above the KCR row that we treat as inconclusive at a single seed (§6.8). What GRPO does teach is the location tag. Among detected anomalies the SFT+GRPO checkpoint places it in the correct grid cell 82.4 % of the time on DS-MVTec against 73.2 % after SFT (§6.10.4).
4. **The KCR recipe, keep, correct and rewrite, and its teacher ablation**. The SFT+GRPO policy is rolled out eight times over the 10,236 training images, a teacher judge keeps 3,557 traces, corrects 812 and rewrites 1,631, and a fresh model is trained from base weights on the resulting 6,000 traces (Chapter 5, §6.6). Three arms with the images held fixed, kept only, kept plus corrected, and all three, isolate the effect of teacher intervention under the RAFT, STaR and STaR+Saunders critique-and-revise recipes. Arm C reaches **82.80 % / 72.07 %** on DS-MVTec / VisA with SFT alone, the best Qwen2.5-VL detector in the thesis and above the released IAD-R1 checkpoint on both benchmarks (+0.88 / +0.73 pp). It is the headline released checkpoint.
5. A **replication of the final recipe on a second backbone**. The KCR keep, correct and rewrite recipe was rerun end to end on LLaVA-OneVision-7B-SI [34], the backbone on which IAD-R1 reports its own headline result. Training that backbone on its own judged and corrected rollouts reaches **87.32 % / 72.65 %** on DS-MVTec / VisA at the selected epoch against the 87.66 % / 72.38 % of the SFT+GRPO model on the same backbone, and 74.29 % on VisA by epoch 4. The supervised stage therefore matches the reinforcement-learned one on both backbones and exceeds it on VisA (§6.9, which also states a contamination caveat that applies to the DS-MVTec column on this backbone but not to VisA).
6. An **evaluation of the explanation itself**, not only of the verdict. On one shared set of correctly detected anomalies, 90 on DS-MVTec and 47 on VisA, we score every model’s traces with a reference-guided judge that is blind to which model wrote the trace, next to mechanical location and type scores (§6.11). Our own models score 2 to 3 points out of 10 above the released IAD-R1 checkpoint on identical images, with 95 % intervals that exclude zero, and among themselves the supervised-only models explain least well on VisA, with the reinforcement-learning stage and the corpus built from its rollouts ahead. The judge measures grounding consistency against reference evidence rather than mechanistic faithfulness, and every score is the median of three judge samples. A fuller decomposed rubric with a human-validation

protocol is proposed in Appendix K as future work.

7. A **diagnosis of where the final system gains and where it still fails** (§6.10.2, §7.6). Against the untrained backbone the KCR model gains 13.7 pp on DS-MVTec and 18.3 pp on VisA, concentrated on the products the base model handled worst (metal_nut +35.6, toothbrush +29.2 and hazelnut +24.8 on DS-MVTec, pipe_fryum +42.2 on VisA), while the three DS-MVTec regressions (bottle −3.2, wood −2.5, pill −1.7) are within 3.2 points on products where the base already scored 60 to 98. The remaining errors fall into five categories, of which small-area defects and multi-object layout reasoning dominate and set the agenda for the next iteration.

1.5. Thesis Outline

The remainder of this thesis is organised as follows.

Chapter 2 surveys the technical background in three layers: (i) classical industrial anomaly detection from PatchCore through EfficientAD, (ii) vision-language models from CLIP through Qwen2.5-VL, and (iii) RL post-training from PPO through GRPO/GSPO and OpenVLThinker. It ends with a focused review of the published VLM-based IAD methods and identifies the gap this thesis fills.

Chapter 3 describes the reasoning-trace dataset, **AnomalyThink**: the six-phase prompt structure, the controlled vocabularies for defect type and location, the three-image input format used for anomalies (query image, red-mask overlay, reference normal) against the single image used for normals, the per-product stratified sampling strategy, and the generation-time quality controls. It ends with quantitative statistics of the resulting $\approx 14.5\text{K}$ -trace dataset.

Chapter 4 presents the supervised fine-tuning methodology: model family, hyperparameters, the LlamaFactory ShareGPT data format, the DeepSpeed ZeRO-3 configuration with CPU offload, and the four-factor ablation (3B/7B, frozen/unfrozen, 6K/15K, epochs 1–4). The unfrozen encoder is run at 15K, plus one matched-learning-rate control at 6K, giving 6 configurations and a control.

Chapter 5 derives the GRPO objective from first principles and reports the two reward functions, which are summed unweighted and decompose into four bounded sub-signals (format, verdict, type, location, max 3.0). It also details the training pipeline (group size, KL coefficient, sample budget).

Chapter 6 reports the results. It starts with the base-model baseline, the four-factor SFT grid and the GRPO run. It then covers the iterative SFT $\leftrightarrow$ RL and alternative RL variants, the teacher ablation that yields the best supervised model, and the replication of that recipe on a second backbone. Three further sections treat GRPO on top of that best supervised baseline, a post-GRPO refinement run on cleaned data, and explanation quality. The chapter closes with the head-to-head against the state of the art, the per-product breakdown on DS-MVTec and VisA, and the training and reward dynamics.

Chapter 7 discusses why frozen vision encoders win, why the smaller split wins, and why VisA remains harder than DS-MVTec. It then treats checkpoint selection and reward over-optimisation, revisits the value of GRPO in the light of the teacher ablation, analyses where the remaining errors come from, and states the limitations of the study.

Chapter 8 summarises and lays out the next steps: iterative SFT $\rightarrow$ RL cycles, curriculum sampling, and the agentic inspection pipeline.

Appendices A–N hold the supporting material: the full hyperparameter tables, the verbatim prompts, the per-product tables on DS-MVTec and on VisA, the reward-function reference, the qualitative trace gallery, the teacher-versus-student traces, the training curves, the datasheet for the AnomalyThink dataset, the evaluation harness and its prompt-mode robustness check, the executed explainability-scoring protocol and its proposed extension, the held-out Real-IAD evaluation, the corrected LLaVA-OneVision corpus, and a reference run of GRPO with a frozen vision encoder.

2

Background and Related Work

This chapter establishes the technical foundations of the thesis and surveys the literature it builds on. §2.1 reviews *classical* industrial anomaly detection (IAD), tracing the four main methodological families from memory banks to real-time student-teacher distillation, and identifies the limitations that motivate the move to language-aware models. §2.2 introduces the vision-language model (VLM) stack on which our system is built, with particular attention to the architecture of Qwen2.5-VL. §2.3 reviews chain-of-thought reasoning and its multimodal extensions, which directly motivate our six-phase trace structure. §2.4 derives the necessary RL background and the GRPO objective. §2.5 surveys the rapidly growing body of work that applies VLMs and RL to industrial anomaly detection (Anomaly-OV [75], AnomalyR1 [9], EMIT [20], LR-IAD [84] and IAD-R1 [39]), and positions this thesis within that landscape. §2.6 introduces the evaluation benchmarks used throughout the rest of the thesis.

2.1. Classical Industrial Anomaly Detection

Modern unsupervised IAD methods share three structural assumptions. The training set contains only normal images, defects are rare in the test data, and the task is to assign an anomaly score to each image such that defective samples are pushed away from the normal distribution. Within that framework, four families have come to dominate the MVTec AD leaderboard.

2.1.1. Memory-bank and feature-embedding methods

Memory-bank methods store a compact bank of normal patch features from a frozen pretrained backbone and score a test patch by its distance to that bank. SPADE [10] retrieves nearest-neighbour normal patches, PaDiM [12] fits a per-location multivariate Gaussian and scores by the Mahalanobis distance to the fitted normal, and PatchCore [58] subsamples the feature bank into a representative *coreset* and scores by k -nearest-neighbour distance, reaching 99.6 % image-level and 98.4 % pixel-level AUROC on MVTec AD with no task-specific training of the backbone. The family is accurate and trains no backbone, but it stores and searches a separate memory bank per object category.

2.1.2. Normalising-flow methods

Normalising-flow methods learn an invertible mapping from normal patch features to a simple base density, so that a low likelihood under the trained flow flags an anomaly. CFLOW-AD [21] conditions per-pyramid-level flows on a positional encoding, FastFlow [79] uses a 2D flow for full-resolution localisation, CS-Flow [59] couples cross-scale feature maps in a single flow, and PyramidFlow [33] performs high-resolution contrastive localisation in a pyramid of flows. These methods produce calibrated, fully-localised anomaly maps (CFLOW-AD reports 98.3 % image-AUROC on MVTec AD), and like memory banks they are fitted per category.

2.1.3. Reconstruction and distillation methods

Reconstruction and knowledge-distillation methods learn to reproduce normal appearance, or normal features, and flag regions they cannot reproduce. Generative reconstructors such as GANomaly [2] and f-AnoGAN [63] score by reconstruction residual, while DRAEM [82] instead trains a discriminative sub-network on synthetically-corrupted images for pixel-precise segmentation (98.0 % image-AUROC on MVTec AD). Distillation variants

score the divergence between a frozen teacher and a student trained only on normal data: Reverse Distillation [13] reconstructs multi-scale teacher features from a one-class bottleneck (98.5 % image-AUROC), Multiresolution Knowledge Distillation [60] matches features across resolutions, and EfficientAD [5] pushes the same idea to roughly 2 ms inference and 600 FPS on commodity GPUs. EfficientAD is the canonical efficiency-oriented baseline against which any VLM-based method must justify its higher computational cost.

2.1.4. Self-supervised and synthetic-defect methods

Self-supervised and synthetic-defect methods manufacture pseudo-anomalies to turn the one-class problem into a supervised one. CutPaste [35] cuts a patch from a normal image and pastes it at a random location, then trains a classifier to detect the splice (96.6 % image-AUROC on MVTec AD with no defective training data). SimpleNet [42] adds Gaussian perturbations to normal feature vectors and trains a small discriminator, reaching 99.6 % image-AUROC at 77 FPS. Most recently, AnomalyDiffusion [25] synthesises realistic defect images with a text-to-image diffusion model from a handful of real samples. These methods avoid per-category memory banks but still require category-specific training, and the synthesis-based variants need a few real defects to seed.

2.1.5. Limitations of classical IAD methods

On MVTec AD all four families saturate near 99 % image-AUROC and the benchmark no longer discriminates between methods. Three operational limitations remain, however, and motivate the move to VLMs.

- *No language explanation.* Every classical method outputs only a scalar and a heatmap, and none of them can articulate “a 3 mm horizontal scratch near the upper-right edge of the threaded cylinder”. This opacity is a recognised barrier to trust and adoption in manufacturing [47], and is the explanation gap addressed in this thesis.
- *Per-category training and per-category memory.* Each of PatchCore, PaDiM, CFLOW-AD, DRAEM, RD4AD, EfficientAD and SimpleNet requires a separate training run, or at least a separate coreset/memory bank, per object category. Real-IAD Variety [93] showed that scaling these methods from 30 to 160 categories drops mean AUROC by 10 to 20 points.
- *Saturation on structural-only benchmarks.* Image-level AUROC > 99 % on MVTec AD masks the fact that classical methods struggle with *logical* anomalies (wrong-number-of-pushpins, swapped components) introduced in MVTec LOCO [7], where they regress by 10 to 30 points.

2.2. Vision-Language Models

A vision-language model maps a pair (I, x) of an image I and a text prompt x to a sequence of text tokens. Modern VLMs share a three-component recipe: (i) a pretrained image encoder that turns I into a sequence of visual tokens, (ii) a connector that projects those visual tokens into the embedding space of a pretrained language model (LM), and (iii) the LM itself, which autoregressively generates the response conditioned on the concatenation of visual and text tokens.

2.2.1. From CLIP to instruction-tuned VLMs

The first system to establish the textual-supervision paradigm at internet scale was **CLIP** [53], a dual-encoder model trained on 400 M (image, text) pairs by maximising the cosine similarity of matched pairs and minimising it for the $N^2 - N$ in-batch negatives. CLIP performs no generation but provides a strong zero-shot classification interface and an image encoder that has since been reused throughout the VLM stack. **BLIP-2** [36] introduced the now-canonical recipe: a frozen ViT image encoder is connected to a frozen LLM through a small Q-Former that contains a fixed set of learnable query tokens that cross-attend to image features. **LLaVA** [41] simplified the connector to a single linear projection, and performed the first *visual instruction tuning* by using GPT-4 to synthesise 158K multimodal instruction-following examples. The resulting model reaches 92.5 % accuracy on ScienceQA (a new state of the art in its GPT-4-augmented setting). **LLaVA-OneVision** [34] extended the recipe to a unified single-image / multi-image / video model.

2.2.2. The Qwen-VL family

The Qwen line [3, 4, 70] provides the backbone for this thesis. **Qwen-VL** introduced a position-aware adapter that compresses image features to 256 tokens. **Qwen2-VL** [70] added two changes critical for industrial in-

spection: *Naive Dynamic Resolution*, which encodes images at native resolution and yields a variable number of visual tokens, and *Multimodal Rotary Position Embedding* (M-RoPE), which decomposes positional encoding into temporal/height/width components so that text, image and video share a single position space. **Qwen2.5-VL** [4] extends Qwen2-VL with: a redesigned ViT using *window attention* for efficiency, absolute time encoding for long video, improved object grounding (bounding boxes and points), and stronger document parsing. We use the 3B and 7B variants throughout this thesis. Qwen2.5-VL’s native dynamic-resolution behaviour is particularly important for IAD because industrial defects can occur at arbitrary scales, ranging from sub-millimetre scratches to fully missing components.

The closest open-weight alternative is **InternVL3** [92], which uses a *native multimodal* pretraining paradigm and Variable Visual Position Encoding (V2PE). On general multimodal benchmarks InternVL3-78B reaches 72.2 on MMMU, the highest open-source score at release time. Proprietary models appear in this thesis in two roles. Google’s **Gemini 2.5-Flash** [18] is the trace-generation model of Chapter 3 and a zero-shot reference in Chapter 6, and OpenAI’s GPT-5-mini [50] is a second zero-shot reference there (§6.12).

2.3. Chain-of-Thought Reasoning

Chain-of-Thought (CoT) prompting [72] is the technique of supplying a few exemplars of intermediate reasoning steps, so that a sufficiently large LM decomposes a problem into a chain of textual sub-claims before emitting a final answer. CoT improves arithmetic, commonsense and symbolic reasoning. In the 2022 experiments the gains appeared only beyond ~ 100 B parameters, as a few-shot prompting effect. Later work showed that the behaviour can be trained into much smaller models by fine-tuning on reasoning traces and by reinforcement learning [11], which is the route this thesis takes at 3B and 7B. **Self-Consistency** [71] samples N chains under temperature decoding and marginalises over the final answer, yielding double-digit gains on GSM8K and other reasoning benchmarks. **Tree of Thoughts** [77] generalises the linear chain to a search tree over thoughts and traverses it with BFS/DFS plus self-evaluation, lifting Game-of-24 accuracy from 4 % to 74 %.

For visual tasks, **Multimodal-CoT** [86] introduced a two-stage architecture that first generates a textual rationale conditioned on image and question, then conditions answer inference on image, question and rationale. The central finding is that grounding the rationale in visual features mitigates hallucination, a property we rely on in our six-phase trace template. Structured outputs of this kind, with explicit verdict, type and location fields, also appear in concurrent VLM-IAD systems such as IAD-R1 [39] and AnomalyR1 [9]. This thesis pushes the idea further by enforcing a *structured* chain-of-thought with a fixed six-phase XML schema (`<think>`, `<location>`, `<type>`, `<answer>`), which has three practical advantages: (i) the structure is machine-checkable, enabling format rewards in RL, (ii) it forces evidence before conclusion, which lowers the false-alarm rate against a verdict-only control (§6.3), and (iii) it makes downstream parsing trivial.

2.4. Reinforcement Learning for Language Models

The RL background relevant to this thesis is the alignment of an already-pretrained policy π_θ with a scalar reward signal $r(\cdot)$. We summarise the relevant algorithms in chronological order and then derive GRPO in the form actually used in Chapter 5.

2.4.1. From PPO to RLHF

Proximal Policy Optimisation [65] maximises a clipped surrogate

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right], \quad (2.1)$$

where $r_t(\theta) = \pi_\theta(a_t | s_t) / \pi_{\theta_{\text{old}}}(a_t | s_t)$ is the importance ratio and $\hat{A}_t$ is the generalised advantage estimate, requiring a learned value function. PPO became the workhorse of language-model alignment via the *Reinforcement Learning from Human Feedback* (RLHF) pipeline introduced by Ouyang et al. [52]: a reference policy is first SFT’d on demonstrations, a reward model is fitted on pairwise human preferences with a Bradley-Terry loss, and the SFT policy is then fine-tuned against the reward model with PPO and a KL penalty to the reference. **Direct Preference Optimisation** (DPO, 54) showed that the same KL-constrained optimum can be reached without explicit reward modelling, by reparameterising the reward in terms of the policy: a simple binary cross-entropy loss on preferred/dispreferred completions suffices.

2.4.2. Group Relative Policy Optimisation (GRPO)

GRPO was introduced as part of DeepSeekMath [66] and popularised for reasoning by DeepSeek-R1 [11]. Its central simplification is to *replace the learned value function* of PPO with a Monte-Carlo group baseline: for each prompt q , sample a group of G completions $\{o_1, \dots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$, score each with the reward function to obtain $\{r_1, \dots, r_G\}$, and compute a group-relative advantage

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_1, \dots, r_G\})}{\text{std}(\{r_1, \dots, r_G\})}, \quad (2.2)$$

which is shared by every token t of the trajectory o_i . The GRPO surrogate is then

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{q \sim \mathcal{P}(Q) \\ \{o_i\} \sim \pi_{\theta_{\text{old}}}(\cdot|q)}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min(\rho_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(\rho_{i,t}(\theta), 1 - \varepsilon, 1 + \varepsilon) \hat{A}_{i,t}) \right. \right. \\ \left. \left. - \beta \text{KL}[\pi_{\theta} \parallel \pi_{\text{ref}}] \right\} \right], \quad (2.3)$$

with token-level ratio $\rho_{i,t}(\theta) = \pi_{\theta}(o_{i,t} | q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t} | q, o_{i,<t})$. The KL term, when used, is placed inside the loss (weighted by β) and computed via Schulman’s unbiased k_3 estimator [64]. In this thesis we set $\beta=0$, so it is monitored but not added to the loss (see Chapter 5):

$$\text{KL}[\pi_{\theta} \parallel \pi_{\text{ref}}] \approx \frac{\pi_{\text{ref}}(o_{i,t} | \cdot)}{\pi_{\theta}(o_{i,t} | \cdot)} - \log \frac{\pi_{\text{ref}}(o_{i,t} | \cdot)}{\pi_{\theta}(o_{i,t} | \cdot)} - 1, \quad (2.4)$$

which is always non-negative. The hyperparameters governing the algorithm are G (group size), ε (clipping range), and β (KL coefficient).

GRPO is attractive for VLM-IAD for three reasons. (i) The verifier-style reward is sparse and binary-ish (correct/incorrect verdict), exactly the regime PPO’s critic struggles with. (ii) The group baseline saves the entire value-network memory cost, which is significant on a 7B model. (iii) The reward function can be *decomposed* into structured components (format, accuracy, type, location) and summed without changing the algorithm, a property we exploit in Chapter 5.

The Qwen team’s follow-up **GSPO** [88] replaces GRPO’s token-level importance ratio with a sequence-level ratio for stability at MoE scale. We discuss it in Chapter 8 as a candidate for future work but do not use it here. Related critic-free algorithms include RLOO [1] and REINFORCE++ [24]. The most directly relevant prior work to our SFT→GRPO recipe is **OpenVLThinker** [14], which showed that iterative SFT↔RL cycles outperform single-round SFT+RL on multimodal reasoning, a roadmap item we instantiate as an IAD-specific ablation in §5.6 and to which we return in §8.4. Its sequel **OpenVLThinker v2** [26] introduces **G²RPO**, a non-parametric replacement for the z -score advantage of (2.2) that maps within-group reward ranks onto the inverse standard-normal CDF. We evaluate this variant on IAD in §5.7.

2.4.3. Self-training: rejection sampling, STaR, and self-distillation

A recurring design question when training a reasoning model is *who writes the rationale*: an external teacher, or the model itself. The knowledge-distillation and self-improvement literature offers a spectrum of recipes, each granting an external teacher a different level of intervention over the model’s own outputs. The teacher ablation of §6.6 instantiates three of them.

- **Hinton-style external distillation** [22]. The classical setting: a strong external teacher writes all training labels, and the student imitates them. The 6K Gemini-only SFT headline of this thesis (§6.3) is a pure external-teacher baseline.
- **Self-distillation** [85]. The model trains on its own outputs without any external teacher. Theoretical and empirical support: when model capacity is sufficient and the training distribution is well-aligned, the student can match or exceed teacher-distilled performance.
- **Rejection-sampling SFT** (RAFT [15] and ReST [67]). Sample multiple candidates per prompt, retain only those scoring above a reward threshold, train. The teacher is used only as a scorer, never as a writer. In our setting the reward is simply ground-truth answer-correctness. How close this comes to policy-gradient RL has been measured directly. On Qwen2.5-Math-7B, rejection-sampling fine-tuning reaches

56.1 against 56.3 for GRPO, and the remaining gap is attributed to GRPO’s use of negative samples [74]. An online variant of rejection-sampling fine-tuning tracks GRPO early in training and is overtaken later, because it does not penalise incorrect responses and reinforces every correct one equally [66]. The arms of \$5.8 sit on this spectrum, with the teacher supplying, on the failed items, the signal that negative samples supply in GRPO.

- **STaR, the Self-Taught Reasoner** [83]. Combines rejection sampling with a *rationalisation-on-failure* step. In STaR the model first tries to answer on its own. When it gets the answer wrong, it is asked again with the correct answer given as a hint, and it writes the rationale itself. No stronger model is involved. The authors find this step is the single most important component of the recipe.
- **Self-critique / critique-and-revise** [61]. Uses an external critic to flag low-quality outputs and produce revised versions, adding a revision step on top of rejection sampling and rationalisation.

2.5. VLM-Based Industrial Anomaly Detection

VLM-based IAD has crystallised in the last two years along three paradigms: *zero-shot CLIP-based* methods, *supervised fine-tuning* methods, and *SFT+RL* methods.

2.5.1. Zero-shot CLIP-based methods

WinCLIP [29] is the canonical zero-shot CLIP-based detector: a compositional prompt ensemble over state words (“normal”, “damaged”) and template strings is paired with a multi-window aggregation of CLIP features to yield patch/window/image-level anomaly scores. It reaches 91.8 % image-AUROC on MVTec AD without any training. **AnomalyCLIP** [91] learns two *object-agnostic* prompt templates shared across all categories, using auxiliary annotated data. The resulting prompts transfer better across domains than category-specific ones. **PromptAD** [38] targets the few-shot setting by appending defect suffixes to learned normal prompts (*semantic concatenation*), and **AdaCLIP** [8] introduces hybrid (static + dynamic) prompts. All of these methods operate on frozen CLIP and produce only a score plus an attention map, with no language-level explanation.

2.5.2. Supervised fine-tuning methods

AnomalyGPT [19] is an early industrial AD VLM built on PandaGPT, which pairs an ImageBind visual encoder with a Vicuna language model. A lightweight pixel decoder is added on top of the ImageBind features, and learnable prompt embeddings (rather than full LLM weights) are tuned on simulated and few real defects. With one normal reference shot, it reaches 86.1 % accuracy and 94.1 % image-AUROC on MVTec AD. **Anomaly-OV** [75] is a LLaVA-based industrial AD VLM. It is built on LLaVA-OneVision and introduces a Look-Twice Feature Matching mechanism that re-attends to suspicious regions, together with the Anomaly-Instruct-125K instruction-tuning dataset and the VisA-D&R detection-plus-reasoning benchmark. These methods do not use RL.

2.5.3. SFT+RL methods

GRPO on top of an SFT cold start has become the dominant recipe for reasoning-based IAD, and recent methods differ mainly in how they shape the reward rather than in the underlying algorithm. AnomalyR1 [9] introduced the recipe as the first end-to-end MLLM IAD method to use GRPO, scoring a Qwen2.5-VL-3B policy with a single Reasoned Outcome Alignment Metric (ROAM) reward over reasoning fidelity and localisation. The later methods each target a specific weakness of this recipe: class imbalance through a focal-loss-inspired reward (LR-IAD [84], on Qwen2-VL-7B, no pixel masks), zero-signal training batches through difficulty-aware re-sampling of groups with no correct rollout (EMIT [20], on InternVL3-8B), bounding-box localisation through a re-think stage with GloU rewards (AD-FM [40]), and threshold-free detection through a Text-as-Mask encoding that predicts anomalies by text generation (OmniAD [87], which reaches 79.1 % average accuracy on MMAD, above Qwen2.5-VL-7B and GPT-4o).

The most relevant comparison for this thesis is **IAD-R1** [39], a recent open-weight SFT+GRPO baseline and the one of these systems that releases a checkpoint on a backbone we can match. Its pipeline has two stages: PA-SFT (Perception Activation SFT) on a curated 2.9K-sample subset of their *Expert-AD* dataset, followed by SC-GRPO (Structured Control GRPO) on the remaining 3K samples with four rewards: R_{acc} (binary verdict), R_{loc} (3×3 grid match), R_{type} (graded type similarity), R_{con} (format consistency). IAD-R1 is evaluated by its authors zero-shot on six datasets (MVTec, MPDD, VisA, DAGM, DTD, SDD), and the LLaVA-OneVision-7B

backbone is the headline of that work. We do not quote the IAD-R1 paper’s own evaluation numbers in this thesis. For all comparisons against IAD-R1 we instead re-run the publicly released Qwen2.5-VL-7B IAD-R1 checkpoint through the same evaluation harness used for our own models, so that the comparison is on a strictly common footing (see §6.12).

2.6. Benchmark Datasets

This thesis uses two datasets: Real-IAD for training, with a held-out split of unseen images reserved for in-domain out-of-sample evaluation, and MMAD (specifically the DS-MVTec and VisA subsets) for primary evaluation.

Real-IAD [69] contains 150K+ high-resolution images across 30 product categories captured from five viewpoints, with ~50K anomalous samples and pixel-precise masks. It is roughly an order of magnitude larger than MVTec AD and re-introduces meaningful headroom over classical methods (mean AUROC drops from > 99 % on MVTec to ~85 % on Real-IAD for state-of-the-art methods). Real-IAD Variety [93] extends this to 160 categories across 28 industries. We do not evaluate on it in this thesis. It is listed as future work in §8.4.

MVTec AD [6] is the de facto standard for unsupervised IAD: 5,354 images of 15 categories, 70+ defect types, pixel-precise masks. **VisA** [94] contains 10,821 images of 12 multi-object scenes (PCBs, capsules, snacks) and is generally harder for classical detectors because it requires reasoning about object-level structure rather than only surface texture.

MMAD [30] is the first comprehensive VLM benchmark for IAD. It defines seven subtasks (anomaly discrimination, defect classification/localisation/description/analysis, object classification/structure) and provides 39,672 questions over 8,366 images drawn from MVTec AD, MVTec LOCO, VisA, GoodsAD and the Defect Spectrum re-annotation of MVTec (our DS-MVTec subset, detailed below). GPT-4o reaches only 74.9 % average accuracy, far below industrial deployment requirements. We use the DS-MVTec subset of MMAD (1,670 samples) and the VisA subset (2,141 samples) as our primary evaluation pair, consistent with IAD-R1’s reporting.

What DS-MVTec is. The DS-MVTec subset is the *Defect Spectrum* re-annotation [76] of the 15 MVTec AD categories (hence the “DS” prefix), repackaged by MMAD as multiple-choice question answering. It is therefore distinct from the original MVTec AD benchmark above in two ways. First, its images carry finer-grained, semantically richer defect labels. Second, and more importantly for this thesis, we evaluate it in the MMAD question-answering setting, scoring the binary anomaly-detection verdict with balanced accuracy, rather than in the original unsupervised image-level AUROC setting. The near-perfect AUROC figures quoted for MVTec AD and the balanced-accuracy figures we report on DS-MVTec are therefore measured under different protocols and not directly comparable.

3

Reasoning Trace Generation

This chapter describes how the supervised training data of this thesis is produced. The dataset, which we name **AnomalyThink**, consists of **14,472** structured chain-of-thought (CoT) reasoning traces generated over the Real-IAD dataset using Gemini 2.5-Flash. It is divided into three disjoint, product- and class-stratified splits: a **6K supervised fine-tuning (SFT) split (AnomalyThink-6K)**, a **4K GRPO split**, and a **4K held-out split**. We also experiment with the full union of all three splits ($\approx 14.5\text{K}$), which we call **AnomalyThink-15K**, as a data-quantity ablation in Chapter 6. The chapter is organised top-down. §3.1 introduces the source data. §3.2 specifies the output format and the six-phase reasoning template. §3.3 details the prompt and the generation-time quality controls, and §3.4 describes those controls in full. §3.5 explains how the traces are partitioned into three stratified splits with strict overlap prevention. §3.6 reports descriptive statistics. §3.7 positions **AnomalyThink** against existing reasoning-trace datasets. Three figures map this chapter. Figure 3.1 shows what is read from Real-IAD and what the teacher is given. Figure 3.2 shows how one trace is written and vetted. Figure 3.3 shows what a finished instance contains and how the corpus is split.

3.1. Source Dataset

Figure 3.1 shows the selection rule and the prompt payload used in this section. All reasoning traces are generated from Real-IAD [69] (§2.6). For trace generation we use two of its properties directly: its per-view organisation, five camera viewpoints per physical part labelled C1 to C5 of which this thesis uses only the top-down C1 view, and the pixel-precise binary defect masks on the anomalous samples. Each sample is grouped under a directory of the form

Product/{OK,NG}/{defect_code}/S####/{image}.jpg

where OK marks defect-free items, NG (*No-Good*) marks defective items, the `defect_code` is one of eight two-letter Real-IAD anomaly codes mapped to a readable defect type (Table 3.1), and `S####` is the sample identifier. Camera angle is encoded in the filename (`_C1_`, ..., `_C5_`).

Table 3.1: The eight Real-IAD two-letter anomaly codes and their readable defect types (OK denotes a normal sample).

Code	Type	Code	Type
AK	Pit	PS	Damage
BX	Deformation	QS	Missing Parts
CH	Abrasion	YW	Foreign Objects
HS	Scratch	ZW	Contamination

For trace generation we use all 30 Real-IAD products and treat as anomalous only the NG samples that have a ground-truth defect mask at the C1 angle. An NG image without a mask at that angle shows no visible defect from the top-down view, so it is relabelled *normal* and enters the normal pool. The verdict label is recomputed from the visible evidence rather than inherited from the folder. A pre-computed mask-analysis pipeline additionally labels each retained mask as *Single* (one localised region), *Multi-Close* (several nearby regions) or *Spread* (scattered non-contiguous regions), but we use this taxonomy only *descriptively* and do *not*

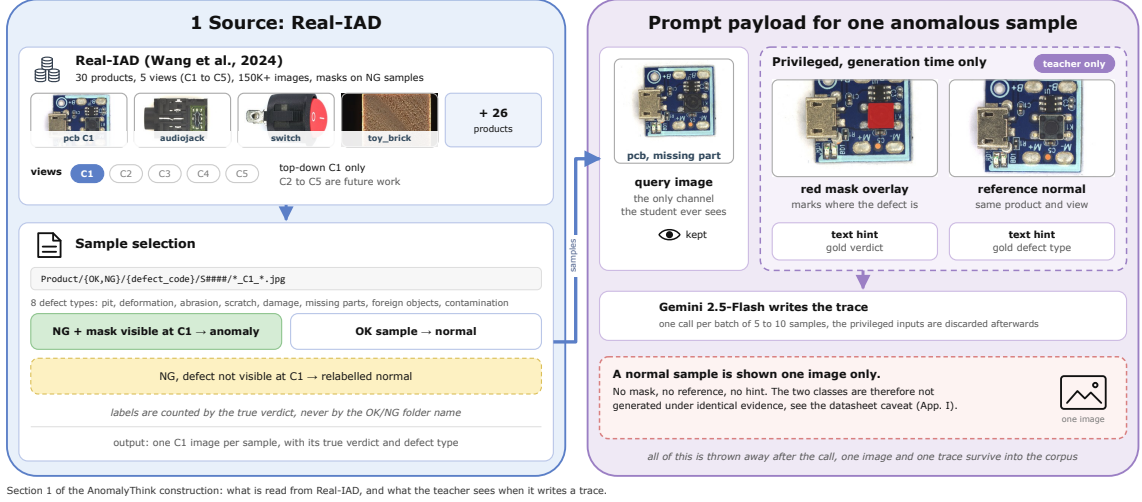


Figure 3.1: What is read from Real-IAD, and what the teacher is given. Only the top-down C1 view is used. An anomalous sample reaches the generator as three images. These are the query image, the same image with the ground-truth defect mask painted red, and a reference normal of the same product and view. Two text hints carry the gold verdict and the gold defect type. All of that is privileged and exists only at generation time. The trained student receives the query image and nothing else.

filter, weight or stratify the dataset on it (inclusion depends solely on mask presence). For normal samples no mask filter is applied. We sample uniformly across the available OK directories.

3.2. Output Format: Structured Six-Phase Trace

A worked example of both tag layouts is on the left of Figure 3.3. Every generated and predicted trace sticks to a fixed XML output format, which is machine-checkable and parseable with simple regular expressions. It lends itself to per-component reward design (§5.2). For anomalous samples the trace has four tags:

```
<think>
[120-200 word inspection rationale following the six-phase template]
</think>
<location>top-right, center</location>
<type>Scratch</type>
<answer>Yes</answer>
```

For normal samples the trace ends at <answer> with no <location> or <type>:

```
<think>
[justification why the inspected image is defect-free]
</think>
<answer>No</answer>
```

Acknowledgement of shared schema with IAD-R1. This four-tag schema is identical to the one introduced by IAD-R1 [39] for SC-GRPO. Specifically, the four-component reward decomposition R_{acc} , R_{loc} , R_{type} , R_{con} of §5.2 maps one-to-one onto the four tags, which we adopt unchanged so that our results on DS-MVTec and VisA are directly comparable to the released IAD-R1 checkpoint re-run under our own harness (Table 6.16). Our contributions *over* IAD-R1 are (i) the *trace content*, in particular the six-phase reasoning template described below, (ii) the *source data*, the openly-released **AnomalyThink** dataset over Real-IAD (§3.7) rather than the gated, effectively-unavailable Expert-AD dataset, and (iii) the GRPO configuration of Chapter 5 (group size, KL coefficient, reward composition, learning rate). The surface XML format is shared on purpose, so that downstream comparisons isolate these three factors as the locus of the contribution.

The contents of <think> follow the **six-phase template**:

1. *Framing*. Name the product and the viewing angle, recall the dominant nominal features (e.g. “threaded cylinder, four notches, central hole”).

2. *Scan*. Sweep the image surface-by-surface, briefly noting any region of interest without yet asserting a defect.
3. *Focus*. Zoom in on the most suspicious region identified in the scan and describe the visual cue (“sharp specular highlight cutting across the texture”).
4. *Evaluate*. Compare the cue to a plausible defect category: is it deeper than the surrounding micro-texture? Does it align with a known manufacturing failure mode?
5. *Alternatives*. Briefly rule out competing explanations, such as specular glare, dust, or a feature that is supposed to be there.
6. *Decide*. Conclude: defect / no defect. If defect, name the type and location using the controlled vocabulary below.

The total budget is 120–200 words. This explicit target replaces an earlier, looser “3–5 sentences” instruction used in pilot generation. It is what produces the tightly-clustered trace lengths reported in §3.6 (mean $\approx$ 137 words).

Location vocabulary. A short list of canonical positions is enforced to make the <location> tag easy to score with a 3×3 grid-style reward (cf. §5.2). The nine allowed values map directly onto the grid cells, drawn in Figure 3.2. Comma-separated compounds such as *top-right*, *center* denote defects spanning multiple cells.

Free-form descriptions (“next to the lower screw”) and bare single-axis terms (*left*, *top*, ...) are disallowed by the generation prompt.

Defect-type vocabulary. The <type> tag draws from a controlled list of 26 industrial defect categories, organised into four groups:

Surface: Scratch, Crack, Dent, Pit, Abrasion, Discoloration, Contamination, Corrosion, Stain, Burr, Flash.

Structural: Broken, Fracture, Deformation, Warping, Chip, Damage, Bend.

Completeness: Missing Parts, Missing Component, Extra Component, Incomplete Assembly.

Other: Misalignment, Foreign Object, Poor Finish, Overmolding.

This vocabulary is intentionally broader than the Real-IAD-internal two-letter codes (BX, HS, ZW, ...): translating from product-specific codes to a generic vocabulary is what enables zero-shot transfer to MVTec AD/VisA defect categories at evaluation time. The translation table is implemented in the trace-generation prompt.

3.3. Generation Pipeline

Inputs to the generator. The Gemini call is structured as a batch of ≤ 10 samples to amortise API latency. For each *anomalous* sample, three images are attached: (a) the original anomalous photo, (b) the same photo with the binary mask overlaid in red, and (c) a *reference normal* image from the same product and, when available, the same camera view (*_C1_*, ..., *_C5_*). For each *normal* sample only the original photo is attached.

The privileged inputs are the overlay, which marks *where* the defect is, together with two internal text hints: the ground-truth verdict (anomalous or normal) and, for anomalies, the gold defect type. The generator’s task is therefore to write a plausible six-phase explanation of an already-known defect, not to detect it from scratch. All of these privileged inputs are *discarded* at inference: the trained student model sees only the test image, never the mask, the verdict, or the type.

System prompt. The Gemini system prompt is reproduced verbatim in Appendix B and consists of six parts: (i) the role and inspection mandate, (ii) the six-phase template described in §3.2, (iii) the two controlled vocabularies, (iv) the required JSON output schema, (v) a list of *auto-reject rules*, and (vi) an *acceptance filter*, a numeric self-scoring rubric the model applies to its own output before returning it (both detailed in §3.4). The auto-reject rules are:

1. **No batch leakage.** The trace must not refer to “another”, “next”, “previous”, “final” images or use ordinal positions.
2. **No metadata leakage.** No mention of labels, ground-truth, annotations, or the privileged mask overlay.

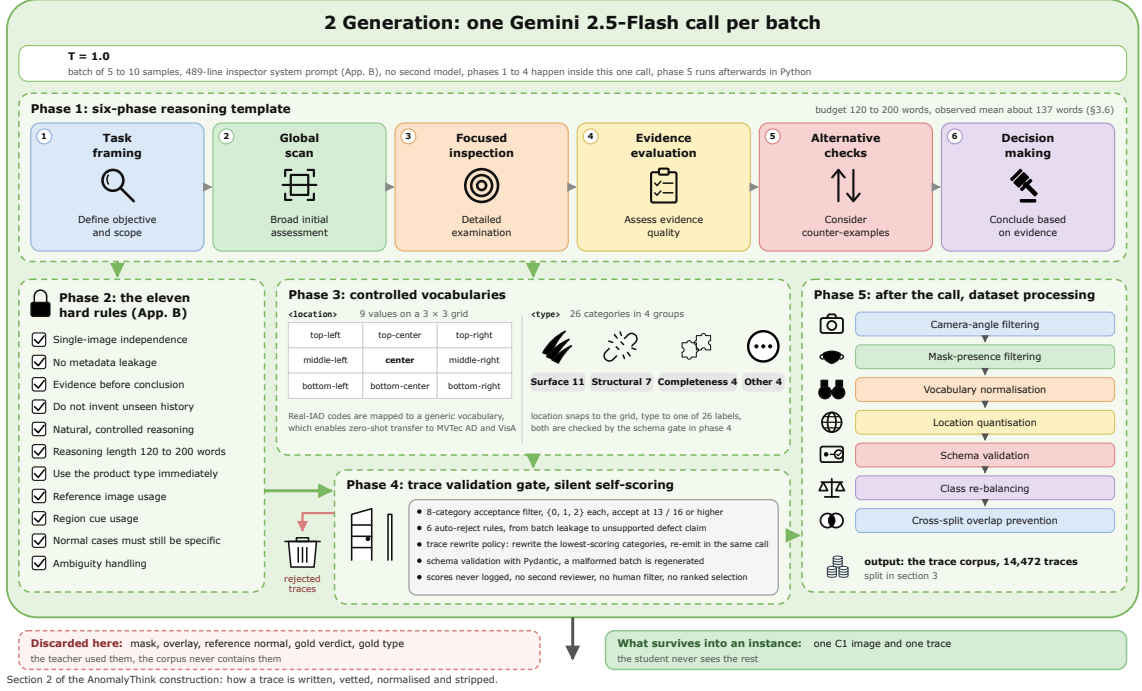


Figure 3.2: How one trace is written and vetted. A batch of five to ten samples goes to Gemini 2.5-Flash in a single call. The six-phase template, the two controlled vocabularies and the three quality gates all run inside that one call. No second reviewer model is used and no human filtering is applied, so **AnomalyThink-6K** is a stratified partition and not a top-scored subset. The privileged inputs are discarded once the call returns.

- Evidence before conclusion.** The defect type must be named *after* the visual cue is described, never before.
- Output structure correctness.** Valid JSON with all four tags in the correct order for anomalies, and `<think>` and `<answer>` only for normal samples.
- Causal restraint.** Any speculation about manufacturing causes must be tentative (“possibly an over-cured edge”) rather than definitive.
- Unsupported defect claim.** A defect must not be asserted without a concrete visible cue described in the reasoning. An unsupported anomaly claim is rejected.

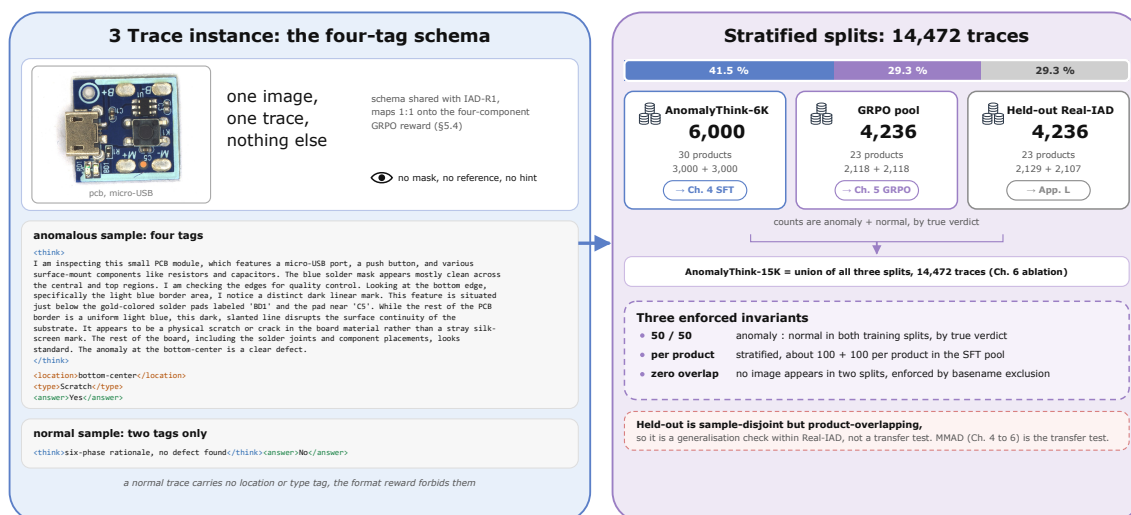
Generation engine. Gemini 2.5-Flash [18] is chosen for two reasons. First, it accepts multi-image inputs natively, which is critical for the reference-overlay pair. Second, batching up to ten samples per call keeps the cost of generating the full dataset small.

3.4. Generation-Time Quality Control

The three gates are the Phase 4 box of Figure 3.2. All of them run inside the same generation call. Quality control is applied *during* generation, inside the single Gemini call, rather than by a separate post-hoc reviewer model. The dataset uses *no second model* and applies *no post-hoc, score-ranked selection* across traces. The generator instead self-assesses each trace against a numeric rubric *within* that same call, as detailed below.

Prompt-level rules. The generation system prompt (§3.3, reproduced in Appendix B) instructs Gemini to emit the fixed tag structure of §3.2, to draw `<location>` and `<type>` values only from the controlled vocabularies, and to obey the six auto-reject rules listed above.

In-prompt acceptance filter. Beyond the hard auto-reject rules, the prompt instructs the generator to *silently* self-score each trace before returning it, on an eight-category rubric (visual grounding, evidence-before-conclusion, image specificity, single-image independence, auxiliary-input discipline, causal restraint, normal/anomalous decision quality, and output-structure correctness), each scored 0, 1 or 2 for a maximum of sixteen. A trace is returned only if it scores at least 13/16 (with visual grounding, evidence-before-conclusion,



Section 3 of the AnomalyThink construction: what one instance contains, and how the 14,472 traces are split.

Figure 3.3: What one training instance contains, and how the corpus is split. An instance is one image and one trace. An anomalous trace carries four tags and a normal trace carries two, because the format reward of \$5.2 forbids a `<location>` or `<type>` tag on a normal sample. The 14,472 traces are partitioned by product and by class into three disjoint splits.

single-image independence, output-structure correctness and normal/anomalous decision quality each required to score the full 2). Otherwise the model is instructed to rewrite the lowest-scoring categories and re-emit. The rubric scores are never output or logged: the entire filter runs inside the *single* generation call, performed by the same model that writes the trace.

Parse-level validation. Each batch response is validated as well-formed JSON matching the expected `{traces: [{image_id, reasoning}]}` schema (via a Pydantic model), and malformed batches trigger a retry. Traces that parse successfully enter the dataset directly. Listing 3.1 shows the shape of one such response. The trace is a single JSON string field, and the XML tags of §3.2 live inside that string, so the JSON layer only checks that every image in the batch came back with a string. The tag order inside the string is the generator’s own in-prompt rule (auto-reject rule 4), not a second parser. The element shown is a real record from the generation archive, shortened in the middle.

Listing 3.1: One element of a stored batch response. The reasoning trace is a JSON string that carries the XML tags. Real-IAD button_battery 0042, shortened.

```
{
  "traces": [
    {
      "image_id": "button_battery_NG_ZW_S0042..._C1_20230921165124",
      "reasoning": "<think>\nI am inspecting a button battery presented in a\nside-view image. [...] As I scan the surface, my attention is drawn to\na small, dark speck located slightly off-center. [...] This observation\nleads me to classify the defect as contamination.\n</think>\n<location>center</location>\n<type>Contamination</type>\n<answer>Yes</answer>",
      ...
    }
  ]
}
```

Crucially, the in-prompt self-assessment is applied *uniformly* to every trace, and is *not* used to rank-and-select a high-scoring subset. The headline **AnomalyThink-6K** is therefore the stratified SFT *partition* defined in §3.5, not a top-scored subset drawn from a larger pool. The 6K-beats-15K result of Chapter 6 is accordingly a *data-count* and *composition* effect under the same four-epoch schedule, since both sets pass through the identical in-prompt filter.

3.5. Dataset Assembly

The generated traces are partitioned into three disjoint splits, stratified by product and balanced by class: a 6K SFT split (**AnomalyThink-6K**), a 4K GRPO split (Chapter 5), and a 4K held-out split. The partition enforces three invariants: class balance (exactly 3,000 / 3,000 and 2,118 / 2,118 anomalous / normal in the two training

splits, and 2,129 / 2,107 in the held-out split), per-product stratification (so no single high-count product dominates), and zero image overlap across the splits. **AnomalyThink-15K** ($\approx 14.5K$) is their union, used as a single SFT set for the data-quantity ablation. Since that ablation is evaluated only on DS-MVTec and VisA, pooling the held-out split into it does not affect any reported held-out number.

Camera-angle filtering. All splits are restricted to the top-down `_C1_` view. Extending to the other four views is left as future work.

File format. Each split is stored in the LlamaFactory ShareGPT JSON format (for SFT) and a custom GRPO format. The exact schemas are in Appendix A.

3.6. Dataset Statistics

The split sizes and the three enforced invariants are shown in Figure 3.3. The full per-split composition, product coverage and class balance (every split is $\approx 50/50$ anomalous/normal, the SFT split spans all 30 Real-IAD products and the GRPO and held-out splits 23 each) is recorded in the datasheet of Appendix I. The traces themselves are short and tightly clustered: the mean length is ≈ 165 tokens (≈ 137 words), with a 5th–95th percentile of 142–187 tokens and a full range of 121–223 tokens, comfortably inside the 120–200-word generation target.

The controlled type vocabulary is respected in practice: the anomalous traces of the final corpus use 36 distinct `<type>` strings, variants of the controlled families rather than a generic “defect” token. §5.2 (Table 5.1, Figure 5.2) shows how the embedding-based type reward separates them.

Example: anomalous trace. Figure 3.4 shows one real anomalous trace from the SFT split, for a Real-IAD woodstick, next to the image it describes. The green cells are where the ground-truth mask lies, and the shaded sentence is where the trace commits to the defect. The four tags in the header are the ones the reward functions of Chapter 5 read, and the location tag agrees with the mask cell.

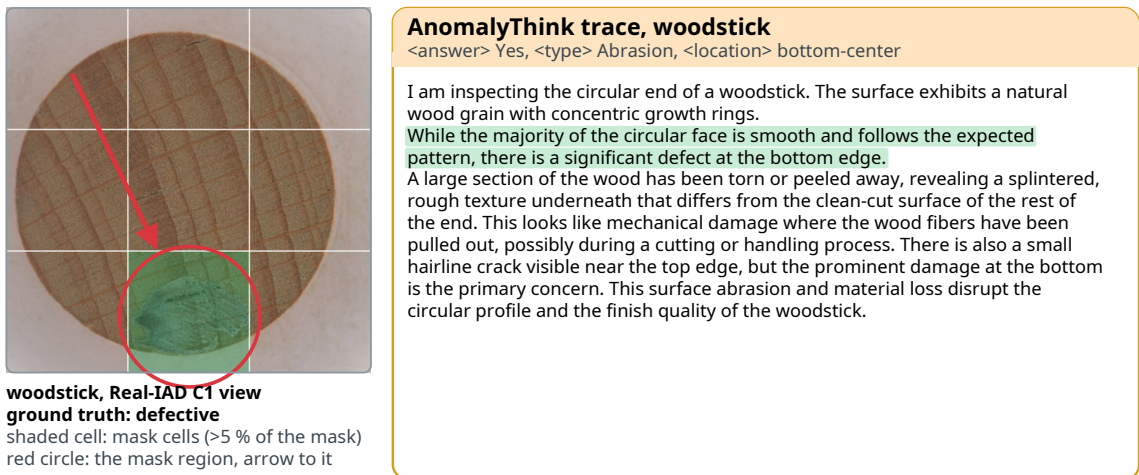


Figure 3.4: One anomalous **AnomalyThink** trace from the SFT split, Real-IAD woodstick 0028, an abrasion on the cut face. Left, the image with the 3×3 location grid, the ground-truth mask cell shaded green and the red circle enclosing the mask region, with the arrow pointing to it. Right, the trace in full with its four output tags in the header. The shaded sentence is where the trace commits to the defect. The same layout is used for the model outputs in Figure 6.10 and Appendix E.5.

Example: normal trace. Figure 3.5 shows a real normal trace from the same split, for a Real-IAD transistor, in the same layout. A normal trace carries no `<location>` or `<type>` tag, and the shaded sentence is where it commits to the verdict.

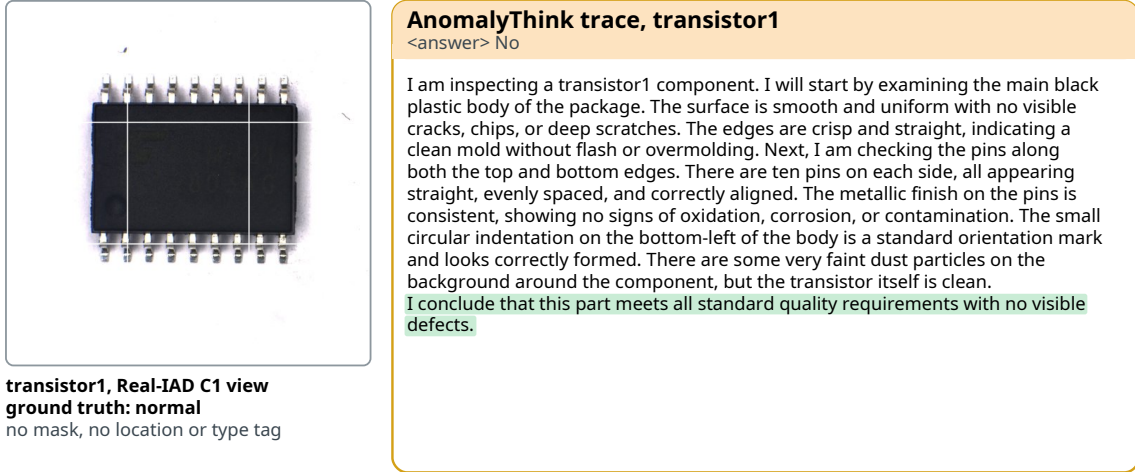


Figure 3.5: One normal **AnomalyThink** trace from the SFT split, Real-IAD transistor1 0234. Left, the image with the 3×3 location grid and no mask, because the sample is defect-free. Right, the trace in full with its single output tag in the header. The shaded sentence is where the trace commits to the verdict.

3.7. Positioning vs. Existing Reasoning-Trace Datasets

Two prior datasets are the natural reference points for **AnomalyThink**: **Anomaly-Instruct-125k** [75] and **Expert-AD** [39]. We differ from both in source data, output format, and intended downstream use. The comparison is summarised in Table 3.2.

Table 3.2: Positioning of **AnomalyThink** against the two reference reasoning-trace datasets. “Public release” refers to the dataset itself (not whether the underlying images are public). Expert-AD shares the same Real-IAD source, structured output and SFT+GRPO purpose, so the axes on which **AnomalyThink** is novel are its *open availability* and its *fully automated, disclosed* generation-and-quality-control pipeline (no human filtering, no second model).

	Anomaly-Instruct-125k [75]	Expert-AD [39]	AnomalyThink (ours)
Size	125K	~5.9K	~14.5K (union) / 6K (SFT split)
Image sources	Industrial+medical+in-the-wild+3D	Real-IAD only	Real-IAD only
Industrial subset	MVTec, VisA, etc. (not Real-IAD)	30 Real-IAD products	30 Real-IAD products
Output format	Multi-turn QA conversations	Structured CoT	Structured XML six-phase CoT
Designed for	Instruction-tuning (vanilla SFT)	SFT + GRPO	SFT + GRPO
Trace generator	GPT-4o	Qwen-VL-Max	Gemini 2.5-Flash
Privileged input	Bounding box on image	Mask region + coarse label + reference normal	Red mask overlay + reference normal (§3.3)
Quality control	In-context examples	Manual expert filtering	In-prompt auto-reject + schema validation, no second model (§3.4)
Public release	Yes	Gated, no access	Yes (this work)

Versus Anomaly-Instruct-125k. It is openly released and larger than ours (125K), but its industrial subset is drawn from MVTec and VisA, exactly the benchmarks we hold out for evaluation. Its multi-turn Q&A format also exposes no machine-parseable <location>, <type> or <answer> tags, so a conversation cannot be consumed directly by the structured four-component GRPO reward of \$5.2. The two are complementary: extending our six-phase template to its medical and in-the-wild splits is left as future work (§8.4).

Versus Expert-AD. It is the closest comparable: like **AnomalyThink** it is built on Real-IAD, uses the same four-tag structured output, and is purpose-built for SFT + GRPO. Its generation pipeline is described in full in the paper: a five-step process that prompts Qwen-VL-Max with the query image, a reference normal, and mask-derived location hints, followed by manual expert filtering. It differs from ours in three ways that matter in practice. First, availability: Expert-AD is behind a gated release whose access requests are not granted, so in practice only the trained IAD-R1 checkpoints can be used, whereas **AnomalyThink** is openly downloadable. Second, quality control: their curation relies on human experts filtering the generated traces, whereas ours is fully automated inside the generation call (in-prompt auto-reject rules and self-scoring, no second model and no manual pass). Third, scale and generator: **AnomalyThink** provides $\approx 14.5\text{K}$ traces from Gemini 2.5-Flash against Expert-AD’s 5.9K from Qwen-VL-Max. To the best of our knowledge **AnomalyThink** is therefore the first *openly available* Real-IAD-based reasoning-trace dataset that is structured for direct GRPO consumption and produced by a fully automated, disclosed pipeline, with its prompt (Appendix B), quality controls (§3.4), and split logic (§3.5) all released alongside the thesis.

4

Supervised Fine-Tuning

This chapter describes the SFT stage. In this stage the Qwen2.5-VL backbone is trained to emit the structured six-phase trace of Chapter 3. The roadmap is as follows. §4.1 gives the training objective, §4.2 the model family and the frozen-versus-unfrozen vision encoder choice, §4.3 the data formatting, §4.4 the distributed training infrastructure (DeepSpeed ZeRO-3 with CPU offload on two A6000s), §4.5 the full hyperparameters, §4.6 the four-factor ablation design (evaluated in Chapter 6), and §4.7 the evaluation protocol. Figure 1.1 situates this stage within the end-to-end pipeline. Figure 4.1 summarises the two training stages side by side: what each stage consumes, which blocks of the backbone receive gradient in each, what each optimises, and which checkpoint each produces.

4.1. Training Objective

SFT minimises the standard auto-regressive negative-log-likelihood on the trace tokens. For an example (I, y) , with test image I and gold trace $y = (y_1, \dots, y_T)$ tokenised with the Qwen2.5-VL tokenizer, the loss is

$$\mathcal{L}_{\text{SFT}}(\theta) = - \sum_{t=1}^T \log \pi_{\theta}(y_t \mid I, x_{\text{usr}}, y_{<t}), \quad (4.1)$$

where x_{usr} is the fixed user question and only the assistant tokens contribute to the loss (user tokens are masked). The model outputs *the entire trace, including all four tags*, so the supervision covers both the natural-language reasoning and the structured tags. Figure 4.1 sets this stage next to the GRPO stage of Chapter 5, so the two can be read side by side.

4.2. Model and Vision Encoder Treatment

Backbone. We use Qwen2.5-VL [4] at two sizes: Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct (open weights, Alibaba). Both share a three-component architecture: a window-attention ViT encoder, then a multimodal projector, then a Qwen LM trunk. The 3B model has 36 layers, 16 attention heads and a 2048 hidden dimension. The 7B model has 28 layers, 28 heads and a 3584 hidden dimension. Native dynamic resolution splits an image into a variable number of 14×14 patches, emitted as a variable-length sequence of visual tokens. We cap this budget at `image_max_pixels = 262,144` ($\approx 512 \times 512$) and apply the same cap at evaluation for train/test consistency.

Why full fine-tuning, not LoRA. We follow the recommendation of Li et al. [39], who report that LoRA [23] fails to convey enough information for long reasoning traces: the `<think>` block collapses to short, generic templates because the low-rank adapter cannot represent the six-phase diversity. Our own early LoRA experiments likewise showed no improvement over the base model. We therefore perform *full* SFT of the LM trunk, and treat the visual encoder as an ablation factor.

Frozen vs. unfrozen vision encoder. The pretrained Qwen2.5-VL visual encoders are strong feature extractors for natural images, but industrial images are out-of-distribution (uniform backgrounds, macro shots, unfamiliar materials). In the *frozen* setting only the LM trunk and the projector are updated and the ViT stays

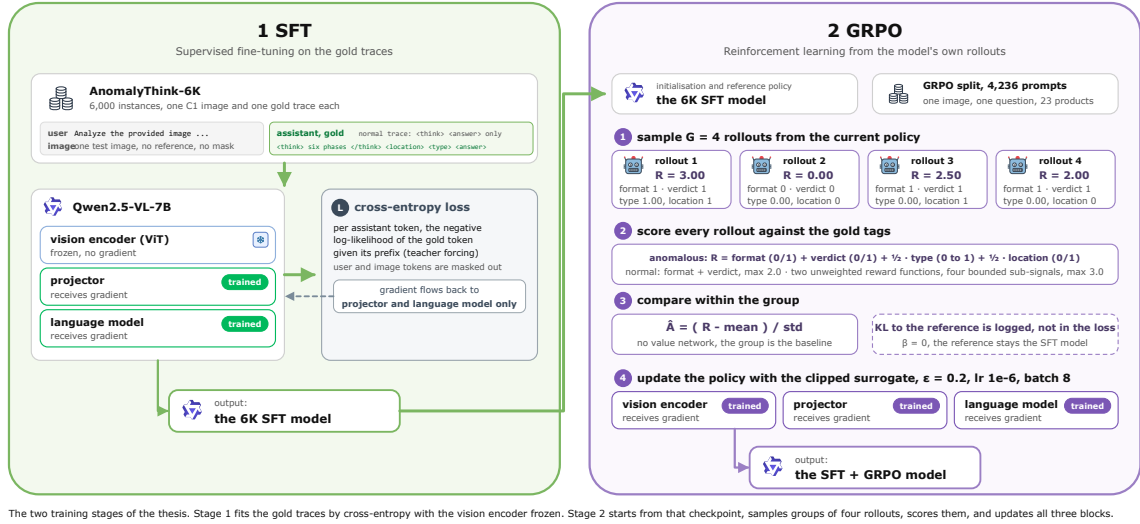


Figure 4.1: The two training stages side by side. (A) Supervised fine-tuning learns the gold six-phase trace by cross entropy on the assistant tokens, with the vision encoder frozen. (B) GRPO starts from the checkpoint that (A) produces and learns from the model's own sampled rollouts, and here all three blocks receive gradient. Panel (B) draws one real group of four rollouts of the SFT+GRPO policy on a Real-IAD button_battery item, with the logged sub-scores, so the four rewards follow the formula in the panel.

fixed, which preserves the pretrained representation and avoids catastrophic forgetting or overfitting. In the *unfrozen* setting the ViT is updated end-to-end. It adapts to industrial statistics but risks overfitting on the small (6K–15K) training set. We treat the choice as an ablation factor. As reported in Chapter 6, frozen wins at both 6K and 15K, on VisA at 15K and on DS-MVTec at 6K (§6.3, §7.1).

4.3. Data Format and Prompt

Each example uses a multimodal ShareGPT-style conversation (the format accepted by LlamaFactory and the Hugging Face trl SFT trainer). The setup is *zero-shot* by construction: a single test image, no reference image, two turns, and *no* system turn at training time:

User turn. The test image followed by the fixed product-conditioned question:

```
<image>
Analyze the provided image of the {product_name}. Determine if there
are any anomalies present. If an anomaly is detected, specify its type
and location, and provide a detailed reasoning for your conclusion.
```

Assistant turn. The gold trace as produced by Gemini under the structured generation prompt: the <think> block followed by (for anomalies) the <location> and <type> tags and the final <answer> tag.

User tokens are masked from the loss, so only the assistant tokens contribute. Images are loaded lazily to avoid materialising the entire image set in RAM.

Prompt consistency (training to evaluation). The evaluation harness (§4.7) re-uses this exact user question, adding only a one-line system message (“Please answer by yes or no”) to anchor the verdict. This is the *train-prompt* evaluation mode used for every headline number in Chapter 6. Keeping the prompt identical between training and inference measures the model's capability, not its instruction-following robustness, and avoids prompt-drift regressions.

4.4. Training Infrastructure

Hardware. All headline experiments run on a single node with two NVIDIA RTX A6000 GPUs (48 GB GDDR6 each) and ample system RAM, with shared NVMe storage. The 3B unfrozen configuration would fit on a sin-

gle GPU at batch size 2 with gradient checkpointing. We still ran it on both GPUs at per-device batch size 4, like every other cell. The 7B unfrozen configuration does *not* fit on either single GPU memory or even both GPUs without offloading, motivating the use of DeepSpeed ZeRO-3 [55, 57] with CPU offload. Every SFT run reported in this thesis uses both GPUs under that configuration.

DeepSpeed ZeRO-3 with CPU offload. We partition gradients, optimizer states, and 16-bit parameters across the two GPUs and offload optimizer states and parameters to pinned CPU memory. The relevant configuration values are reproduced in Listing 4.1. The full JSON is Listing A.1 in Appendix A.5. Mixed precision is bf16, which we found more stable than fp16 for Qwen2.5-VL-7B and avoids the loss-scalar tuning otherwise required.

Listing 4.1: Key DeepSpeed ZeRO-3 settings. The full file is Listing A.1.

```
"zero_optimization": {
  "stage": 3,
  "offload_optimizer": {"device": "cpu", "pin_memory": true},
  "offload_param": {"device": "cpu", "pin_memory": true},
  "overlap_comm": true,
  "contiguous_gradients": true,
  "stage3_gather_16bit_weights_on_model_save": true
},
"bf16": {"enabled": "auto"},
"fp16": {"enabled": "auto"}
```

Wall-clock cost. A 7B-frozen-6K run for four epochs takes ≈ 6.9 hours across the two A6000s. A 7B-unfrozen-15K run takes ≈ 12.2 hours, and the 3B-frozen-6K run takes ≈ 1.6 hours over four epochs. The Arm-C run of Chapter 6 takes ≈ 5.1 hours for four epochs, on the same two GPUs but with a different batch geometry. The 7B-unfrozen-6K control of Table 6.3 takes ≈ 7.7 hours for four epochs, 12 % more than the frozen run at the same number of steps.

4.5. Hyperparameters

The full SFT hyperparameter table is reported in Appendix A. The headline values are summarised below. They follow standard full-fine-tuning defaults, refined in informal preliminary checks and validated by the single-run 6K-scale results of Chapter 6.

Table 4.1: Hyperparameters used in the SFT runs.

Hyperparameter	Frozen-ViT runs	Unfrozen-ViT runs
Optimizer	AdamW [44]	AdamW
β_1, β_2	0.9, 0.999	0.9, 0.999
Learning rate	1×10^{-5}	1×10^{-6}
LR schedule	Cosine, 20-step (6K) / 50-step (15K) warmup	Cosine, 20-step (6K control) / 50-step (15K) warmup
Weight decay	0	0
Effective batch size	32	32
Sequence length cap	12,144 tokens	12,144 tokens
Mixed precision	bf16	bf16
Gradient checkpointing	Enabled	Enabled
Gradient clipping	1.0	1.0
Epochs	4	4
Random seed	42	42

Per-device batch size and gradient accumulation vary by cell (4/4 for the 6K and all 3B cells, 16/1 for the 7B-15K cells, and 8/2 for the Arm-C run of Chapter 6), but always combine to an effective batch size of 32 on two GPUs. The per-cell values are listed in Appendix A.3.

Why a low, encoder-dependent learning rate. We train the language-model trunk (and, in the frozen cells, the projector) at 1×10^{-5} . When the visual encoder is *unfrozen* we drop the learning rate by an order of magnitude, to 1×10^{-6} . A 7B-unfrozen run at the frozen rate (1×10^{-5}) on the 6K dataset peaks at 75.97 % DS-MVTec at epoch 1 and then *degrades* with further training, staying 4 to 9 points behind the frozen run at every epoch

(Table 6.3, §6.3). The smaller step is needed to keep the pretrained ViT stable. The frozen cells, which never update the ViT, train robustly at 1×10^{-5} .

Learning-rate schedule. All runs use a cosine schedule with a short warmup (20 steps for the 6K dataset, 50 for 15K), the standard default for full fine-tuning at this scale.

4.6. Ablation Design

We organise the SFT experiments as a controlled four-factor grid:

- **Model size** $\in \{3B, 7B\}$,
- **Vision encoder** $\in \{\text{frozen}, \text{unfrozen}\}$,
- **Dataset size** $\in \{6K, 15K\}$, and
- **Epoch** $\in \{1, 2, 3, 4\}$.

A full factorial would produce $2 \times 2 \times 2 \times 4 = 32$ evaluation cells per benchmark. We evaluate the *unfrozen* encoder at 15K at both sizes, and at 6K only at 7B and only at the frozen cells’ learning rate, as the control that isolates the freezing flag (Table 6.3). The 15K-unfrozen cells already show that unfreezing yields no DS-MVTec gain and degrades VisA (§6.3), consistent with the frozen-ViT recommendation of Li et al. [39], so the 3B-unfrozen-6K cell is excluded from the formal evaluation grid. This leaves **6 configurations** plus the control (28 epoch-cells per benchmark). To keep wall-clock cost tractable we run a single seed per cell but average over the full evaluation set (1,670 DS-MVTec samples and 2,141 VisA samples). Per-cell results are tabulated in Chapter 6 and Appendix C.

The grid is chosen so that one factor moves at a time between two adjacent cells, enabling “effect-of-X” comparisons (Table 4.2). The one exception, the learning rate that moves with the freezing flag, is closed by the matched-rate control row. The corresponding results are reported in §6.3.

Table 4.2: Each comparison changes one design factor and holds the rest fixed. The vision-encoder rows use a tenfold lower learning rate in the unfrozen cell, because the encoder degrades within an epoch at the frozen cells’ rate (§4.5, §6.3), so that comparison is between the two recipes as they can be trained. The matched-rate row holds the learning rate at the frozen value and changes freezing alone. The dataset-size comparison is made at the matched epoch-3 checkpoint of both cells. The model-size and vision-encoder comparisons are made at each cell’s best epoch, as reported in Table 6.3. The tuned-rate freezing comparison is made at the 15K data point, the matched-rate one at 6K.

Effect of	Cell A	Cell B
Model size	3B-frozen-6K	7B-frozen-6K
Vision encoder	7B-frozen-15K	7B-unfrozen-15K
Vision encoder, matched rate	7B-frozen-6K	7B-unfrozen-6K at 1×10^{-5}
Dataset size	7B-frozen-6K-ep3	7B-frozen-15K-ep3

4.7. Evaluation Protocol

Inference inputs. At test time the trained model receives a single test image and the same user-turn question as during training (zero-shot, no reference image), preceded by the one-line system message “Please answer by yes or no”. The image is processed at the training resolution cap, `image_max_pixels = 262,144`.

Decoding. We use greedy decoding (`do_sample=False`) with maximum 1024 new tokens. Greedy decoding gives reproducible numbers, and in informal comparisons sampling-based decoding did not materially change balanced accuracy.

Answer extraction. The model’s output is parsed by a deterministic regex that extracts `<think>`, `<location>`, `<type>`, `<answer>` in that order. Failure to emit a parseable `<answer>` tag is counted as an incorrect prediction. Empirically less than 0.5 % of post-SFT outputs fail this parse.

Metrics. We report:

- **Balanced accuracy** $\text{bal-acc} = \frac{1}{2}(\text{TPR} + \text{TNR})$, where $\text{TPR} = \frac{TP}{TP+FN}$ and $\text{TNR} = \frac{TN}{TN+FP}$. This is the primary

metric throughout because IAD class distributions vary across products and a model that “always says yes” otherwise scores deceptively high accuracy.

- Plain accuracy, precision, recall, F1 (for comparison with the literature).
- Per-product balanced accuracy on the 15 DS-MVTec products (Appendix C) and the 12 VisA products (Appendix D).

All percentages in Chapter 6 are reported to two decimal places.

5

GRPO Reinforcement Learning

Post-SFT RL stage. We take the best SFT checkpoint of Chapter 4 and refine it with Group Relative Policy Optimisation (GRPO, §2.4.2). This chapter covers the algorithm as implemented (§5.1), the two reward functions and their unweighted aggregation (§5.2), data preparation (§5.3), the training loop and hyperparameters (§5.4), and implementation details and run statistics (§5.5). Panel (B) of Figure 4.1 on page 22 summarises the stage as implemented, next to the SFT stage it starts from.

5.1. GRPO as Implemented

GRPO replaces PPO’s value network with a Monte-Carlo group baseline. For a given prompt q , a group of G completions $\{o_1, \dots, o_G\}$ is sampled from the rollout policy $\pi_{\theta_{\text{old}}}$. Then we score each completion with $R(o_i; q)$ and form the group-relative advantage

$$\hat{A}_{i,t} = \frac{R(o_i; q) - \mu_G}{\sigma_G + 10^{-4}}, \quad \mu_G = \frac{1}{G} \sum_{j=1}^G R(o_j; q), \quad \sigma_G = \sqrt{\frac{1}{G-1} \sum_{j=1}^G (R(o_j; q) - \mu_G)^2}, \quad (5.1)$$

broadcast as a constant over the tokens of o_i . The policy is then updated by the clipped surrogate of (2.3):

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q, \{o_i\}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min(\rho_{i,t} \hat{A}_{i,t}, \text{clip}(\rho_{i,t}, 1 - \varepsilon, 1 + \varepsilon) \hat{A}_{i,t}) - \beta \text{KL}_t^{k_3} \right\} \right], \quad (5.2)$$

with token-level importance ratio $\rho_{i,t}(\theta) = \pi_{\theta}(o_{i,t} \mid q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$ and the Schulman k_3 KL term $\text{KL}_t^{k_3}$ of (2.4). It is placed in the loss (weighted by β) rather than a reward bonus.

Hyperparameter choices. We use $G = 4$ rollouts per prompt. This trades a less-noisy baseline against the cost of full Qwen2.5-VL-7B generations of up to 512 completion tokens on a 4096-token multimodal prompt (Table 5.2), on two A6000s. Clipping $\varepsilon = 0.2$ (the PPO default). KL coefficient $\beta = 0$. The k_3 KL of (2.4) is *monitored* (stays small) but *not* added to the loss. Reference policy π_{ref} fixed to the SFT checkpoint. It is never updated.

Algorithm: GRPO training loop (single iteration)

Require: SFT checkpoint θ_0 , dataset $\mathcal{D}_{\text{grpo}}$, reward R , group size G , clip ε , KL coef. β , steps T

1. $\theta_{\text{ref}} \leftarrow \theta_0, \theta \leftarrow \theta_0$
2. **for** $t = 1, \dots, T$ **do**
3. Sample prompt $q \sim \mathcal{D}_{\text{grpo}}$ (test image + question)
4. Sample group $\{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q)$ with $\theta_{\text{old}} = \theta$
5. Score each completion: $r_i \leftarrow R(o_i; q)$ // four-component, §5.2
6. Compute group statistics μ_G, σ_G , then advantages $\hat{A}_{i,t}$ via (5.1)
7. Compute surrogate $\mathcal{J}_{\text{GRPO}}$ via (5.2)
8. $\theta \leftarrow \theta + \eta \nabla_{\theta} \mathcal{J}_{\text{GRPO}}$ // a single AdamW step
9. **end for**

Figure 5.1: GRPO training loop, single iteration.

5.2. Reward Function

Following the IAD-R1 reward design [39] our trainer is built on, the per-rollout reward is the *unweighted* sum of two reward functions: a *consistency* (format) reward and an *accuracy* reward. This makes it consist of four sub-signals (format, verdict, type, location):

$$R(o; q) = R_{\text{cons}}(o; q) + R_{\text{acc}}(o; q), \quad R_{\text{cons}} \in [0, 1], \quad R_{\text{acc}} \in [0, 2], \quad (5.3)$$

So the maximum reward is 3.0. No per-component weight tuning: the two functions are summed with equal unit weight, exactly as in the IAD-R1 code we adapt. Each is detailed below.

5.2.1. Consistency (format) reward R_{cons}

$R_{\text{cons}} \in \{0, 1\}$. A binary reward on the full match of the XML schema, and there is a difference between the anomalous and normal output schemas. For an anomalous trace the output must contain `<think>`, `<location>`, `<type>`, `<answer>` in order. For a normal trace it must contain `<think>` then `<answer>` and must *not* contain a `<location>` or `<type>` tag: two negative look-aheads forbid both (Appendix E.1). The schema enforced is selected from the *gold* verdict, not the model’s own. For this binary reward a missing, misordered or forbidden tag scores 0 and the intact schema scores 1. Text before the first tag or after the last is ignored by the pattern (Appendix E.1). No partial credit is given. This catches the two failure modes seen in mid-training rollouts: free-form outputs with no XML, and normal samples that emit loc/type tags.

5.2.2. Accuracy reward R_{acc}

$R_{\text{acc}} \in [0, 2]$. This reward rewards the final verdict, the type of anomaly, and the location. We compare the lower-cased, stripped contents of the `<answer>` tag against the gold verdict exactly (the permissive alias table belongs to the *evaluation* harness of Appendix J, not to the reward), then compute:

- *Normal samples* (gt = no): $R_{\text{acc}} = 1$ if the predicted verdict is “no”, else 0.
- *Anomalous samples* (gt = yes): $R_{\text{acc}} = \frac{1}{2}(s_{\text{typ}} + s_{\text{loc}})$, plus 1.0 if the predicted verdict is “yes”. So a unit reward for the correct verdict, plus the *average* of a type and a location sub-score. The type/location credit is granted regardless of the verdict.

The *type* sub-score $s_{\text{typ}} \in [0, 1]$ is the semantic similarity between the predicted and ground-truth `<type>` strings. It uses the Nomic embedding model [48] (nomic-embed-text-v2-moe, served on CPU), and maps the cosine similarity to fixed bins: $\geq 0.90 \rightarrow 1.0$, $\geq 0.80 \rightarrow 0.9$, $\geq 0.70 \rightarrow 0.7$, $\geq 0.55 \rightarrow 0.5$, $\geq 0.40 \rightarrow 0.2$, else 0. These boundaries are fixed heuristic bins, chosen so that exact matches, close synonyms and same-category terms receive descending credit. We did not run a sensitivity analysis on the exact cut-points, which we leave to future work. It is 0 unless both prediction and gold emit a `<type>` tag. The *location* sub-score $s_{\text{loc}} \in \{0, 1\}$ is 1 if and only if the predicted `<location>` maps to the same cell of a 3×3 region grid as gold. Else 0. The mapping reads the free-text tag by substring: “left” or “right” sets the column and “top”, “upper”, “bottom” or “lower” the row, “left” taking precedence over “right” and “top” over “bottom”, and a tag with none of these words maps to the centre. It therefore represents one cell, so a multi-cell answer such as “top-right, bottom-left”, which the generation prompt allows, is scored as top-left, and a descriptive location without a grid word is scored as the centre (Appendix E). It is 0 unless both emit a `<location>` tag.

Whether an embedding similarity is a meaningful reward depends on how the type vocabulary sits in that embedding space, and the final corpus makes the question small: its anomalous traces use 36 distinct <type> strings, 28 once the reward has lower-cased them. Table 5.1 shows what the reward sees for one gold type, computed through the reward’s own code path (the `search_query` prefix and masked mean pooling of the embedding server, Appendix E). Spelling and phrasing variants of the gold string score 0.76 to 0.92 and receive 0.7 to 1.0 of the type credit. The closest other family, *crack*, scores 0.44 and receives 0.2, and every other family scores below 0.40 and receives nothing, and so does an unrelated word. The same holds for the other families: *missing part* and *missing component* score 0.92 and 0.81 against *missing parts*, *contaminant* 0.81 against *contamination*, *pits* 0.90 against *pit*, while all 45 pairs among the ten most frequent distinct families score between 0.11 and 0.38, below every bin. The binned reward therefore gives graded credit for the gold vocabulary word and its variants and none for a wrong family, which is the behaviour the type term relies on. Its limit is the other side of the same property: a true synonym phrased differently (*scuff mark* or *abrasion* for a scratch, *stain* for contamination) also receives nothing, so the term rewards the controlled vocabulary word rather than the meaning. Figure E.1 in Appendix E gives the full matrix for the twelve most frequent strings: the only pair above the lowest bin is the singular and plural of one family (*foreign object* and *foreign objects*, 0.96), every other pair sits between 0.11 and 0.38. Figure 5.2 shows the vocabulary as a t-SNE projection through the reward’s own embedding path, the 28 distinct lower-cased corpus strings plus added variants: phrasing variants of one family cluster, and the pairs that share a modifier word (*surface*, *mark*, *area*) are the ones that approach the lowest bin from a different family.

Table 5.1: Cosine similarity and binned type score against the gold string *scratch*, computed with the reward’s embedding path. Variants of the gold word are credited, other defect families and an unrelated word are not.

Predicted <type>	Cosine	Type score
a scratch	0.92	1.0
scratches	0.84	0.9
surface scratch	0.83	0.9
scratched surface	0.76	0.7
crack	0.44	0.2
scuff mark	0.37	0
chip	0.31	0
abrasion	0.23	0
contamination	0.22	0
pit	0.21	0
banana	0.19	0
missing parts	0.15	0

5.2.3. Total reward and run variants

Summed without weighting (Eq. 5.3), this gives four effective sub-signals. Maximum contributions: format = 1, verdict = 1, type = 0.5, location = 0.5 (total 3.0). The verdict and the XML each carry unit weight. Type and location are auxiliary, at half weight through the average inside R_{acc} . All sub-scores are bounded, so the group-normalised advantage of (5.1) stays well-scaled. The SFT+GRPO run uses exactly these two reward functions. The GRPO-on-C and estimator-ablation probes (§6.8) additionally register a third *reasoning* reward scored by an external judge server (Appendix E). That server was not running in the reported probes, so the term returned its neutral fallback of 0.5 for every scored rollout (logged range 0.484–0.500). Being constant within a group it cancels in (5.1), so those runs optimise the same effective signal as the production configuration, offset by a constant. The Gemini-judge variant adds a Gemini-scored reasoning reward. All are combined in the same unweighted way.

5.3. Data Preparation

Source. The *complementary* 4K GRPO split (cf. §3.5) is drawn from the same trace dataset as SFT. There is no overlap with the SFT training set. The split covers 23 of the 30 Real-IAD products. The other seven products had too few anomalous C1 captures to fill a 50/50-balanced GRPO split once the SFT pool was allocated first, so they appear only in SFT (Appendix A). We use these 4,236 prompts, stratified to 50/50 anomalous/normal both overall and within each product (the per-product totals are not equalised and range from 8 to 302). As

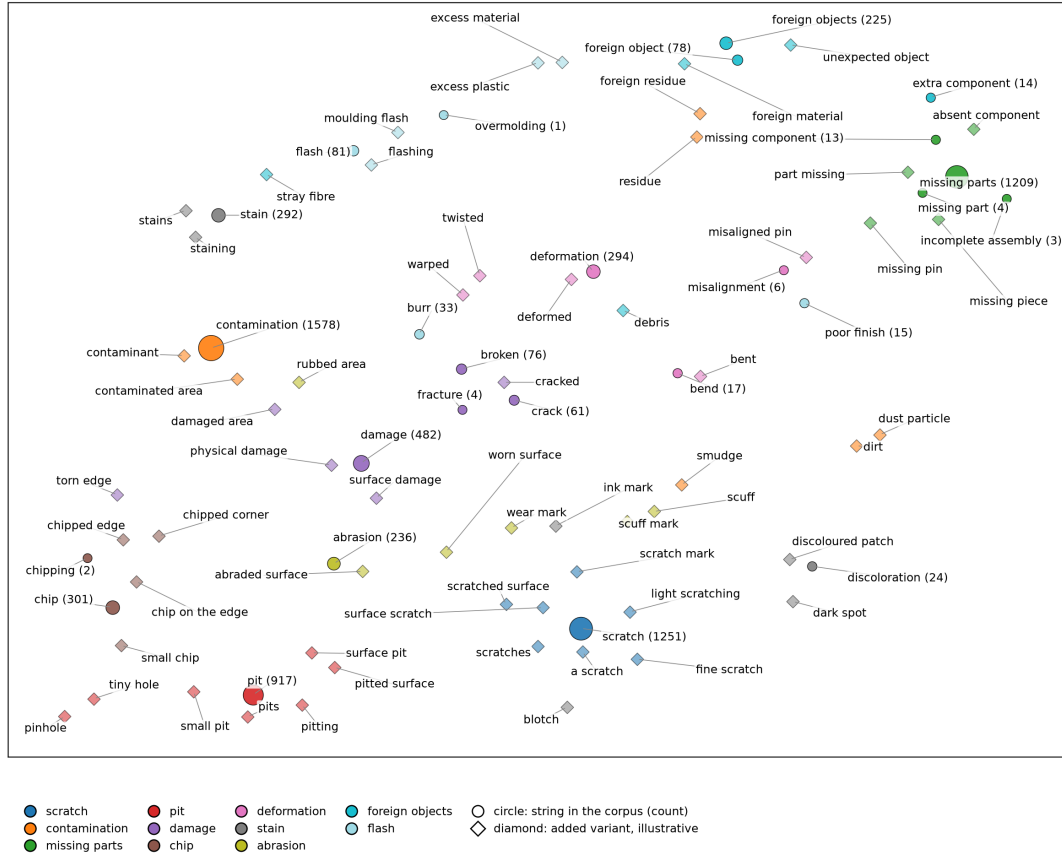


Figure 5.2: Two-dimensional t-SNE projection of <type> strings embedded exactly as the type reward embeds them (nomic-embed-text-v2-moe, search_query prefix, masked mean pooling, one string at a time, Appendix E). Circles are the 28 distinct lower-cased strings of the final corpus, sized by their count. Diamonds are 57 added variants a model might emit, included for illustration only. Colour is the defect family the string belongs to. Variants of one family sit together, and the reward credits them: *a scratch*, *scratches* and *surface scratch* all score above 0.80 against *scratch*. Families whose strings share a modifier word also drift towards each other, *surface scratch* against *surface damage* scores 0.61 and *scratch mark* against *scuff mark* 0.67, which is the lexical side of the same embedding. The distances in the projection are not cosines. The cosines are in Table 5.1 and Figure E.1.

in §3.5, we focus on the top-down C1 camera angle.

Image input. The same *zero-shot, single-image* protocol as SFT (Chapter 4): one test image, no reference image (trainer flag `single_img = 1`). The dynamic-resolution pixel budget (Table 5.2) downscales each image if needed to bound the number of visual tokens the encoder produces. No system message (`use_system_prompt = false`). The user turn is the fixed question reproduced verbatim in Appendix B (“...Are there any defects in the query image?”). The single-image format keeps the per-rollout cost low, one image-token sequence per prompt. That matters when every training step generates G full completions.

5.4. Training Loop and Hyperparameters

Table 5.2: Hyperparameters used in the SFT+GRPO run.

Hyperparameter	Symbol	Value
Base policy (initial θ_0)	—	best 7B-frozen-6K SFT ckpt-564
Reference policy π_{ref}	—	same, fixed throughout
Group size	G	4
Clipping range	ϵ	0.2
KL coefficient	β	0 (monitored, not added to loss)
Learning rate	η	1×10^{-6}
LR schedule	—	linear
Optimizer	—	AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$
Weight decay	—	0
Per-device batch size	—	1
Gradient accumulation	—	4
Effective batch size	—	8 (2 GPUs)
Max prompt length	—	4096 tokens
Max completion length	—	512 tokens
Max image pixels	—	480,000 (single test image, dynamic res.)
Max training steps	T	1,060 (2 epochs over 4,236 prompts)
Save every	—	530 steps (end of each epoch)
Checkpoint evaluation	—	post hoc on DS-MVTec/VisA (§4.7)
Mixed precision	—	bf16
Random seed	—	42

Why $G = 4$. Larger group sizes give a less noisy advantage estimate and are standard in DeepSeekMath ($G = 64$). But on a 7B VLM with dynamic-resolution image inputs (capped at the ≈ 0.48 MP `max_pixels` budget of §5.3) and CPU-offloaded ZeRO-3, each rollout costs on the order of 20–30 seconds. So we use $G = 4$, matching the group size in the IAD-R1 configuration our trainer builds on [39]: enough rollouts for a stable group baseline while staying within the per-step memory and wall-clock budget per GPU.

Why $\eta = 1 \times 10^{-6}$. GRPO is far more sensitive to learning rate than SFT. At higher rates the policy drifts from the reference rapidly and the reward signal becomes unstable. At $\eta = 1 \times 10^{-6}$ the monitored KL stays small through epoch 1 while reward climbs. It roughly doubles in epoch 2, alongside the checkpoint regression reported below. This low rate is standard for GRPO-style RL post-training, where the policy is held close to its reference by design [11, 66].

5.5. Implementation Details and Production-Run Statistics

Framework. Hugging Face’s `trl` library’s GRPO trainer. We modified it to support the custom four-component reward function and the Nomic embedding-based type similarity, building on the IAD-R1 implementation [39] that our code is based on.

Nomic embedding on CPU. The Nomic model (`nomic-embed-text-v2-moe`, ≈ 475 M parameters) is loaded in float32, adding approximately 1.9 GB to host RAM. We keep it on CPU to leave VRAM for the rollouts. The controlled vocabulary’s defect-type embeddings are pre-computed once at trainer start. Only the predicted type string is embedded on each rollout. The per-step latency overhead is < 50 ms.

Image lazy-loading. Images are loaded on the fly via PIL. The first rollout of a given image caches it for the duration of the group, then frees it from memory.

Run statistics. The SFT+GRPO run, the second of two identical launches of this configuration (§7.7), processed 2,118 anomaly + 2,118 normal samples (4,236 total) over 2 epochs (1,060 update steps). A checkpoint is saved at the end of each epoch (every 530 steps). Key numbers from the training log:

- *Mean combined reward* (the unweighted sum of §5.2, max 3.0): rose from 1.66 (step 10) to 2.22 (step 100), then stabilised at 2.13 (step 530).

- *Accuracy reward* (the bundled verdict, type and location signal of §5.2): from 0.875 to 1.25, a +42.9% relative improvement.
- *Format consistency reward*: from 0.781 to 0.875 (+12.0%).
- *Mean KL divergence*: through epoch 1 (up to ckpt-530) it stays bounded in [0.0, 0.117], exceeding 0.095 on only 13 of those 530 steps. It climbs further in epoch 2, consistent with the slight epoch-2 regression below. At the selected checkpoint the policy never moves dangerously far from the SFT reference.
- *Completion length*: stayed in the ~153–186-token band (mean ~167), a slow drift from about 175 to 163 tokens, a negligible length collapse.
- *Wall-clock cost*: ≈ 11.3 hours to the headline ckpt-530 (end of epoch 1). The full two-epoch run took ≈ 24.7 hours on $2 \times A6000$ with ZeRO-3 CPU offload.

Listing 5.1 shows how one rollout’s reward decomposes under the bins of Appendix E, followed by the batch-level quantities the trainer actually logs at every step (here step 319 of the SFT+GRPO run). Per-rollout decompositions are not logged. The first two lines are an illustration with the bin values, not a logged record.

Listing 5.1: Reward decomposition of one anomalous rollout, illustrated with the bin values of Appendix E, and the per-step batch means the trainer logs (SFT+GRPO run, step 319). The trainer sums the two reward functions unweighted.

```
R = 2.85 = consistency(1.0) + accuracy(1.85) # one anomalous rollout, illustration
    accuracy 1.85 = verdict(1.0) + (type 0.7 + location 1.0)/2 # type: cosine in [0.70, 0.80) -> 0.7
step 319 (logged): reward 2.266 accuracy_reward 1.297 consistency_reward 0.969
                  reward_std 0.192 kl 0.059 epoch 0.60
```

Checkpoint selection. We evaluate both saved checkpoints (end of epoch 1 and epoch 2) on the DS-MVTEC and VisA test sets and pick the highest balanced accuracy. For this run it is checkpoint-530 (end of epoch 1), which is best on both (DS-MVTEC 82.73, VisA 70.39). checkpoint-1060 (end of epoch 2) underperforms it by 0.70 pp on DS-MVTEC (and 0.53 pp on VisA). That suggests mild over-optimisation, discussed in §7.4.

5.6. Iterative SFT $\leftrightarrow$ RL: A Self-Distillation Cycle

The single SFT $\rightarrow$ GRPO pass of §5.4 treats the GRPO policy as the end product. Deng et al. [14] instead *iterate* the two stages. The RL-improved policy generates fresh reasoning traces. These are filtered for correctness and fed into a new SFT round. Then RL is run again. The intuition is self-distillation: once the policy reasons better than the original teacher on a subset of inputs, its own outputs become a higher-quality SFT dataset than the teacher’s. This is not guaranteed to hold, however: language models are limited at improving their own reasoning without external feedback [27], so a pure self-distillation loop may plateau below what an external teacher can supply, a possibility the teacher-intervention arms of §5.8 test directly. We implemented one such cycle, referred to as **iter-2**. It tests directly whether the loop that helps general multimodal reasoning also helps the narrow IAD task. We report it as a controlled experiment, not the headline recipe, for reasons the results of §6.5 make clear.

Procedure. Starting from the SFT+GRPO checkpoint (ckpt-530), we

1. roll out completions on the GRPO prompt set with the iter-2 policy,
2. apply a *rollout-and-filter* (rejection-sampling) step that keeps only the trajectories whose `<answer>` verdict matches the gold label *and* whose XML structure passes the format regex of §5.2, yielding 4,026 self-generated traces,
3. *reset to the Qwen2.5-VL-7B base model* and run a fresh SFT on these filtered self-traces (frozen ViT, $\eta = 10^{-5}$, batch 4, 4 epochs, identical to the SFT recipe of Chapter 4), with checkpoints saved at the end of epoch 3 (ckpt-378) and epoch 4 (ckpt-504), and
4. run GRPO again from the iter-2 SFT checkpoint, this time for 2 epochs with the same reward function and `num_generations=4`, saving every 265 steps (ckpt-265 through ckpt-1060, i.e. epochs 0.5–2.0).

The full sequence base $\rightarrow$ SFT₀ $\rightarrow$ GRPO₀ $\rightarrow$ rollout-and-filter $\rightarrow$ base $\rightarrow$ SFT₁ $\rightarrow$ GRPO₁ is the IAD instantiation of the OpenVLThinker self-improvement loop [14]. The rejection-sampling filter plays the role of the verifier.

Two variants. We ran the cycle twice. *iter-2 v1* used the first rollout-and-filter pass. *iter-2 v2* tightened

the filter (stricter format matching and de-duplication of near-identical traces) before the second SFT. Both are reported in §6.5. The takeaway, previewed: the iterative cycle improves over its own intermediate iter-2 SFT checkpoint, but does *not* surpass the simpler single-stage GRPO of §5.4. We therefore retain the single-stage SFT+GRPO checkpoint (ckpt-530) as the strongest *GRPO* checkpoint. Whether any GRPO checkpoint should be the deployed model, as opposed to a well-curated SFT checkpoint, is revisited in §6.8.

5.7. Gaussian-Advantage GRPO (G²RPO)

We evaluate two modifications to the vanilla advantage estimator of (5.1). The first is the mean-centred variant of **Dr.GRPO** [43]: it keeps the group mean but omits the σ_G division (i.e. $\hat{A}_{i,t} = R(o_i; q) - \mu_G$), so an advantage keeps the scale of the reward gap that produced it. The published method also removes the per-response length normalisation of the loss. Our arm changes the advantage only, hence “variant”. The second, which we call G²RPO, was introduced by OpenVLThinker v2 [26]. The vanilla GRPO advantage in (5.1) normalises group-relative rewards by their empirical mean and standard deviation. With $G=4$ rollouts and a reward taking only a few discrete values in $[0, 3]$ (format, verdict and location are binary, the type sub-signal is binned), it is common for three of the four rollouts to share a value while the fourth differs by a small margin. The z -score is bounded, at most $(G-1)/\sqrt{G} = 1.5$ in magnitude for $G=4$ with the sample standard deviation of (5.1), so it cannot explode. Its weakness is the reverse. The standardisation is scale-free, so a rollout that beats its three siblings by 0.05 receives the same full-size advantage of 1.5 as one that beats them by 2.0, and that lone outlier receives a larger advantage than any rollout in an evenly spread group ($\pm 1.16, \pm 0.39$). The gradient cannot tell a trivial distinction from a decisive one, and on a quantised reward such near-ties are the common case.

The modification. G²RPO replaces the z -score with a non-parametric rank-to-quantile transform onto the standard normal. For each group of G rewards $\{r_1, \dots, r_G\}$, let $\text{rank}(r_i)$ denote the rank of r_i among the group (with ties handled by averaging, see below). The G²RPO advantage is

$$\hat{A}_i^{\text{G}^2\text{RPO}} = \Phi^{-1}\left(\frac{\text{rank}(r_i) - 0.5}{G}\right) = \sqrt{2} \operatorname{erfinv}\left(\frac{2\text{rank}(r_i) - G - 1}{G}\right), \quad (5.4)$$

where Φ^{-1} is the inverse standard-normal CDF, implemented in PyTorch through the inverse error function. The -0.5 midpoint places the smallest reward at quantile $1/(2G)$ rather than 0, avoiding the $\pm\infty$ tails. So the worst rollout always receives $\Phi^{-1}(1/(2G)) \approx -1.15$ (for $G=4$), regardless of the absolute reward gap. The best receives $\Phi^{-1}((2G-1)/(2G)) \approx +1.15$, and the middle two sit at ± 0.32 , so the outer gaps (0.83) are wider than the inner one (0.64). The estimator is *scale-free* by construction, as the z -score already is, so it does not remove the near-tie problem above. What it changes is the spacing: the four advantages are fixed quantiles of the standard normal whatever the reward values, the extremes are ± 1.15 rather than up to ± 1.5 , and ties are handled explicitly (below) rather than through the standard deviation.

Tie handling. For our discrete-valued reward vector $\mathbf{r} \in [0, 3]^G$, ties are common and matter. If two of the four rollouts both receive $r = 2.0$, naive rank assignment would give them advantages $\Phi^{-1}(3/8)$ and $\Phi^{-1}(5/8)$, which are not equal. We instead average the target quantiles of tied rewards. So algebraically-equal rewards receive algebraically-equal advantages. A small but important fix: without it, the gradient would prefer one trivially-indistinguishable rollout over another.

Implementation. The flag `--use_g2rpo=true` swaps the advantage in (5.1) for (5.4). The surrogate loss (5.2) and the KL term are unchanged. The estimator is a direct port of OpenVLThinker v2’s per-group rank-to-quantile transform. Everything else is identical to the vanilla GRPO of §5.4 (SFT initialisation, reward functions, KL coefficient β , group size $G=4$, prompt format, dataset, hyperparameters), so the experiment is a clean A/B on the advantage estimator alone.

Optional outer whitening. A secondary flag `--g2rpo_outer_whitening=true` adds a rank-to-quantile pass over the entire batch (across groups) after the per-group transform. This is OpenVLThinker v2’s headline configuration. It is a second whitening that makes the cross-group distribution of advantages also match $\mathcal{N}(0, 1)$. We tested both variants in pilot runs and saw minimal difference on IAD. So the reported G²RPO numbers (the 120-step estimator probe of §6.8, Table 6.12) use per-group OT without outer whitening.

Why we tested it. The reward of §5.2 is discrete-valued in $[0, 3]$, exactly the quantised heavy-tailed regime that motivated G²RPO. The hypothesis: the rank-based estimator would give a cleaner training signal on IAD groups where most rollouts cluster at a few reward values, and that would translate into a better final checkpoint. As we report in §6.5, the variant improved over the vanilla advantage in some pilots but did not beat the single-stage GRPO headline. The effect size of the advantage estimator is small relative to the reward design and data quality of §5.2.

5.8. Teacher-Intervention Arms: Constructing Self-Distilled SFT Datasets

§5.6 described one self-distillation loop, the iterative SFT→RL cycle. Here we define a second, more controlled family, built on the SFT+GRPO checkpoint: three *teacher-intervention arms*, which are three data-construction recipes over the same image set that vary the level of external-teacher intervention applied to the model’s own rollouts. Because a recipe that revises a weak rollout retains an item that a recipe which only filters would discard, the arms also differ in size, so intervention and corpus size move together. The arms instantiate the self-training recipes reviewed in §2.4.3 (rejection sampling, STaR, and critique-and-revise), and their results are reported in §6.6.

Three arms. The three arms are constructed by holding the underlying image set fixed (the 6,000-image SFT pool of §6.3) and varying the level of teacher intervention applied to the rollouts of the SFT+GRPO checkpoint (ckpt-530) on those images:

- **Arm A, pure rejection sampling**, following RAFT [15] and the kept-only variant of STaR [83]. The GRPO model is rolled out on the headline SFT image pool. Only rollouts whose final answer matches the ground truth are retained. Across the eight rollouts per image, 84.6% of individual rollouts give the correct verdict and 90.8% of images have at least one correct rollout. We then sample a subset of 2,978 items, stratified by product and balanced on the Real-IAD acquisition folder (1,489 NG and 1,489 OK, up to availability of confident NG rollouts). By the verdict the trace itself teaches it is 1,298 anomalous and 1,680 normal, because a kept rollout on an NG capture whose defect is not visible answers no.
- **Arm B, STaR-style with rationalisation on failure** [83]. Items whose rollout is correct are kept as-is (3,557 items, “kept”). Items where the model failed all rollouts are augmented with a Gemini-written corrected trace (812 items, “corrected”). Total 4,369 items, natural balance $\approx 31\%$ NG / 69% OK (the imbalance is intrinsic to the recipe, because correct OK rollouts dominate for a precision-biased model). Arm B discards the 1,631 correct-but-weakly-grounded items that Arm C revises and keeps.
- **Arm C, STaR with external critique and revision** [61]. In addition to the rationalisation-on-failure step, items whose rollout was correct but weakly grounded are revised by Gemini-3-Flash (1,631 items, “rewritten”). An item counts as weakly grounded if its accuracy reward passes but stays below the weak-but-correct threshold of 1.8, or if the faithfulness judge of Figure 5.3 scores it below ACCEPTABLE. Total 6,000 items, 50/50 balanced. We call this recipe Keep-Correct-Revise (KCR): keep the rollouts the policy already gets right, have a teacher correct the ones it gets wrong, and have it revise the ones that are right but weakly grounded. This is identical in image set and recipe to the iter-2-clean dataset used in §6.7, but trained from *base* here rather than from a GRPO checkpoint. This from-base training is the crucial difference from the regressing §6.7 experiments. On essentially the same cleaned $\approx 6\text{K}$ dataset, resetting to base (Arm C, 82.80%) beats continuing from the GRPO checkpoint (81.07%) by +1.73 pp. This is exactly the reset-to-base design of the OpenVLThinker self-improvement loop [14], in which every SFT iteration restarts from the base model and only the data carries forward.

All three arms share the same image distribution, the same SFT recipe (Qwen2.5-VL-7B, frozen vision tower, 4 epochs, $\text{LR } 10^{-5}$, cosine schedule, DeepSpeed ZeRO-3 CPU offload, per-device batch 8 with gradient accumulation 2 for an effective batch size of 32), and the same evaluation protocol on the DS-MVTec (1,670 samples) and VisA (2,141 samples) MMAD subsets. The per-epoch balanced accuracy and the best checkpoint per arm are reported in §6.6. The rollout pool is all 10,236 training items. The three arms are built on the 6,000 SFT images only, so no model in this thesis is trained on the full 10,236. Figure 5.3 shows the four stages that turn those rollouts into the arms. The policy is sampled at $k=8$, every rollout is scored locally, a Gemini-3-Flash faithfulness judge re-scores the survivors, and a corrector repairs the traces along two paths. Local scoring keeps 8,872 items and sends the other 1,364 for correction, so those two buckets partition the pool exactly. A further 2,406 items pass but score below the weak-but-correct threshold and are marked for rewrite, which makes the rewrite set a subset of the kept set rather than a third slice of the pool.

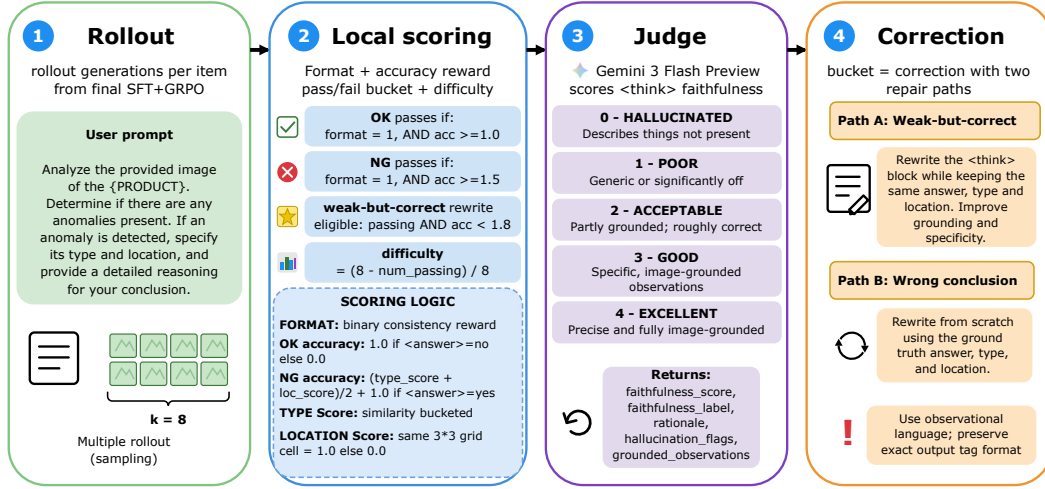


Figure 5.3: How the rollouts of the SFT+GRPO checkpoint become corrected training traces. (1) The policy is sampled $k=8$ times per image. (2) Local rewards route each item to keep or to correct, and mark the weak but correct ones for rewrite. (3) A Gemini-3-Flash judge re-scores every item for faithfulness. (4) The same model repairs the traces along two paths. All counts are items, not rollouts.

The same loop on a second backbone. The keep, correct and rewrite rule above was also run end to end on LLaVA-OneVision-7B-SI [34], over the same 10,236 images and with the same judge. That run gives 9,179 kept and 1,057 corrected, of which the judge confirms 9,045 as faithful and demotes 133 into a combined rewrite set of 2,513, with one kept item dropped after the judge call failed on it. Supervised fine-tuning on that backbone froze the vision tower and trained the projector and the language trunk, and we confirmed from the saved weights that the tower does not move across the run. The GRPO stage on that backbone trained every parameter, including the vision tower. One evaluation detail is specific to it. Its chat template renders the system turn empty, so its rows are scored on the user-turn instruction alone, under the same harness and the same prompt mode as Qwen2.5-VL. The results are reported in §6.9.

Prespecified outcomes. Before running the experiment we wrote down four possible outcomes [14, 15, 83, 85]. (O1) “ $A \geq \text{Headline}$ ”: self-distillation matches or exceeds external Gemini at fixed image distribution, consistent with Zhang et al. [85] and the strong form of Zelikman et al. [83]. (O2) “ B beats both A and C ”: rationalisation-on-failure adds value but critique-and-revise adds noise (the prediction of Dong et al. [15] and the OpenVLThinker design [14]). (O3) “ C beats B beats A ”: each layer of teacher intervention adds value (the conservative “no harm from more help” prediction). (O4) “ $A \approx B \approx C < \text{Headline}$ ”: teacher choice does not matter at fixed image distribution, and the VisA win seen with iter-2-v2 (§6.5) was an image-distribution artefact.

6

Results and Analysis

This chapter reports what the models achieve and how they achieve it. It is organised around questions rather than around the order in which the experiments were run. §6.1 fixes the measurement and states the caveats once, and §6.2 establishes where the untrained backbones start. §6.3 asks how far supervised fine-tuning goes on its own. §6.4 and §6.5 ask what reinforcement learning adds on a weak supervised baseline, and find that the plain recipe beats the elaborate ones. §6.6 then asks who should write the training traces, and produces the best Qwen2.5-VL model in this thesis. §6.7 explains why that corpus has to be trained from base, and §6.8 returns to reinforcement learning and finds that on the stronger initialisation it no longer helps. §6.9 runs the whole loop again on a second backbone. §6.10 looks past the headline number at what the models actually do, §6.11 scores the explanations, §6.12 places the work against the state of the art, and §6.13 reports the training dynamics. A held-out Real-IAD check is in Appendix L.

All percentages are given to two decimal places. Bold rows in tables mark our headline configuration. “Bal-acc” abbreviates balanced accuracy.

6.1. Evaluation Protocol and Metrics

Every number in this chapter comes from one evaluation harness applied to one fixed sample set. The benchmarks are the DS-MVTec and VisA subsets of MMAD, at 1,670 and 2,141 images. Ours and the reference checkpoints alike are run through the same harness, the same image and token budgets and the same answer parser, so the rows are comparable to each other. Every row uses the training prompt of the supervised stage with its system message, with one stated exception: the SFT+GRPO checkpoints of §6.4 are scored under the prompt the policy was trained on, the fixed GRPO question without a system message. Under the supervised prompt the same headline checkpoint scores 80.74% on DS-MVTec and 70.50% on VisA (Appendix J.2), so the prompt accounts for up to two points of that row on DS-MVTec and nothing on VisA. They are not directly comparable to numbers quoted in other papers, which we discuss in §6.12. The parser reads the verdict from the <answer> tag and, when no tag is present, from a bare yes or no in the response. A generation from which neither yields a verdict is counted as wrong in every balanced accuracy of this thesis.

Why balanced accuracy. Both benchmarks are class-imbalanced, and both failure modes are easy to reach by accident. A model that answers yes to everything scores well on plain accuracy where defects are common. A model that answers no to everything scores well where they are rare. Balanced accuracy is the mean of sensitivity and specificity, so it is at chance for both degenerate policies. §6.2 shows that this is not a hypothetical concern, because both untrained backbones fall into one of the two traps. Plain accuracy, precision, recall and F1 are reported alongside where they add something.

Checkpoint selection. Unless stated otherwise, each configuration is evaluated at every epoch and the reported row is the epoch with the highest DS-MVTec balanced accuracy. The VisA column then reports that same checkpoint rather than the VisA best, so the two columns are one model and not two. Where a configuration’s VisA peak falls at a different epoch we say so.

Four names. Four models recur through the chapter and the rest of the thesis, and the figures label them the same way. Table 6.1 fixes the names once. Inside the teacher ablation of §6.6 the third model is also called Arm C, because there it is compared with Arms A and B.

Table 6.1: The four names used for the Qwen2.5-VL-7B models in this thesis, the corpus each is trained on, the reported checkpoint and the section that produces it. The same four names label the LLaVA-OneVision-7B-SI ladder of §6.9.

Name	Training	Checkpoint	Section
Base	none, the released instruct model	as released	§6.2
SFT	the 6K Gemini-written corpus, vision encoder frozen	ckpt-564, epoch 3	§6.3
SFT+GRPO	GRPO from the SFT checkpoint, all three blocks trained	ckpt-530, epoch 1	§6.4
KCR	the Keep-Correct-Revise corpus built from the SFT+GRPO rollouts, trained from base	ckpt-376, epoch 2	§6.6

Two caveats, stated once. First, every model evaluated on public DS-MVTec and VisA images may have encountered those images during pretraining, and we cannot rule this out for any backbone. Second, and more specifically, the public LLaVA-OneVision instruction mixture (lmms-lab/LLaVA-OneVision-Data) contains entries whose sample identifiers name MVTec-AD, and part of them are anomaly-detection instructions. We found such entries but cannot say how many distinct MVTec-AD images they correspond to, and they may have been part of the backbone’s pretraining. The same search finds no VisA entries. Every DS-MVTec number on the LLaVA-OneVision backbone therefore carries a possible train-on-test contamination, and every VisA number on that backbone does not. Where a claim in this chapter depends on the LLaVA-OneVision rows, we make it on VisA.

The chapter in one figure. Figure 6.1 gives the whole result before the argument that builds it. Two backbones run the same four-stage ladder, and on both of them the last and best stage is supervised. The rest of this chapter establishes each rung and then asks what the final one is actually doing.

6.2. Zero-Shot Baseline Performance

Table 6.2 reports the three backbones without any fine-tuning, under the same prompt and parser as every trained model.

Table 6.2: Zero-shot performance of the untrained backbones on the MMAD DS-MVTec and VisA subsets. The two 7B models are precision-dominated and the 3B model is the opposite. On VisA both 7B backbones collapse onto almost the same trivial policy. [†] LLaVA-OneVision does not attend to the system message, so its instruction is carried in the user turn, for the reason given below the table.

Model	Accuracy	Bal-acc	F1	Precision	Recall
<i>DS-MVTec (1,670 samples)</i>					
Qwen2.5-VL-3B (base)	76.53 %	56.14 %	86.18 %	75.90 %	99.67 %
Qwen2.5-VL-7B (base)	55.03 %	69.08 %	56.06 %	99.17 %	39.07 %
LLaVA-OneVision-7B-SI (base) [†]	64.37 %	75.66 %	67.99 %	99.84 %	51.55 %
<i>VisA (2,141 samples)</i>					
Qwen2.5-VL-3B (base)	56.10 %	50.21 %	71.81 %	56.01 %	100.00 %
Qwen2.5-VL-7B (base)	48.34 %	53.79 %	14.26 %	98.92 %	7.69 %
LLaVA-OneVision-7B-SI (base) [†]	48.34 %	53.80 %	14.13 %	100.00 %	7.60 %

Three things follow. The 3B model answers yes to almost everything. Its DS-MVTec recall is 99.67 % and its VisA recall is exactly 100 %, which inflates plain accuracy and leaves balanced accuracy at the chance line. The two 7B models do the opposite. They are so conservative that on VisA they catch 7.69 % and 7.60 % of true defects respectively, at essentially perfect precision. That the two 7B backbones land within 0.01 pp of each other on VisA, from different pretraining data and different architectures, is worth noting. It is not agreement about the images. It is two models independently discovering that answering no is the safe response when

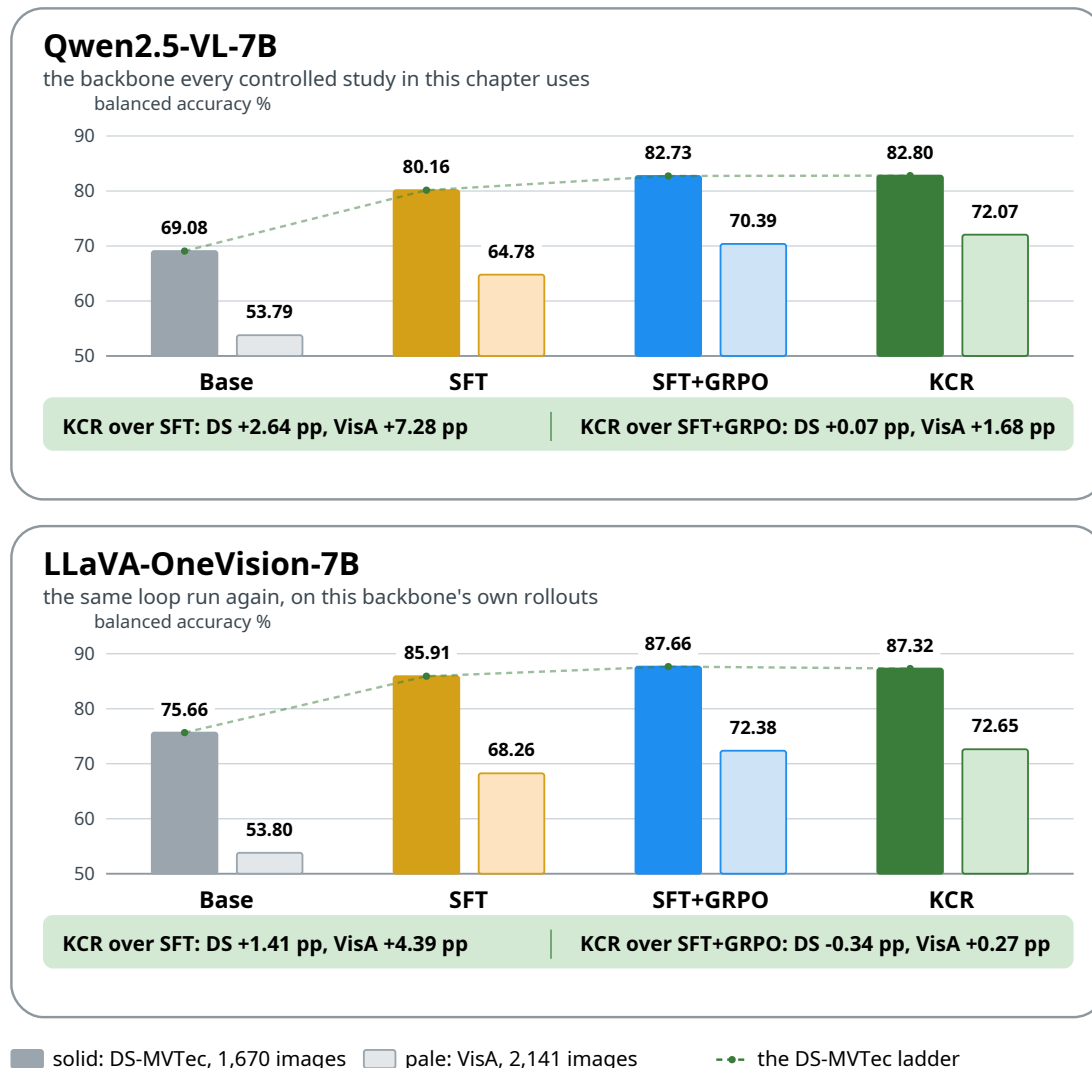


Figure 6.1: Balanced accuracy at each stage of the training ladder, for both backbones and both benchmarks. Solid bars are DS-MVTec and pale bars VisA. The axis starts at 50, which is chance for balanced accuracy. Each stage is its best epoch selected on DS-MVTec. The corrective KCR corpus of \$6.6 matches the SFT+GRPO stage on both backbones with no reinforcement-learning step and leads on VisA. The DS-MVTec cells of the lower panel carry the contamination caveat above. VisA does not.

the task is unfamiliar. Balanced accuracy exposes both pathologies, which is why it is the primary metric for the rest of this chapter.

The LLaVA-OneVision base row and the system turn. LLaVA-OneVision-7B-SI does not attend to the system message. Under the system-turn placement that every other row in this chapter uses, it never emits the four-tag schema. It answers in free text that often names the defect, for example “*The anomaly is glue strip. It is located at the center of the image*”. The harness’s fallback for tag-less text reads a bare yes or no from the response, and because the word “anomaly” contains “no”, it maps almost every such response to a no verdict: 1,632 of the 1,670 DS-MVTec generations and all 2,141 VisA generations, the remaining 38 carrying no verdict at all. The row therefore collapses to an all-no policy, 49.21% on DS-MVTec and 50.00% on VisA. That is a format-compliance failure and not blindness to defects. Moving the identical instruction to the user turn (S.J.1) makes the model answer with a bare yes or no, which the same fallback reads as intended, and gives the row above. After fine-tuning the placement no longer matters, because the trained model emits the schema either way, so the four trained LLaVA-OneVision rows in this chapter keep the system-turn form.

6.3. SFT Ablation: Four-Factor Grid

The first question is how much of the gap in Table 6.2 closes with supervised fine-tuning on Gemini-written traces, and which design choices matter. The ablation varies four factors: model size (3B against 7B), vision-encoder treatment (frozen against unfrozen), training-set size (6K against 15K) and training epoch (1 to 4). The first three give six populated configurations plus one control, because the unfrozen encoder is evaluated at 15K, and at 6K only as the matched-learning-rate control described below. Figure 6.2 and Table 6.3 summarise the grid, the first showing every epoch of the matched 15K cells and the best epoch of every populated cell.

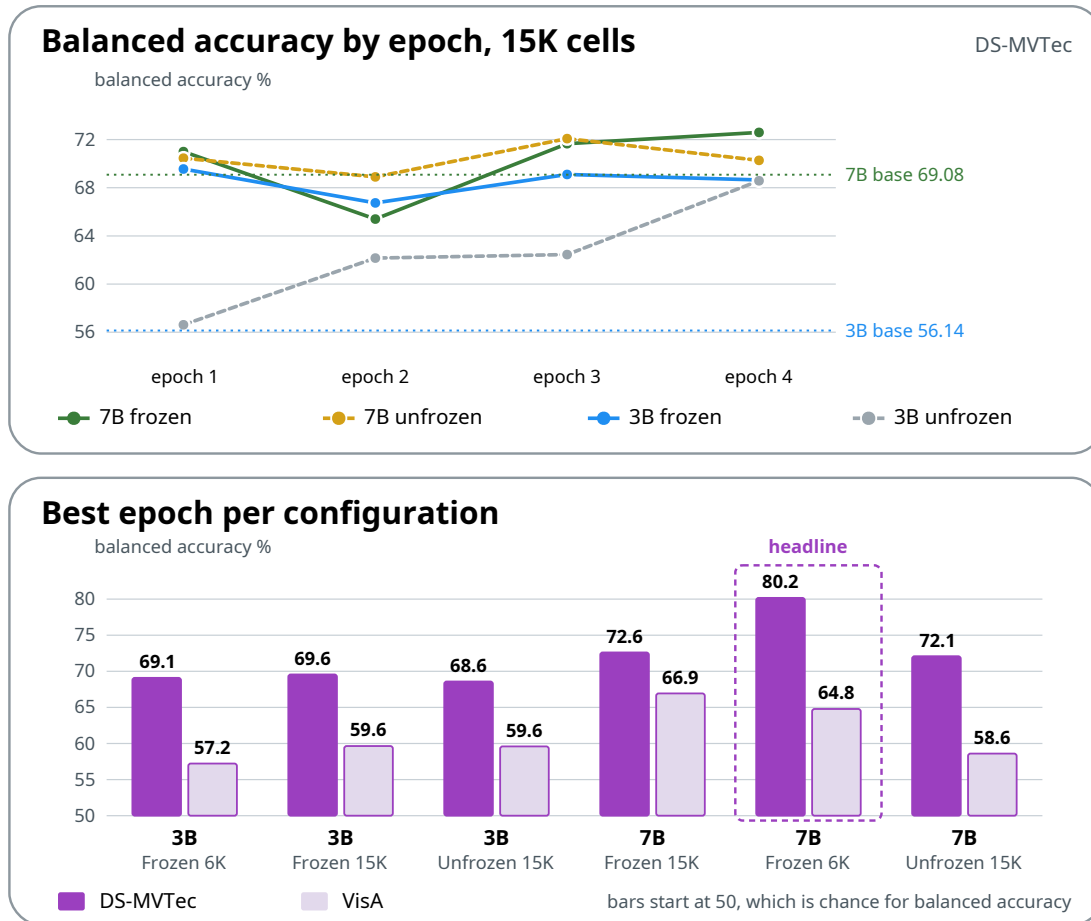


Figure 6.2: The four-factor SFT ablation. **Top:** balanced accuracy at every epoch for the four matched 15K cells, against each backbone’s zero-shot baseline. On DS-MVTec the frozen and unfrozen curves stay close, and the decisive difference appears on VisA in Table 6.3. **Bottom:** the DS-MVTec-best epoch of each of the six populated cells, with the VisA bar reporting that same checkpoint. The 7B cells dominate the 3B cells and 7B-frozen-6K is the DS-MVTec best. The matched-rate control of Table 6.3 is not drawn.

Table 6.3: Best SFT checkpoint per cell, where “best” is the epoch with the highest DS-MVTec balanced accuracy. The DS-MVTec and VisA columns are balanced accuracy and the last two are plain accuracy and F1 on DS-MVTec, all for that same DS-MVTec-selected checkpoint, so they are not independently maximised. “Frozen” means the vision tower is fixed and “Unfrozen” means full end-to-end fine-tuning at the lower learning rate of 1×10^{-6} (§4.5). [‡] Trained at the frozen cells’ learning rate of 1×10^{-5} , the same schedule as the bold row, so that pair differs in the freezing flag only. Bold marks the thesis headline SFT configuration.

Model size	Vision encoder	Train set	Best epoch	Balanced accuracy		DS-MVTec	
				DS-MVTec	VisA	Accuracy	F1
3B	Frozen	6K	4	69.08 %	57.22 %	72.22 %	80.02 %
3B	Frozen	15K	1	69.56 %	59.65 %	68.80 %	76.18 %
3B	Unfrozen	15K	4	68.58 %	59.60 %	69.58 %	77.34 %
7B	Frozen	15K	4	72.60 %	66.94 %	78.86 %	85.66 %
7B	Frozen	6K	3	80.16 %	64.78 %	80.36 %	85.76 %
7B	Unfrozen	15K	3	72.08 %	58.60 %	68.50 %	75.02 %
7B	Unfrozen [‡]	6K	1	75.97 %	60.13 %	71.68 %	77.59 %

Four findings follow.

More data does not mean better performance at this stage. The 7B-frozen-6K model reaches 80.16% on DS-MVTEC. The same configuration at 15K training samples reaches 71.66% at the same epoch-3 checkpoint, and 72.60% even at its own best epoch. That is the largest single effect in the ablation and it runs against the usual intuition. It is not a checkpoint-selection artefact. The 15K run trails the 6K run at every epoch and its best checkpoint is the last one, with training loss converging to the same floor. It is also not a trace-length effect. The 6K SFT split is a subset of the roughly 14.5K set and both have the same mean trace length of about 137 words. What differs is composition. The 6K split is the clean stratified SFT partition, whereas the 15K set folds the GRPO and held-out splits in as SFT data. The effect is also specific. At 7B the curated split is far ahead on DS-MVTEC, by 7.56 points at each configuration’s best epoch, and behind on VisA by 2.16. At 3B the union is slightly ahead on both. The headline gain from the smaller set is therefore a 7B and DS-MVTEC effect, and not a general claim that less data is better.

Freezing the vision encoder wins. Freezing beats unfreezing on DS-MVTEC at both sizes, 69.56 against 68.58 at 3B and 72.60 against 72.08 at 7B. The effect is far larger on VisA at 7B, 66.94% frozen against 58.60% unfrozen. We read this as overfitting. The encoder’s parameters are expressive and a small industrial training set cannot supervise them robustly. The last row of Table 6.3 removes the learning-rate confound: a 7B-unfrozen run on the 6K split at the frozen cells’ rate of 1×10^{-5} , the same four-epoch schedule as the headline SFT row, so the two differ in the freezing flag only. It peaks at epoch 1, 75.97% on DS-MVTEC and 60.13% on VisA, and sits 4.2 to 9.4 points behind the frozen run on DS-MVTEC at every epoch (70.80, 75.46 and 73.94% at epochs 2 to 4) while its VisA scores stay level with the frozen run (65.45, 65.12 and 64.17% against 64.78%). Its training loss falls normally, from 2.18 to 0.43, so this is not a divergence but weaker generalisation. An earlier two-epoch pilot of the same configuration peaked at 72.82% and fell to 69.27%. Freezing therefore wins at both learning rates, and the unfrozen arm is otherwise evaluated at 15K only, in line with Li et al. [39].

Size helps everywhere. On every matched cell of data and encoder treatment, the 7B model beats the 3B model by 3 to 11 points of balanced accuracy on DS-MVTEC. The gap is largest in the frozen cells, which suggests the larger trunk extracts more from the same frozen visual features.

The trace buys precision, not accuracy everywhere. We trained one control on the same 6,000 images with the verdict only and no reasoning trace. On DS-MVTEC the structured model is ahead at every epoch, by 1.2 to 3.7 points, and it is far less trigger-happy. At epoch 3 the false-alarm rate is 20.3% with reasoning and 34.0% without. On VisA the control is ahead at the one epoch where both were scored, 69.39 against 64.78. So the trace is not free accuracy. It buys calibration on DS-MVTEC, and the case for it rests as much on the explanation it produces, which §6.11 measures directly.

6.4. GRPO on the SFT Baseline

Starting from the best SFT checkpoint of §6.3 (7B-frozen-6K, ckpt-564), we run GRPO and evaluate every saved checkpoint. Table 6.4 reports the result.

Table 6.4: Balanced accuracy of the GRPO checkpoints, with the SFT initialisation as the reference row. The last two columns are plain accuracy and F1 on DS-MVTEC only. The GRPO rows are scored under the GRPO training prompt (§6.1), the SFT row under the supervised prompt. Bold marks the headline checkpoint.

Checkpoint	DS-MVTEC	VisA	Accuracy	F1
ckpt-530	82.73 %	70.39 %	81.08 %	86.01 %
ckpt-1060	82.03 %	69.86 %	80.36 %	85.44 %
<i>SFT only, best</i>	80.16 %	64.78 %	80.36 %	85.76 %
<i>Δ GRPO against SFT</i>	+2.57 pp	+5.61 pp	+0.72 pp	+0.25 pp

The best checkpoint is ckpt-530 at the end of epoch 1, at 82.73% on DS-MVTEC (precision 94.09%, sensitivity 79.20%, specificity 86.27%, F1 86.01%) and 70.39% on VisA (sensitivity 63.66%, specificity 77.13%). Two observations follow.

The gain is larger where the baseline was weaker. GRPO adds 2.57 points on DS-MVTEC and 5.61 points on VisA. The SFT baseline was much weaker on VisA, at 64.78%, so there was more headroom for the reward to exploit. This asymmetry recurs throughout the chapter, and it is the first hint of the pattern that §6.8 makes explicit.

Training longer makes it worse. The further-trained ckpt-1060 at the end of epoch 2 underperforms ckpt-530 by 0.70 points on DS-MVTEC and 0.53 on VisA. We read this as mild reward over-optimisation and early-stop at ckpt-530. The dynamics behind that choice are in §6.13 and the selection criterion is discussed in §7.4.

6.5. Iterative SFT $\leftrightarrow$ RL and Alternative RL Variants

§5.6 described two strategies more elaborate than the single-stage recipe: the OpenVLThinker-style iterative SFT $\leftrightarrow$ RL cycle (iter-2), and a G²RPO variant that replaces the linear group z -score advantage of (5.1) with the Gaussian rank-to-quantile transform of Hu et al. [26]. Both ran on the same backbone. Table 6.5 gives the full iterative progression and Table 6.6 the comparison.

Table 6.5: Full balanced-accuracy progression of the iterative cycle. “SFT₁” is the fresh SFT on 4,026 self-filtered traces after resetting to the Qwen2.5-VL-7B base, and “GRPO₁” is the subsequent RL stage. The *Prompt* column records the evaluation prompt, **T** for the train prompt and **G** for the GRPO prompt.

Variant	Stage and checkpoint (epoch)	Prompt	DS-MVTEC	VisA
iter-2	SFT ₁ ckpt-378 (ep 3)	T	77.61 %	72.00 %
	SFT ₁ ckpt-504 (ep 4)	T	79.54 %	72.02 %
	GRPO ₁ ckpt-265 (ep 0.5)	G	78.38 %	70.70 %
	GRPO₁ ckpt-530 (ep 1)	G	80.64 %	68.65 %
	GRPO ₁ ckpt-795 (ep 1.5)	G	79.55 %	68.40 %
	GRPO ₁ ckpt-1060 (ep 2)	G	79.37 %	68.67 %

Two patterns from the single-stage run recur. The iterative GRPO stage peaks again at the end of epoch 1 and declines afterwards. It also lifts DS-MVTEC over its own SFT base, 80.64 against 79.54, while costing several points on VisA, 68.65 against 72.02. That is the same precision-for-recall rebalancing GRPO performs elsewhere in this chapter.

Table 6.6: Best checkpoint of each RL strategy on Qwen2.5-VL-7B. The single-stage and G²RPO rows start from the headline SFT checkpoint. The iterative row starts from its own SFT₁ checkpoint of Table 6.5, which is trained on self-filtered rollouts after a reset to base. The G²RPO row is the two-epoch run evaluated on both benchmarks at every half epoch, selected on DS-MVTEC like the other rows. An earlier run of the same configuration, evaluated on DS-MVTEC only, reached 81.94 % at ckpt-530. Both elaborate variants are below the single-stage recipe on both benchmarks.

RL strategy	Best checkpoint	DS-MVTEC	VisA
Single-stage GRPO	ckpt-530	82.73 %	70.39 %
G ² RPO [26]	ckpt-795	81.87 %	69.75 %
Iterative SFT $\leftrightarrow$ RL (iter-2)	GRPO ₁ ckpt-530	80.64 %	68.65 %

Neither the iterative cycle at 80.64 % / 68.65 % nor the Gaussian-advantage variant at 81.87 % / 69.75 % beats the plain single-stage recipe at 82.73 % / 70.39 %, on either benchmark. We read this as a scale-and-task effect rather than a refutation of either method. Deng et al. [14] and Hu et al. [26] demonstrate their gains on broad open-ended multimodal reasoning, where the teacher distribution is the binding constraint. Ours is a narrow binary detection task on which one well-tuned GRPO pass already approaches the reachable ceiling for a 7B model. The rollout-and-filter step also concentrates the self-generated dataset on easy, already-correct examples, which is a known failure mode of naive rejection-sampling self-training. §6.6 shows what happens when that filter is replaced by a corrector.

Caveat on prompt formats. Every GRPO checkpoint in this section, the iter-2 ones and the single-stage headline alike, is evaluated under the GRPO prompt, and the supervised rows under the

supervised prompt (§6.1). On checkpoints evaluated under both, the two agree to within about two points, so the ranking in Table 6.6 is robust to the difference. The absolute numbers should still be read with it in mind.

6.6. Teacher Ablation: External vs. Self-Distillation

§6.3 varied how much data and which architecture. This section varies a third thing, which is *who writes the trace*. Every run so far trained on traces written by Gemini 2.5-Flash. The three arms defined in §5.8 are three data-construction recipes applied to the same rollout pool over the same 6,000 SFT images. They differ in how much the external teacher intervenes in the model's own rollouts, and as a direct consequence they also differ in how many items survive, from 2,978 when only already-correct rollouts are kept to 6,000 when weak ones are revised rather than discarded. Recipe and corpus size are therefore not separable here, and the comparison should be read as one between curation strategies rather than as a clean isolation of teacher intervention on its own. Arm A keeps only the rollouts that are already correct, which is pure rejection sampling. Arm B adds a teacher-written trace wherever the model failed every rollout. Arm C additionally has the teacher revise the rollouts that were correct but weakly grounded, which is the recipe we call Keep-Correct-Revise. The construction and the routing counts are in Figure 5.3.

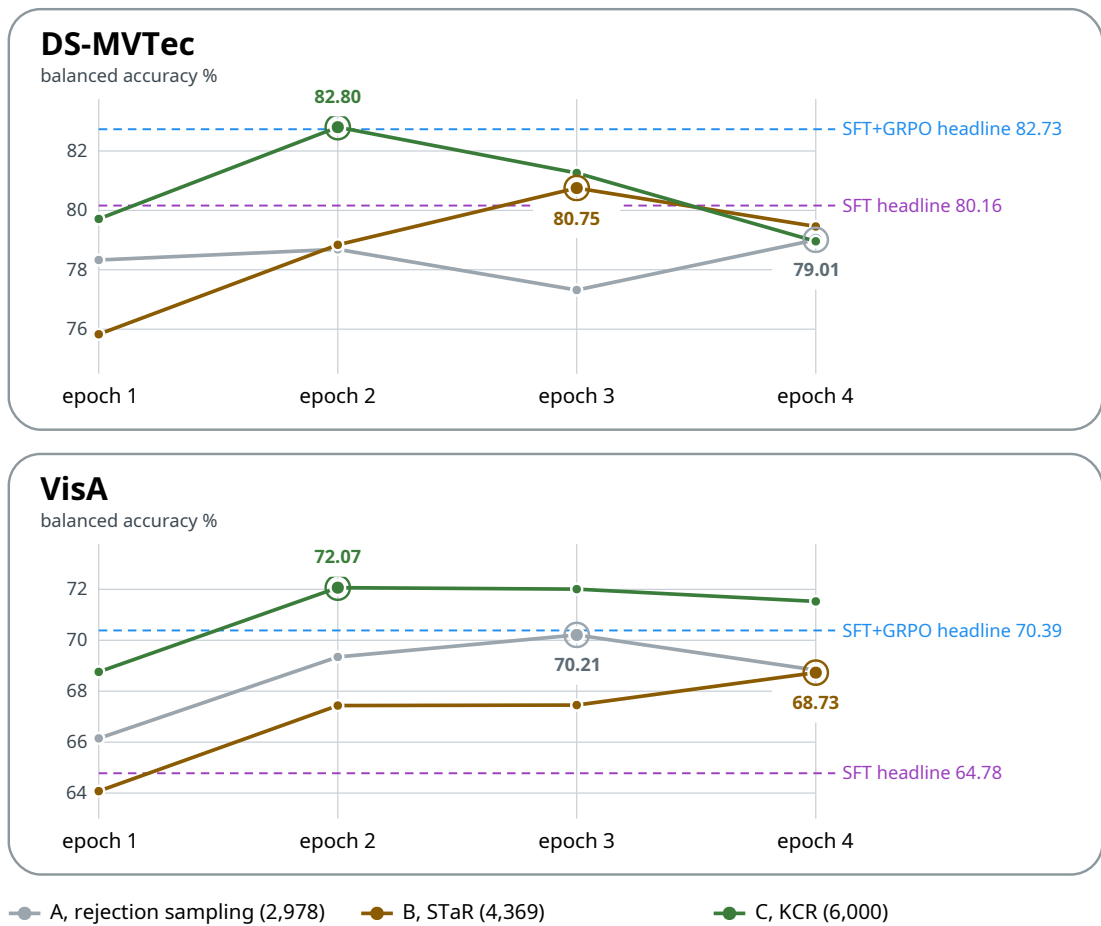


Figure 6.3: Balanced accuracy of the three arms at each epoch. The arms are drawn from the same 6,000 images with the level of teacher intervention changing, and with it the corpus size: they keep 2,978, 4,369 and 6,000 items, so size and curation move together. The ringed marker is each arm's best epoch and the dashed lines are the SFT and SFT+GRPO results of this chapter. Arm C is the only arm that clears both, and it does so at epoch 2 on both benchmarks. The per-epoch figures are in Table 6.7.

Table 6.7: Per-epoch balanced accuracy of the three arms. The best epoch per arm is bold. All arms trained from the Qwen2.5-VL-7B base, frozen vision tower, 4 epochs, effective batch 32.

Arm	Checkpoint (epoch)	n items	DS-MVTec	VisA
A, pure rejection sampling	ckpt-94 (ep1)	2,978	78.33 %	66.15 %
	ckpt-188 (ep2)	2,978	78.69 %	69.35 %
	ckpt-282 (ep3)	2,978	77.32 %	70.21 %
	ckpt-376 (ep4)	2,978	79.01 %	68.83 %
B, kept plus corrected	ckpt-137 (ep1)	4,369	75.83 %	64.08 %
	ckpt-274 (ep2)	4,369	78.84 %	67.44 %
	ckpt-411 (ep3)	4,369	80.75 %	67.46 %
	ckpt-548 (ep4)	4,369	79.46 %	68.73 %
C, Keep-Correct-Revise	ckpt-188 (ep1)	6,000	79.71 %	68.76 %
	ckpt-376 (ep2)	6,000	82.80 %	72.07 %
	ckpt-564 (ep3)	6,000	81.26 %	72.01 %
	ckpt-752 (ep4)	6,000	78.96 %	71.53 %

Table 6.8: Each arm’s DS-MVTec-best checkpoint against the two baselines of §6.3 and §6.4. Positive values mean better than the headline SFT baseline.

Model	DS-MVTec	VisA	Δ_{DS}	Δ_{VisA}
SFT, 6K Gemini corpus (§6.3, ckpt-564)	80.16 %	64.78 %	reference	reference
SFT+GRPO (§6.4, ckpt-530)	82.73 %	70.39 %	+2.57	+5.61
A best, ckpt-376 (ep4)	79.01 %	68.83 %	−1.15	+4.05
B best, ckpt-411 (ep3)	80.75 %	67.46 %	+0.59	+2.68
C best, ckpt-376 (ep2)	82.80 %	72.07 %	+2.64	+7.29

More teacher help is monotonically better on DS-MVTec. The best-epoch ordering is C at 82.80%, then B at 80.75%, then A at 79.01%. Each additional layer of intervention, from none, to correction on failure, to correction plus revision, buys roughly two points. This is the conservative prespecified outcome, that more help does no harm.

The best arm needs no reinforcement-learning step at all. Arm C reaches 72.07% on VisA, which is 1.68 points above the published SFT+GRPO headline of 70.39%, and it matches that headline on DS-MVTec to within 0.07 points. It does so with supervised fine-tuning alone. A sufficiently well-constructed SFT corpus therefore reaches, at the supervised stage, what the full pipeline reached only after reinforcement learning. §6.8 asks the obvious follow-up.

Self-distillation alone already beats the Gemini-only headline on VisA. Arm A uses 2,978 items written entirely by the model, with no teacher intervention anywhere, and it beats the 64.78% headline on VisA at every epoch. The gain is +4.05 points at its DS-MVTec-best checkpoint and +5.43 points at its VisA-best epoch. On DS-MVTec it trails the headline by 1.15 points. That asymmetry is the clearest evidence we have that the Gemini-written traces carry something DS-MVTec-favourable and VisA-unfavourable in their style, and that the model’s own rollouts carry something VisA-favourable that Gemini does not.

On VisA the middle arm is the worst. The VisA ordering is C, then A, then B, not C, B, A. Arm B underperforms Arm A at every epoch despite having more items and an extra correction step. The two differ in two ways. Arm B keeps all 3,557 correct rollouts and adds 812 teacher-corrected traces, whereas Arm A subsamples the same kept pool down to 2,978 items to balance the classes. Two readings are available. The corrected traces, written from scratch by the teacher, may introduce the same stylistic shift that hurts on VisA. Or B’s class imbalance, at 31 % anomalous, may over-train the model on visible defects and under-train it on the missing-object distribution that dominates VisA. Separating the two needs a balance-matched control, which we leave to future work.

6.7. Post-GRPO SFT Refinement on Cleaned Data

Arm C’s corpus works. It matters where you start from when you train on it. Before the arms were run we tried the same idea in the obvious way, by continuing to train the GRPO checkpoint on cleaned, teacher-patched data. All three attempts regressed, and understanding why explains the reset-to-base design of §6.6.

The motivation was two data flaws found after the fact (Chapter 3). About 30 % of anomalous <type> tags in the original 6K pool carried raw Real-IAD defect codes rather than readable names, and the <location> vocabulary mixed two spellings of the same grid cell. Both were corrected. Starting from the GRPO production checkpoint, the cleaned 6K pool plus the full 4K GRPO pool was rolled out at $k=8$, giving 81,888 rollouts over 10,236 items. The routing and the correction pipeline are those of Figure 5.3. This earlier variant of it kept 3,557 items unchanged and patched 2,443, for the same 6,000-item total. The patched pool came out at 96.1 % anomalous (2,347 of 2,443), which reflects the model’s failure profile in Table 6.9.

Table 6.9: Per-item failure profile of SFT+GRPO ckpt-530 rolling out the 10,236-item training pool. Failures are heavily skewed to the anomalous class, because answering no is the trivial response.

	Anomalous items	Normal items	Total
At least 1 of 8 rollouts wrong	1,541 (30.1 %)	871 (17.0 %)	2,412
All 8 of 8 rollouts wrong	807 (15.8 %)	176 (3.4 %)	983

Table 6.10: Best-epoch balanced accuracy of three refinement compositions, all continuing from SFT+GRPO ckpt-530. “Avg” is the mean of the two benchmarks. None of the three matches the baseline.

Variant	Dataset composition	DS-MVTec	VisA	Avg	Δ
Baseline	SFT+GRPO ckpt-530, no further SFT	82.73 %	70.39 %	76.56 %	—
6K pool	3,557 kept plus 2,443 patched	81.07 %	70.05 %	75.56 %	−1.00
Balanced-192	96 normal plus 96 anomalous patches	71.30 %	60.52 %	65.91 %	−10.65
Held-out plus $\frac{197}{197}$	4,236 held-out items plus balanced patches	75.77 %	66.41 %	71.09 %	−5.47

The cause is a loss asymmetry, not the data. The 6K pool variant is balanced at the item level, 3,000 anomalous and 3,000 normal, but the loss is not. We measured the per-item cross-entropy of the GRPO checkpoint on every assistant token of the pool (Appendix H.3). The kept items, the model’s own rollouts, score 0.44 nats per token, the teacher-corrected items 1.47 and the teacher-rewritten items 0.85. The 2,443 patched items, 96 % of them anomalous, therefore carry 64 % of the summed loss while being 41 % of the items, and the anomalous half of the pool produces 67 % of the loss. The kept items are not gradient-free: their own-sample loss is a third of the total. But the signal the first epoch sees is two thirds anomalous on a corpus that is half anomalous, which drives anomaly recall up and erodes normal recall. Figure 6.4 shows the consequence product by product, and every regression falls on a product whose normal class is subtle. Subsampling the patches, the Balanced-192 attempt, does not rescue the recipe because 192 items cannot anchor a 7B model. At epoch 1 it produced 322 of 1,670 unparsable outputs on DS-MVTec. Adding out-of-distribution rehearsal data, the Held-out variant, does not rescue it either, because the same skewed signal dominates.

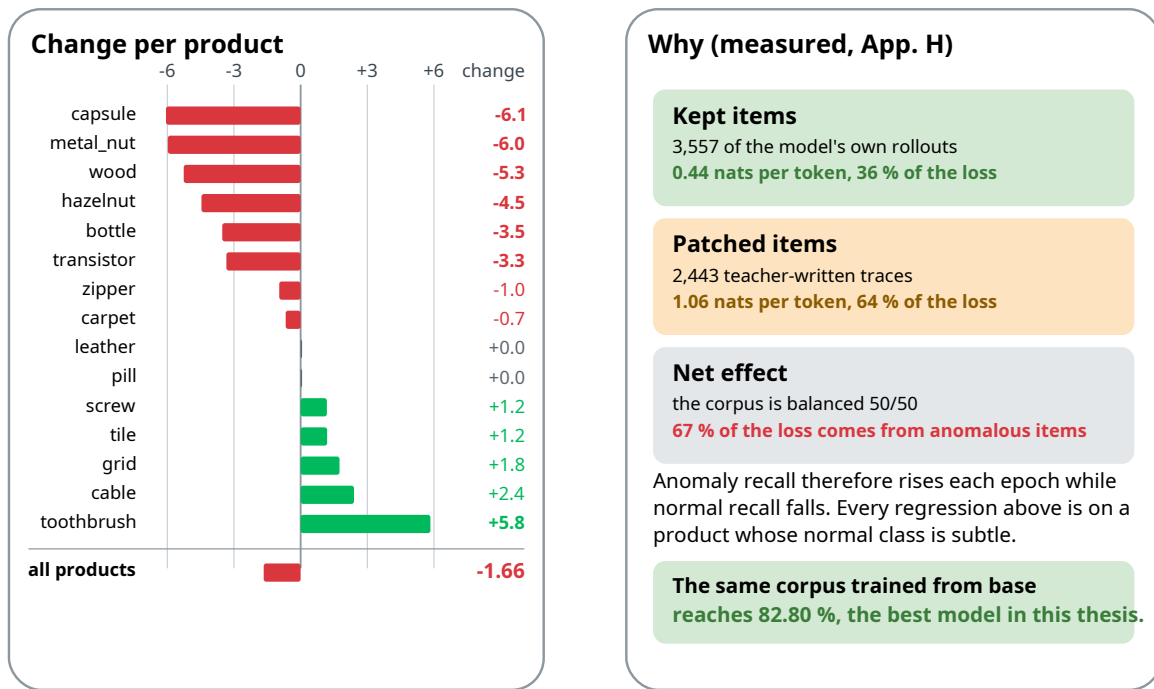


Figure 6.4: Change in per-product balanced accuracy on DS-MVtec when the cleaned 6K pool is trained on top of the GRPO checkpoint, against that checkpoint as the baseline. Five products improve, eight regress and two are unchanged, for a net -1.66 points. The right-hand panel gives the measured cause (Appendix H.3): the kept items score 0.44 nats per token under the GRPO checkpoint and the patched items 1.06, so 67 % of the loss comes from anomalous items on a corpus that is half anomalous.

The lesson is the design decision of §6.6. The same 6,000-item corpus that regresses to 81.07 % when training continues from the reinforcement-learned policy reaches 82.80 % when a fresh model is trained on it from base. Refinement on self-distilled data has to restart. That is the design of the OpenVLThinker loop [14] and, earlier, of ReST-EM, whose improve step fine-tunes the base model rather than the previous iteration's model, to limit drift and preserve transfer [67].

6.8. GRPO on the KCR Baseline

§6.4 found that GRPO lifts a weak supervised baseline by 2.57 and 5.61 points. §6.6 then produced a supervised checkpoint that already matches that result. The natural question is whether the reinforcement-learning stage still earns its place once the initialisation is strong. We ran the identical GRPO recipe from KCR ckpt-376 instead of the original SFT checkpoint, with a denser save cadence of 265 steps.

Table 6.11: GRPO trained on top of the KCR checkpoint, full-subset evaluation. The Δ columns are relative to the KCR initialisation. Every checkpoint sits below that initialisation on DS-MVtec.

Checkpoint	DS-MVtec	VisA	Δ_{DS}	Δ_{VisA}
<i>Reference rows</i>				
KCR ckpt-376, SFT only, the initialisation	82.80 %	72.07 %	—	—
SFT+GRPO, ckpt-530	82.73 %	70.39 %		
IAD-R1 released checkpoint, our harness	81.92 %	71.34 %		
<i>GRPO on KCR</i>				
ckpt-265 (ep 0.5)	80.61 %	72.68 %	-2.19	+0.61
ckpt-530 (ep 1.0)	80.32 %	70.34 %	-2.48	-1.73
ckpt-1060 (ep 2.0)	81.20 %	69.17 %	-1.60	-2.90
ckpt-1590 (ep 3.0)	80.70 %	68.70 %	-2.10	-3.37
ckpt-1855 (ep 3.5)	78.81 %	67.58 %	-3.99	-4.49
ckpt-2120 (ep 4.0)	80.38 %	67.86 %	-2.42	-4.21

It does not. Every checkpoint lands below the initialisation on DS-MVTeC, by 1.6 to 4.0 points, while the training reward rose over the same span. The mean group reward went from about 1.9 to 2.4 over the 2,120 logged steps, with a single-step peak of 3.25. On VisA only the earliest checkpoint is above the initialisation, by 0.61 points, and every later one falls below it. This is the mirror image of §6.4, where the same recipe lifted a weak baseline clearly. The reward rises and the task metric does not follow, which is the signature of a reward and metric decoupling [16]. One further run belongs on record. A variant from the same initialisation with the evaluation prompt as the training prompt and a KL coefficient of $\beta = 0.1$, trained for three epochs, touched 82.95 % and 72.62 % at one of its fifteen checkpoints and fell back below the initialisation afterwards. A single-seed margin of 0.15 and 0.55 points is inside the run-to-run variation of §7.7, so we read that run as inconclusive and leave a prompt-aligned GRPO stage to future work (§8.4).

The advantage estimator is not the explanation. One natural hypothesis is that the degradation is an artefact of group-relative normalisation. At $G = 4$ with a discrete, heavily tied reward, three rollouts often tie and the fourth wins by a small margin on whichever trivial component still varies, and the scale-free z -score gives that narrow winner a full-size advantage (§5.7) [43]. We tested this directly with three short GRPO bursts from the same KCR checkpoint, identical except for the estimator: vanilla group z -score, the mean-centred variant of Dr.GRPO, and G^2 RPO with the tie-robust rank-to-quantile transform. Each ran 120 steps, which at an effective batch of 8 prompts is 0.23 of an epoch over the 4,236-prompt set, with a checkpoint every 20, evaluated on a fixed seed-deterministic 400-sample probe.

Table 6.12: Closed-loop estimator ablation, balanced accuracy on a fixed 400-sample probe (DS-MVTeC / VisA) per 20-step checkpoint, all from KCR ckpt-376. The probe reads about 1.7 points above the full evaluation, so these figures are not comparable to the full-subset numbers elsewhere in this chapter.

Step	CTRL (z -score)	Dr.GRPO (no std)	G^2 RPO (rank)
0 (init)	84.52 / 71.89	84.52 / 71.89	84.52 / 71.89
20	82.87 / 72.22	84.23 / 70.18	85.27 / 70.63
40	81.74 / 70.76	81.19 / 66.82	83.07 / 69.80
60	82.25 / 71.82	81.69 / 69.92	83.55 / 68.38
80	83.67 / 70.45	82.06 / 69.82	82.61 / 68.99
100	83.02 / 70.78	82.03 / 70.71	84.23 / 68.76
120	81.89 / 69.87	81.91 / 70.83	82.22 / 68.99

All three behave the same way, as Figure 6.5 shows. Of the eighteen post-initialisation checkpoints only one edges above the initialisation on DS-MVTeC, G^2 RPO at step 20 by 0.75 points, and it is back below by step 40. On VisA every arm finishes below the initialisation. Averaged over the six post-initialisation checkpoints the plain z -score is the strongest of the three, at 70.98 against 69.71 and 69.26, although Dr.GRPO is the highest of them at the final step. Removing the standard-deviation division does not rescue the drop, and neither does replacing it. We therefore find no support for the estimator hypothesis. We bound the claim deliberately. The probe shows that no estimator escapes the plateau within 120 steps on this saturated initialisation, and it is not a general verdict on the estimators. One difference from the published G^2 RPO configuration belongs here. OpenVLThinker v2 pairs the estimator with dynamic sampling that discards groups whose rollouts are uniformly correct or uniformly incorrect [26]. Our probe ports the estimator and not the sampling, and on a saturated initialisation most groups are uniformly correct, so the half we did not port is the one that carries the gradient. The more likely explanation is limited headroom above a well-constructed supervised checkpoint. That is consistent with two recent findings. On a strong initialisation, reinforcement learning with verifiable rewards sharpens rather than extends capability [81]. And for the strongest open vision-language models the supervised stage is the binding constraint [80].

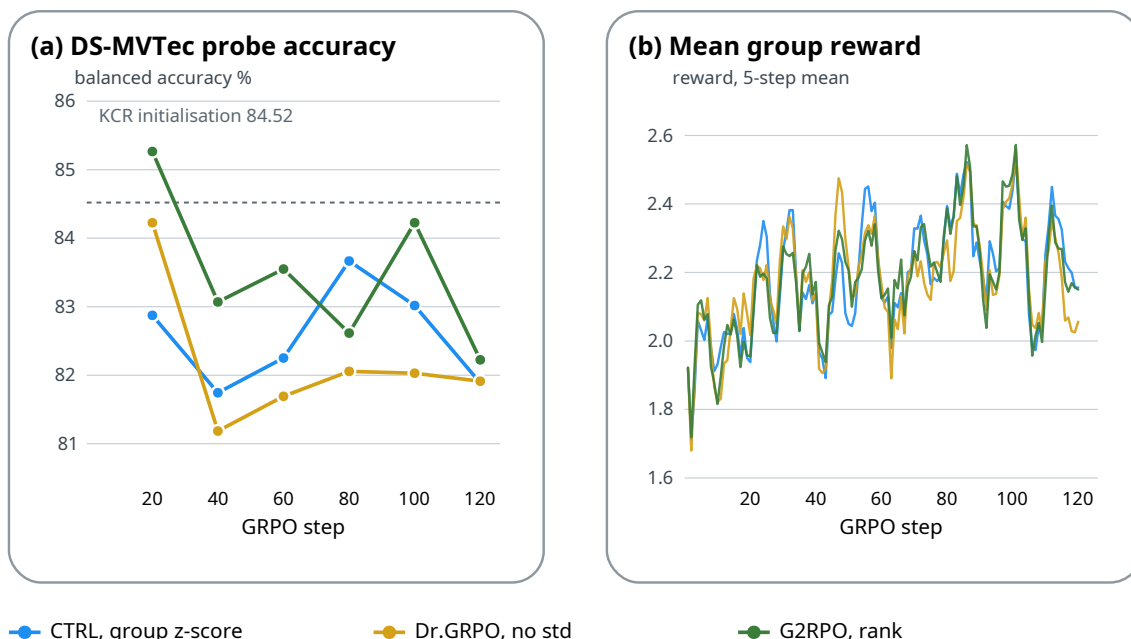


Figure 6.5: Reward and accuracy decoupling on the KCR initialisation for all three estimators over the 120-step probe. **(a)** DS-MVTec balanced accuracy per 20-step checkpoint against the initialisation, the dotted line at 84.52%. **(b)** Mean group reward over the same steps. Only one of the eighteen post-initialisation checkpoints rises above the initialisation on DS-MVTec, and it is back below by step 40, while the training reward rises for all three estimators over the same span.

Methodological note. The ablation used a tight closed loop by design, with 120-step bursts, a checkpoint every 20 steps and a fixed 400-sample probe. The probe takes about four minutes per evaluation against about thirty-five for the full subsets, which is what makes it possible to read the trajectory at fine resolution. The probe was validated against the full subsets on four reference checkpoints before use. It reproduces their ranking and tracks balanced accuracy to within about two points, running one to two points high in absolute terms while preserving the gap between the initialisation and the GRPO checkpoints. The caveats are a single seed per arm and a 120-step horizon. A first G²RPO attempt terminated at step 100 with a truncated checkpoint and was re-run to completion under the identical configuration.

Reading §6.4 and this section together. The two results are not in conflict. GRPO added 2.57 and 5.61 points to a supervised checkpoint at 80.16 and 64.78. It subtracts from one at 82.80 and 72.07. What the reinforcement-learning stage bought, in our runs, was not the final model but the rollouts from which the final corpus was built. Every item in the KCR corpus is a rollout of the GRPO policy or a teacher’s edit of one, so GRPO sits upstream of the best model rather than outside it. A second pass of the policy-gradient objective on that policy adds nothing, whereas its curated rollouts, trained from base, do. That is the central claim of this chapter, and §6.9 tests whether it holds on a second backbone.

6.9. Replication on a Second Backbone

Everything above uses one backbone. To test whether the finding depends on that choice, we ran the whole loop again on LLaVA-OneVision-7B-SI [34], the backbone on which IAD-R1 reports its own headline result [39]. Nothing about the recipe changes. That backbone’s own SFT+GRPO policy is rolled out at $k=8$ over the same Real-IAD images, every item is routed into keep, correct or rewrite, and a fresh model is trained from base on the resulting corpus. Only the model that wrote the rollouts is different.

Table 6.13: The Keep-Correct-Revise loop rerun on LLaVA-OneVision-7B-SI, scored on the identical 1,670 DS-MVtec and 2,141 VisA samples at the best epoch of each configuration. The reference row is the released IAD-R1 Qwen2.5-VL-7B checkpoint. The final block is a third backbone trained on the corpus Qwen2.5-VL wrote. [†] Base row only, the instruction is carried in the user turn (§6.2). [‡] Six of this row’s 2,141 generations carry no parsable verdict and are counted as wrong, as everywhere else in this chapter. Scored over only the parsable 2,135 the figure is 72.58%, and no ordering changes. [§] Selected on DS-MVtec as every row. The VisA score of this run rises at every epoch and reaches 74.29% at epoch 4 (Appendix M). Every DS-MVtec cell on this backbone carries the contamination caveat of §6.1.

Training corpus	Checkpoint	DS-MVtec	VisA
LLaVA-OneVision-7B-SI base, no fine-tuning	as released	75.66 % [†]	53.80 % [†]
SFT, 6K Gemini corpus	ep1	85.91 %	68.26 %
SFT, KCR corpus carried over from Qwen2.5-VL	ep1	87.70 %	70.97 %
SFT ep1, then GRPO	ckpt-530	87.66 %	72.38 % [‡]
SFT, corpus from this backbone’s own rollouts (KCR)	ep2	87.32 %	72.65 %[§]
Released IAD-R1 checkpoint (Qwen2.5-VL-7B), reference	as released	81.92 %	71.34 %
<i>A third backbone, Qwen3-VL-8B, by corpus transfer only</i>			
Qwen3-VL-8B base, no fine-tuning	as released	78.68 %	64.45 %
Qwen3-VL-8B trained on the Qwen2.5-VL KCR corpus	ep2	85.82 %	76.45 %

The parity reproduces. The model trained on this backbone’s own judged and corrected rollouts reaches 87.32% and 72.65% at the DS-MVtec-selected epoch. That is within 0.34 points of the SFT+GRPO model on DS-MVtec and 0.27 above it on VisA, a tie on both, reached with no reinforcement-learning step. A second GRPO checkpoint run as a restart reaches 87.86% and 71.89%, so the tie does not depend on which GRPO checkpoint is chosen. The corrective corpus then keeps improving where the GRPO run does not. Its VisA score rises at every epoch, to 74.29% at epoch 4, 1.91 points above the GRPO checkpoint, while the GRPO run’s own later checkpoint falls to 71.89%. We report the DS-MVtec-selected epoch so that VisA stays the held-out column (§6.1). On this backbone too the supervised stage reaches the reinforcement-learned level without reinforcement learning, and on the clean benchmark exceeds it at later epochs. The result therefore reproduces on a second backbone. On the held-out Real-IAD split of Appendix L the same checkpoint reaches 84.03%, above every Qwen2.5-VL row there.

Three qualifications belong with the table. First, not every configuration clears the released IAD-R1 checkpoint. On VisA the Gemini-written corpus reaches 68.26% and the corpus carried over from Qwen2.5-VL reaches 70.97%, and both sit below IAD-R1’s 71.34%. Only the two later configurations clear it. Second, that released checkpoint is a Qwen2.5-VL-7B model, so the reference row compares our LLaVA-OneVision models against IAD-R1’s public artefact and not against an IAD-R1 model on the same backbone. No full-benchmark score for an IAD-R1 LLaVA-OneVision checkpoint exists under our harness. Third, the corrective corpus’s DS-MVtec score peaks at epoch 2 and its VisA score at epoch 4, so no single epoch is best on both, and the DS-MVtec rule picks the earlier one.

A defect in the corpus behind the winning row. The corpus used for the winning row is not constructed the way the Qwen2.5-VL KCR corpus is, and the difference matters for the claim this section makes. It was drawn from the whole 10,236-image rollout pool rather than from the 6,000-image SFT split, so only 3,516 of its 6,000 images are shared with the SFT split and the remaining 2,484 come from the GRPO split. It was also balanced on whether the file path contains an anomalous folder name rather than on the verdict the trace itself teaches, which leaves it at 45.0% anomalous rather than the intended even split. The Qwen2.5-VL KCR corpus has neither problem. All 6,000 of its images lie inside the SFT split and it is exactly balanced. This section therefore compares two backbones whose corpora differ in image set as well as in author, which is one factor more than intended. A rebuilt corpus that matches the Qwen2.5-VL construction exactly, 6,000 images all inside the SFT split and balanced on the trace verdict, has been trained and is reported in Appendix M. At epoch 4 it reaches 74.29% on VisA against 74.25% for the first build, and 86.60% against 88.45% on DS-MVtec. The VisA replication therefore stands on a clean corpus. The rows above use the corrected build, and the first build is kept on record in Appendix M.

On the two generation paths. Some rows were produced through the HuggingFace generation path and some through vLLM, so we measured the two against each other on identical weights over the full benchmarks. On the four LLaVA-OneVision rows they agree on 93.9 to 98.4% of per-sample verdicts and every row moves by less than 0.6 points between paths. The VisA ordering is identical under both. On DS-MVTec the two middle rows exchange places, 87.70 against 87.74 under HuggingFace and 87.84 against 87.66 under vLLM, a margin of at most 0.18 points that should be read as a tie. The first and last rows, which carry the argument, are unaffected. Agreement is lower on the Qwen2.5-VL and Qwen3-VL checkpoints, at 79.6 to 94.3% over the fourteen evaluation files run through both paths, so this parity statement is scoped to this backbone. The likely cause is a decoding difference, the repetition penalty of 1.05 that the HuggingFace path applies from the Qwen2.5-VL release configuration and the vLLM path does not (Appendix J.2).

A third backbone, by corpus transfer only. The last block adds Qwen3-VL-8B. That backbone was never rolled out from, so it has no corpus of its own and is trained on the corpus Qwen2.5-VL wrote. The block therefore tests whether the corpus transfers to a new backbone, not whether the loop reproduces on one. The lift over its own base is large on both benchmarks, over seven points on DS-MVTec and over twelve on VisA. Its 76.45% is the highest VisA figure anywhere in this thesis, above the 75.18% of the zero-shot Gemini 2.5-Flash teacher in Table 6.16 and above every model we trained on the other two backbones.

Framing. This section is a replication, not a new headline. The headline of this thesis stays the Qwen2.5-VL KCR checkpoint of \$6.6, which is the backbone every controlled study uses and whose DS-MVTec column carries no contamination caveat.

6.10. Model Behaviour: Operating Point, Per Product and Cases

Balanced accuracy compresses two numbers into one, and the two can move in opposite directions without the summary noticing. This section looks past the headline at the operating point, at the per-product distribution, and at individual images. It uses the per-sample verdicts of the same evaluation files that produce every table above.

6.10.1. The Operating Point

Figure 6.6 decomposes every row of Figure 6.1 into sensitivity, the share of defects caught, and specificity, the share of good parts passed.

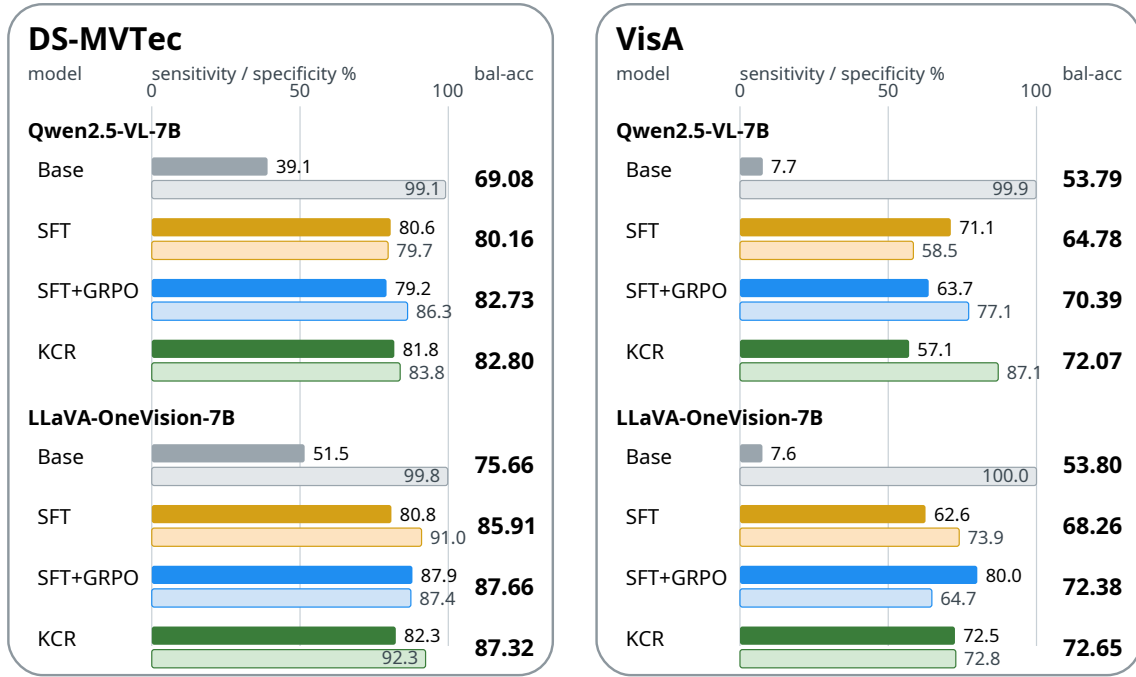


Figure 6.6: Sensitivity and specificity for every stage and both backbones. In each pair the upper solid bar is sensitivity, the share of defects caught, and the lower pale bar is specificity, the share of good parts passed. Balanced accuracy is their mean. On VisA the two KCR models reach a similar score from different operating points, 57.1 % and 87.1 % on Qwen2.5-VL against 72.5 % and 72.8 % on LLaVA-OneVision.

Two observations follow, and neither is visible in the headline table.

The two base models fail in the same specific way. On VisA both untrained 7B backbones sit at about 7.6 % sensitivity and essentially 100 % specificity. They are not making different mistakes. They have both defaulted to answering no.

The two KCR models reach a similar score from different operating points. On VisA the Qwen2.5-VL KCR checkpoint catches 57.1 % of defects and passes 87.1 % of good parts. The LLaVA-OneVision KCR checkpoint is balanced, at 72.5 % and 72.8 %. Their balanced accuracies differ by only 0.58 points, so the summary metric treats them as near neighbours, while in a factory they would behave differently. The Qwen2.5-VL model would let more defects through and rarely stop a good part. The LLaVA-OneVision model would catch more defects and halt the line more often. Which is preferable depends on the relative cost of a missed defect and a false alarm, which this benchmark does not encode. The same caution applies within a backbone. Going from SFT+GRPO to KCR on Qwen2.5-VL raises VisA balanced accuracy by 1.68 points, and it does so by trading 6.6 points of sensitivity for 10.0 points of specificity. That is a real change in behaviour reported as a small change in score.

6.10.2. Per Product

Figure 6.7 shows that the aggregate gain is not evenly spread. On the Qwen2.5-VL backbone the VisA gain over SFT is +7.3 points overall, but it ranges from +23.2 on pcb4 and +10.5 on macaroni2 down to +0.5 on fryum. On DS-MVTec the overall gain is +2.6 points and six of fifteen products regress, with pill losing 17.7 points, transistor 7.9 and leather 7.8. The LLaVA-OneVision backbone is steadier, with smaller deltas that are more often positive, though pcb3 loses 7.5 points on VisA. On VisA the Qwen2.5-VL backbone is the one exception to the pattern, with all twelve products improving. Two things follow. The corrective corpus is not a uniform improvement, and a product-level regression of that size would matter to a deployment even when the aggregate improves. The two backbones also do not gain on the same products, which is consistent with each corpus being written by the backbone that produced it. Full per-product tables for every stage are in Appendices C and D.

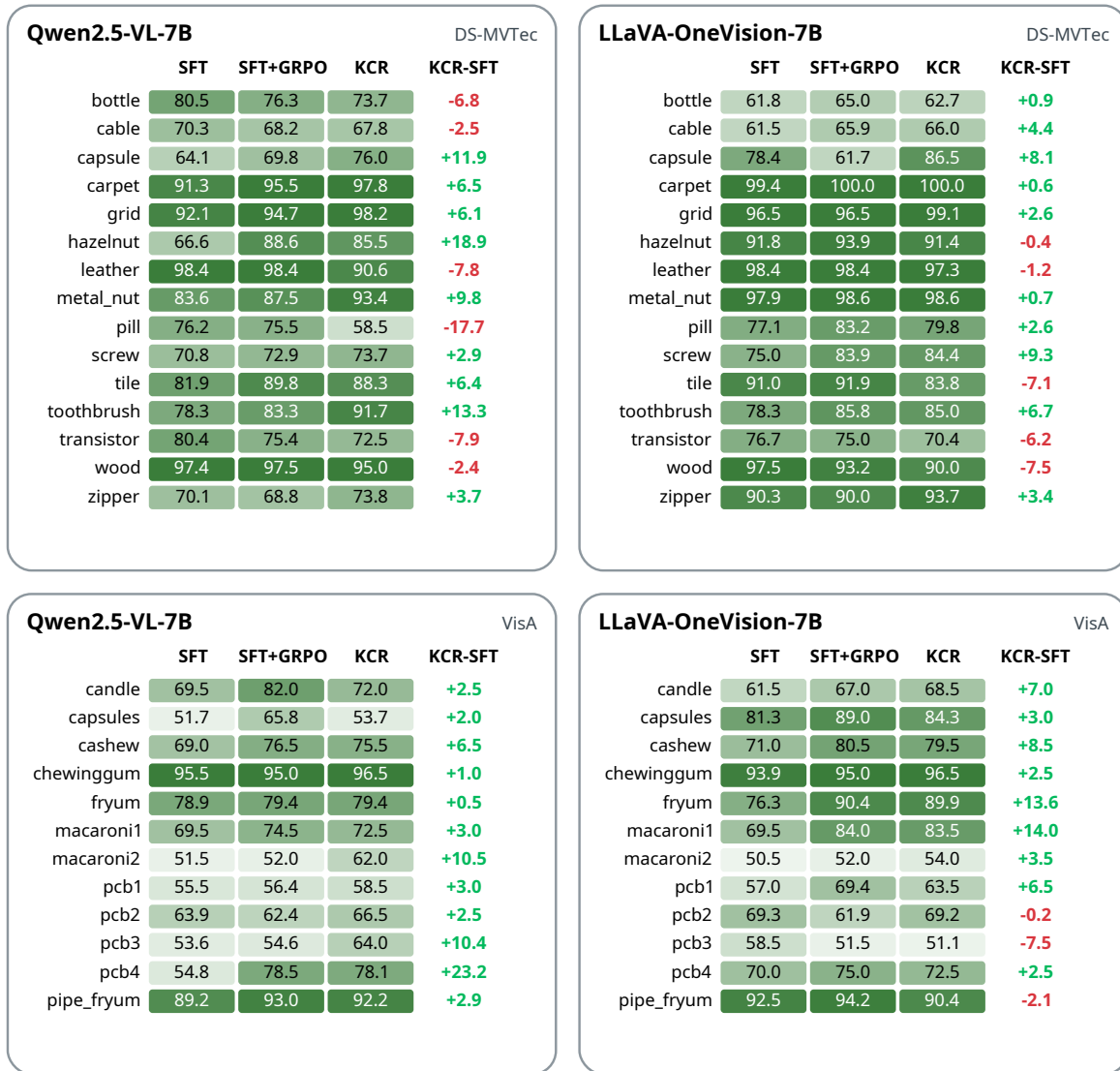


Figure 6.7: Per-product balanced accuracy for the three trained stages, both backbones and both benchmarks. Each row is one product, the three cells are the three trained stages, shaded from 50 (white) to 95 (dark green). The right-hand column is the KCR gain over SFT in percentage points, green when positive. The overall gain is positive in all four panels, and individual products move in both directions, which the overall number hides.



Figure 6.8: Per-product balanced accuracy of the untrained Qwen2.5-VL-7B and of the KCR model on DS-MVTec (top) and VisA (bottom), sorted by gain, with the gain in percentage points above each pair. Strict scoring on the same evaluation files as Tables 6.2 and 6.8. On DS-MVTec twelve of fifteen products improve, led by `metal_nut` (+35.6), `toothbrush` (+29.2) and `hazelnut` (+24.8), and three regress by at most 3.2 points on products the base model already scored high. On VisA every product improves, and the base sits at chance on nine of the twelve, the all-no behaviour of \$6.2.

Figure 6.8 puts the same products against the untrained backbone. It is the whole distance the pipeline travels per product. The largest gains are on the products the base model handled worst, `metal_nut`, `toothbrush` and `hazelnut` on DS-MVTec and `pipe_fryum`, `chewinggum` and `fryum` on VisA, and the three DS-MVTec regressions, `pill`, `wood` and `bottle`, are within 3.2 points on products where the base already scored 60 to 98.

6.10.3. Case by Case

Aggregates and distributions still say nothing about individual decisions. Figure 6.9 shows six images on which the three trained Qwen2.5-VL stages disagree. Every one carries a real defect.

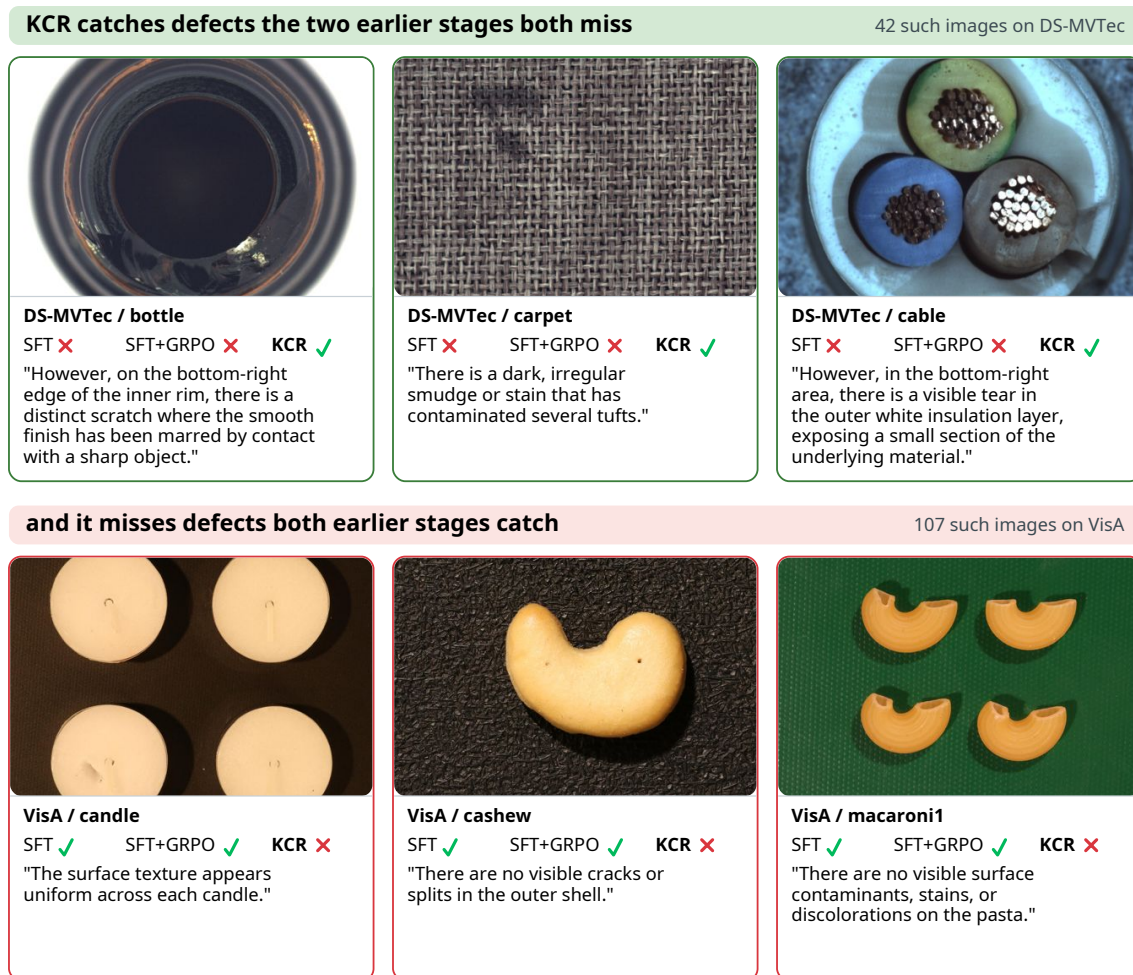
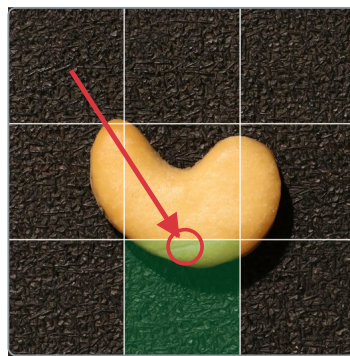


Figure 6.9: Six Qwen2.5-VL cases where the three trained stages disagree, with the model's own trace quoted verbatim. In the upper row the KCR checkpoint names specific visual evidence and the two earlier stages miss the defect. In the lower row it asserts uniformity on an image that carries one. There are 42 images of the first kind on DS-MVTec and 107 of the second on VisA.

The counts behind the figure make the trade concrete. On DS-MVTec, 42 images are called correctly by the KCR checkpoint and by neither earlier stage, and 42 go the other way. On VisA the figures are 121 and 107. Those VisA sets are not symmetric in composition. Of the 121 images the KCR checkpoint uniquely gets right, most are normal images it correctly passes. Of the 107 it uniquely gets wrong, almost all are defects it misses. That is the same sensitivity-for-specificity trade seen in Figure 6.6, now visible one image at a time, and it is why the aggregate improvement should not be read as a uniform improvement. Figure 6.10 shows two of these disagreements with both traces in full, chosen from the 65 anomalous images on which the supervised model answers no, the corrective model answers yes, and the corrective model's `<location>` tag overlaps the mask's cells. Appendix F.5 gives 21 of them, selected by hand from that pool to cover twelve products.



cashew, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting a single cashew nut. The overall shape is a characteristic semi-circular or crescent form with a smooth, rounded surface. I'll begin by checking the outer perimeter; the edges are clean and show no signs of chipping, cracking, or irregular molding. The top surface has a fine, granular texture typical of this product, and I don't see any deep scratches, pits, or foreign material embedded in the shell. The color is a consistent light tan/yellowish hue without any dark spots that might indicate mold or contamination. There are no visible cracks radiating from the center, which would suggest structural failure. The symmetry of the nut is maintained, and the thickness appears uniform across the visible face.

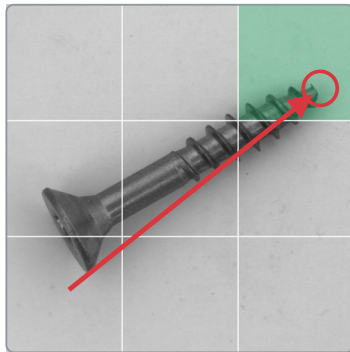
Based on these observations, the cashew meets all standard quality requirements.

KCR

<answer> yes, <type> missing parts, <location> bottom-center

I am inspecting a single cashew nut against a dark textured background. The overall shape of the nut is a standard half-moon form with a smooth, light-colored surface. I'll scan the perimeter first to check for structural integrity.

On the bottom edge, near the center, there is a clear indentation or missing piece on the otherwise smooth curve. This area shows a slight flattening and a break in the continuity of the outer shell, indicating that the nut has been chipped or broken. The rest of the surface appears uniform without any visible cracks, stains, or discoloration. The defect on the lower edge disrupts the expected symmetrical profile of the cashew.



screw, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this screw. I'll start with the head; the flat surface is smooth and the outer edge is well-defined without any burrs or stripping marks from a screwdriver. The cross-recess pattern on the top-left is clearly visible and correctly formed. Moving to the shank, the metal has a consistent texture and color, showing no signs of corrosion or oxidation. The spiral grooves along the length of the shank are uniform and free of debris or manufacturing defects. I'm checking the overall straightness of the screw; it shows no signs of bending or twisting. There are no scratches or dents on the side surfaces.

The part appears to be in excellent condition.

KCR

<answer> yes, <type> chip, <location> top-right

I am inspecting a metal screw with a hexagonal head and a spiral drive. I'll start by checking the overall shape; it appears straight and properly formed. The hexagonal head has clean edges without any signs of deformation or crushing. Looking at the side profile, the spiral grooves on the neck are uniform in depth and spacing.

However, upon closer inspection of the top-right edge of the screw's neck, there is a clear structural defect.

A small piece of the metal has been chipped away from the outer edge of one of the spiral ridges. This missing material breaks the continuity of the drive pattern and would likely interfere with the screw's ability to grip a surface. The rest of the surface shows some minor pitting typical of industrial metal, but this specific chip on the drive edge is a critical failure.

Figure 6.10: Two images on which the supervised model misses the defect and the corrective model finds it, with both traces in full. Top, VisA cashew 086, a chipped edge. Bottom, DS-MVTec screw 017, a chipped thread. The red circle encloses the ground-truth mask, one circle per separated region, with the arrow pointing to the largest, and the green shading marks its 3×3 cells. The shaded sentence is where each trace commits: the supervised trace scans the surface, finds it uniform and calls the part clean, and the corrective trace names the cue, the type and the cell. Appendix F.5 holds 21 such pairs.

6.10.4. Localisation

The <location> tag is the one output that balanced accuracy never sees, and it is the one an operator would act on. Table 6.14 scores it on the full benchmarks, with the same rule as the judge of §6.11: the predicted 3×3 cells must overlap the cells that hold more than five percent of the ground-truth mask. The first column of each benchmark conditions on the anomalies the model detected, so it isolates localisation from detection. The second counts a missed defect as a missed location.

Table 6.14: Share of anomalies whose predicted `<location>` overlaps the ground-truth mask cells, on the full DS-MVTec (1,226 anomalies) and VisA (1,197) sets, for the three trained stages of each backbone at the checkpoints of Figure 6.1. “Of detected” conditions on the model’s own true positives. “Of all” divides by every anomaly. [†] The DS-MVTec cells on this backbone carry the contamination caveat of §6.1.

Stage	DS-MVTec		VisA	
	of detected	of all	of detected	of all
Qwen2.5-VL-7B				
SFT	73.2 %	59.0 %	38.1 %	27.1 %
SFT+GRPO	82.4 %	65.3 %	55.9 %	35.6 %
KCR	74.9 %	61.3 %	52.7 %	30.1 %
LLaVA-OneVision-7B [†]				
SFT	77.5 %	62.6 %	54.2 %	33.9 %
SFT+GRPO	83.1 %	73.1 %	71.2 %	57.0 %
KCR	83.1 %	68.4 %	63.7 %	46.2 %

Two things follow. The reinforcement-learning stage is the one that teaches localisation. On both backbones and both benchmarks the SFT+GRPO checkpoint has the highest hit rate among its detections, by 9.2 and 17.8 points over the supervised model on Qwen2.5-VL and by 5.6 and 17.0 on LLaVA-OneVision. That is the location sub-reward of §5.2 doing its job, and it is a contribution of GRPO that the verdict metric never records. The corrective corpus keeps part of it. KCR sits between the supervised and the reinforcement-learned checkpoint in every cell but one, where it ties, so the localisation learned under the reward survives the rewrite and retrain only in part. Read together with §6.8, this is the most precise statement of what the reinforcement-learning stage adds on this task. It does not raise the verdict on a strong initialisation, and it does raise the location, which the corpus built from its rollouts then carries forward at a discount. The `<type>` tag does not follow the same pattern. Scored the same way on the full benchmarks, its similarity to the MMAD label is flat across the three stages, because the models name Real-IAD’s eight classes and MMAD labels defects in its own product-specific terms (Appendix J.5).

6.11. Explanation Quality: a Reasoning-Faithfulness Comparison

Everything so far measures the binary verdict. A central claim of this thesis is that the model produces an explainable structured rationale, so the explanation itself has to be measured. We score seven of our own models, spread over three backbones, against the released IAD-R1 Qwen2.5-VL-7B checkpoint [39], all on one shared set of images that every one of them detects correctly, which isolates reasoning quality from detection accuracy and makes every comparison a paired one.

Protocol. The shared set is drawn from the anomalies that all nine rows of Table 6.15 detect correctly, 718 images on DS-MVTec and 390 on VisA. From that intersection we draw six images per product on DS-MVTec, 90 images over the fifteen products, and four per product on VisA, 47 images over the twelve products because `pcb1` has only three, with a fixed seed. Each trace is scored by a Gemini-3-Flash judge, a newer model generation than the Gemini 2.5-Flash trace teacher, so the judge did not write the traces the SFT rows were trained on. It is the same model family that corrected and rewrote the KCR corpus (§5.8), so for the KCR rows the judge shares a lineage with the corrector, and a preference for its own phrasing cannot be excluded. The shared-image design and the mechanical *Loc* and *Type* columns do not depend on the judge. The judge sees the original image, an overlay with the true defect region in red, and the ground-truth defect type from the MMAD annotation. It is blind to which model wrote the trace and is told the verdict is already correct. It therefore grades only reasoning quality, on five axes scored 0 to 2. The axes are visual grounding, defect faithfulness, evidence before conclusion, coherence and conciseness, reported as their sum out of 10, and each score is the median of three judge samples. Separately, and mechanically rather than by the judge, we score *Location* as the predicted 3×3 cell against the ground-truth mask, with the rule of Table 6.14, and *Type* as `nomic-embed-text-v1.5` [49] similarity between the predicted `<type>` and the MMAD defect type. Where MMAD carries no ground-truth type, five of the 90 DS-MVTec images and none on VisA, the *Type* score falls back to the label from the image folder name, identically for every model.

Why a protocol of our own. No published benchmark scores the faithfulness of a free-text inspection rationale. MMAD’s defect-description and defect-analysis subtasks are multiple choice and cannot score a trace, and the VisA-D&R benchmark of Anomaly-OV [75] lets a language model grade descriptions without the defect mask and without separate axes. The protocol above is therefore ours. Its ingredients follow recent judging practice: a reference-guided, decomposed rubric [32, 56], error scoring against a ground-truth reference as in radiology report evaluation [51], a vision-language model as the critic [73], and the faithfulness against fluency framing of Jacovi and Goldberg [28] and Khalifa et al. [31].

Table 6.15: Explanation quality on the shared image set, 90 DS-MVTec and 47 VisA anomalies that every row detects correctly. *Judge* is the blind Gemini-3-Flash score, the median of three samples per trace. *Loc* and *Type* are scored mechanically. Rows are grouped by backbone and a corpus written by a different model is named in brackets. The LLaVA-OneVision KCR row is the corrected corpus of Appendix M. [†] The DS-MVTec cells of the LLaVA-OneVision rows carry the contamination caveat of §6.1.

Model	DS-MVTec ($n = 90$)			VisA ($n = 47$)		
	Judge / 10	Loc	Type	Judge / 10	Loc	Type
Qwen2.5-VL-7B, SFT	9.17	0.77	0.56	8.70	0.38	0.55
Qwen2.5-VL-7B, SFT+GRPO	9.33	0.82	0.56	9.23	0.64	0.54
Qwen2.5-VL-7B, KCR	9.09	0.78	0.53	9.11	0.55	0.56
Qwen3-VL-8B (Qwen2.5-VL KCR corpus)	9.41	0.82	0.56	9.53	0.62	0.54
LLaVA-OneVision-7B, SFT [†]	9.36	0.78	0.58	8.57	0.49	0.56
LLaVA-OneVision-7B, SFT+GRPO [†]	9.47	0.81	0.55	9.43	0.70	0.53
LLaVA-OneVision-7B, KCR [†]	9.61	0.86	0.56	9.34	0.60	0.55
IAD-R1, our prompt	6.13	0.74	0.55	6.96	0.68	0.55
IAD-R1, its own prompt	4.36	0.34	0.30	6.49	0.62	0.53

The gap to IAD-R1 is large and it survives the caveats. Because every row is scored on the same images, a difference between two rows is the mean of per-image differences, which we report with its 95% interval: the mean plus or minus 1.96 standard errors of those paired differences over the 90 or 47 images, each image scored by the median of the three judge samples. Against the Qwen2.5-VL KCR checkpoint, IAD-R1 under our prompt scores 2.96 points lower on DS-MVTec, interval 2.41 to 3.50, and 2.15 lower on VisA, interval 1.47 to 2.83. The VisA half carries no contamination caveat, so the gap is not explained by what any backbone saw during pretraining. IAD-R1 is given our prompt because its own released prompt does not ask for reasoning at all. Under its own prompt it scores 4.36 and 6.49, and 42 of its 90 shared DS-MVTec outputs are the single word “Yes” with no reasoning to score. Both prompts are in the table and the prompt-matched row is the fair one. Scoring a model on reasoning it was never asked to produce is not a fair comparison.

Among our own models the differences are small, and the ones that hold up follow the training stage. The seven models we trained sit within 0.5 points of each other on DS-MVTec and within one point on VisA. Four of the five axes are near their maximum for every trained model, and evidence before conclusion sits between 1.89 and 2.00, so the rubric has little room left to discriminate. Two differences survive the paired comparison. On VisA, the benchmark without a contamination caveat, the two supervised-only rows are the weakest explainer, 8.70 for Qwen2.5-VL and 8.57 for LLaVA-OneVision, and every row that went through the reinforcement-learning stage or the corpus built from its rollouts sits between 9.11 and 9.53. On LLaVA-OneVision the step from SFT to SFT+GRPO is worth 0.85 points on the same 47 images, interval 0.26 to 1.44, and the step to KCR 0.77, interval 0.17 to 1.37. The mechanical location column moves the same way, 0.38 to 0.64 to 0.55 on Qwen2.5-VL and 0.49 to 0.70 to 0.60 on LLaVA-OneVision, which is the ladder of Table 6.14 measured on identical images. On DS-MVTec the LLaVA-OneVision KCR row is the highest, 0.52 above the Qwen2.5-VL KCR row, interval 0.19 to 0.85, under the contamination caveat. Beyond these, the remaining pairs are within their intervals of zero, so we claim no ranking among the Qwen rows, and no difference between a Gemini-written corpus and a corrective one on DS-MVTec.

One matched case. Figure 6.11 works the whole protocol through on a single VisA image that both the Qwen2.5-VL KCR checkpoint and IAD-R1 call correctly, so only the explanation differs. It shows what the

aggregate numbers are made of.

Limits. Beyond the contamination caveat we stress three limits. The judge measures grounding consistency, meaning plausibility against the ground-truth evidence, and not mechanistic faithfulness. An input-perturbation flip test is left to future work. Each score is the median of three judge samples, which damps the judge’s variance without removing it. And the shared set holds the defects that all nine rows detect, which are the easier ones, so the absolute scores are higher than they would be on each model’s full set of detections and the comparison is one of explanation quality at matched difficulty rather than of coverage. An earlier version of this experiment scored each model on its own 100 correctly detected anomalies and gave the same ordering against IAD-R1. The per-axis scores and raw outputs are in `outputs/explainability_shared/`. Appendix K sets out an extended rubric that generalises the five-axis protocol used here. It is proposed as future work and was not run for the numbers above.

6.12. Comparison to the State of the Art

Table 6.16: Comparison on the MMAD DS-MVTec and VisA subsets. Every row, ours and the references alike, is evaluated under one identical harness on the same 1,670 and 2,141 images, strict scoring. Dates in parentheses are release dates. The four Gemini 3.x Flash rows are models released between May and September 2026, evaluated in September 2026 at the lowest thinking level they accept (“low”). Bold marks our headline model. [†] LLaVA-OneVision rows carry the DS-MVTec contamination caveat of §6.1. ^{*} The first KCR build on that backbone, assembled from a different 6K pool than the corrected build (Appendix M), still the same Keep-Correct-Revise recipe, epoch 4. [¶] Scored under the GRPO training prompt. Under the supervised prompt this checkpoint scores 80.74 % and 70.50 % (§6.1).

Method	Training paradigm	DS-MVTec	VisA
Qwen2.5-VL-7B (base)	no fine-tuning	69.08 %	53.79 %
Gemini 2.5-Flash (trace teacher, April 2025)	zero-shot, structured prompt	81.52 %	75.18 %
GPT-5-mini [50]	zero-shot, structured prompt	77.10 %	68.23 %
Gemini 3.5-Flash (May 2026)	zero-shot, structured prompt	89.28 %	76.44 %
Gemini 3.6-Flash (July 2026)	zero-shot, structured prompt	90.60 %	82.01 %
Gemini 3.7-Flash (August 2026)	zero-shot, structured prompt	89.32 %	81.04 %
Gemini 3.8-Flash (September 2026)	zero-shot, structured prompt	89.52 %	79.66 %
Ours (15K traces, ep3)	SFT only	71.66 %	64.28 %
Ours (6K traces)	SFT only	80.16 %	64.78 %
IAD-R1 [39], released ckpt	SFT + GRPO	81.92 %	71.34 %
Ours (6K + GRPO) [¶]	SFT + GRPO	82.73 %	70.39 %
Ours (KCR, corrective 6K)	SFT only	82.80 %	72.07 %
Ours, LLaVA-OneVision-7B (6K + GRPO) [†]	SFT + GRPO	87.66 %	72.38 %
Ours, LLaVA-OneVision-7B (KCR, corrective 6K) [†]	SFT only	87.32 %	72.65 %
Ours, LLaVA-OneVision-7B (KCR, first 6K build) ^{†*}	SFT only	88.45 %	74.25 %

On the IAD-R1 numbers. All IAD-R1 numbers in this thesis come from running the released checkpoint through the same harness we use for our own models. We do not quote the numbers from the IAD-R1 paper, because we cannot reproduce the evaluation conditions used there, meaning the sub-sample selection, the prompt template, the answer parser and the decoding hyperparameters. A comparison against numbers obtained under a different harness is not a controlled one. The rows above describe how the publicly available checkpoint behaves under our pipeline, which is the only comparison we can run cleanly.

- Our best model, the KCR checkpoint, exceeds the released IAD-R1 checkpoint on both benchmarks under a common harness, by 0.88 points on DS-MVTec and 0.73 on VisA, using supervised fine-tuning alone.
- Our SFT+GRPO model exceeds IAD-R1 on DS-MVTec by 0.81 points but trails it on VisA by 0.95. The KCR checkpoint closes that gap and reverses it, which is what §6.8 would lead one to expect. On this backbone the reinforcement-learning stage is not what carries the VisA column.
- Even the plain 6K supervised model reaches within about 1.8 points of IAD-R1 on DS-MVTec with no reinforcement-learning step at all.

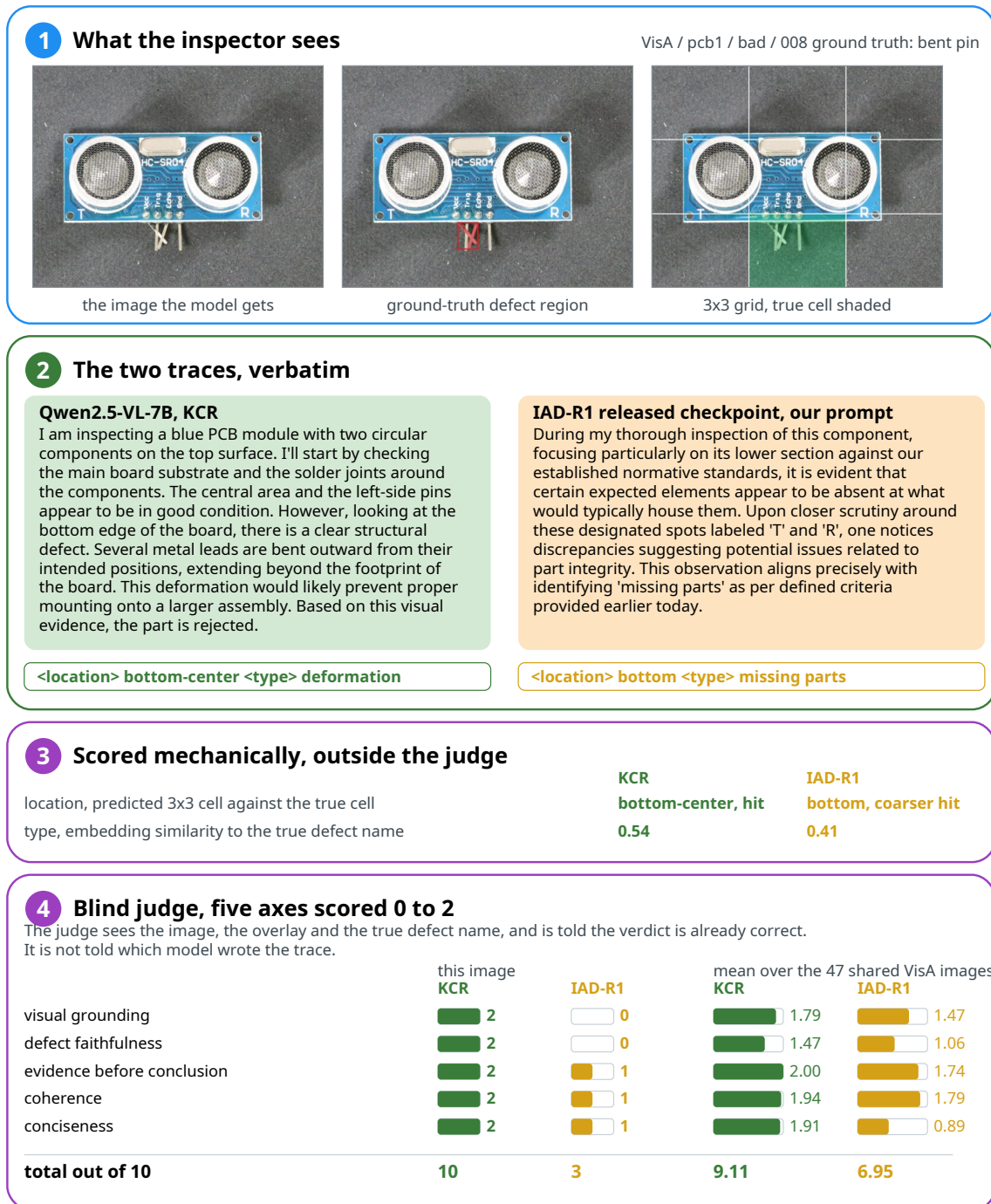


Figure 6.11: The explanation protocol on one matched VisA image, pcb1/bad/008, whose ground-truth defect is a bent pin and which both models call correctly. Our trace names the region and the visual cue before concluding, and its <location> tag hits the exact ground-truth cell. The IAD-R1 trace reports missing parts, which is not the defect the mask shows, and the blind judge scores it zero on visual grounding and zero on defect faithfulness. The per-axis means beside the single-case scores are over the 47 shared VisA images that underlie Table 6.15.

The student passes its teacher on one benchmark. We benchmark the trace-generation teacher itself, Gemini 2.5-Flash, zero-shot under the same harness on the identical sample set, with a structured prompt and no thinking budget. At inference the teacher sees only the test image, because the privileged mask overlay used during trace generation (§3.3) is not available, so this is a like-for-like comparison. The teacher reaches 81.52% on DS-MVTec and 75.18% on VisA. On DS-MVTec that sits below our best 7B students. The KCR checkpoint at 82.80% and the single-stage GRPO model at 82.73% both exceed the proprietary model that generated their training traces, and even the plain 6K model lands within 1.4 points of it. On VisA the teacher still leads both headline Qwen2.5-VL models, by 3.11 points over the KCR checkpoint and 4.79 over the SFT+GRPO one, and it does so by being markedly conservative, at 60.3% sensitivity and 90.0% specificity. It is not ahead of everything we trained. The Qwen3-VL-8B model of \$6.9 reaches 76.45% on VisA, above the teacher, on a corpus the teacher never wrote. Distillation therefore closes the gap to the teacher on the texture and structure regime and reverses it on DS-MVTec, but not on the multi-object VisA regime.

A second proprietary reference. GPT-5-mini [50], called as gpt-5-mini through the OpenAI API in June 2026 with reasoning effort low and a 2,000-token output cap, under the identical harness, prompt and sample set, zero-shot, reaches 77.10% and 68.23%. That is below both Gemini 2.5-Flash and our KCR model on both subsets. Its error profile is the mirror image of Gemini’s. Where Gemini is conservative, GPT-5-mini overcalls, at 96.1% DS-MVTec sensitivity and only 58.1% specificity. The two references bracket the task from opposite failure modes. A 7B model fine-tuned on 6,000 corrective in-distribution traces beats GPT-5-mini on both subsets, by 5.70 and 3.84 points, and beats Gemini 2.5-Flash on DS-MVTec by 1.28. It does not beat Gemini 2.5-Flash on VisA, where it trails by 3.11. The pretraining caveat of \$6.1 applies to every row.

Recently released references. Four Gemini 3.x Flash models were released while this thesis was being written, between May and September 2026, and were evaluated under the identical harness and prompt shortly before submission, at the lowest thinking level they accept. They reach 89.3 to 90.6% on DS-MVTec and 76.4 to 82.0% on VisA, with sensitivity between 94 and 97% and specificity between 82 and 87% on DS-MVTec. That places them 6.5 to 7.8 points above the Qwen2.5-VL KCR model on DS-MVTec and 4.4 to 9.9 points above it on VisA, and 1.1 to 3.3 points above the LLaVA-OneVision KCR row on DS-MVTec, where that row carries the contamination caveat. The same caveat as for every row applies, both benchmarks are public and may have been seen in pretraining. Our student therefore beats its own teacher and the proprietary references of its generation, and trails the frontier models released during the last four months of the project by the margin one would expect between a fine-tuned 7B open model and a frontier model. It does so with a rationale for every verdict, on two GPUs and at no per-image API cost, and the released corpus is the tool for narrowing that margin further.

6.13. Training and Reward Dynamics

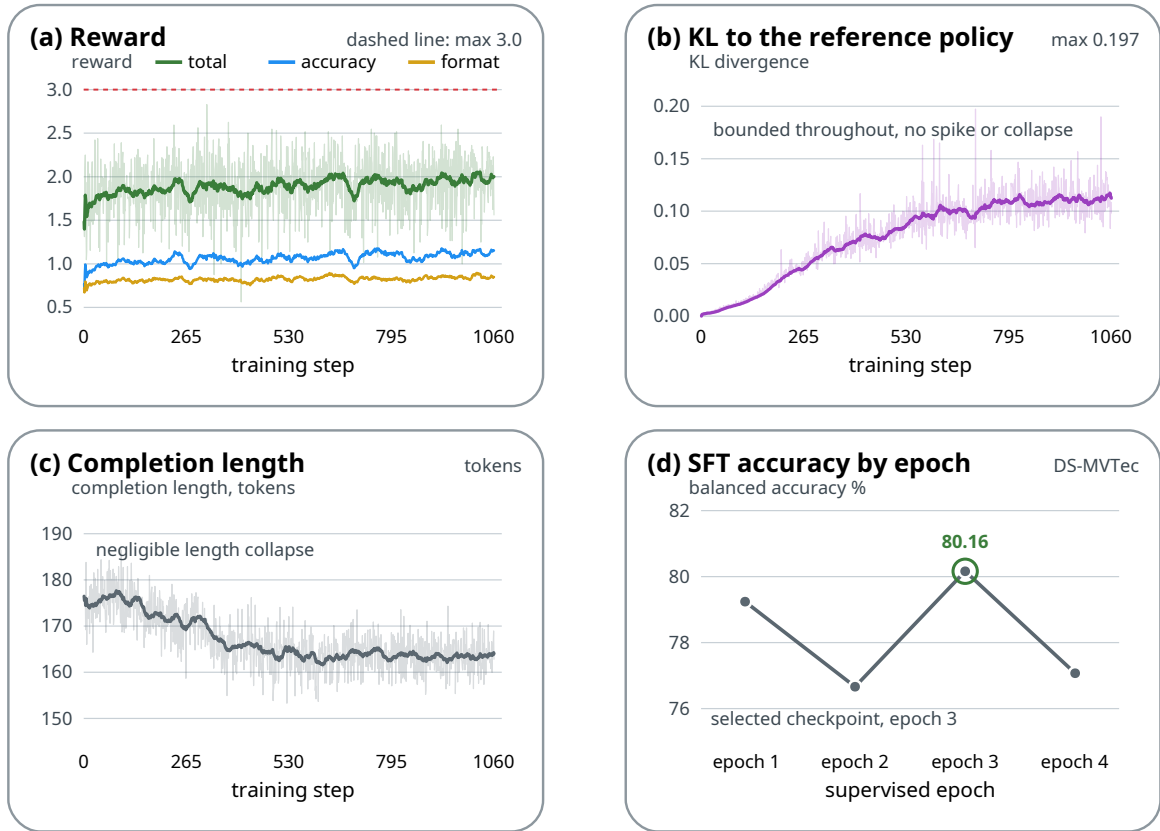


Figure 6.12: Dynamics of the SFT+GRPO run and of the supervised stage that initialises it. **(a)** The total reward and its two components, against the maximum of 3.0. **(b)** The KL to the reference policy, which stays bounded with no spike or collapse. **(c)** Completion length, with a negligible length collapse. **(d)** Validation balanced accuracy of the supervised run by epoch, with the selected checkpoint ringed. Faint traces are the raw per-step values and solid lines a 25-step trailing mean.

Figure 6.12 shows all four series.

Supervised loss. For the headline 7B-frozen-6K run the cross-entropy loss falls from 1.14 at step 50, through 0.58 at step 500, to 0.34 at step 752 at the end of epoch 4. Four roughly linear phases are visible: a rapid descent over the first 200 steps, a plateau between steps 250 and 550, a second descent between 600 and 700, and saturation in the last 50. Validation balanced accuracy on DS-MVTec tracks the loss and peaks at epoch 3. Epoch 4 is slightly worse, which is mild overfitting and motivates selecting ckpt-564.

Reward. For the SFT+GRPO run the reward and its two components evolve as follows. The combined reward goes $1.66 \rightarrow 2.22 \rightarrow 2.13$ across steps 10, 100 and 530, against a maximum of 3.0. The accuracy reward rises from 0.875 to 1.25, a relative gain of 42.9%, and the format reward from 0.781 to 0.875. The KL divergence rises from zero and stays bounded, stabilising between 0.10 and 0.20 with a full-run maximum of 0.197, with no spike or collapse. Completion length stays in the 153 to 186 token band, drifting from about 175 to 163 tokens over the run, a negligible length collapse. A combined reward that saturates while the KL stays bounded is the standard healthy profile. The policy finds higher-reward regions while staying close to its reference.

Why stop at ckpt-530. The further-trained ckpt-1060 underperforms ckpt-530 by about 0.7 points while its smoothed training reward is slightly higher. We attribute this to mild over-optimisation of the format reward at the expense of the type and location rewards, which the evaluation metric does not reward directly but which proxy generalisation. §6.8 shows the same decoupling in a controlled setting. Termination criteria

are discussed in §7.4, where we argue for stopping on held-out balanced accuracy rather than on a held-out reward verifier.

6.14. Chapter Summary

Five results carry this chapter.

Supervised fine-tuning does most of the work. The Qwen2.5-VL-7B base model reaches 69.08 % and 53.79 % balanced accuracy on DS-MVTec and VisA. Supervised fine-tuning on 6,000 Gemini-written traces takes that to 80.16 % and 64.78 % (§6.3). The single largest factor in that stage is not model size or encoder treatment but which traces are in the corpus.

What reinforcement learning adds depends on where it starts. On the weak supervised baseline GRPO adds 2.57 and 5.61 points (§6.4). On the strong KCR baseline the same recipe subtracts from every checkpoint on DS-MVTec while the training reward rises, and a three-way estimator ablation finds the drop is not an artefact of the advantage normalisation (§6.8).

The best model is supervised. Of the three teacher-intervention recipes, the one that corrects failed rollouts and revises weakly grounded ones reaches 82.80 % and 72.07 % with no reinforcement-learning step (§6.6). That matches the published SFT+GRPO headline on DS-MVTec to within 0.07 points and exceeds it on VisA by 1.68. The corpus has to be trained from base rather than on top of the reinforcement-learned policy (§6.7).

The parity reproduces on a second backbone. Run again on LLaVA-OneVision-7B-SI, the corrective corpus matches that backbone's own SFT+GRPO model at the selected epoch, -0.34 and $+0.27$ points, and exceeds it on VisA by 1.91 points at epoch 4 (§6.9). The VisA half of that carries no contamination caveat.

The aggregate number hides the behaviour. The two best models reach a similar VisA score from opposite operating points, and the corrective corpus trades sensitivity for specificity rather than improving both (§6.10). Explanation quality separates our models from the released IAD-R1 checkpoint by 2.2 to 3.0 points on identical images, while our own models sit within one point of each other and the supervised-only rows explain least well on VisA (§6.11).

What reinforcement learning does add. On both backbones the SFT+GRPO checkpoint localises best among its detections, by 9 to 18 points over the supervised model, and the corrective corpus retains part of that gain (§6.10.4). The verdict metric does not see this, and it is the clearest contribution of the reinforcement-learning stage in this thesis.

A note on how these numbers were selected. Every checkpoint in this chapter is chosen on DS-MVTec balanced accuracy, which makes the DS-MVTec column a development-set figure rather than a held-out estimate. VisA is never selected on itself. Limitation (L7) in §7.7 states the size of that effect, and the two claims above that carry the thesis are both made on VisA.

7

Discussion

The numbers reported in Chapter 6 answer the four research questions of §1.3 (**RQ2** conditionally rather than affirmatively), but they also surface a number of design choices that deserve explanation, and several failure modes that future work must address. This chapter discusses the strongest patterns in turn: why a frozen vision encoder wins (§7.1), why 6K traces beat 15K (§7.2), why VisA remains harder than DS-MVTec even after GRPO (§7.3), why early stopping mid-GRPO outperforms longer training (§7.4), why the teacher ablation forces us to re-read the value of the GRPO stage itself (§7.5), and the failure-mode taxonomy we observed on the residual error set (§7.6). §7.7 states limitations of the present work.

7.1. Why Frozen > Unfrozen Vision Encoder

Across the matched configurations in Table 6.3, freezing the ViT never loses and on VisA wins decisively, but the effect is strongly asymmetric. At the best-epoch checkpoints the DS-MVTec gap is small (~ 0.5 – 1 pp: 7B 72.60 vs. 72.08, 3B 69.56 vs. 68.58), whereas on VisA freezing helps by ~ 8.3 pp at 7B (66.94 vs. 58.60) and is negligible at 3B. (The much larger 80-vs-72 gap one might recall is the *6K-vs-15K* data-size effect of §7.2, a different axis and not a freezing effect.) The matched-rate control of Table 6.3 removes the learning-rate confound from this reading: 7B-unfrozen on the 6K split at the frozen cells’ rate peaks at 75.97% against 80.16% frozen on DS-MVTec and is level on VisA, so freezing wins whether the unfrozen encoder is trained at its own best rate or at the frozen one. The VisA pattern is the opposite of what naive intuition (“more parameters to update is more capacity”) predicts.

We propose two converging explanations.

First, the visual encoder of Qwen2.5-VL ships with ~ 675 M parameters [4] pre-trained on hundreds of millions of image-text pairs. Even the 15K trace dataset is two orders of magnitude smaller and contains roughly thirty visual modes (one per product), which is too coarse a supervisory signal to retrain that many parameters without overfitting.

Second, and this is the most actionable observation for industrial inspection, the VisA task requires fine-grained reasoning about *multi-object* scenes (PCBs, capsules, snacks). The ViT’s strength on these scenes comes from its CLIP-style pretraining. Unfreezing it during fine-tuning on the (largely single-object) Real-IAD dataset erodes that strength. The most visible symptom in our logs is that the unfrozen 7B model collapses on VisA recall (it predicts “yes” on only 34–41 % of true defects), while the frozen variant maintains ~ 80 % recall.

When would unfreezing be the right choice? If the training dataset were $100\times$ larger (e.g. 1.5M traces) and crossed the threshold where the ViT can no longer be assumed in-distribution, for example a custom hyperspectral imaging modality, then end-to-end fine-tuning could outperform frozen. For 6K–15K Real-IAD traces on RGB inspection imagery, the cost-benefit is clearly in favour of freezing.

7.2. Why the Smaller Split Wins

The 8.5-point gap between 6K and 15K SFT bal-acc on DS-MVTec is the single largest effect in the ablation grid and the most surprising result of the SFT chapter. Three factors together explain it.

(a) A count-and-composition effect, not a length or quality effect. The 6K is a same-length 100% subset of the $\approx 14.5\text{K}$ union (§6.3, §3.5), so the gap is not a “concise vs. verbose” contrast. Nor is it a data-quality effect: an independent GPT-5-mini verifier passes 97.8% of the dataset, and re-training from base on only the answer-correct traces ties the unfiltered 6K to within noise (79.60/65.99 vs. 80.16/64.78 on DS-MVTec/VisA), so there is no low-quality pool for a filter to remove. What differs between 6K and 15K is therefore only *count and composition*: the 15K arm folds the GRPO and held-out splits in as extra SFT data, and at a fixed 4-epoch budget those $\approx 8.5\text{K}$ traces add no benefit.

(b) Epochs vs. examples. At a fixed budget of 4 epochs, each 6K sample is seen 4 times, and each 15K sample is seen 4 times too, but the gradient updates are now $2.4\times$ more numerous. The training logs do not support an under-training reading: the 15K run’s loss plateaus at about 0.58 from epoch 3 onward, while the 6K run reaches 0.57 at epoch 3 and 0.34 at epoch 4. The extra optimiser steps therefore do not buy a lower training loss. Whether a longer schedule or a higher learning rate would change the picture is untested, and at roughly $2\times$ the wall-clock cost the same compute would buy a GRPO stage on top of the 6K SFT.

(c) Structured chain-of-thought generalisation. The 6K traces are, in aggregate, more structurally uniform: their `<think>` block consistently follows the six-phase template, while the 15K-only extension samples are marginally more variable in length and structure (though of essentially the same mean length). The model’s posterior over reasoning trajectories therefore concentrates on a tighter mode on 6K, which is closer to the test-time decoding distribution and improves generalisation.

This observation is operationally important: *the natural way to scale this pipeline is not “generate more traces” but “generate higher-quality traces” or “filter more aggressively”*. It is consistent with the broader trend in instruction tuning toward smaller, curated datasets, of which LIMA’s 1,000-example fine-tuning [90] is the canonical reference. A 65B model tuned on 1,000 curated examples was preferred by human annotators over the same base model tuned on 52,000. Their data-scaling ablation flattened out over a $16\times$ increase in examples. LIMA is a language-only result on open-ended generation quality, so we read it as an analogy rather than as evidence for our benchmark. The direction is the same in both settings: past a point, more examples of the same kind stop paying.

7.3. Why VisA Remains Harder than DS-MVTec

Across every model we evaluate, VisA scores sit well below DS-MVTec: our best model (KCR) reaches 82.80 % on DS-MVTec but 72.07 % on VisA (a 10.7-point internal gap), and our SFT+GRPO model 82.73 % / 70.39 %. This is *not* a deficit relative to the state of the art. Under our common evaluation harness (§6.12) the released IAD-R1 [39] Qwen2.5-VL-7B checkpoint scores 81.92 % / 71.34 %. It exhibits the same large internal gap between DS-MVTec and VisA (10.6 pp) and its absolute VisA score (71.34 %) is no higher than ours. Indeed our best model *exceeds* IAD-R1 on VisA (+0.73 pp) as well as on DS-MVTec (+0.88 pp). (An earlier, non-common-footing evaluation of the IAD-R1 checkpoint suggested a ~ 3.5 pp VisA deficit. That gap was an artefact of differing sample selection and prompt, and disappears once every model is scored on the identical 1,670 / 2,141 set.) The open problem is therefore not closing a gap to IAD-R1 on VisA, but the harder, field-wide challenge of raising *absolute* VisA accuracy for VLM detectors. Three properties of VisA explain why it is intrinsically harder.

First, VisA is dominated by *multi-object scenes*: PCBs with many components, capsules in trays, fryums of varied shape. Detection in such scenes requires the model to count, compare, and reason about layout, not just identify a surface defect. Our six-phase prompt was designed primarily around single-object MVTec-like products and does not explicitly steer the model toward layout reasoning.

Second, VisA defects are *smaller relative to image area* (mean defect area $\approx 1.0\%$ of the image on VisA vs. $\approx 3.8\%$ on DS-MVTec, measured over the 1,197 VisA and 1,226 DS-MVTec ground-truth anomaly masks). The dynamic-resolution Qwen2.5-VL ViT preserves these defects only if the image is encoded at sufficient resolu-

tion. Our pixel budget caps each image at `image_max_pixels = 262,144` ($\approx 512 \times 512$) at SFT and evaluation time, which downsamples the higher-resolution VisA images by roughly $2.5\times$ per side.

Third, our location reward is a binary cell match on the 3×3 grid (§5.2): a tag in the right cell earns the full credit and a tag in the neighbouring cell earns nothing. A graded spatial reward that credits adjacent cells, as IAD-R1’s SC-GRPO does [39], would encourage the precise spatial reasoning that pays off most where defects are small and scenes are cluttered, which is exactly VisA. This is a lever we did not exploit, not a property of the benchmark.

Concrete remediation. Three changes would, we believe, raise absolute VisA accuracy further: (i) raise the per-image pixel budget from 262,144 ($\approx 512^2$) to $\approx 802,816$ ($\approx 896^2$), (ii) replace the binary cell-match location reward with a graded one that credits adjacent cells, as IAD-R1’s SC-GRPO does [39], and (iii) augment the training set with synthesised multi-object VisA-style images. Each targets the small-object / multi-object difficulty above rather than any deficit relative to a competing system.

7.4. Checkpoint Selection and Reward Over-Optimisation

A surprising finding is that the SFT+GRPO checkpoint `ckpt-530` (end of epoch 1) outperforms `ckpt-1060` (end of epoch 2) by ~ 0.7 pp on DS-MVTec, despite the latter achieving a slightly higher smoothed training reward. The classical name for this phenomenon is *reward hacking*. The policy keeps climbing the reward surface, but in directions that no longer improve balanced accuracy, for example by over-optimising the saturated format term at the expense of the type and location sub-signals. (Note it is *not* length inflation: mean completion length is flat over the run, 176 tokens at step 1 and 163 at step 1060, Appendix H.)

Three lessons follow. First, the reward function should be regarded as a proxy for the deployment metric, and the proxy gap will widen as the policy is trained further. A held-out balanced-accuracy verifier, the deployment metric itself and not the reward, is the right termination criterion, precisely because held-out reward inherits the proxy gap. Second, the four-component reward is not perfectly calibrated. In particular the binary format (consistency) reward saturates quickly at its maximum of 1.0, removing its useful gradient and leaving the policy to over-optimize the other sub-signals. Reward shaping that down-weights the format term once it saturates (or gates it) is an obvious next step. Third, the headline run used *no* KL penalty ($\beta = 0$): the k_3 KL was monitored and stayed small even without it, so the policy did not drift over the two-epoch budget. On longer training, where reward hacking becomes more likely, adding a small KL penalty ($\beta \approx 0.05$ – 0.2) is an obvious safeguard worth investigating.

7.5. Revisiting the Value of GRPO

The published headline of this thesis is that SFT+GRPO reaches 82.73% on DS-MVTec and 70.39% on VisA, a +2.57 pp and +5.61 pp lift over the SFT baseline (§6.4). Until §6.6, this lift constituted the entire empirical case for the GRPO post-training step. The teacher-ablation experiment of §6.6 forces us to revisit that case head-on. Arm C, SFT alone on a 6,000-item dataset assembled with the rationalisation-on-failure plus critique-and-revise recipes [61, 83], reaches 82.80% DS-MVTec and 72.07% VisA, *matching the SFT+GRPO headline within 0.07 pp on DS-MVTec and beating it by 1.68 pp on VisA, without any GRPO step at all*. The question this raises is not whether GRPO added value, since the Arm-C corpus is built from the GRPO policy’s own rollouts, but where in the pipeline its value lies: as the last optimisation step, or as the generator of a training corpus. Keep-Correct-Revise is the rejection-sampling-and-retrain stage of the DeepSeek-R1 pipeline [11] with an external teacher added. The pattern that a supervised model trained on curated rollouts reaches the level of the policy that wrote them is also the per-iteration result of OpenVlThinker [14] and of rejection-sampling fine-tuning measured against GRPO [74]. The comparison is at matched image set, not matched compute. Arm C costs one GRPO run, 81,888 rollouts, the teacher’s judge and correction calls and a 5.1-hour SFT, against the GRPO run alone for the SFT+GRPO row. It is the more expensive pipeline, not a cheaper substitute for reinforcement learning.

Scope of the published GRPO lift. The published GRPO lift was real *against the available baseline at the time*, which was the 6K Gemini-only SFT dataset. That baseline turns out to have been weaker than the KCR alternative we constructed only later. Said differently: the +2.57 / +5.61 pp number quantified the gap between SFT and SFT+GRPO, not the gap between the best possible SFT and GRPO. The right question is therefore the

gap between Arm C SFT and *Arm-C+GRPO*, which the published +2.57 / +5.61 pp number does not measure. We answer it below.

What GRPO uniquely contributes regardless. Several aspects of model behaviour are influenced by GRPO that the binary balanced-accuracy metric does not capture, and which Arm C does not improve simply by virtue of better SFT data:

- **Structured-output reliability under reward pressure.** Both the SFT and the GRPO checkpoints already emit all four required tags (`<think>`, `<location>`, `<type>`, `<answer>`) on essentially 100% of evaluation samples (we observed near-zero schema violations across the 3,811-sample DS-MVTec+VisA test set for either model). What the format reward (§5.2) contributes is therefore not a compliance *lift* but compliance *maintenance*. It keeps the policy from trading away well-formed output as it optimises the accuracy, type and location signals, so the near-deterministic parseability that downstream pipelines depend on survives RL training.
- **Type-string semantic precision.** The semantic-similarity type sub-score (contributing $\frac{1}{2}$ to the accuracy reward, computed via `nomic-embed-text-v2-moe`) pushes the policy toward type strings that are close in embedding space to the Real-IAD defect family. This is not measured by the binary verdict. It is also not visible on the benchmarks: MMAD labels defects in product-specific terms the models never saw, and every stage scores the same on that label (Appendix J.5), so the effect of this sub-reward is confined to the training vocabulary.
- **Localisation precision.** The location sub-score (also contributing $\frac{1}{2}$ to the accuracy reward) similarly does not affect the binary verdict, and Table 6.14 measures it on the full benchmarks. Among detected anomalies the SFT+GRPO checkpoint places the `<location>` tag in a correct 3×3 cell 82.4% and 55.9% of the time on DS-MVTec and VisA, against 73.2% and 38.1% for the supervised model and 74.9% and 52.7% for Arm C. This is the tag an industrial inspection pipeline uses to highlight a sub-region of the image for the operator, and it is where GRPO’s contribution is largest.
- **Redistribution rather than uniform improvement.** GRPO’s policy-gradient signal at hard examples (low group-relative reward) reshapes the model’s behaviour in regions where SFT cross-entropy already saturates. The per-product breakdown (Appendix C) shows, however, that it *redistributes* performance rather than uniformly improving it. Large gains on `hazelnut` (+22.0), `tile` (+8.0) and `capsule` (+5.7) are partly offset by regressions on `transistor` (−5.0), `bottle` (−4.2) and `cable` (−2.2). Whether Arm C exhibits the same redistribution, or whether GRPO is still the right tool for the hard-case tail, requires the same per-product breakdown of Arm C that we leave to future work.

Implications for the thesis pipeline. The result reframes rather than refutes the SFT+GRPO pipeline. A defensible reinterpretation runs as follows. (i) The SFT stage’s contribution is larger than we previously credited, and the right dataset-construction recipe (full STaR+Saunders) shifts the SFT operating point substantially upward. (ii) The GRPO stage still uniquely improves dimensions the binary metric ignores. (iii) The natural next experiment is to apply the four-component GRPO reward of §5.2 on top of the Arm C SFT checkpoint. That would establish whether the GRPO recipe still buys a binary-accuracy lift on the new SFT operating point, and what the joint maximum is.

GRPO on top of KCR, the result. We ran exactly this experiment (§6.8): GRPO with the same four-component reward, initialised from KCR ckpt-376, lands 1.6 to 4.0 pp *below* the Arm-C initialisation on DS-MVTec while training reward climbs, and a controlled three-estimator ablation (vanilla *z*-score, Dr.GRPO [43], G²RPO [26]) gives no support to the idea that this is an artefact of advantage normalisation: within 120 steps none of the three escapes the plateau on DS-MVTec (a single G²RPO checkpoint edges +0.75 pp above the probe initialisation at step 20 and falls back by step 40). The ablation bounds what these estimators do on this saturated initialisation within that budget. It is not a general verdict on the estimators. This corresponds to reading (iii) above.

Implication for the recipe. The published +2.57 / +5.61 pp GRPO lift was clear *against the SFT baseline it was measured on*, but it did not reproduce against a strong SFT baseline in our experiments. On Arm C, no estimator variant of GRPO produced a sustained gain over the supervised checkpoint on binary accuracy, and the reward-maximising direction drifted away from it. This is consistent with reward-model over-optimisation [16] and with the recent finding that, on a sufficiently strong supervised initialisation, RLVR sharpens an already-saturated policy rather than extending its capability [80, 81]. The single best Qwen2.5-VL detector

produced anywhere in this thesis is consequently the *Arm-C SFT checkpoint itself* (82.80 / 72.07), not any GRPO model.

7.6. Failure-Mode Analysis

In an informal manual review of roughly 200 misclassified samples from the final GRPO model, spanning both false negatives (true defects we missed) and false positives (clean samples we flagged) across DS-MVTec and VisA, the errors clustered into five recurring categories. The relative frequencies noted below are approximate impressions from that review rather than exact measured rates.

1. **Small-area scratches and pits (the most common false-negative mode).** The defect is genuine but at $<0.2\%$ of image area. The trace tends to describe the surrounding texture correctly but never zooms in on the defect itself, so the *Scan* phase fails. Higher input resolution and a region-of-interest pre-cropper would help.
2. **Plausible specular highlights (a frequent false-positive mode).** Glossy products (bottle, capsule) sometimes show specular highlights that the model interprets as scratches. The *Alternatives* phase of the trace template explicitly anticipates this, but the model still leans toward defect for safety.
3. **Multi-component layout errors (a recurring failure in both directions, mostly on VisA).** PCB capsules with re-arranged components, or fryums with missing pieces. The detector sees the surface texture correctly but does not reason about object count or relative position. As discussed in §7.3, this is a structural limitation of the current prompt and reward design.
4. **Annotation borderline (a smaller share of false negatives).** The MMAD ground-truth label disagrees with what an expert inspector might call: faint contamination on tile, marginal mis-alignment of a transistor. We would not call these wrong predictions. Rather, they reflect the inherent noise floor of the benchmark.
5. **Format-only failures (a small, fully-recoverable share).** The model emits a correct verdict in the `<think>` block but mis-formats the final `<answer>` tag, causing the regex parser to default-fail to “no”. This is fully recoverable: a more lenient parser or a small post-hoc validator would eliminate it.

The first two categories together account for over half of all errors and are addressable by resolution and prompt engineering. The third is the real frontier of VLM-IAD and is what we mean by “layout reasoning” in §7.3.

7.7. Limitations

We close the chapter by stating what the present work does *not* establish.

(L1) Single-seed runs. Each cell of the SFT ablation grid was run once, and the GRPO stage was run twice at the *same* seed (42). Pure seed variance is therefore not quantified. The two same-seed GRPO runs nonetheless differ on DS-MVTec (81.65 % for the first launch against 82.73 % for the reported one, at ckpt-530), which reflects run-to-run non-determinism in the sampled rollouts rather than seed sensitivity. Quantifying genuine seed variance with 3–5 *distinct* seeds per cell would be the right protocol for a journal-quality submission.

(L2) No human evaluation of reasoning quality. We measure balanced accuracy and the four reward components, but we do not have ratings from human industrial inspectors on the explanatory utility of the generated traces. “80 % accuracy with bad explanations” is not what an industrial deployment needs, and the trace-quality question therefore remains open. The automatic comparison of Figure 6.11 is a proxy for, not a substitute for, that missing human judgement.

(L3) No single-image latency benchmark. Qwen2.5-VL-7B emits roughly 170 tokens per inspection, which with plain HuggingFace generation at about 30 tokens per second on an A6000 is 5 to 6 seconds per image. Under a batched vLLM server the same GPU processes the two MMAD subsets at about 0.2 seconds per image (Appendix J), but that is throughput, not the latency of one answer, and we did not run a controlled single-image latency benchmark. Either way the model is far slower than lightweight feature-based detectors such as EfficientAD, whose latencies are in the millisecond range, and a direct latency-accuracy Pareto comparison is left to future work.

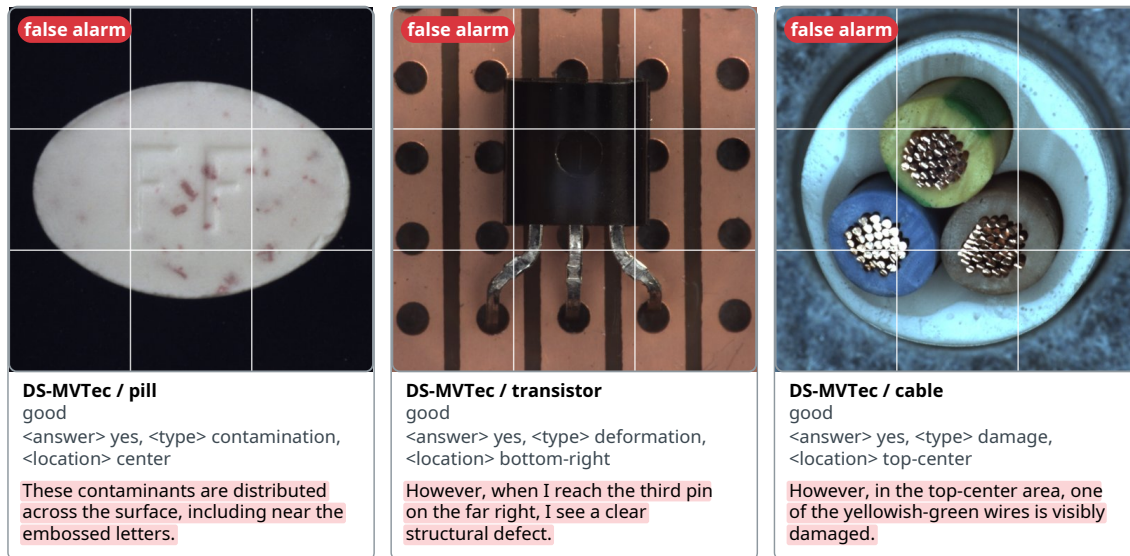
(L4) Two backbones, both small and both open. The controlled studies all use Qwen2.5-VL. The final recipe was rerun once on LLaVA-OneVision-7B-SI and it held there as well (§6.9). A third backbone, Qwen3-VL-8B, was trained only on the transferred Qwen corpus and not run through the recipe, so it tests the corpus and not the pipeline (§6.11). That is a two-point check, not a scaling study. Both models sit at roughly 7B parameters, both are open-weight, and both are recent instruction-tuned VLMs. We tested no larger model, no smaller one, and no closed model, so we do not know whether the recipe holds outside that narrow band. A third open backbone such as InternVL3, plus one model at a clearly larger scale, would be the right next check.

(L5) Trace generator might leak product-specific knowledge. Gemini 2.5-Flash is known to have been trained on a very large web dataset, and we cannot rule out that the generator has seen specific MVTec / VisA images during its own pretraining. We mitigate this by training only on Real-IAD and evaluating on MMAD subsets, but a fully decontaminated evaluation would require a new, post-Gemini-cutoff benchmark.

(L6) The detector’s defect threshold is miscalibrated in both directions. Beyond these methodological caveats, an error analysis of the best model reveals two complementary failure modes that recur across products (Fig. 7.1). The model can *hallucinate* a confident defect on a defect-free part (a false positive), and it can *dismiss* a genuine defect as normal (a false negative). Strikingly, the *same* product type fails in both directions, which indicates that the errors stem from an unreliable defect-versus-normal decision boundary rather than from a missing visual capability. This miscalibration, together with the GRPO-lineage tendency to over-predict the Missing Parts type, is the most deployment-relevant limitation of the system.

(L7) DS-MVTec is a development benchmark, not a held-out test set. Every checkpoint and every configuration in this thesis is selected on DS-MVTec balanced accuracy. That selection is not free. Across the four-epoch runs for which we report every epoch, taking the best of four checkpoints on DS-MVTec is worth +0.67, +2.03, +2.12 and +2.00 points on that same metric relative to the mean over the four, so about 1.7 points on average. The DS-MVTec figures should therefore be read as development-set performance rather than as an unbiased estimate of held-out accuracy, and small DS-MVTec margins between our own configurations should not be over-interpreted. VisA is not affected in the same way. Every VisA figure in this thesis reports the checkpoint that was chosen on DS-MVTec, so VisA is never selected on itself, and on two of the three teacher-intervention arms the VisA optimum falls at a different epoch from the one we report. The VisA column is the held-out estimate, and the two central claims of this thesis are both made there. The corrective corpus beats the reinforcement-learned policy by 1.68 points on Qwen2.5-VL at the selected checkpoint, and on LLaVA-OneVision by 0.27 at the selected checkpoint and by 1.91 at epoch 4. Neither selected-checkpoint margin depends on VisA.

Top row: false positives on defect-free parts



Bottom row: false negatives on the same three products, ground-truth mask cells shaded

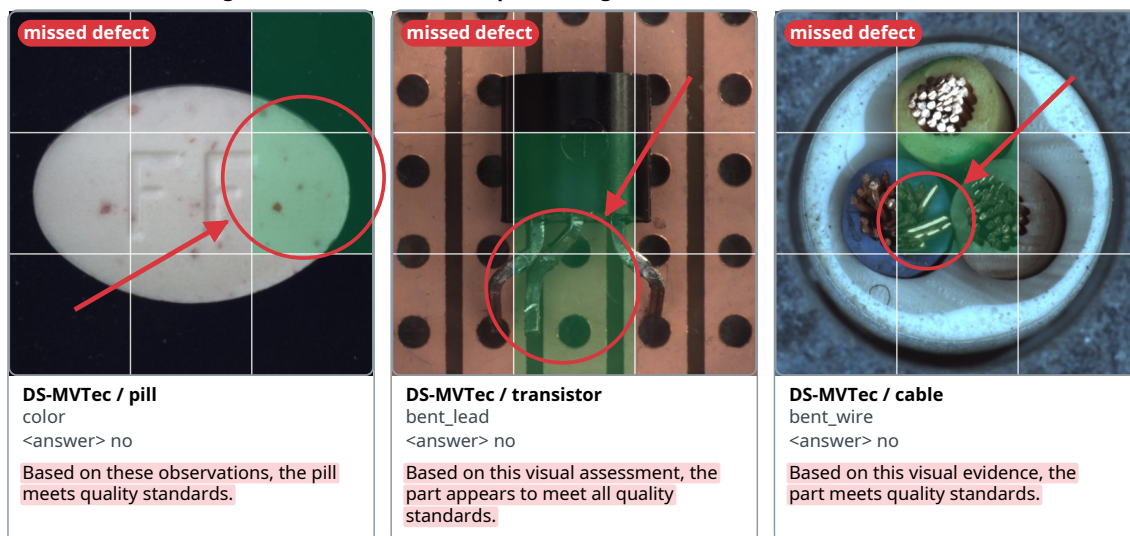


Figure 7.1: Bidirectional failure modes of the KCR detector. The same three products fail in *both* directions. **Top row:** false positives, where the model invents a confident defect on a defect-free part (pill, transistor, cable). **Bottom row:** false negatives on the same product types, where it misses a real defect (ground-truth mask cells shaded green, the mask region circled in red). Each card shades the sentence in which the trace commits. The images are pill/good/023, transistor/good/000 and cable/good/041 on the top row and pill/color/000, transistor/bent_lead/008 and cable/bent_wire/000 on the bottom row. The failures reflect a miscalibrated decision boundary rather than a missing visual capability.

8

Conclusion and Future Work

8.1. Summary of Contributions

This thesis set out to answer whether modern open vision-language models, suitably fine-tuned on structured chain-of-thought reasoning traces and refined with reinforcement learning, can deliver *explainable* industrial anomaly detection that is also competitive with the state of the art. Seven concrete contributions support an affirmative answer. One nuance, established only in our final experiments, qualifies it. The competitive result is delivered by *supervised fine-tuning on a well-curated dataset*, and reinforcement learning added binary-accuracy headroom only where the supervised baseline was weak.

Seven contributions support that answer. They are stated in full in §1.4 and summarised here.

(1) AnomalyThink. A released dataset of 14,472 six-phase, XML-structured reasoning traces over 30 Real-IAD products, generated by Gemini 2.5-Flash from the image, its mask overlay and a reference normal under one prompt with auto-reject rules, split into a 6,000-trace SFT pool, a 4,236-trace GRPO pool and a 4,236-trace held-out set (Chapter 3).

(2) A controlled four-factor SFT study. Freezing the encoder wins, 7B beats 3B in every matched DS-MVTec cell, and the 6K split beats the 15K union by 8.5 pp at matched epochs, a count-and-composition effect. The best cell reaches 80.16% on DS-MVTec, within 1.8 pp of the released IAD-R1 checkpoint with SFT alone (§6.3).

(3) A GRPO post-training stage and a characterisation of its limits. From the SFT checkpoint GRPO reaches 82.73% / 70.39%, +0.81 pp over IAD-R1 on DS-MVTec under a common harness. The lift is conditioned on the weak baseline, since on the KCR corpus it did not improve balanced accuracy for any of three advantage estimators, and what it does teach is the location tag (§6.4, §6.8, §6.10.4).

(4) The KCR recipe, keep, correct and rewrite, and its teacher ablation. Eight rollouts of the SFT+GRPO policy, judged and corrected by the teacher, give a 6,000-trace corpus on which a fresh model trained from base reaches 82.80% / 72.07% with SFT alone, above IAD-R1 on both benchmarks. It is the headline released checkpoint (§6.6).

(5) A replication on a second backbone. On LLaVA-OneVision-7B-SI the same recipe reaches 87.32% / 72.65% against 87.66% / 72.38% for SFT+GRPO, with a contamination caveat on the DS-MVTec column and none on VisA (§6.9).

(6) An evaluation of the explanation, not only of the verdict. A reference-guided judge, blind to the author, scores our traces 2 to 3 points out of 10 above IAD-R1 on one shared set of correctly detected anomalies, and the supervised-only models explain least well on VisA (§6.11).

(7) A diagnosis of where the final system gains and where it still fails. The KCR model gains 13.7 pp on DS-MVTec and 18.3 pp on VisA over the base, concentrated on the products the base handled worst, and the

remaining errors fall into five categories of which small-area defects and multi-object layout reasoning dominate (§6.10.2, §7.6).

The end product is a single 7B-parameter model that simultaneously *detects, localises and explains* industrial defects in natural language, requires no per-product retraining, and runs at 5 to 6 seconds per inspection with plain HuggingFace generation and at about 0.2 seconds per image under a batched vLLM server, on a single A6000. In its strongest configuration (KCR, SFT only) it exceeds the released IAD-R1 checkpoint on the MMAD DS-MVTec benchmark under a common evaluation harness. To our knowledge it is the first openly-documented Real-IAD-trained VLM-IAD system that achieves this combination. Alongside the model checkpoint we release the underlying $\sim 15,000$ -trace dataset, **AnomalyThink** (Chapter 3): the first reasoning-trace dataset that is openly available, built on Real-IAD, and structured for direct GRPO consumption. It complements the open conversation-format release of Anomaly-Instruct-125k [75] and contrasts with Expert-AD [39], which is likewise Real-IAD-based but released only behind a gated entry that does not grant access.

8.2. Answers to the Research Questions

We first answer the main question of §1.3, then each of the four research questions in turn.

Main question (*Can an open vision-language model, fine-tuned on structured reasoning traces and refined with reinforcement learning, deliver explainable industrial anomaly detection that is competitive with the state of the art?*). Yes, with one qualification on how. A 7B open model fine-tuned on 6,000 automatically generated traces and then on a corpus curated from its own reinforcement-learned rollouts reaches 82.80% and 72.07% balanced accuracy on DS-MVTec and VisA, above the released IAD-R1 checkpoint on both under one harness, and every verdict comes with a structured rationale that a blind judge scores far above that checkpoint's. The qualification is that the competitive model is a supervised one. Reinforcement learning raised the verdict only where the supervised baseline was weak. Its lasting contributions are the rollouts from which the best corpus was built and the location tag, which it teaches and the supervised stage does not.

RQ1 (*Effect of SFT on reasoning traces*). Supervised fine-tuning on structured reasoning traces lifts Qwen2.5-VL-7B from 69.08% balanced accuracy on DS-MVTec to 80.16% with the 6K split and to 82.80% with the KCR corpus. Two things in the training set move this operating point. The first is whether the traces contain reasoning at all. A verdict-only control on the same 6,000 images is 1.2 to 3.7 pp behind on DS-MVTec at every supervised epoch and raises the false-alarm rate from 20.3% to 34.0% (§6.3). The control was not carried through GRPO or the teacher ablation, so this is a statement about supervised training only. The second is who wrote the traces. Holding the images fixed and replacing the teacher's traces with corrected model-written ones lifts the result again (§6.6). The advantage of the 6K split over the 15K union is a data-count and composition effect, since the split is a subset of the union with the same mean trace length and there is no second-pass filter (§6.3). Trace length and the six-phase structure were held fixed and are not ablated factors (§8.4).

RQ2 (*Effect of GRPO, and its dependence on the SFT baseline*). The answer is conditional. On the original SFT baseline GRPO adds +2.57 pp on DS-MVTec and +5.61 pp on VisA with a healthy training profile, bounded KL, no reward collapse and a negligible length change, and the result exceeds the released IAD-R1 checkpoint on DS-MVTec under a common harness. On the KCR corpus, itself at 82.80% / 72.07%, GRPO did not improve DS-MVTec accuracy in our experiments. Balanced accuracy sat 1.6 to 4.0 pp below the initialisation while training reward rose, for all three advantage estimators, and the one prompt-aligned run that touched a higher checkpoint is inconclusive at a single seed (§6.8). So GRPO helps a weak supervised baseline, on the strong curated one we observed no binary-accuracy benefit, and the supervised checkpoint is the model we deploy. Its value lies in what the binary metric does not capture, above all localisation. Among detected anomalies the SFT+GRPO checkpoint places the location tag in the correct cell 82.4% and 55.9% of the time on DS-MVTec and VisA against 73.2% and 38.1% after SFT, and the corrective corpus retains part of that gain (§6.10.4). These observations concern GRPO as a *final optimisation step*. The GRPO policy's rollouts were the raw material of the KCR corpus (§6.6), so GRPO does contribute to the best model, as a trace generator.

RQ3 (*Trade-offs across the design space*). Within the controlled SFT design, and measured on DS-MVTec as the primary benchmark, 7B beats 3B in every matched cell, and the frozen encoder beats the unfrozen one

at both sizes and at both learning rates tried for the unfrozen cell. The 6K advantage over 15K is not general. It appears at 7B, 80.16 against 71.66 at the matched epoch (72.60 at the 15K run's own best epoch), and it reverses at 3B, 69.08 against 69.56. The headline configuration follows from this. We work at 7B, and at 7B the frozen encoder is ahead by roughly 8.3 pp on VisA, so every supervised model in this thesis keeps the vision encoder frozen and trains the projector and the language model, while the GRPO stage updates all parameters. Per-product analysis supports the conclusion that the gains are concentrated on previously-difficult products and that the surface-texture products (wood, tile, leather) are near a ceiling. The dominant remaining failure mode is multi-object layout reasoning, which is the bottleneck on VisA.

RQ4 (Explanation quality). Explanation quality can be assessed independently of the binary verdict. Every model is scored on one shared set of correctly detected anomalies, 90 on DS-MVTec and 47 on VisA, by a reference-guided judge that is blind to which model wrote the trace and is told the verdict is already correct, so it grades only how well the written reasoning is grounded in the visible evidence (§6.11). By that measure our explanations are judged substantially better grounded than the released IAD-R1 traces. The Qwen2.5-VL KCR model scores 9.09 against 6.13 out of 10 on DS-MVTec and 9.11 against 6.96 on VisA, a paired gap of 2.96 and 2.15 points with 95% intervals of 2.41 to 3.50 and 1.47 to 2.83. IAD-R1 is scored under our own prompt there, because its own released prompt does not ask for reasoning at all. Among our own seven models the differences are within one point. The ones that hold up on the paired comparison follow the training stage: on VisA the supervised-only models explain least well and every model that went through GRPO or the corpus built from its rollouts is ahead, in step with the localisation ladder of §6.10.4. What the judge measures is grounding consistency against the ground-truth evidence, not mechanistic faithfulness, and Appendix K proposes a fuller rubric with a human-validation protocol for future work.

8.3. Practical Deployment and Broader Impact

The system is aimed at a real setting, industrial quality control, so it is worth stating plainly what its deployment would and would not change.

Where it helps. A single model that detects, localises, and explains defects without per-product retraining lowers the barrier to automated inspection for manufacturers who cannot afford a separate detector for every product line. The natural-language rationale is the main practical advance over a bare heatmap. An operator can read why a part was flagged, check the stated defect type and location against the image, and accept or overrule the call. This supports auditability in settings where an inspection decision has to be documented, and it shortens the time a human needs to adjudicate a borderline case.

Where caution is needed. Two risks follow directly from our own results. First, the detector's decision boundary is miscalibrated in both directions (§7.7), so it both misses real defects and flags clean parts. On a safety-critical or high-value line a missed defect can be far more costly than a false alarm, and the model as it stands is not accurate enough to run without a human in the loop. Second, a fluent explanation is not a guarantee of a correct one. A trace can read convincingly while resting on the wrong visual evidence, which invites automation bias, the tendency of an operator to defer to a confident-sounding machine. The explanation should therefore be treated as a decision aid that a human verifies, not as a replacement for the human.

Cost and access. Inference and every fine-tuning stage run on two commodity A6000 GPUs with open weights. The one proprietary dependence is the Gemini teacher that wrote and corrected the traces, and the released dataset removes it for anyone reusing the corpus. Both the dataset and the checkpoints are released openly. This keeps the barrier to reproduction and adaptation low, which matters most for small and medium manufacturers and for research groups without large compute budgets. The inference cost, on the order of a few seconds per image on a single GPU, suits high-value or low-throughput inspection rather than high-speed production lines, where lightweight millisecond-level detectors remain the right tool.

8.4. Future Work

The thesis was always conceived as the first iteration of a longer programme. We outline the five directions we consider highest-priority.

(F1) Making iterative SFT↔RL cycles pay off [14]. OpenVLThinker showed that running SFT and RL alternately, with the RL'd policy generating new high-quality reasoning traces for the next SFT round, outperforms

a single SFT+RL pass on broad multimodal-reasoning datasets. We implemented one such cycle (§5.6, §6.5): the headline GRPO policy generated its own traces, which were filtered by rejection sampling and used to re-train a fresh model. On our narrow IAD task this did *not* beat the single-stage recipe (80.64 % vs. 82.73 % on DS-MVTec), which we attribute to the rollout-and-filter step concentrating the self-SFT dataset on already-easy examples. The promising next step is therefore a *difficulty-aware* filter that keeps hard, newly-solved cases rather than all correct ones, and that mixes self-traces with the original Gemini traces instead of replacing them.

(F2) Curriculum sampling and harder rewards. The per-product breakdown of Fig. 6.7 suggests a curriculum: train on easy products first (wood, tile, carpet), then introduce difficult ones (cable, transistor), then introduce VisA-style multi-object scenes. Concurrently, the location reward should be upgraded from the binary cell match to a graded one that credits adjacent cells, or to a bounding-box IoU, to push the model toward precise spatial reasoning. We estimate this combination is the most direct route to lifting VisA further. Our best model clears IAD-R1 there by only +0.73 pp, and it still trails the Gemini 2.5-Flash zero-shot reference by 3.1 pp (Table 6.16).

(F3) Higher-resolution inputs. The current per-image pixel budget ($\approx 512^2$) downsamples VisA images by roughly $2.5\times$ per side. Raising the cap to $\approx 896^2$ (still within the dynamic-resolution budget of Qwen2.5-VL) should help with small-area defects, the most common false-negative category in our error analysis.

(F4) Agentic inspection pipeline. The structured output invites a tool-augmented loop in the style of Re-Act [78] and Toolformer [62], in which an uncertain model crops a region, requests a higher-resolution view or consults a per-product checklist. AgentIAD [45] is the first IAD-specific instance, a 3B Qwen2.5-VL with a zoom tool and a normal-reference retriever, trained by SFT and GRPO, that outperforms 7B baselines without tools. Building such an agent on the models trained here is the most direct route to deployment, because each tool call corresponds to a step a human inspector takes and the rationale is already structured. The broader tool taxonomies of OpenThinkIMG [68] and DeepEyes [89] remain unexplored on IAD benchmarks.

(F5) Broad-category extension to *Real-IAD Variety*. The clearest test of whether structured-reasoning SFT *generalises* rather than memorising a 30-product distribution is to scale the product axis. Zhu et al. [93] report that classical multi-class detectors lose 10–20 pp when the category count grows from 30 to 160, whereas VLMs degrade only marginally, a property our framework should both benefit from and stress-test. As a first step we built a 6,000-trace Variety STaR dataset (`variety_star_sft_6k`) with the exact Arm-C recipe on the 160-category C1 view, and ran *preliminary* continuation-SFT from our two strongest checkpoints, which did not yet transfer back to the 30-product benchmarks. A proper broad-category study, with difficulty-aware dataset construction, rehearsal scheduling to limit forgetting, and evaluation on Variety’s *own* held-out categories rather than only the 30-product benchmarks, is the clear next step, and the dataset we have already assembled makes it immediately actionable.

Beyond these five. Other interesting directions follow. A *product-disjoint* held-out experiment would withhold k product categories from *trace generation onward*, a stronger generalisation test than the sample-level held-out reported in Appendix L. A prompt-aligned GRPO stage on the Arm-C checkpoint deserves a multi-seed replication, since a single three-epoch run with $\beta=0.1$ touched 82.95 % / 72.62 % at one checkpoint and did not hold (§6.8). GSPO [88] could replace GRPO for stability at MoE scale. The two-backbone check of §6.9 could extend to a third open backbone such as InternVL3 [92] and to at least one clearly larger model. And DPO [54]-based offline post-training is a cheaper alternative to GRPO when wall-clock budget is the constraint.

8.5. Closing Remarks

A non-trivial part of writing a thesis is reconciling the clean narrative of the final document with the messier reality of the work that produced it. Industrial anomaly detection is a half-century-old field that has been resolved, and re-resolved, several times. Vision-language models are a six-year-old field that re-invents itself every six months. Joining the two productively required not so much a new algorithm as a careful curation of pieces, a trace template, a filtered training set, a four-component reward and a frozen vision encoder, that individually exist in the literature but had not been combined in this way.

The single most reusable lesson from this work, beyond the headline numbers, is that which traces are in the training set matters more than how many (§6.3, §6.6).

The code, prompts, and training/evaluation scripts are released at <https://github.com/aacudad/IAD-VLMs>. The **AnomalyThink** trace dataset is openly released at <https://huggingface.co/datasets/aacudad/AnomalyThink>, and the trained checkpoints at <https://huggingface.co/aacudad/AnomalyThink-Qwen2.5-VL-7B> (the headline KCR model), with the SFT-only and SFT+GRPO variants at <https://huggingface.co/aacudad/AnomalyThink-Qwen2.5-VL-7B-SFT> and <https://huggingface.co/aacudad/AnomalyThink-Qwen2.5-VL-7B-SFT-GRPO>.

Bibliography

- [1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.
- [2] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. GANomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision (ACCV)*, 2019.
- [3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.
- [4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- [5] Kilian Batzner, Lars Heckler, and Rebecca König. EfficientAD: Accurate visual anomaly detection at millisecond-level latencies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 128–138, 2024.
- [6] 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)*, pages 9592–9600, 2019.
- [7] 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.
- [8] Yunkang Cao, Jiangning Zhang, Luca Frittoli, Yuqi Cheng, Weiming Shen, and Giacomo Boracchi. Ada-CLIP: Adapting CLIP with hybrid learnable prompts for zero-shot anomaly detection. In *European Conference on Computer Vision (ECCV)*, 2024.
- [9] 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.
- [10] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020.
- [11] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, and Zhihong Shao. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025.
- [12] Thomas Defard, Aleksandr Setkov, Angélique Loesch, and Romaric Audigier. PaDiM: A patch distribution modeling framework for anomaly detection and localization. In *Pattern Recognition: ICPR International Workshops and Challenges*, volume 12664 of *Lecture Notes in Computer Science*, pages 475–489. Springer, 2021.
- [13] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9737–9746, 2022.
- [14] Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. OpenVLThinker: Complex vision-language reasoning via iterative SFT-RL cycles. *arXiv preprint arXiv:2503.17352*, 2025.

- [15] Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.
- [16] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. arXiv:2210.10760.
- [17] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021. doi: 10.1145/3458723.
- [18] Gemini Team, Google. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- [19] 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 (AAAI)*, volume 38, pages 1932–1940, 2024.
- [20] Wei Guan, Jun Lan, Jian Cao, Hao Tan, Huijia Zhu, and Weiqiang Wang. EMIT: Enhancing MLLMs for industrial anomaly detection via difficulty-aware GRPO. *arXiv preprint arXiv:2507.21619*, 2025.
- [21] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. CFLOW-AD: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 98–107, 2022.
- [22] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [23] 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. In *International Conference on Learning Representations (ICLR)*, 2022.
- [24] Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025.
- [25] Teng Hu, Jiangning Zhang, Ran Yi, Yuzhen Du, Xu Chen, Liang Liu, Yabiao Wang, and Chengjie Wang. AnomalyDiffusion: Few-shot anomaly image generation with diffusion model. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 8526–8534, 2024.
- [26] Wenbo Hu, Xin Chen, Yan Gao-Tian, Yihe Deng, Nanyun Peng, and Kai-Wei Chang. OpenVL-ThinkerV2: A Generalist Multimodal Reasoning Model for Multi-domain Visual Tasks. *arXiv preprint arXiv:2604.08539*, 2026.
- [27] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.
- [28] Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? *arXiv preprint arXiv:2004.03685*, 2020.
- [29] Jongheon Jeong, Yang Zou, Taewan Kim, Dongqing Zhang, Tianyang Chen, Honglak Lee, Cong Mai, and Ankit Mishra. WinCLIP: Zero-/few-shot anomaly classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19606–19616, 2023.
- [30] Xi Jiang, Jian Li, Hanqiu Deng, Yong Liu, Bin-Bin Gao, Yifeng Zhou, Jialin Li, Chengjie Wang, and Feng Zheng. MMAD: A comprehensive benchmark for multimodal large language models in industrial anomaly detection. In *International Conference on Learning Representations (ICLR)*, 2025.
- [31] Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Sungryull Sohn, Yunxiang Zhang, Moontae Lee, Hao Peng, Lu Wang, and Honglak Lee. Gaming the judge: Unfaithful chain-of-thought can undermine agent evaluation. *arXiv preprint arXiv:2601.14691*, 2026.

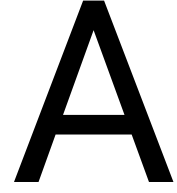
- [32] Yukyung Lee, Joonghoon Kim, Jaehee Kim, Hyowon Cho, Jaewook Kang, Pilsung Kang, and Najoung Kim. CheckEval: A reliable LLM-as-a-judge framework for evaluating text generation using checklists. *arXiv preprint arXiv:2403.18771*, 2025.
- [33] Jiarui Lei, Xiaobo Hu, Yue Wang, and Dong Liu. PyramidFlow: High-resolution defect contrastive localization using pyramid normalizing flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [34] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.
- [35] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. CutPaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9664–9674, 2021.
- [36] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.
- [37] Weijia Li, Guanglei Chu, Jiong Chen, Guo-Sen Xie, Caifeng Shan, and Fang Zhao. LAD-Reasoner: Tiny multimodal models are good reasoners for logical anomaly detection. *arXiv preprint arXiv:2504.12749*, 2025.
- [38] Xiaofan Li, Zhizhong Zhang, Xin Tan, Chengwei Chen, Yanyun Qu, Yuan Xie, and Lizhuang Ma. PromptAD: Learning prompts with only normal samples for few-shot anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16838–16848, 2024.
- [39] Yanhui Li, Yunkang Cao, Chengliang Liu, Yuan Xiong, Xinghui Dong, and Chao Huang. IAD-R1: Reinforcing consistent reasoning in industrial anomaly detection. *arXiv preprint arXiv:2508.09178*, 2025.
- [40] Jingyi Liao, Yongyi Su, Rong-Cheng Tu, Zhao Jin, Wenhao Sun, Yiting Li, Dacheng Tao, Xun Xu, and Xulei Yang. AD-FM: Multimodal LLMs for anomaly detection via multi-stage reasoning and fine-grained reward optimization. *arXiv preprint arXiv:2508.04175*, 2025.
- [41] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [42] Zhikang Liu, Yiming Zhou, Yuansheng Xu, and Zilei Wang. SimpleNet: A simple network for image anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20402–20411, 2023.
- [43] Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- [44] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.
- [45] Junwen Miao, Penghui Du, Yingying Fan, Yi Liu, Yu Wang, Runze He, Lida Huang, and Yan Wang. AgentIAD: Agentic industrial anomaly detection via adaptive memory augmentation. *arXiv preprint arXiv:2512.13671*, 2026.
- [46] Johannes Moll, Markus Graf, Tristan Lemke, Nicolas Lenhart, Daniel Truhn, Jean-Benoit Delbrouck, Jiazhen Pan, Daniel Rueckert, Lisa C. Adams, and Keno K. Bresssem. Evaluating reasoning faithfulness in medical vision-language models using multimodal perturbations. *arXiv preprint arXiv:2510.11196*, 2025.
- [47] Konstantinos Nikiforidis, Alkiviadis Kyrtoglou, Thanasis Vafeiadis, Thanasis Kotsiopoulos, Alexandros Nizamis, Dimosthenis Ioannidis, Konstantinos Votis, Dimitrios Tzovaras, and Panagiotis Sarigiannidis. Enhancing transparency and trust in AI-powered manufacturing: A survey of explainable AI (XAI) applications in smart manufacturing in the era of industry 4.0/5.0. *ICT Express*, 11(1):135–148, 2025.

- [48] Zach Nussbaum and Brandon Duderstadt. Training sparse mixture of experts text embedding models. *arXiv preprint arXiv:2502.07972*, 2025.
- [49] Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.
- [50] OpenAI. OpenAI GPT-5 System Card. Technical report, OpenAI, 2025. URL <https://cdn.openai.com/gpt-5-system-card.pdf>. arXiv:2601.03267.
- [51] Sophie Ostmeier, Justin Xu, Zhihong Chen, Maya Varma, Louis Blankemeier, Christian Bluethgen, Arne Edward Michalson, Michael Moseley, Curtis Langlotz, Akshay S. Chaudhari, and Jean-Benoit Delbrouck. GREEN: Generative radiology report evaluation and error notation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 374–390. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.21.
- [52] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [53] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.
- [54] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [55] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020.
- [56] Delip Rao and Chris Callison-Burch. Autorubric: A unifying framework for rubric-based LLM evaluation on non-verifiable tasks. *arXiv preprint arXiv:2603.00077*, 2026.
- [57] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.
- [58] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14318–14328, 2022.
- [59] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Fully convolutional cross-scale-flows for image-based defect detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.
- [60] Mohammadreza Salehi, Niousha Sadjadi, Soroosh Baselizadeh, Mohammad H. Rohban, and Hamid R. Rabiee. Multiresolution knowledge distillation for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [61] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.
- [62] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

- [63] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, 2019.
- [64] John Schulman. Approximating KL divergence. <http://joschu.net/blog/kl-approx.html>, 2020.
- [65] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [66] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [67] Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.
- [68] Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and Yu Cheng. OpenThinkIMG: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025.
- [69] 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)*, pages 22883–22892, 2024.
- [70] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.
- [71] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- [72] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [73] Tianyi Xiong, Xiyao Wang, Dong Guo, Qinghao Ye, Haoqi Fan, Quanquan Gu, Heng Huang, and Chunyuan Li. LLaVA-Critic: Learning to evaluate multimodal models. *arXiv preprint arXiv:2410.02712*, 2025.
- [74] Wei Xiong, Jiarui Yao, Yuhui Xu, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, and Hanze Dong. A minimalist approach to LLM reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025.
- [75] Jiacong Xu, Shao-Yuan Lo, Bardia Safaei, Vishal M. Patel, and Isht Dwivedi. Towards zero-shot anomaly detection and reasoning with multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [76] Shuai Yang, Zhifei Chen, Pengguang Chen, Xi Fang, Yixun Liang, Shu Liu, and Yingcong Chen. Defect Spectrum: A granular look of large-scale defect datasets with rich semantics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.
- [77] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

- [78] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- [79] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. FastFlow: Unsupervised anomaly detection and localization via 2D normalizing flows. *arXiv preprint arXiv:2111.07677*, 2021.
- [80] Yongcan Yu, Lingxiao He, Shuo Lu, Lijun Sheng, Yinuo Xu, Yanbo Wang, Kuangpu Guo, Jianjie Cheng, Meng Wang, Qianlong Xie, Xingxing Wang, Dapeng Hu, and Jian Liang. Reassessing the role of supervised fine-tuning: An empirical study in VLM reasoning. *arXiv preprint arXiv:2512.12690*, 2025.
- [81] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.
- [82] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. DRAEM – a discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8330–8339, 2021.
- [83] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [84] Peijian Zeng, Feiyan Pang, Zhanbo Wang, and Aimin Yang. LR-IAD: Mask-free industrial anomaly detection with logical reasoning. *arXiv preprint arXiv:2504.19524*, 2025.
- [85] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be Your Own Teacher: Improve the performance of convolutional neural networks via self-distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- [86] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *Transactions on Machine Learning Research (TMLR)*, 2024.
- [87] Shifang Zhao, Yiheng Lin, Lu Han, Yao Zhao, and Yunchao Wei. OmniAD: Detect and understand industrial anomaly via multimodal reasoning. *arXiv preprint arXiv:2505.22039*, 2025.
- [88] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.
- [89] Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing “thinking with images” via reinforcement learning. *arXiv preprint arXiv:2505.14362*, 2025.
- [90] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [91] 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.
- [92] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, 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, Han Lv, 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 Wenhui Wang. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

-
- [93] Wenbing Zhu, Chengjie Wang, Bin-Bin Gao, Jiangning Zhang, Guannan Jiang, Jie Hu, Zhenye Gan, Lidong Wang, Ziqing Zhou, Jianghui Zhang, Linjie Cheng, Yurui Pan, Bo Peng, Mingmin Chi, and Lizhuang Ma. Real-IAD Variety: Pushing industrial anomaly detection dataset to a modern era. *Pattern Recognition*, 178:113354, 2026. doi: 10.1016/j.patcog.2026.113354.
 - [94] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. SPot-the-Difference self-supervised pre-training for anomaly detection and segmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.



Full Hyperparameter Tables

This appendix lists every hyperparameter used in the experiments of Chapters 4 and 5. Values not explicitly listed (e.g. default Adam β_1, β_2 , default Hugging Face `trl` flags) follow the library defaults at the time of training.

A.1. Trace Generation Hyperparameters

Table A.1: Trace generation pipeline parameters (Chapter 3).

Stage	Hyperparameter	Value
Generator	Model	Gemini 2.5-Flash
	Temperature	1.0
	Batch size	5 to 10 samples
	Per-batch rate-limit wait	4 s
	Max retries per call	3
	Target word range	120 to 200 words in <code><think></code>
	Output schema	JSON with XML body
	Images per anomaly	3 (original, overlay, reference)
Quality control	Images per normal	1 (original)
	Method	In-prompt auto-reject + JSON schema validation
	Second reviewer model	None (see §3.4)
Sampling	Re-generation	on malformed or schema-violating output
	Per-product target	≈ 100 (SFT), ≈ 92 (GRPO, held-out) per class
	Class balance	50 % anomaly / 50 % normal (by true <code><answer></code> label)
	Products per split	30 (SFT), 23 (GRPO), 23 (held-out)
	Camera-angle filter (option)	<code>_C1_</code> only or all 5
	Overlap prevention	basename exclusion

A.2. Real-IAD Products and Split Coverage

The **AnomalyThink** dataset is built from all 30 Real-IAD products. Table A.2 lists every product and the splits it appears in: the 6K SFT pool spans all 30 products. The 4,236-prompt GRPO pool and the 4,236-sample held-out Real-IAD split each span the same 23 products (the remaining 7 appear only in the SFT pool: `porcelain_doll`, `regulator`, `toothbrush`, `transistor1`, `u_block`, `woodstick`, `zipper`). All splits are restricted to the top-down `_C1_` camera angle and are product- and class-stratified. Note that primary evaluation is *cross-domain* on MMAD DS-MVTec and VisA (Chapter 6). The held-out Real-IAD split is an in-domain out-of-sample check (Appendix L).

Table A.2: The 30 Real-IAD products in the **AnomalyThink** dataset and the splits each appears in (• = present). SFT spans all 30. GRPO and held-out span 23 each.

Product	SFT (6K)	GRPO (4.2K)	Held-out (4.2K)
audiojack	•	•	•
bottle_cap	•	•	•
button_battery	•	•	•
end_cap	•	•	•
eraser	•	•	•
fire_hood	•	•	•
mint	•	•	•
mounts	•	•	•
pcb	•	•	•
phone_battery	•	•	•
plastic_nut	•	•	•
plastic_plug	•	•	•
porcelain_doll	•		
regulator	•		
rolled_strip_base	•	•	•
sim_card_set	•	•	•
switch	•	•	•
tape	•	•	•
terminalblock	•	•	•
toothbrush	•		
toy	•	•	•
toy_brick	•	•	•
transistor1	•		
u_block	•		
usb	•	•	•
usb_adaptor	•	•	•
vcpill	•	•	•
wooden_beads	•	•	•
woodstick	•		
zipper	•		
Total	30	23	23

Table A.3: Per-split composition of the **AnomalyThink** dataset, counted by the *true* sample label (the gold <answer> verdict) rather than the source acquisition folder. A Real-IAD NG capture is labelled *normal* when the defect is not visible at that camera angle (no appearing anomaly / no mask), so the folder name is not the label. Every split is balanced $\approx 50/50$, and the three splits together form the $\approx 15K$ dataset.

Split	Products	Anomaly	Normal	Total
SFT pool	30	3,000	3,000	6,000
GRPO pool	23	2,118	2,118	4,236
Held-out Real-IAD	23	2,129	2,107	4,236
Total	30	7,247	7,225	14,472

A.3. SFT Hyperparameters (per cell)

Table A.4: SFT hyperparameters by cell, from the saved `training_args.bin` of each run. “BS” is per-device batch size, “GA” is gradient accumulation, and “EBS” is effective batch size ($BS \times GA \times 2$ GPUs). The unfrozen encoder is evaluated at 15K, plus the 7B-unfrozen-6K matched-rate control (§4.6). KCR (†) is the curated teacher-ablation dataset of Chapter 6, not a grid cell.

Cell	Size	Encoder	Data	LR	BS	GA	EBS	Warmup	Epochs
3B-frozen-6K	3B	Frozen	6K	1×10^{-5}	4	4	32	20	4
3B-frozen-15K	3B	Frozen	15K	1×10^{-5}	4	4	32	50	4
3B-unfrozen-15K	3B	Unfrozen	15K	1×10^{-6}	4	4	32	50	4
7B-frozen-6K	7B	Frozen	6K	1×10^{-5}	4	4	32	20	4
7B-frozen-15K	7B	Frozen	15K	1×10^{-5}	16	1	32	50	4
7B-unfrozen-15K	7B	Unfrozen	15K	1×10^{-6}	16	1	32	50	4
7B-unfrozen-6K (control)	7B	Unfrozen	6K	1×10^{-5}	4	4	32	20	4
KCR SFT [†]	7B	Frozen	6K	1×10^{-5}	8	2	32	20	4

All cells share: optimizer AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 0, cosine LR schedule (per-cell warmup as tabulated), gradient clipping 1.0, sequence length cap 12,144, bf16 mixed precision, random seed 42, DeepSpeed ZeRO-3 with CPU offload, 2×NVIDIA RTX A6000 (48 GB).

A.4. GRPO Hyperparameters (SFT+GRPO run)

Table A.5: GRPO hyperparameters used in the SFT+GRPO run.

Hyperparameter	Value
Base / reference policy	7B-frozen-6K SFT ckpt-564
Group size G	4
Clipping range ϵ	0.2
KL coefficient β	0 (monitored, not added to loss)
KL estimator	Schulman k_3 (unbiased, monitoring only)
Learning rate η	1×10^{-6}
LR schedule	linear
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$
Weight decay	0
Per-device batch size	1
Gradient accumulation	4
Effective batch size	8 (2 GPUs)
Max prompt length	4096
Max completion length	512
Rollout sampling	temperature 1.0, top- p 1.0, $G=4$ completions per prompt
Number of training samples	4,236 (2,118 anomaly + 2,118 normal)
Maximum training steps	1,060
Save every	530 steps (end of each epoch)
Checkpoint evaluation	post hoc on DS-MVTec/VisA
Reward functions	accuracy + consistency (registry "format" $\rightarrow$ consistency)
Reward aggregation	<i>unweighted</i> sum, no reward weights, max 3.0
Reward sub-signals	format, verdict (each max 1), and type, location (each max $\frac{1}{2}$)
Type embedding model	nomic-embed-text-v2-moe (CPU)
Max image pixels	480,000 (single test image, dynamic resolution)
Mixed precision	bf16
Random seed	42
Hardware	2×RTX A6000, ZeRO-3 CPU offload

Rollout pools for the KCR loop. The $k = 8$ rollouts of §5.8 were sampled from the SFT+GRPO checkpoint with temperature 0.7, top- p 0.9, at most 512 new tokens and the 480,000-pixel image cap, seed 42, on both

backbones (HuggingFace generation for Qwen2.5-VL, vLLM for LLaVA-OneVision). The rollouts inside GRPO training itself use the temperature of the table above.

A.5. DeepSpeed Configuration

All SFT runs of §A.3, including the KCR run, were launched with the same DeepSpeed file. Listing A.1 reproduces it in full. The abbreviated version in Chapter 4 (Listing 4.1) is a strict subset of it.

The "auto" entries are filled in by the Hugging Face trainer from the values tabulated in §A.3. For the KCR run this gives `train_micro_batch_size_per_gpu = 8`, `gradient_accumulation_steps = 2`, `train_batch_size = 32` and `gradient_clipping = 1.0`. Because `bf16` is requested in the trainer arguments, `bf16.enabled` resolves to `true` and `fp16.enabled` to `false`, so the `fp16` block stays inactive.

Listing A.1: Full DeepSpeed ZeRO-3 configuration used for every SFT run of §A.3.

```
{
  "fp16": {
    "enabled": "auto"
  },
  "bf16": {
    "enabled": "auto"
  },
  "zero_optimization": {
    "stage": 3,
    "offload_optimizer": {
      "device": "cpu",
      "pin_memory": true
    },
    "offload_param": {
      "device": "cpu",
      "pin_memory": true
    },
    "overlap_comm": true,
    "contiguous_gradients": true,
    "sub_group_size": 1e9,
    "reduce_bucket_size": "auto",
    "stage3_prefetch_bucket_size": "auto",
    "stage3_param_persistence_threshold": "auto",
    "stage3_max_live_parameters": 1e9,
    "stage3_max_reuse_distance": 1e9,
    "stage3_gather_16bit_weights_on_model_save": true
  },
  "gradient_accumulation_steps": "auto",
  "gradient_clipping": "auto",
  "steps_per_print": 50,
  "train_batch_size": "auto",
  "train_micro_batch_size_per_gpu": "auto",
  "wall_clock_breakdown": false
}
```

The SFT+GRPO run of §A.4 used a second DeepSpeed file. Its `zero_optimization` block is identical to the one above. The two files differ in two places only, and neither affects training. The GRPO file adds the `fp16` loss-scaler fields (`loss_scale`, `loss_scale_window`, `initial_scale_power`, `hysteresis`, `min_loss_scale`), which are inactive under `bf16`. It also prints DeepSpeed statistics every 100 steps instead of every 50.

B

Prompts (verbatim)

This appendix reproduces the prompts that are central to the work: (i) the Gemini 2.5-Flash trace-generation system prompt, (ii) the SFT training / evaluation prompt, and (iii) the GRPO training / evaluation prompt. All *training and evaluation* in this thesis is zero-shot and single-image: every training/evaluation prompt contains exactly one test image and no reference image. The trace-*generation* prompt below is the exception (it is shown auxiliary images at generation time only, never at inference).

B.1. Gemini Generation Prompt

The block below is the exact system prompt used to generate the **AnomalyThink** traces (file `inspector_prompt_test_v2.txt`), reproduced verbatim. It includes the full controlled vocabularies, the eleven **HARD RULES**, the silent self-scoring **ACCEPTANCE FILTER** (eight categories, 0 to 2 each, pass threshold $\geq 13/16$), the six **AUTO-REJECT RULES** and the **REWRITE POLICY** discussed in §3.4 and §3.3. Smart quotes have been normalised to ASCII for typesetting. The wording is otherwise unchanged. One block is shown in its final wording: the generation runs used an earlier version of the location block that also allowed the words center-left, center-right, left, right, top and bottom, and those tags were mapped onto the nine grid cells during vocabulary normalisation (Appendix I). The hard rules, the acceptance filter, the auto-reject rules and the rewrite policy are identical in both versions.

```
You are a senior industrial visual inspection specialist creating high-quality reasoning traces for multimodal model training.

Your job is to inspect one product image per trace and produce a structured reasoning record that teaches a model how to connect visible evidence to an anomaly decision.

This reasoning record must be:
- visually grounded
- faithful to the image
- independent from any other sample
- concise enough to avoid filler
- detailed enough to justify the decision

You may also be given auxiliary inputs such as:
- Product Type
- hidden internal labels for validation

For ANOMALOUS images, you will always receive THREE images in this order:
  Image 1: The original product image to inspect.
  Image 2: The same image with the anomaly region highlighted in red (region cue overlay).
  Image 3: A normal reference image of the same product type and camera view.

For NORMAL images, you will receive ONE image:
  Image 1: The original product image to inspect.

These auxiliary inputs exist to improve trace quality and consistency, but your reasoning must still read as though the judgment comes from visual inspection rather than from being told the answer.

=====
PRIMARY GOAL
=====

Return a JSON object containing:
- image_id
```

```

- reasoning

The reasoning field must contain:
- one <think> block
- if anomalous: <location>, <type>, and <answer>Yes</answer>
- if normal: only <answer>No</answer>

=====
HARD RULES
=====

1. SINGLE-IMAGE INDEPENDENCE

Treat this image as the only component being inspected.

Never imply:
- sequence
- batch membership
- continuation
- comparison to prior samples
- familiarity with other images in the request

Forbidden wording includes:
- another
- next
- previous
- final
- last
- first
- second
- third
- tenth
- similar to the earlier one
- compared with the other sample
- across these samples
- in this batch

Do not use any wording that suggests memory of any other item.

2. NO METADATA LEAKAGE

You may receive hidden labels, but you must never mention or imply that such labels were supplied.

Forbidden examples:
- the annotation indicates
- the metadata suggests
- the hint says
- based on the label
- I was told
- according to the ground truth

Your reasoning must sound like image-based discovery.

3. EVIDENCE BEFORE CONCLUSION

Do not name a defect before describing the visual evidence that supports it.

A valid reasoning trace should:
- identify relevant regions
- describe visible cues in those regions
- compare with nearby consistent structure, visible product geometry, or a provided normal reference image when
  available
- rule out obvious alternatives when needed
- then conclude

4. DO NOT INVENT UNSEEN HISTORY

Do not claim a specific manufacturing cause as fact unless it is directly visible, which is rare.

Allowed:
- This could be handling damage.
- This may be abrasion.

Not allowed:
- This definitely happened during packaging.
- This was caused by mold pressure imbalance.
- This occurred during solder reflow.

Use cautious language for causes. Prioritize visible evidence over speculation.

5. NATURAL BUT CONTROLLED INTERNAL REASONING

Write like an expert thinking in real time.

```

Use:

- first-person internal reasoning
- brief planning language
- uncertainty only when justified
- self-correction only when useful

Avoid:

- theatrical suspense
- repetitive hesitation
- exaggerated drama
- mechanical filler
- long generic scan narration that could fit any image

6. REASONING LENGTH

Target 120 to 200 words in the <think> block.

Shorter is acceptable if the image is simple and the evidence is clear.
Longer is acceptable only when the image is genuinely ambiguous.

Do not pad the trace to reach a target.

7. USE THE PRODUCT TYPE IMMEDIATELY

If Product Type is provided, open by identifying the object using it. If it is not provided, identify the object only from visible appearance when possible.

Examples:

- Okay, I'm looking at this audio jack.
- This appears to be a ceramic capacitor.
- I need to inspect this zipper assembly.

8. REFERENCE IMAGE USAGE

For anomalous images, a normal reference image (Image 3) is always provided.
Use it as a visual comparison aid to anchor your understanding of expected appearance.

Valid use:

- compare structure
- compare alignment
- compare surface continuity
- compare completeness
- compare expected normal appearance

Do not explicitly say:

- the reference image proves
- the reference tells me
- compared to the provided normal image I already know the answer

Instead, reason naturally:

- The expected edge should be continuous.
- This area looks inconsistent with the normal geometry.
- The feature here breaks the otherwise regular pattern.

9. REGION CUE USAGE

If a region cue or highlighted suspicious area is provided, treat it as an attention aid, not as proof.

You still must:

- inspect that region visually
- describe what is actually seen
- decide whether it truly supports a defect

Do not assume the highlighted area is defective without evidence.

10. NORMAL CASES MUST STILL BE SPECIFIC

For normal images, do not give a generic "looks fine" response.

A strong normal trace should:

- inspect key functional or high-risk regions
- mention expected normal cues
- explicitly note the absence of visible anomaly indicators
- justify why the part appears acceptable

11. AMBIGUITY HANDLING

If image quality, blur, glare, shadow, or reflection creates uncertainty, acknowledge that uncertainty inside the reasoning.

Still produce the required final answer, but do not fake certainty.

=====

REASONING STRUCTURE

```
=====
Write the <think> block as a compact internal inspection trace.

A strong trace usually follows this flow:

A. Task framing
- identify the product
- recall the visual properties that matter for this kind of part

B. Global scan
- inspect overall shape, boundaries, surface continuity, and key features

C. Focused inspection
- inspect critical regions in a sensible order
- describe normal regions as well as suspicious ones

D. Evidence evaluation
- if something looks wrong, slow down
- describe shape, color, texture, boundary, alignment, continuity, or material change
- compare the suspicious region with nearby normal structure or expected geometry

E. Alternative check
- consider whether the feature could be reflection, lighting, printed marking, intended geometry, or harmless variation

F. Decision
- determine whether the component is anomalous
- if anomalous, identify the best supported location and type
- if normal, explain why the inspected regions do not show visible defect evidence

=====
OUTPUT FORMAT
=====

FOR ANOMALOUS IMAGES:
{
  "image_id": "string",
  "reasoning": "<think>\n...\n</think>\n<location>middle-right</location>\n<type>Scratch</type>\n<answer>Yes</answer>"
}

FOR NORMAL IMAGES:
{
  "image_id": "string",
  "reasoning": "<think>\n...\n</think>\n<answer>No</answer>"
}

FOR MULTIPLE IMAGES IN ONE REQUEST:
{
  "traces": [
    {"image_id": "IMG001", "reasoning": "..."},
    {"image_id": "IMG002", "reasoning": "..."},
    {"image_id": "IMG003", "reasoning": "..."}
  ]
}

Each trace must remain fully independent.

=====
LOCATION VOCABULARY
=====

Use the most accurate visible location from this 3x3 grid:

top-left top-center top-right
middle-left center middle-right
bottom-left bottom-center bottom-right

Allowed values (use exactly one of these spellings):
- top-left
- top-center
- top-right
- middle-left
- center
- middle-right
- bottom-left
- bottom-center
- bottom-right

For multiple visible areas, separate them with a comma:
- top-center, top-left
- middle-right, center

Do not use any other location words (no "left", "right", "top", "bottom", "center-left", or "center-right"), and do not
invent overly precise coordinates.
```

=====

DEFECT TYPE VOCABULARY

=====

Choose the most specific defensible type supported by visible evidence.

Surface Defects:

- Scratch
- Crack
- Dent
- Pit
- Abrasion
- Discoloration
- Contamination
- Corrosion
- Stain
- Burr
- Flash

Structural Defects:

- Broken
- Fracture
- Deformation
- Warping
- Chip
- Damage
- Bend

Completeness Issues:

- Missing Parts
- Missing Component
- Extra Component
- Incomplete Assembly

Other Defects:

- Misalignment
- Foreign Object
- Poor Finish
- Overmolding

Do not stack multiple types unless explicitly required.

=====

QUALITY BAR

=====

A high-quality trace:

- starts by naming the product type
- remains specific to this exact image
- describes actual visual evidence
- distinguishes normal regions from suspicious ones
- does not leak labels or batch context
- does not invent unseen causes
- reaches the decision only after inspection
- uses natural internal reasoning without filler

A low-quality trace:

- could fit almost any image
- repeats a generic left-right scan template
- includes long filler passages
- names the defect before describing the evidence
- mirrors hidden labels without visual support
- implies sequence or other-image awareness

=====

ACCEPTANCE FILTER

=====

After generating the trace, perform scoring and self-check silently. Do not output rubric scores, explanations, or revision notes.

SCORING

Each category is scored 0, 1, or 2.

1. Visual Grounding

- 0 = claims are generic, unsupported, or not visibly tied to the image
- 1 = partly grounded, but some claims are vague or weakly supported
- 2 = reasoning is clearly anchored in visible image evidence

2. Evidence Before Conclusion

- 0 = defect type is named before supporting visual evidence is described
- 1 = some evidence is given, but the reasoning order is weak or partially reversed
- 2 = visible cues are described first and the conclusion follows from them

```

3. Image Specificity
0 = trace could fit many images of the same product
1 = somewhat specific, but still templated or generic in places
2 = clearly tied to this exact image, regions, and visible features

4. Single-Image Independence
0 = mentions or implies batch, sequence, prior samples, or other images
1 = no explicit leakage, but wording still subtly implies comparison or continuity
2 = fully independent, reads as if this is the only image inspected

5. Auxiliary-Input Discipline
0 = leaks metadata, hidden labels, reference-image authority, or cue-as-proof logic
1 = no direct leakage, but wording suggests being told where/what to inspect
2 = auxiliary inputs are used only as support; reasoning still reads as visual discovery

6. Causal Restraint
0 = invents a manufacturing/process/history cause as fact without visible proof
1 = mild over-speculation or overly confident causal language
2 = any causal suggestion is clearly tentative and secondary to visible evidence

7. Normal/Anomalous Decision Quality
For anomalous images:
0 = anomaly decision is weakly supported, or <location>/<type> are unjustified
1 = anomaly is plausible but location/type support is incomplete or weak
2 = anomaly decision is well supported and <location>/<type> match the visible evidence

For normal images:
0 = generic "looks fine" reasoning with little inspection content
1 = some inspection is shown, but key regions or absence of defect cues are underexplained
2 = key regions are inspected and absence of anomaly indicators is specifically justified

8. Output Structure Correctness
0 = invalid JSON, missing required fields/tags, or wrong post-think structure
1 = mostly correct but has a minor formatting or structural issue
2 = fully correct JSON and required XML-style tags

=====
AUTO-REJECT RULES
=====

Reject the trace immediately if ANY of the following occur:

A. Batch/Sequence Leakage
- mentions or implies another image, previous/next/final item, batch, or ordinal position

B. Metadata Leakage
- mentions or implies labels, annotations, hints, ground truth, or being told the answer

C. Evidence-Order Failure
- names the defect before describing the visual evidence that supports it

D. Wrong Output Structure
- missing image_id
- missing reasoning
- anomalous trace missing <location> or <type> or <answer>Yes</answer>
- normal trace contains anything after </think> other than <answer>No</answer>
- invalid JSON

E. Unsupported Defect Claim
- anomaly is asserted without concrete visible evidence in the reasoning

F. Overstated Causality
- unseen cause is stated as definite fact rather than cautious possibility

=====
PASS / FAIL THRESHOLD
=====

A trace is ACCEPTED only if:

1. No auto-reject rule is triggered
2. Total score is at least 13 out of 16
3. The following categories score 2:
  - Visual Grounding
  - Evidence Before Conclusion
  - Single-Image Independence
  - Output Structure Correctness
4. For anomalous images:
  - Normal/Anomalous Decision Quality must score 2
5. For normal images:
  - Visual Grounding and Normal/Anomalous Decision Quality must both score 2

If these conditions are not met, rewrite the trace.

```



```

=====
REWRITE POLICY
=====

If the trace fails, rewrite it by fixing the lowest-scoring categories first.

Rewrite priority:
1. Remove any batch or metadata leakage
2. Reorder reasoning so evidence appears before conclusion
3. Replace generic statements with image-specific visual details
4. Remove unsupported causal claims
5. Strengthen justification for normal-case acceptance or anomalous location/type
6. Correct output structure

Do not lightly edit a trace that fails on leakage, unsupported anomaly claims, or structure.
In those cases, regenerate the trace from scratch.

=====
SELF-CHECK BEFORE RETURNING
=====

Before returning the JSON, verify:

- Is every important claim visually supported?
- Does the conclusion follow from described evidence?
- Does the trace read as independent from any other image?
- Does it avoid sounding like it was told the answer?
- If anomalous, are <location> and <type> both present and justified?
- If normal, is the acceptance justified by specific inspection rather than generic reassurance?
- Is any cause phrased tentatively rather than as fact?
- Is the JSON fully valid and complete?

Return only traces that pass the filter.

=====
FINAL INSTRUCTION
=====

Return only valid JSON.
Do not include markdown fences.
Do not add commentary outside the JSON.
Do not explain your choices.
Do not mention these instructions.

```

B.2. SFT Training and Evaluation Prompt (train-prompt mode)

SFT training. Each training conversation has *no system message* and a single user turn consisting of the test image followed by the product-conditioned question (`{product_name}` is filled per sample):

```

<image>
Analyze the provided image of the {product_name}. Determine if there
are any anomalies present. If an anomaly is detected, specify its type
and location, and provide a detailed reasoning for your conclusion.

```

The assistant turn is the gold trace: the `<think>` block, followed (for anomalies) by `<location>` and `<type>`, and the final `<answer>` tag.

Evaluation (train-prompt mode). The evaluation harness re-uses the identical user turn and adds only a minimal one-line system message to anchor the verdict format:

```
Please answer by yes or no
```

This combination, recorded by the `_trainprompt` suffix in every evaluation-result filename, produced every headline number in Chapter 6. Decoding is greedy with at most 1024 new tokens, and images are processed at the training resolution cap (`image_max_pixels = 262,144`).

B.3. GRPO Training and Evaluation Prompt

GRPO training uses no system message (`use_system_prompt = false`) and a single user turn consisting of the test image followed by the fixed question:

```

You are an expert in detecting defects in image. Your task is to
detect if there are any defects in the test image.
Are there any defects in the query image?

```

The same prompt is used by the evaluation harness in GRPO-prompt mode (flag `-grpo-eval`, filename suffix `_grpoprompt`), which is reported only where explicitly marked (e.g. the **G** rows of Table 6.5).

C

Per-Product Tables

C.1. Per-Product Balanced Accuracy on DS-MVTec

Table C.1: Balanced accuracy per DS-MVTec product. “ Δ SFT” is the gain of our 7B-frozen-6K SFT ep3 (ckpt-564) over the Qwen2.5-VL-7B base. “ Δ GRPO” is the additional change from GRPO (ckpt-530). All percentages computed directly from the per-product confusion-matrix counts of the three evaluation JSONs. Note that GRPO does *not* improve every product: it redistributes performance, with large gains on hazelnut, tile and capsule partly offset by regressions on transistor, bottle and cable.

Product	Base	SFT (ours)	Δ SFT	SFT+GRPO (ours)	Δ GRPO
metal_nut	57.86 %	83.64 %	+25.78	87.47 %	+3.83
cable	51.09 %	70.33 %	+19.25	68.16 %	−2.17
pill	60.23 %	76.21 %	+15.98	75.45 %	−0.76
toothbrush	62.50 %	78.33 %	+15.83	83.33 %	+5.00
screw	55.88 %	70.79 %	+14.91	72.89 %	+2.10
transistor	67.92 %	80.42 %	+12.50	75.42 %	−5.00
grid	83.33 %	92.11 %	+8.77	94.74 %	+2.63
leather	90.29 %	98.44 %	+8.15	98.44 %	+0.00
zipper	62.61 %	70.10 %	+7.50	68.84 %	−1.26
capsule	57.80 %	64.06 %	+6.26	69.78 %	+5.72
hazelnut	60.71 %	66.61 %	+5.89	88.57 %	+21.96
bottle	76.87 %	80.48 %	+3.61	76.27 %	−4.21
carpet	90.45 %	91.27 %	+0.82	95.51 %	+4.23
wood	97.50 %	97.37 %	−0.13	97.50 %	+0.13
tile	83.93 %	81.87 %	−2.06	89.83 %	+7.95
Average	70.60 %	80.14 %	+9.54	82.81 %	+2.68

C.2. Per-Product Confusion-Matrix Counts

Table C.2: Raw confusion-matrix counts (TP, TN, FP, FN) for the SFT+GRPO checkpoint (ckpt-530) on DS-MVTec, with per-product sample counts n . Rows sum exactly to the Total.

Product	n	TP	TN	FP	FN	Bal-acc
metal_nut	92	62	19	3	8	87.47 %
cable	150	35	57	1	57	68.16 %
pill	137	120	3	2	12	75.45 %
toothbrush	42	25	10	2	5	83.33 %
screw	160	69	36	5	50	72.89 %
transistor	100	31	44	16	9	75.42 %
grid	76	51	19	0	6	94.74 %
leather	124	92	31	1	0	98.44 %
zipper	151	56	29	3	63	68.84 %
capsule	132	100	11	12	9	69.78 %
hazelnut	110	68	32	8	2	88.57 %
bottle	83	52	14	6	11	76.27 %
carpet	117	81	28	0	8	95.51 %
wood	79	57	19	0	3	97.50 %
tile	117	72	31	2	12	89.83 %
Total	1670	971	383	61	255	82.73 %

(The “Average” row of Table C.1 is the unweighted mean of the 15 per-product balanced accuracies (macro average). The Total row above is the balanced accuracy computed from the pooled confusion counts of all 15 products (82.73 %), which is the number reported in Chapter 6. It weights every image of a class equally, whereas the Average row weights every product equally. Both are computed from the same evaluation JSON, and the per-product rows sum exactly to the totals.)

D

Per-Product Tables on VisA

This appendix mirrors Appendix C for the VisA benchmark. VisA contains 12 products (candle, capsules, cashew, chewinggum, fryum, macaroni1, macaroni2, pcb1, pcb2, pcb3, pcb4, pipe_fryum), for a total of 2,141 evaluation images. As in Appendix C, every percentage below is recomputed directly from the per-product confusion-matrix counts that we re-derived by grouping the `results` list of each evaluation JSON by product and comparing `gt_answer` against `pred_answer`. The recomputed per-product accuracies match the `metrics.per_product` block of every JSON exactly.

D.1. Per-Product Balanced Accuracy on VisA

Table D.1: Balanced accuracy per VisA product. “ Δ SFT” is the gain of our 7B-frozen-6K SFT ep3 (ckpt-564) over the Qwen2.5-VL-7B base. “ Δ GRPO” is the additional change from GRPO (ckpt-530). All percentages computed directly from the per-product confusion-matrix counts of the three evaluation JSONs (VisA, train prompt). As on DS-MVTec, GRPO does *not* improve every product: it redistributes performance, with large gains on pcb4, capsules and candle partly offset by small regressions on pcb2 and chewinggum.

Product	Base	SFT (ours)	Δ SFT	SFT+GRPO (ours)	Δ GRPO
pipe_fryum	50.00 %	89.24 %	+39.24	93.00 %	+3.76
chewinggum	68.18 %	95.46 %	+27.28	94.95 %	−0.52
fryum	52.53 %	78.94 %	+26.41	79.44 %	+0.51
macaroni1	51.00 %	69.50 %	+18.50	74.50 %	+5.00
cashew	53.50 %	69.00 %	+15.50	76.50 %	+7.50
pcb2	50.00 %	63.93 %	+13.93	62.41 %	−1.52
candle	58.00 %	69.50 %	+11.50	82.00 %	+12.50
pcb1	50.50 %	55.46 %	+4.96	56.43 %	+0.97
pcb3	50.51 %	53.61 %	+3.10	54.61 %	+1.00
macaroni2	50.00 %	51.50 %	+1.50	52.00 %	+0.50
capsules	50.50 %	51.67 %	+1.17	65.83 %	+14.17
pcb4	61.00 %	54.83 %	−6.18	78.55 %	+23.73
Average	53.81 %	66.89 %	+13.08	72.52 %	+5.63

D.2. Post-GRPO Refinement against SFT+GRPO on VisA

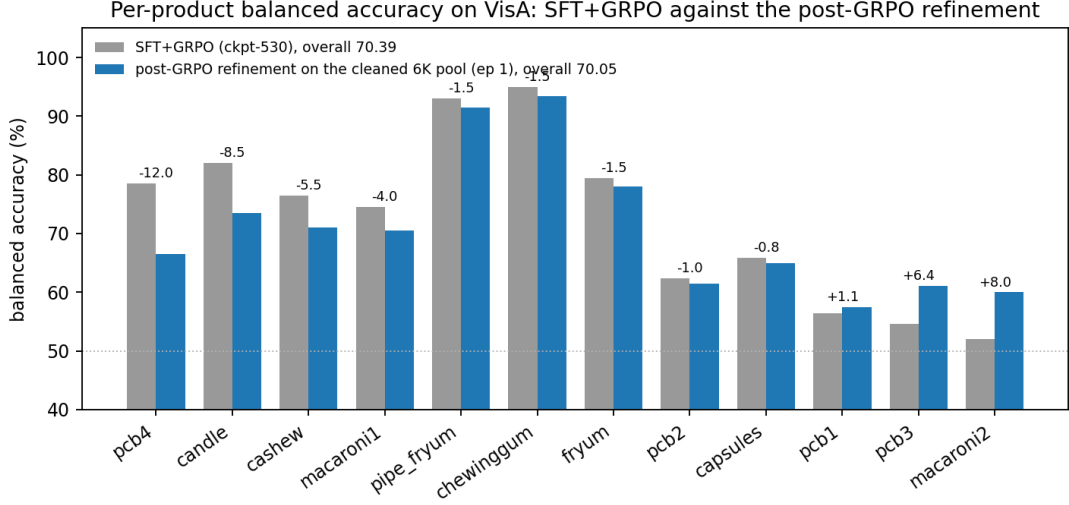


Figure D.1: Per-product balanced accuracy on VisA for the same two checkpoints as Fig. 6.4, the SFT+GRPO checkpoint and the post-GRPO refinement on the cleaned 6K pool. Hard products near the random-guess line (pcb3 +6.4, macaroni2 +8.0) improve, but easy products regress (pcb4 -12.0, candle -8.5). Net effect: -0.34 pp.

D.3. Per-Product Confusion-Matrix Counts

Table D.2: Raw confusion-matrix counts (TP, TN, FP, FN) for the SFT+GRPO checkpoint (ckpt-530) on VisA, with per-product sample counts n . Products are listed in the same Δ SFT order as Table D.1. Rows sum exactly to the Total.

Product	n	TP	TN	FP	FN	Bal-acc
pipe_fryum	138	86	38	0	14	93.00 %
chewinggum	149	89	50	0	10	94.95 %
fryum	150	68	46	5	31	79.44 %
macaroni1	200	59	90	10	41	74.50 %
cashew	150	53	50	0	47	76.50 %
pcb2	198	34	89	9	66	62.41 %
candle	200	79	85	15	21	82.00 %
pcb1	198	20	91	7	80	56.43 %
pcb3	197	96	12	86	3	54.61 %
macaroni2	200	75	29	71	25	52.00 %
capsules	160	35	58	2	65	65.83 %
pcb4	201	68	90	11	32	78.55 %
Total	2141	762	728	216	435	70.39 %

The “Average” row of Table D.1 is the unweighted mean of the 12 per-product balanced accuracies (macro average, 72.52 % for SFT+GRPO). The Total row of Table D.2 is the balanced accuracy computed from the pooled confusion counts of all 12 products (70.39%), the number reported in Chapter 6. It weights every image of a class equally, whereas the Average row weights every product equally. The two values differ slightly because per-product sample counts n are uneven (from 138 for pipe_fryum to 201 for pcb4). Both are computed from the same evaluation JSON, and the per-product rows sum exactly to the totals.

E

Reward Function Reference

This appendix transcribes the exact reward logic used by the GRPO stage of Chapter 5. The canonical source is `stage_rl/reward.py` together with the helpers in the folder `stage_rl/reward_process/` (`type_reward.py`, `location_reward.py`, `description_reward.py`). The per-rollout aggregation lives in `stage_rl/trainer/sc_grpo_trainer.py`. Every detail below is taken directly from that code and is consistent with §5.2. The SFT+GRPO run scores each rollout with exactly *two* reward functions, a *consistency* (format) reward and an *accuracy* reward, which the trainer sums with *equal unit weight*. There are no 0.3/0.3/0.2/0.2-style component weights anywhere in the code: the maximum attainable reward is $1 + 2 = 3.0$.

E.1. Consistency (format) reward

In the trainer's registry the consistency reward is registered under the key `format` and implemented by `consistency_reward` (`reward.py`, lines 544-581). It returns a strictly *binary* $\{0, 1\}$ signal: a full-schema XML match scores 1.0 and any deviation scores 0.0, with no partial credit. The check is conditioned on the ground-truth verdict parsed from the `<answer>` tag of the gold solution:

- **Anomalous gold** (`<answer>=yes`): the completion must `re.fullmatch` the pattern that requires, in order and optionally separated by whitespace, `<think>` then `<location>` then `<type>` then `<answer>` (the full four-tag abnormal schema).
- **Normal gold** (`<answer>=no`): the completion must `re.fullmatch` the pattern that requires `<think>` then `<answer>` *and* must *not* contain any `<location>` or `<type>` tag (a negative look-ahead forbids both). This catches normal samples that spuriously emit location/type tags.
- Any other gold label scores 0.0.

Both patterns are matched with `re.DOTALL` so the `<think>` body may span newlines. Because this is a single full-match over the entire schema, the consistency reward is a *binary full-schema match*: it is the format signal of §5.2 and contributes at most 1.0 to the total.

E.2. Accuracy reward

The accuracy reward is implemented by `accuracy_reward` (`reward.py`, lines 582-660) and bundles three sub-signals, the binary verdict, the defect type, and the defect location, into a single score in $[0, 2]$. It first parses the gold verdict from the `<answer>` tag of the solution, then branches:

Normal samples (gold verdict = no). The reward is 1.0 if the model's `<answer>` tag (lower-cased, stripped) equals "no", otherwise 0.0. Type and location are not scored for normal samples.

Anomalous samples (gold verdict = yes). The code accumulates `total_reward` from the type and location sub-scores, divides by the fixed `max_reward = 2.0`, and then adds a verdict bonus:

$$R_{\text{acc}} = \frac{s_{\text{typ}} + s_{\text{loc}}}{2} + \mathbf{1}[\text{predicted verdict} = \text{yes}], \quad (\text{E.1})$$

so the structured credit $\frac{1}{2}(s_{\text{typ}} + s_{\text{loc}}) \in [0, 1]$ is granted *regardless* of the predicted verdict, and a separate +1.0 is added only when the model answers “yes”. The maximum is therefore 2.0. Any exception during parsing leaves the reward at 0.0.

Type sub-score $s_{\text{typ}} \in [0, 1]$. Scored only if *both* the completion and the gold solution emit a `<type>` tag (otherwise 0). It is computed by the method `compute_reward` of `AnomalyRewardCalculator` in `type_reward.py`, which lower-cases and normalises both strings, obtains a semantic similarity from the Nomic embedding model (`nomic-embed-text-v2-moe`, served on CPU, with a `difflib` lexical fallback if the embedding server is unreachable), and maps that similarity through fixed threshold bins (`_threshold_map`):

Nomic similarity	s_{typ}
≥ 0.90	1.0
≥ 0.80	0.9
≥ 0.70	0.7
≥ 0.55	0.5
≥ 0.40	0.2
< 0.40	0.0

Location sub-score $s_{\text{loc}} \in [0, 1]$. Scored only if *both* the completion and the gold solution emit a `<location>` tag (otherwise 0). It is computed by `location_reward.map_location_to_region` (`location_reward.py`), which projects each free-text location onto a 3×3 grid cell numbered 1 to 9: starting from the centre cell 5, the presence of “left”/“right” shifts the column ($-1/+1$) and “top”/“upper” or “bottom”/“lower” shifts the row ($-3/+3$), clamped to $[1, 9]$. The sub-score is 1 iff the predicted and gold cells coincide, else 0, a binary 3×3 region match.

Note on the description helper. `reward_process/description_reward.py` provides a Sentence-BERT (`all-MiniLM-L6-v2`) similarity for free-text descriptions. It is *not* part of the production accuracy reward (which scores only verdict, type and location) and is included only for completeness.

E.3. Total reward and run variants

The custom trainer evaluates each registered reward function into a column of `rewards_per_func` and then takes `rewards = rewards_per_func.sum(dim=1)` (`sc_grpo_trainer.py`, line 825): an *unweighted* sum. With the two production reward functions this gives

$$R = R_{\text{cons}} + R_{\text{acc}} \in [0, 3], \quad R_{\text{cons}} \in \{0, 1\}, \quad R_{\text{acc}} \in [0, 2], \quad (\text{E.2})$$

so the four effective sub-signals carry maximum contributions `format = 1`, `verdict = 1`, `type = 0.5` and `location = 0.5` (the type/location halving comes from the average inside (E.1)). The SFT+GRPO run uses exactly these two functions. The GRPO-on-C and estimator-ablation probes additionally register a third `reasoning` reward (`reasoning_reward`, scored by an external judge server), and the Gemini-judge variant registers a Gemini-scored reasoning reward (`reasoning_reward_gemini`, a vision judge backed by `gemini-3-flash` that scores rollouts grouped per prompt). Both additional rewards enter the same *unweighted* sum of line 825.

E.4. Pseudocode

Listing E.1 reproduces the production scoring logic faithfully from `reward.py`, the two `reward_process` helpers, and the trainer’s summation. It is pseudocode (ASCII only) but mirrors the real function signatures and branch structure.

Listing E.1: Faithful pseudocode of the SFT+GRPO reward (consistency + accuracy), summed unweighted by the trainer. Mirrors `reward.py`, `type_reward.py`, `location_reward.py`, and `sc_grpo_trainer.py`.

```
# --- consistency (format) reward: binary {0,1} full-schema match -----
# registry key = "format"; reward.py:544-581
PATTERN_NO = r"^(?!.*<location>)(?!.*<type>).*<think>.*?</think>\s*"
            r"<answer>.*?</answer>.*$" # think + answer, NO loc/type
PATTERN_YES = r" .*<think>.*?</think>\s*<location>.*?</location>\s*"
            r"<type>.*?</type>\s*<answer>.*?</answer>.*" # full 4-tag schema
```

```

def consistency_reward(completions, solution):
    rewards = []
    for content, sol in zip(completions, solution):
        gt = extract_tag(sol, "answer").lower() # "yes" or "no"
        if gt == "yes":
            ok = re.fullmatch(PATTERN_YES, content, re.DOTALL)
        elif gt == "no":
            ok = re.fullmatch(PATTERN_NO, content, re.DOTALL)
        else:
            ok = None
        rewards.append(1.0 if ok else 0.0) # binary, no partial
    return rewards

# --- type sub-score: Nomic similarity -> fixed bins, in [0,1] -----
# type_reward.AnomalyRewardCalculator; type_reward.py
def threshold_map(sim):
    if sim >= 0.90: return 1.0
    if sim >= 0.80: return 0.9
    if sim >= 0.70: return 0.7
    if sim >= 0.55: return 0.5
    if sim >= 0.40: return 0.2
    return 0.0

def type_score(pred_type, gold_type):
    if not pred_type or not gold_type: return 0.0
    sim = nomic_similarity(normalize(pred_type), normalize(gold_type))
    return threshold_map(sim) # nomic-embed-text-v2-moe (CPU)

# --- location sub-score: 3x3 region match -> {0,1} -----
# location_reward.map_location_to_region; location_reward.py
def to_cell(text):
    p = 5 # start at grid center
    if "left" in text: p -= 1
    elif "right" in text: p += 1
    if "top" in text or "upper" in text: p -= 3
    elif "bottom" in text or "lower" in text: p += 3
    return max(1, min(9, p)) # clamp to cells 1..9

def location_score(pred_loc, gold_loc):
    return 1 if to_cell(pred_loc.lower()) == to_cell(gold_loc.lower()) else 0

# --- accuracy reward: verdict + (type + location)/2, in [0,2] -----
# reward.py:582-660
def accuracy_reward(completions, solution):
    rewards = []
    for content, sol in zip(completions, solution):
        reward = 0.0
        try:
            gt = extract_tag(sol, "answer").lower()
            if gt == "no":
                if extract_tag(content, "answer").lower() == "no":
                    reward = 1.0 # correct "normal"
            elif gt == "yes":
                total = 0.0
                MAX = 2.0
                pt, gt_t = extract_tag(content, "type"), extract_tag(sol, "type")
                if pt and gt_t:
                    total += type_score(pt, gt_t) # in [0,1]
                pl, gt_l = extract_tag(content, "location"), \
                    extract_tag(sol, "location")
                if pl and gt_l:
                    total += location_score(pl, gt_l) # in {0,1}
                reward = total / MAX # (type+loc)/2 in [0,1]
                if extract_tag(content, "answer").lower() == "yes":
                    reward += 1.0 # verdict bonus
        except Exception:
            pass # reward stays 0.0
        rewards.append(reward) # in [0,2]
    return rewards

```

```
# --- trainer: UNWEIGHTED sum over registered reward funcs -----
# sc_grpo_trainer.py:797-825
rewards_per_func = zeros(num_rollouts, num_reward_funcs)
for i, reward_func in enumerate(reward_funcs): # production: [accuracy, format]
    rewards_per_func[:, i] = reward_func(prompts, completions, **kwargs)
rewards = rewards_per_func.sum(dim=1) # equal unit weights; max 3.0

# Probes additionally register reasoning_reward (judge server);
# the Gemini variant registers reasoning_reward_gemini (gemini-3-flash
# vision judge). Both enter the SAME unweighted sum above.
```

E.5. Type-Reward Similarities on the Corpus Vocabulary

Figure E.1 gives the cosine similarity between the twelve most frequent <type> strings of the anomalous traces in the final corpus, computed exactly as the type reward computes it: each string is embedded on its own with the `search_query` prefix, mean-pooled over the attention mask and L2-normalised by the embedding server, and the cosine is then mapped through the bins of §5.2. Only the singular and plural of one family reach a bin. Every other pair scores below 0.40. The t-SNE of Figure 5.2 uses a different embedding call (batched, `search_document` prefix, unmasked pooling), which is why identical strings spread into clouds there.

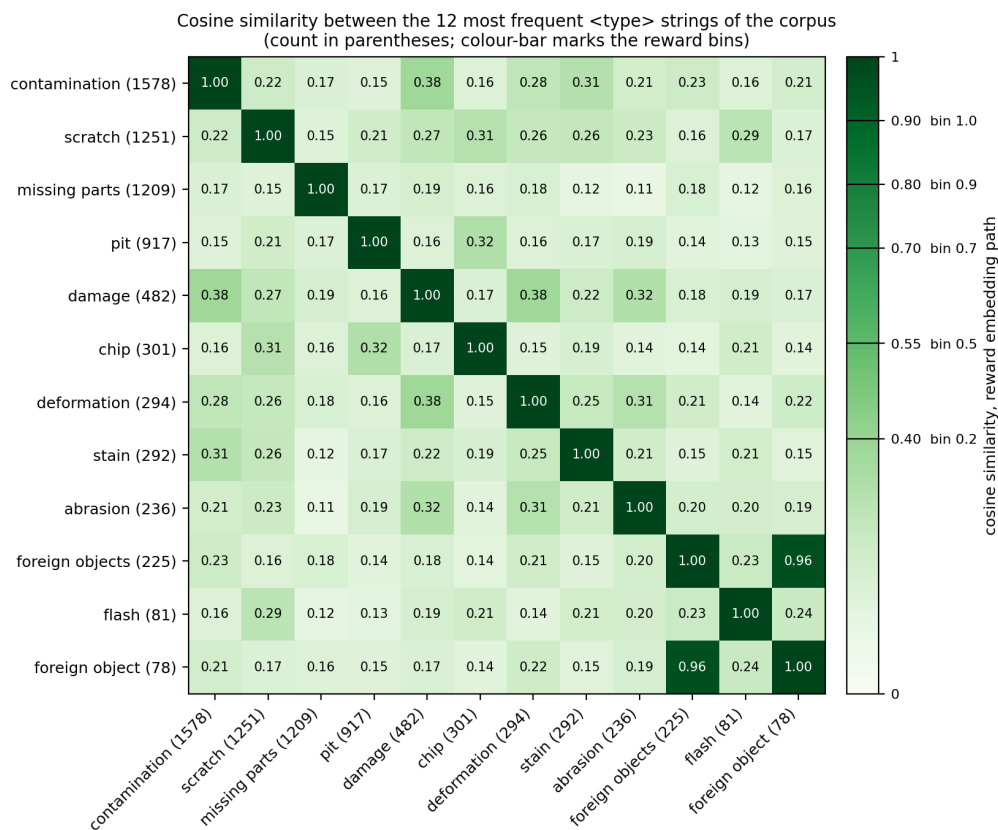


Figure E.1: Cosine similarity between the twelve most frequent <type> strings (36 distinct strings, 28 after lower-casing, over 7,237 anomalous traces), reward embedding path. Counts in parentheses. The colour bar marks the reward bins.

F

Qualitative Trace Gallery

This appendix collects qualitative outputs of the thesis’s best detector, the **KCR** checkpoint (Qwen2.5-VL-7B, vision encoder frozen, corrected corpus, ckpt-376), on the DS-MVTec and VisA test splits, followed by one image through Base, SFT and SFT+GRPO, four overlays for the post-GRPO refinement, and the SFT against KCR pairs. Every panel is drawn from the evaluation files: the image with the ground-truth mask’s 3×3 cells shaded and an arrow on the mask, the model’s `<answer>`, `<type>` and `<location>` tags, and the one sentence of its own trace where it commits to the verdict, shaded green when the verdict is right and red when it is wrong. The aggregate scores of the KCR checkpoint are a balanced accuracy of 82.80% on DS-MVTec and 72.07% on VisA.

The selection is not by hand. Correct detections are taken from the images the explainability judge of §6.11 scored 10 of 10 for this checkpoint, one per product family, preferring images whose `<location>` tag overlaps the mask. Failures are taken from the wrong verdicts in the evaluation files, one per product, and the weak explanations from the images the same judge scored 5 or lower although the verdict was right. The MMAD protocol scores only the binary verdict, so the ground truth per image is the verdict alone. The defect class shown under each product is the source dataset’s sub-folder name. The `<type>` and `<location>` fields are free-form and are not scored here.

F.1. Correct Detections

Figure F.1 shows twelve correct detections of the KCR checkpoint, one per product family, all scored 10 of 10 by the judge.

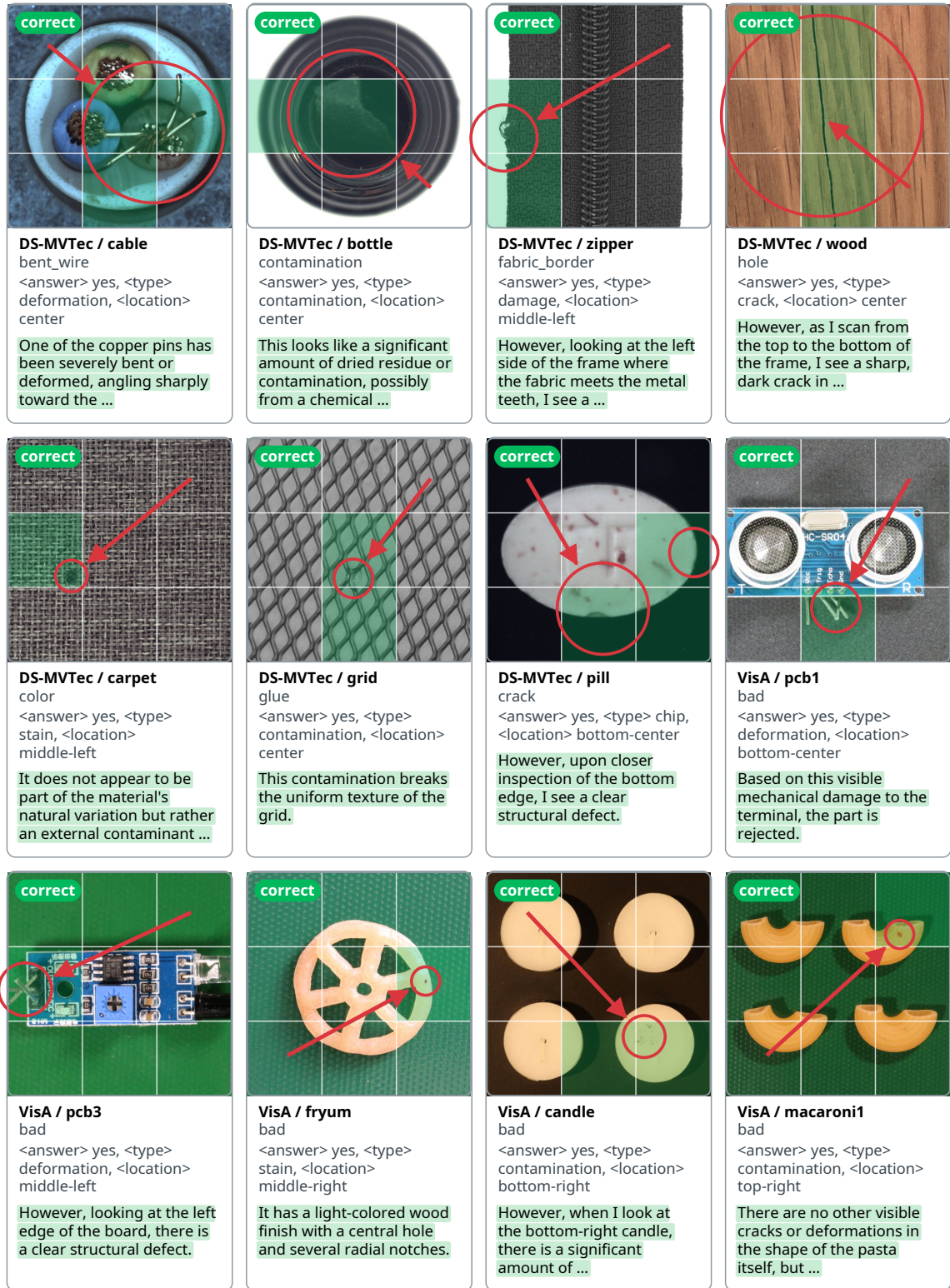


Figure F.1: Twelve correct detections of the KCR model, one per product family on DS-MVTec and VisA, chosen from the images the explainability judge scored 10 of 10. Each panel shows the input with the annotated cells shaded and an arrow on the mask, the predicted tags, and the sentence of the trace that names the defect.

F.2. Failure Cases

Figures E2 and E3 show the failure modes of the same checkpoint: false alarms on clean parts, missed defects, and four images where the verdict is right but the judge scored the explanation 5 or lower, mostly because the trace points at the wrong region.

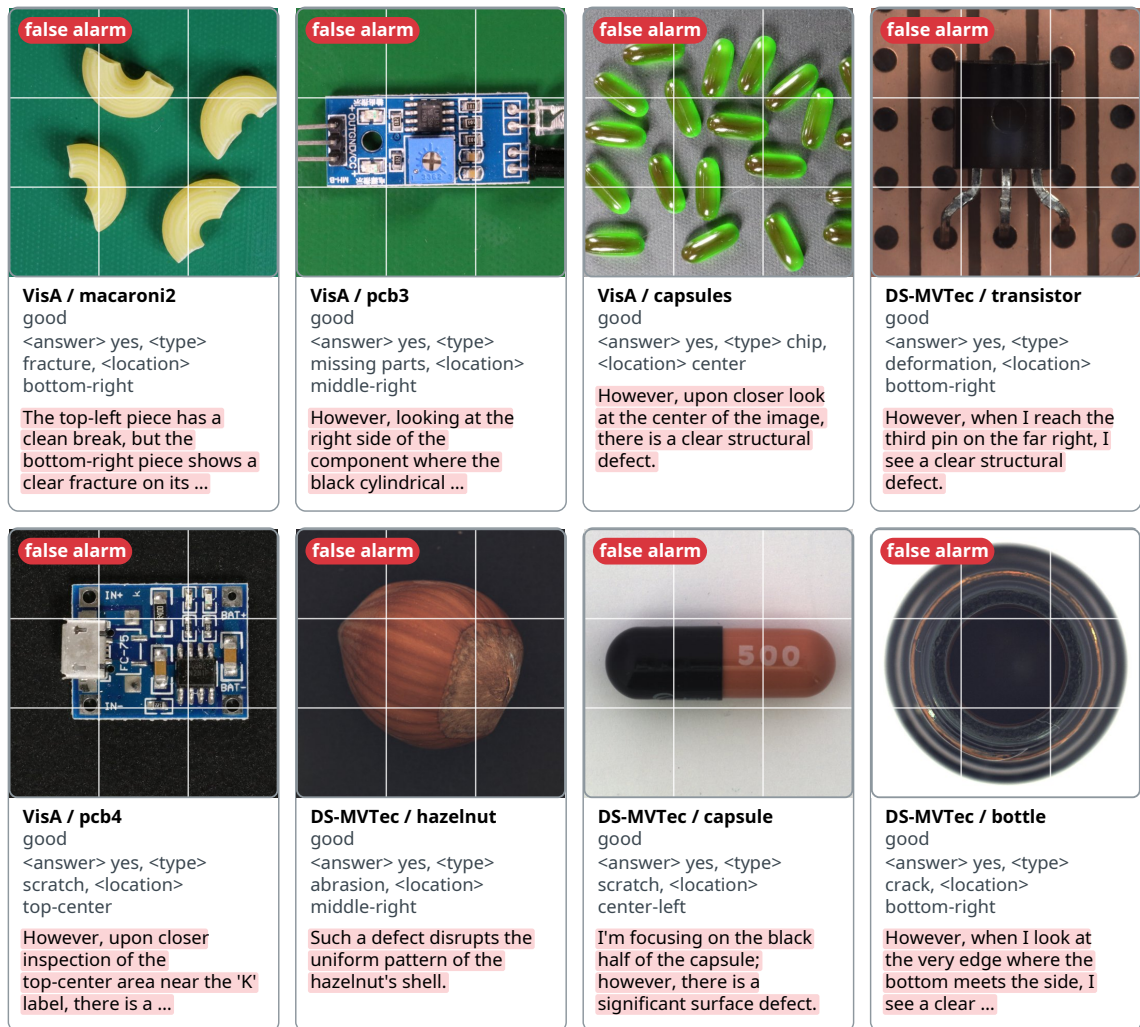


Figure F.2: Eight false alarms of the **KCR** model, one per product, chosen from the products with the most false positives. No mask exists, the part is clean. The shaded sentence is where the trace invents the defect.

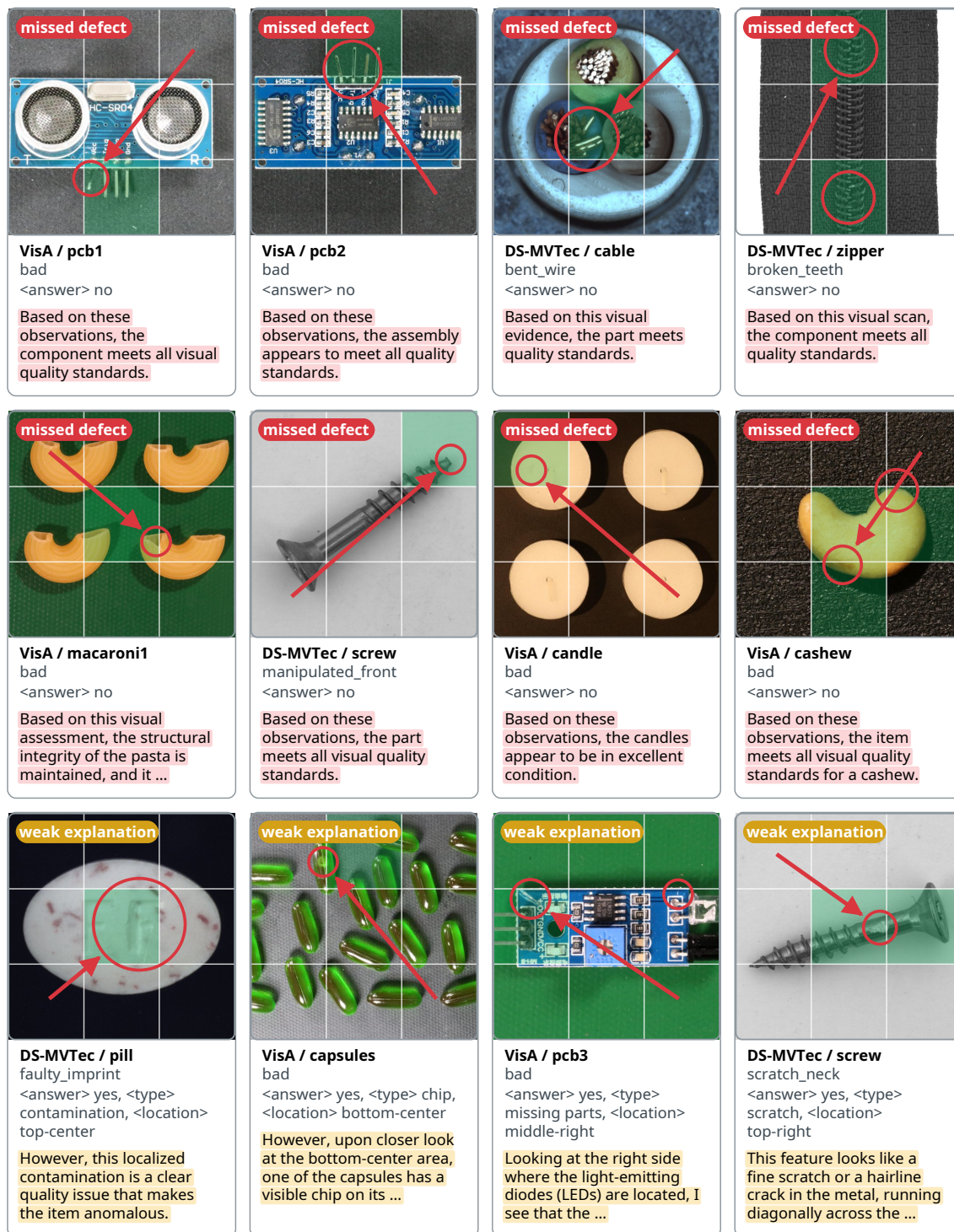
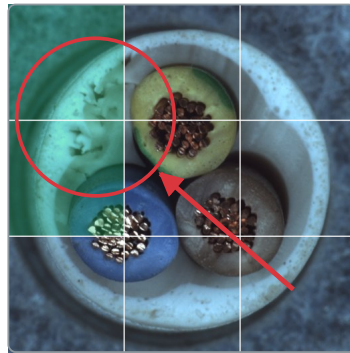


Figure E.3: Top two rows, eight missed defects of the KCR model, one per product, chosen from the products with the most false negatives: the trace scans the surface and calls the part clean while the mask marks a defect. Bottom row, four images with the right verdict where the explainability judge scored the trace 3 or 4 of 10, in every case because the trace describes a region that does not overlap the mask.

F.3. One Image through Base, SFT and SFT+GRPO

Figure F.4 follows one image through the three stages of the pipeline on identical prompts. It is chosen from the images on which the judge scored the SFT+GRPO trace 10 of 10 and on which the SFT model answered no: a DS-MVTec cable whose outer sheath is broken open at the left. The base Qwen2.5-VL-7B emits free text without the schema and finds no anomaly. The SFT checkpoint (ckpt-564) adopts the schema, walks over the cores and the insulation and also calls the cable sound. The SFT+GRPO checkpoint (ckpt-530) reaches the outer wall of the housing, names the missing section on the middle-left side and rejects the part, with a `<location>` tag that overlaps the mask.



cable, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

Base

`<answer>` no

No. The image shows a cross-section of a multi-core cable with three distinct cores, each wrapped in insulation of different colors (green, blue, and brown). The cores appear to be uniformly filled with copper strands, which is typical for such cables. There are no visible signs of damage, irregularities, or inconsistencies that would indicate an anomaly. Each core is neatly arranged within the cable's sheath, and the insulation looks intact without any signs of fraying or damage. **Therefore, based on the visual inspection, there are no anomalies present in this cable.**

SFT

`<answer>` no

I am inspecting this cable assembly, which consists of three distinct colored sections: blue, green, and brown. I'll start by examining the outer white protective casing. The circular boundary is intact with no visible cracks or abrasions. Moving to the internal components, each section contains a dense core of copper-colored metallic filaments. These filaments appear uniformly packed and show no signs of fraying, loose strands, or contamination. The transition between the different colored sections (blue, green, and brown) is sharp and well-defined, indicating a correct manufacturing process without mixing or bleeding of materials. There are no visible air bubbles or voids within the insulation layers. The surface of the cable shows some very minor, typical handling marks that do not compromise the structural integrity. **Overall, the cable appears to be in excellent condition.**

SFT+GRPO

`<answer>` yes, `<type>` missing parts, `<location>` middle-left

I am inspecting a cable assembly, specifically looking at the internal structure within a protective housing. I see three distinct sections of the cable, each with its own color-coded insulation and exposed copper conductors. The conductors are arranged in a dense, circular pattern and appear to be properly seated. My attention then shifts to the outer wall of the housing. **On the middle-left side of the circular rim, there is a significant structural defect.** A large section of the white outer material has been chipped away or was never properly formed, leaving a rough, irregular gap that exposes the underlying material. This missing material compromises the integrity of the seal and the mechanical protection for the cables. The rest of the circumference appears smooth and intact, making this localized damage stand out as a clear failure point.

Figure F.4: DS-MVTec cable/poke_insulation/001, ground truth defective, mask in the top-left and middle-left cells. Base and SFT answer no. SFT+GRPO answers yes with `<type>` missing parts and `<location>` middle-left: the verdict and the cell are right, the type is not the gold *poke insulation*. The shaded sentence in each trace is where it commits.

E.4. Post-GRPO Refinement against SFT+GRPO: Overlays

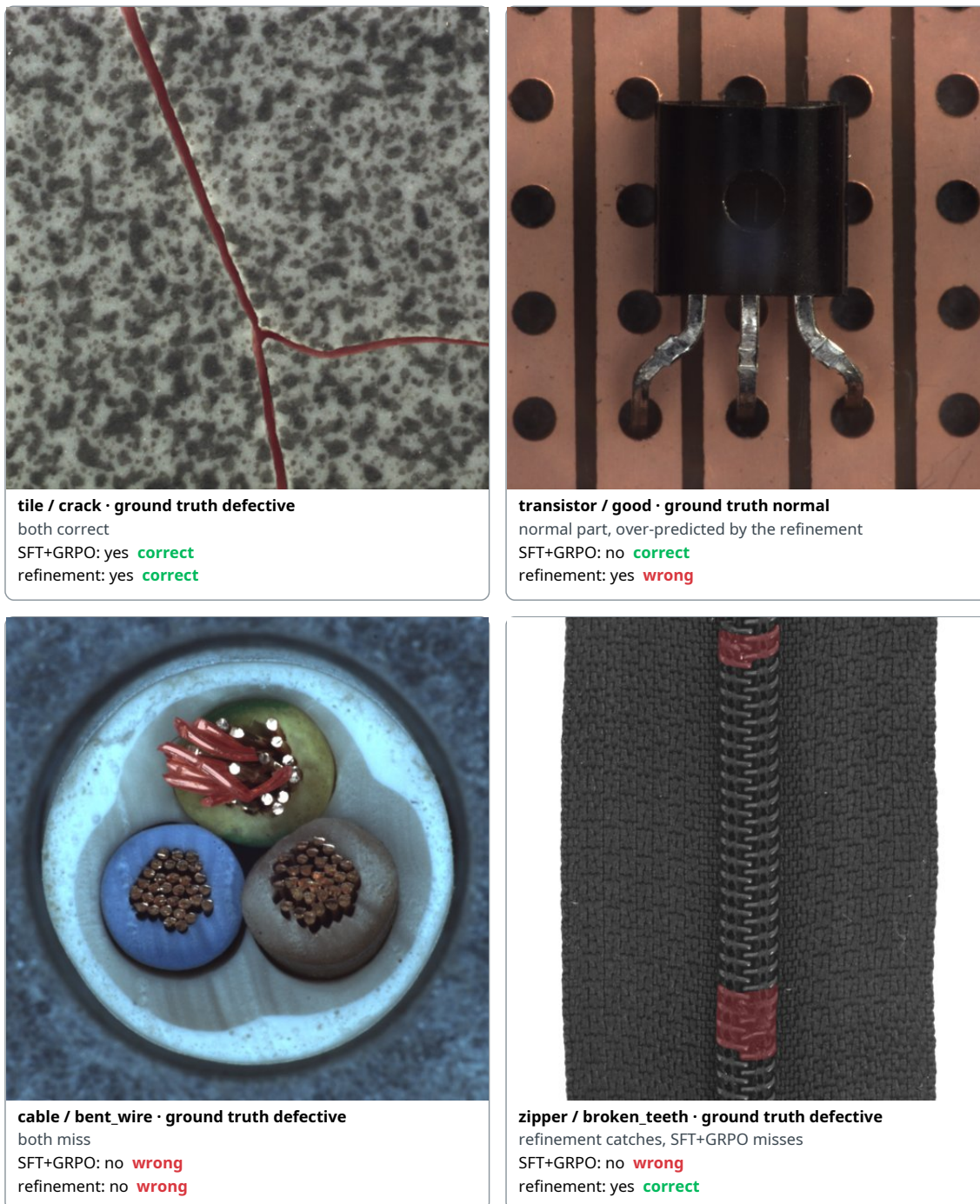
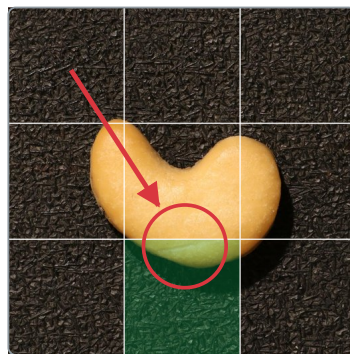


Figure E.5: Four DS-MVTec items with the ground-truth defect mask tinted red, comparing the SFT+GRPO checkpoint (ckpt-530) against the post-GRPO refinement on the cleaned 6K pool of \$6.7 (epoch 1). *Top-left*, a tile crack both checkpoints flag correctly. *Top-right*, a clean transistor where SFT+GRPO correctly says no defect and the refinement over-predicts one, representative of the OK-recall regression that drives the net loss on DS-MVTec. *Bottom-left*, a cable bent-wire case both miss. *Bottom-right*, a zipper broken-teeth case the refinement catches and SFT+GRPO misses.

F.5. SFT Against KCR on the Same Image

Each pair below is one image on which the headline supervised model (7B-frozen-6K, ckpt-564) answers no and the KCR model (ckpt-376) answers yes with a <location> tag that overlaps the ground-truth mask's 3 × 3 cells. Sixty-five images on DS-MVTec and VisA satisfy that rule. The 21 here were chosen by hand to cover twelve products, and the traces are reproduced verbatim from the evaluation files. The red circle encloses the ground-truth mask, one circle per separated region, with the arrow pointing to the largest, and the green shading marks its cells. The supervised trace never carries a <type> or <location> tag, because a trace that concludes no ends at the <answer> tag by construction (§3.2).



cashew, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting a single cashew nut. The overall shape is a characteristic semi-circular or crescent form with a smooth, rounded surface. I'll begin by checking the outer perimeter; the edges are clean and show no signs of chipping, cracking, or irregular molding. The top surface has a fine, granular texture typical of this product, and I don't see any deep scratches, pits, or foreign material embedded in the shell. The color is a consistent light tan/yellowish hue without any dark spots that might indicate mold or contamination. There are no visible cracks radiating from the center, which would suggest structural failure. The symmetry of the nut is maintained, and the thickness appears uniform across the visible face.

Based on these observations, the cashew meets all standard quality requirements.

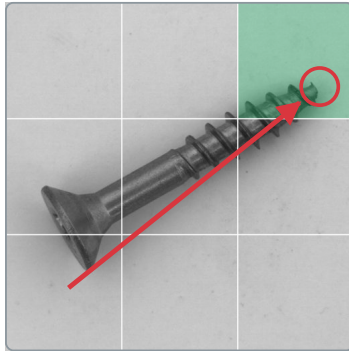
KCR

<answer> yes, <type> missing parts,
<location> bottom-center

I am inspecting a single cashew nut against a dark textured background. The overall shape of the nut is a standard half-moon form with a smooth, light-colored surface. I'll scan the perimeter first to check for structural integrity.

On the bottom edge, near the center, there is a clear indentation or missing piece on the otherwise smooth curve. This area shows a slight flattening and a break in the continuity of the outer shell, indicating that the nut has been chipped or broken. The rest of the surface appears uniform without any visible cracks, stains, or discoloration. The defect on the lower edge disrupts the expected symmetrical profile of the cashew.

Figure F.6: Pair 1: VisA, cashew, 086.JPG, ground truth bad, mask in the bottom-center cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



screw, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this screw. I'll start with the head; the flat surface is smooth and the outer edge is well-defined without any burrs or stripping marks from a screwdriver. The cross-recess pattern on the top-left is clearly visible and correctly formed. Moving to the shank, the metal has a consistent texture and color, showing no signs of corrosion or oxidation. The spiral grooves along the length of the shank are uniform and free of debris or manufacturing defects. I'm checking the overall straightness of the screw; it shows no signs of bending or twisting. There are no scratches or dents on the side surfaces.

The part appears to be in excellent condition.

KCR

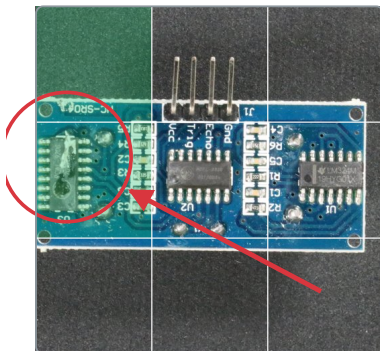
<answer> yes, <type> chip, <location> top-right

I am inspecting a metal screw with a hexagonal head and a spiral drive. I'll start by checking the overall shape; it appears straight and properly formed. The hexagonal head has clean edges without any signs of deformation or crushing. Looking at the side profile, the spiral grooves on the neck are uniform in depth and spacing.

However, upon closer inspection of the top-right edge of the screw's neck, there is a clear structural defect.

A small piece of the metal has been chipped away from the outer edge of one of the spiral ridges. This missing material breaks the continuity of the drive pattern and would likely interfere with the screw's ability to grip a surface. The rest of the surface shows some minor pitting typical of industrial metal, but this specific chip on the drive edge is a critical failure.

Figure E7: Pair 3: DS-MVTec, screw, 017.png, ground truth manipulated front, mask in the top-right cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



pcb2, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this PCB module, which appears to be an ultrasonic sensor interface board. I'll start by checking the main blue substrate; the solder mask is uniform and free from scratches or discoloration. The white silk-screened labels (pins, component names like 'U1', 'C1', etc.) are all sharp and legible without any smearing or missing segments. Moving to the components, the central integrated circuit (U1) and the surrounding resistors and capacitors (R1, C1, C3, C5, C6) are correctly positioned on their pads with good solder wetting. The IC pins are straight and show no signs of bridging. On the left side, the HCSR04 module is securely mounted, and its pins are properly aligned with the board holes. The four metal leads protruding from the top edge are straight and undamaged. No foreign objects or flux residue are visible on the board surface.

The assembly appears complete and correctly manufactured.

KCR

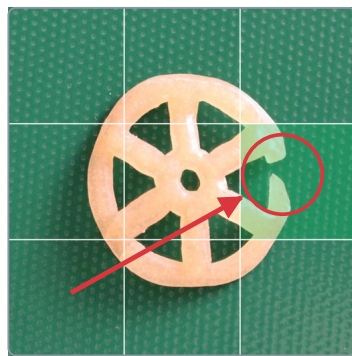
<answer> yes, <type> missing parts, <location> middle-left

I am inspecting a small blue printed circuit board (PCB) assembly. I'll start by checking the main components: the central micro-processor-like IC, the smaller resistors and capacitors labeled C1, C3, R1, etc., and the external pins on the top edge. The solder joints on the surface-mount components look clean and well-formed. However, looking at the left side of the board where the larger integrated circuit (U3) is located, there is a clear structural issue.

A portion of the plastic package housing this component is missing, leaving behind an empty footprint on the board.

This indicates a failure in component placement or damage during handling. Such a defect would likely prevent proper electrical connection with a PCB header. Based on this visual evidence, the part is rejected.

Figure E8: Pair 9: VisA, pcb2, 084.JPG, ground truth bad, mask in the middle-left, top-left cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



fryum, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

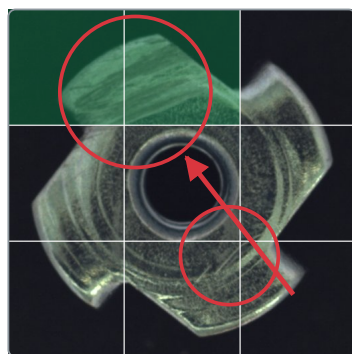
I am inspecting this plastic component, which appears to be a circular spacer or guide with internal structural ribs. I'll start by checking the outer circumference; it is perfectly round without any warping or chips. The central hole is well-defined and centered. Looking at the internal geometry, the six triangular cutouts are uniform in size and shape, showing no signs of molding defects or material blockage. The six outer tabs are also symmetrical and free from deformation. The surface has a slightly textured, matte finish that is consistent across the entire part. There are no visible cracks, scratches, or discoloration on either the top face or the edges. The material color is a consistent light tan/orange hue. Based on this thorough visual check, the part meets all quality standards.

KCR

<answer> yes, <type> missing parts, <location> middle-right

I am inspecting a circular component made of a light-colored material with a textured surface. The part has several triangular cutouts arranged around a central hub. I will scan the outer rim and the internal features for any irregularities. Most of the structure appears intact, but on the middle-right side of the outer circular boundary, there is a clear break in the geometry. A piece of the material is missing from the edge, creating an irregular notch instead of a smooth curve. This indentation disrupts the symmetry of the part's perimeter. The rest of the surface shows some minor pitting, which seems to be normal for this material, but the missing section on the right side is a structural defect.

Figure F.9: Pair 10: VisA, fryum, 014.JPG, ground truth bad, mask in the middle-right cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



metal_nut, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

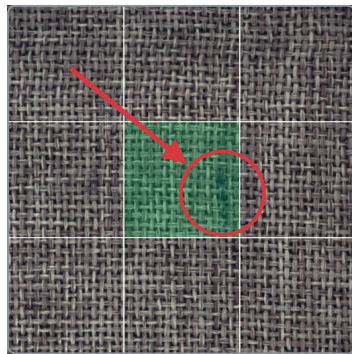
I am inspecting this metal_nut component from a top-down view. The part has a central circular hole and four outer lobes. I'll start by checking the central bore; it appears perfectly round with clean edges and no visible burrs or obstructions. Moving to the outer lobes, each of the four flanges shows consistent geometry. The surface texture across the lobes is uniform, showing a fine metallic grain without any signs of pitting, corrosion, or uneven coating. There are no cracks or fractures visible at the stress points near the inner hole and the outer perimeter. The overall shape is symmetrical and free of any warping or deformation. I've scanned the entire surface area for scratches or dents, and none are found. The material color is consistent throughout, indicating a stable manufacturing process. Based on these observations, the part meets all standard quality requirements.

KCR

<answer> yes, <type> missing parts, <location> top-left

I am inspecting a metal_nut with a hexagonal outer profile and a circular inner hole. I'll start by checking the overall symmetry and surface finish. The central bore appears clean and well-formed. Moving to the outer perimeter, most of the six points show consistent geometry and metallic texture. However, on the top-left point, there is a clear structural defect. A significant portion of the metal has been removed from this corner, creating a jagged, irregular edge instead of the sharp, rounded corner seen on the other five sides. This missing material compromises the part's structural integrity and functional profile. Based on these observations, the component is rejected.

Figure F.10: Pair 12: DS-MVTec, metal nut, 003.png, ground truth scratch, mask in the top-center, top-left cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



carpet, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this carpet sample, focusing on the uniformity of the weave and color. The texture is consistent across the entire frame, with a regular pattern of interlocking fibers. I'm scanning for any signs of loose threads, fraying at the edges, or knots that might be out of place. The color is a natural, earthy brown throughout, without any dark spots that could indicate staining or contamination. There are no visible snags or broken warp threads. The overall density of the weave appears even, and the surface finish is smooth. I don't see any evidence of crushing or deformation in the fibers.

Everything looks to be within normal manufacturing tolerances for this product type.

KCR

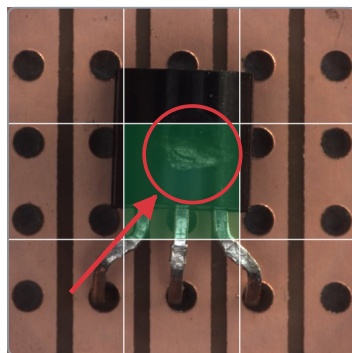
<answer> yes, <type> contamination, <location> center

I am inspecting a section of a brownish-gray woven fabric. The texture shows a regular pattern of interlocking threads. I will scan the surface from left to right.

In the center-right area, I detect a small, dark blue stain that has contaminated the fabric.

This discoloration appears as a tiny spot that breaks the uniform color of the weave. It looks like a localized contamination on the surface. The rest of the fabric maintains its consistent appearance without similar interruptions in the material's color or structure.

Figure F.11: Pair 13: DS-MVTec, carpet, 014.png, ground truth color, mask in the center cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



transistor, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting a black integrated circuit (IC) package with three metallic leads. The body of the IC is rectangular and appears to be made of a matte black plastic. I'll start by checking the body for cracks or chips; the surface looks smooth and intact. Next, I'm examining the three silver-colored metal leads extending from the bottom. Each lead shows a clean, bright finish without signs of corrosion or oxidation. The leads are properly spaced and appear to be straight, with no bending or warping that would interfere with soldering. The alignment of the leads relative to the plastic body is consistent. There are no visible contaminants or residues on either the plastic body or the metal pins. The circular holes in the background, likely part of a PCB, show clean edges without any bridging or debris.

Based on this visual assessment, the component is in good condition.

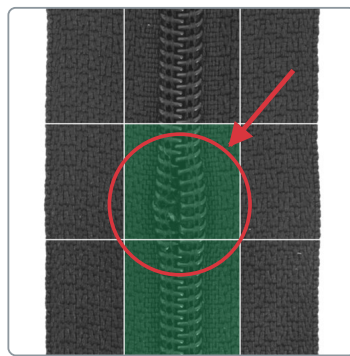
KCR

<answer> yes, <type> deformation, <location> center

I am inspecting a black integrated circuit (IC) package with three metal leads on a copper-colored PCB. The body of the IC appears rectangular and intact. I will now examine each of the three pins extending from the bottom. The left and right pins appear to be straight and properly formed. However, the central pin is severely deformed; it is bent upwards and twisted out of its intended parallel alignment compared to the other two leads.

This structural damage would prevent the component from being correctly mounted onto a board. The solder points where the leads enter the PCB look clean, but the mechanical integrity of the lead frame itself is compromised. Based on this visual evidence, the part is rejected.

Figure F.12: Pair 15: DS-MVTec, transistor, 007.png, ground truth damaged case, mask in the center cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



zipper, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this zipper assembly. I will start by examining the metal teeth along the center. Each tooth is properly seated and aligned with its neighbor, showing no signs of bending, missing elements, or misalignment that would impede function. The spacing between the teeth is consistent throughout the visible section. Next, I'll look at the fabric tape on either side. The weave appears tight and uniform, with no loose threads, fraying, or holes. The dark color is even across both sides of the tape. I'm also checking the transition area where the teeth meet the fabric; the attachment looks secure and clean. There are no signs of corrosion or surface contamination on the metallic parts.

Based on this visual scan, the component meets all quality standards.

KCR

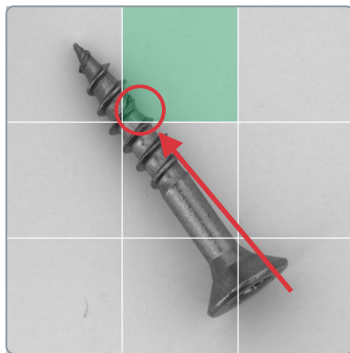
<answer> yes, <type> chip, <location> center

I am inspecting a section of a black zipper mounted on a dark fabric tape. I will focus first on the metal teeth.

While most teeth are evenly spaced and show consistent interlocking geometry, there is a visible chip on one of the teeth in the center of the image.

The metallic finish is otherwise uniform without signs of corrosion or discoloration. Next, I'll examine the fabric tape. The weave appears tight and consistent throughout the visible segment, with no loose threads or fraying along the edges where it meets the metal teeth. There are no stains or discolorations on the fabric surface. The alignment between the teeth and the tape is mostly straight, but the chip on the central tooth indicates a defect.

Figure F.13: Pair 16: DS-MVTec, zipper, 009.png, ground truth split teeth, mask in the bottom-center, center cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



screw, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this screw. I'll start with the head; it has a clean, flat finish with well-defined edges and no signs of stripping or impact damage. The central recessed area is clear. Moving to the shank, the metal surface shows a consistent metallic texture without any visible corrosion, rust, or discoloration. The threads along the neck are uniform and show no signs of crushing or irregular spacing. The transition from the head to the neck is smooth. At the very tip of the screw, the chamfered edge is sharp and correctly formed. There are no burrs or extra bits of material (flash) on the sides. The overall geometry is straight and symmetrical. No scratches or dents are visible on the body.

This part appears to be in excellent condition.

KCR

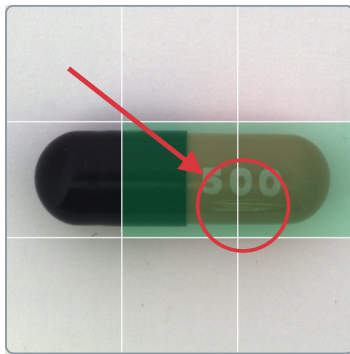
<answer> yes, <type> chip, <location> top-center

I am inspecting a metal screw. I'll start by checking the overall shape; it has a standard conical head with a straight neck and a spiral thread at the top. The surface finish appears to be a matte metallic texture.

I'm looking closely at the threads to ensure they are evenly spaced and undamaged, but there is a visible chip on the tip of the screw head.

The base where the head meets the neck looks clean without any signs of corrosion or discoloration. There are no visible burrs along the edges of the head or the sides of the neck. Based on this visual assessment, the part does not meet quality standards.

Figure F.14: Pair 21: DS-MVTec, screw, 018.png, ground truth thread side, mask in the top-center cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



capsule, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this pharmaceutical capsule. I'll start with the overall shape; it's a standard oval/capsule form. The black and orange segments are well-defined with a clean, straight seam where they meet. Looking at the orange face, the "500" marking is clearly printed in white and is centered correctly. There are no signs of chipping or crumbling around the edges of either the black or orange sections. The surface finish is smooth and consistent across both parts of the pill. I'm checking for any cracks or structural weaknesses, but the material appears solid and intact. No foreign particles or discoloration spots are visible on the surface.

The pill looks to be in perfect condition.

KCR

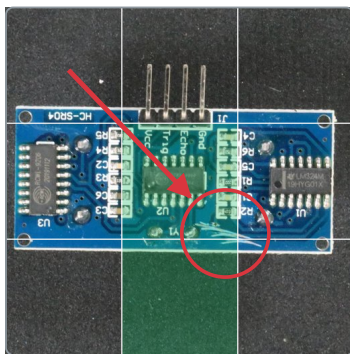
<answer> yes, <type> scratch,
<location> center-right

I am inspecting a black and orange capsule with white markings. The overall shape appears correct, but I'll examine the surface finish and the clarity of the printed text.

On the orange section, specifically on the right side where the "500" is printed, I see a thin, horizontal line that looks like a scratch or a hairline crack in the plastic.

This mark runs diagonally across the "0" in the number. It reflects light differently than the surrounding smooth surface, indicating it has broken the surface continuity. Such a mark could indicate handling damage or contact with a sharp object during packaging. Given this visible surface blemish on the product's outer shell, the item does not meet quality standards.

Figure F.15: Pair 24: DS-MVTec, capsule, 001.png, ground truth scratch, mask in the center, middle-right cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



pcb2, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this blue PCB module, which appears to be an ultrasonic distance sensor interface. I'll start by checking the surface of the board; the solder mask is a consistent blue with no signs of peeling, scratching, or discoloration. The white silkscreen markings (pins, component labels like 'U1', 'C1', etc.) are clear and legible without any smearing or fading. Next, I'm examining the components: the central integrated circuit (U1), the resistors (R1, R2, R3), and the capacitors (C1, C3, C5) are all correctly placed on their respective pads. The solder joints look shiny and well-formed, indicating good wetting. The three pins at the top and the four pins at the bottom are straight and properly aligned for external connection. There are no visible bridges between adjacent pins or components. The small circular mounting hole in the bottom-left corner is clean.

Overall, the assembly appears complete and correctly manufactured.

KCR

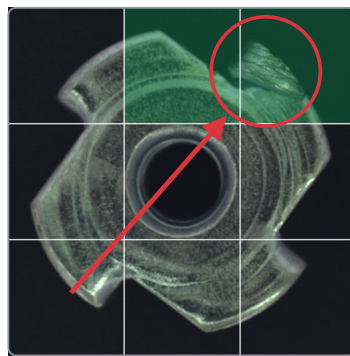
<answer> yes, <type> scratch,
<location> bottom-center

I am inspecting a small blue printed circuit board (PCB) with various surface-mount components and labeled pins. I'll start by checking the main board substrate and then move to the component markings. The central ICs and resistors appear correctly placed.

However, looking at the bottom edge of the PCB, specifically in the center-bottom area, there is a prominent physical scratch.

This mark appears as a bright, irregular line that cuts across the blue solder mask near the label "U1". It does not look like part of the design and represents a clear breach in the protective coating of the circuit traces. Such damage could indicate handling issues or contact with a sharp tool during assembly. Based on this visual evidence, the board is rejected.

Figure F.16: Pair 27: VisA, pcb2, 090.JPG, ground truth bad, mask in the bottom-center, center cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



metal_nut, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this metal_nut component from a top-down perspective. The part has a characteristic star-shaped or scalloped outer profile with a central through-hole. I'll begin by examining the outer perimeter; each of the six lobes appears structurally sound with no signs of deformation or edge chipping. Moving to the central bore, the circularity is well-maintained, and the internal wall surface looks smooth. The metallic finish across the entire body shows a consistent texture with typical machining marks, but no deep scratches or gouges that would indicate handling damage. There are no signs of corrosion, oxidation, or discoloration on the silver-toned surface. The symmetry of the part is preserved, and all structural elements align correctly according to the design's geometry.

Based on these observations, the part meets quality standards.

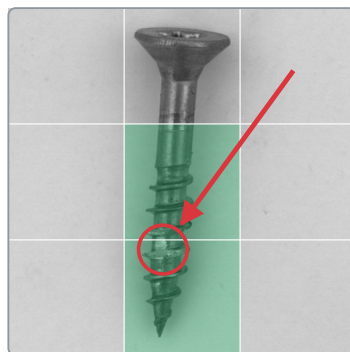
KCR

<answer> yes, <type> missing parts,
<location> top-right

I am inspecting a metallic nut with a hexagonal outer profile and a circular inner hole.

I'll scan each of the six outer points and the central bore. The overall finish has a textured, brushed metallic appearance. As I examine the top-right point of the nut, I see that it is not fully formed; there is a missing section on the outer edge, leaving a jagged, irregular shape where there should be a clean corner. This missing material compromises the symmetry of the part. The other five corners appear to be intact and correctly shaped. The central hole is clear and circular. Based on this visual evidence, the component is rejected.

Figure F.17: Pair 30: DS-MVTec, metal nut, 008.png, ground truth bent, mask in the top-center, top-right cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



screw, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this screw from a top-down view. I'll start by examining the head; the hexagonal shape is well-defined, and the internal threading appears intact with no visible stripping or damage to the recesses. Moving down the shank, the metal surface shows a consistent metallic texture with expected machining marks from the manufacturing process. The spiral grooves along the side are uniform and show no signs of clogging or deformation. I'm checking the outer diameter for any signs of corrosion, rust, or burrs, but the surface remains clean and smooth. The transition from the head to the neck is gradual and shows no evidence of cracking or stress fractures. The overall geometry is straight and aligns perfectly with the intended design. No foreign objects or debris are visible on the surface.

Based on these observations, the part meets all quality standards.

KCR

<answer> yes, <type> scratch,
<location> bottom-center

I am inspecting this metal screw. I'll start by checking the overall shape; it has a standard conical head with a sharp point. The surface finish appears to be a matte metallic texture. Looking closely at the head, the internal threading area looks clean and properly formed. However, upon examining the lower section of the screw's body, specifically the area just above the point, there is a visible scratch on the metal surface. This mark disrupts the uniform finish of the neck. It doesn't appear to be part of the material grain but rather a surface blemish. Given that screws rely on a smooth surface for proper drive into wood, this scratch represents a functional defect.

Figure F.18: Pair 33: DS-MVTec, screw, 013.png, ground truth thread top, mask in the bottom-center, center cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



capsule, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting a black and orange capsule. The orange half of the capsule has "500" printed on it in white. I'll check the surface of the orange section first. There are some very fine, dark linear marks or scratches on the surface, particularly near the bottom edge of the "500" print, but they appear to be superficial and within normal variation for this product type. The transition between the black and orange halves is clean and well-defined. The black half has a smooth, slightly glossy finish with no visible cracks or deformation. The embossed logo on the bottom left of the black section is clear. I don't see any signs of contamination, leaking, or structural damage. The overall shape is consistent with a standard pharmaceutical capsule.

Based on these observations, the item appears to be in good condition.

KCR

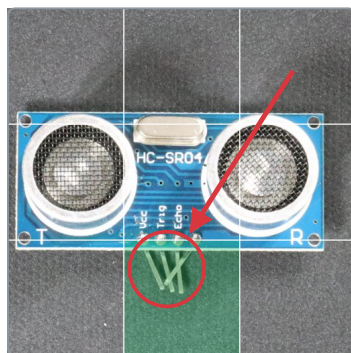
<answer> yes, <type> scratch,
<location> center-right

I am inspecting a black and orange capsule with white markings. The overall shape appears correct, but I'm scanning the surface for any imperfections. On the orange section, specifically on the right side near the "0" in the "500" marking, I see several fine, dark lines.

These look like surface scratches that have disrupted the smooth finish of the pill's coating.

They are oriented diagonally across the surface. Such marks can occur during handling or storage if the pill comes into contact with sharp objects or abrasive surfaces. Since these are visible surface defects on the finished product, it fails inspection.

Figure F.19: Pair 37: DS-MVTec, capsule, 016.png, ground truth scratch, mask in the center, middle-right cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



pcb1, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this PCB module, specifically an ultrasonic distance sensor board. I'll start by checking the main blue substrate; the solder mask is uniform and free from scratches or discoloration. The two large circular metal contacts on the left and right sides are properly seated within their respective housing and show no signs of deformation or oxidation. Looking at the bottom edge, the three metallic leads are straight and exhibit clean, bright solder points where they meet the board. The silkscreen markings, including 'T', 'R', and the component name 'HC-SR04', are sharp and legible without any smearing or ink residue. I'm also scanning the small surface-mount resistors and capacitors; they all appear correctly placed and soldered to their pads. There are no visible bridges, cold joints, or foreign objects on the board surface.

The overall assembly appears robust and follows standard manufacturing quality.

KCR

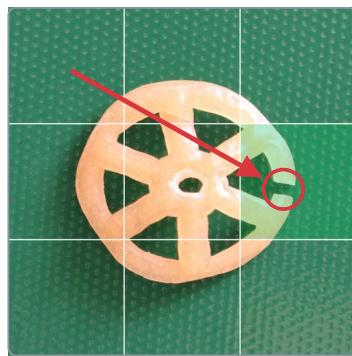
<answer> yes, <type> deformation,
<location> bottom-center

I am inspecting a blue PCB module with two circular metal components on the top surface. I'll check the solder joints and the overall assembly. The main board appears to be in good condition, but looking at the bottom edge, there is a clear structural issue.

Three metal leads are bent outward from the bottom-center edge, extending below the footprint of the board.

These leads are not properly soldered or aligned and protrude significantly beyond the intended boundary. This deformation would likely prevent proper mounting onto a larger assembly. The rest of the board shows standard markings and clean edges. Based on this visible evidence of bent pins, the part is rejected.

Figure F.20: Pair 38: VisA, pcb1, 010.JPG, ground truth bad, mask in the bottom-center cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



fryum, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this circular component, which appears to be a specialized washer or spacer made of a light-colored, possibly composite material. I'll start by checking the outer circumference; it is perfectly round with no chips or flat spots. The six internal structural ribs radiating from the center are evenly spaced and well-defined. I'm looking closely at the central hole and the surrounding area for any cracks or stress marks, but the surface remains smooth. There are some minor variations in color and texture, but these appear to be natural characteristics of the material rather than stains or contamination. The edges of the part are clean, showing no burrs or rough spots from the manufacturing process. Overall, the geometry is consistent and shows no signs of deformation or damage.

This part meets all visual quality standards.

KCR

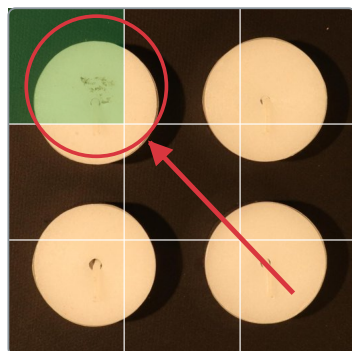
<answer> yes, <type> missing parts,
<location> middle-right

I am inspecting a circular component made of a light-colored material with a textured surface. The part has several radial ribs and a central hub. I'll scan the outer perimeter and the internal structural elements. Most of the part shows a consistent finish and geometry.

However, on the middle-right side of the outer ring, there is a clear structural defect.

A piece of the material is missing from the edge, creating a sharp notch or chip in what should be a smooth circular boundary. This indentation breaks the symmetry of the outer rim. The rest of the part's surface appears clean, but this edge damage is a significant flaw.

Figure F21: Pair 41: VisA, fryum, 016.JPG, ground truth bad, mask in the middle-right cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



candle, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting four circular wooden components, likely the bases of candles or similar items. I'll start by checking the overall shape; they are all perfectly round with clean edges. The surface finish is smooth, showing no signs of rough sanding or unfinished areas. I'm looking closely at the center of each piece where a small hole is located; the holes are cleanly drilled and centered without any splintering or cracking around the entry points. There are some very faint natural wood marks and grain lines visible, which are characteristic of the material and not defects. No cracks, dents, or discolorations are present on any of the surfaces. The color is a consistent light tan across all four parts.

Everything appears to be in excellent condition.

KCR

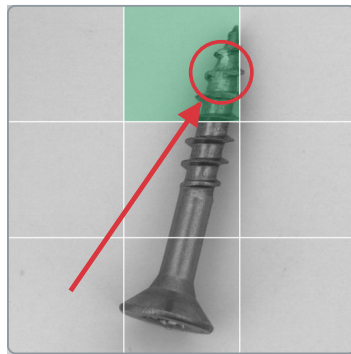
<answer> yes, <type> scratch,
<location> top-left

I am inspecting four circular wooden components. I'll start by checking the overall shape; they appear to be flat and circular as expected. Next, I'm looking at the surface texture and grain pattern. The wood shows a natural light-colored finish with some darker grain lines running through it, which is normal for this material.

However, upon closer inspection of the top-left component, there is a distinct vertical scratch on the left side that disrupts the smooth surface.

The other three components have clean edges and surfaces without any visible cracks or damage. There's also a small number '1' mark on the top-right piece, but it appears to be a standard parting line from the manufacturing process rather than a defect.

Figure F22: Pair 42: VisA, candle, 081.JPG, ground truth bad, mask in the top-left cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



screw, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this screw for surface and structural integrity. Starting with the head, the circular flange appears intact with no visible cracks or deformation in the metal. The central recessed area shows standard machining marks that are consistent across the diameter. Moving to the shank, the metallic finish is uniform throughout the length, showing no signs of corrosion, discoloration, or oxidation. I'm checking the spiral threads along the top; each ridge is well-defined and lacks any signs of stripping or mechanical wear. The transition from the head to the neck is smooth without any stress marks or burrs. There are no scratches or dents on the flat surfaces of the body. The overall geometry is symmetrical and aligns with the expected design for a high-quality fastener.

No anomalies are detected.

KCR

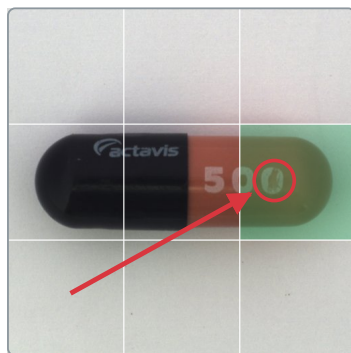
<answer> yes, <type> pit, <location> top-center

I am inspecting a metal screw with a hexagonal head and a spiral drive. I'll start by checking the overall shape; it appears straight and properly formed. The hexagonal neck shows clean edges without any signs of deformation or crushing. Moving to the main body, the metallic finish is uniform across most surfaces.

However, upon closer look at the top edge of the screw head, there is a visible indentation or small pit in the metal.

This irregularity breaks the smooth surface of the hexagonal profile. It looks like a minor surface defect that occurred during handling or processing. Given this visual evidence of a surface pit on the head's edge, the part does not meet quality standards.

Figure E23: Pair 45: DS-MVTec, screw, 016.png, ground truth thread top, mask in the top-center cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



capsule, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting a pharmaceutical capsule. The product consists of a black outer shell and an orange-brown inner core. I'll start by checking the integrity of the outer casing; the surface is smooth and the curvature is consistent without any cracks or chips. The 'Actavis' logo is clearly printed in white on the black section, with no smudging or fading. Moving to the orange-brown section, the color is uniform throughout. The embossed '500' marking is sharp and legible, indicating a clean press during manufacturing. I'm looking for any signs of contamination, such as dark spots or hair embedded in the material, but none are found. The seam where the two halves of the capsule meet is tight and even.

Overall, the structural and cosmetic features of this sample are within normal specifications.

KCR

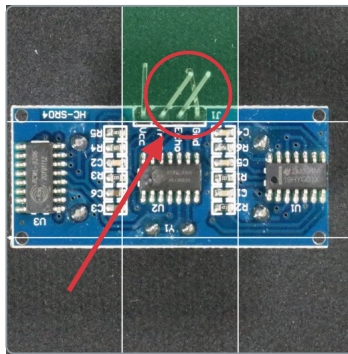
<answer> yes, <type> scratch, <location> middle-right

I am inspecting a black and orange capsule with white markings. The overall shape appears correct, but I'll check the surface finish and the clarity of the printed text. Looking at the orange section where the "500" is printed, I see that the number is clearly legible.

However, upon closer inspection of the right side of the orange half, specifically in the middle-right area, there is a visible scratch on the surface.

This mark disrupts the smooth texture of the pill's outer shell. It looks like a small abrasion or surface blemish that occurred during handling. The rest of the capsule, including the black end and the embossed logo, seems to be in good condition without similar marks.

Figure E24: Pair 49: DS-MVTec, capsule, 018.png, ground truth faulty imprint, mask in the middle-right cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



pcb2, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this PCB module, which appears to be an ultrasonic distance sensor interface. I'll start by checking the main blue substrate; the solder mask is uniform and free from scratches or peeling. The surface-mount components, including the central IC labeled U1 and the smaller resistors/capacitors, are all correctly positioned with no visible tombstoning or misalignment. The solder joints on the pins look clean and well-wetted. Looking at the three pins protruding from the top edge, they are straight and properly spaced. The white silkscreen markings (pins, component labels, and 'HC-SR04') are sharp and legible. There is no evidence of contamination, flux residue, or burnt areas on the board.

The overall assembly quality is high, and all components appear to be in their intended positions without any structural damage.

KCR

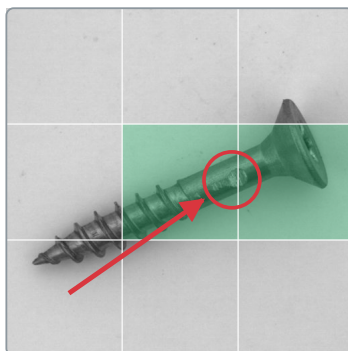
<answer> yes, <type> deformation, <location> top-center

I am inspecting a blue PCB module with various surface-mount components and external pins.

I'll start by checking the main board substrate and the solder joints on the chips.

The central area and the right-side components appear to be in good condition. However, looking at the top edge of the board where the pins are located, there is a clear structural issue. Three metal pins are protruding significantly from the top-center edge, bent outward at sharp angles. These pins are not properly seated into the board's pin header and are sticking out beyond the intended profile. This deformation would prevent the module from being correctly mounted onto a larger assembly. Based on this visual evidence, the part is rejected.

Figure F.25: Pair 60: VisA, pcb2, 008.JPG, ground truth bad, mask in the top-center cell. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.



screw, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

SFT

<answer> no

I am inspecting this screw to ensure it meets quality standards. I'll start with the head; the cross-recess pattern is well-defined and free from stripping or damage. The metal surface has a consistent finish without any signs of corrosion or discoloration. Moving down the neck, the threads appear uniform and properly spaced along the entire length of the shank. There are no visible burrs or deformations on the outer diameter that would interfere with its function. The tip of the screw is sharp and correctly angled for insertion. I've checked for any cracks or material fatigue, particularly at the transition points between the head and the neck, and found none.

The part appears structurally sound and matches the expected geometry for a standard screw.

KCR

<answer> yes, <type> scratch, <location> center-right

I am inspecting a metal screw. I'll start by checking the overall shape; it has a standard conical head with a flat top and a long, straight neck that tapers into a sharp point. The surface finish appears to be a matte metallic texture. I'm looking closely at the head's edges and the transition points where the metal meets the different sections.

On the right side of the head, specifically along the edge of the flat top, there is a visible scratch.

This mark disrupts the uniform finish of the metal surface. It looks like a small abrasion or a shallow gouge in the material. Given this localized surface damage on the head, the part does not meet quality standards.

Figure F.26: Pair 63: DS-MVTec, screw, 004.png, ground truth scratch neck, mask in the center, middle-right cells. The shaded sentences are where the supervised trace calls the part clean and where the corrective trace names the cue.

G

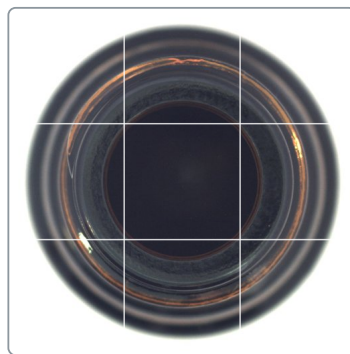
Teacher vs. Student Traces

This appendix places the **Gemini 2.5-Flash** teacher side by side with the distilled 7B student (our SFT+GRPO Qwen2.5-VL-7B, ckpt-530) on the *same* images. The teacher was evaluated zero-shot with `thinking_budget=0`, so its replies are terse. It emits the structured schema (`<think> / <type> / <location> / <answer>`) but rarely more than a single justifying sentence, and on many VisA samples it collapses to a bare `<answer> No </answer>`. The student, trained on the longer **AnomalyThink** traces, produces a multi-step “inspection walk-through” before committing to a verdict. Each figure shows the image with the annotated cells shaded where a mask exists, the teacher’s trace on the left, the student’s on the right, and the sentence where each commits shaded green when its verdict is right and red when it is wrong. Traces are reproduced verbatim from the evaluation files.

The aggregate picture from Chapter 6 motivates the cases shown here: on DS-MVTec the student *surpasses its own teacher* (balanced accuracy 82.73% vs 81.52%), whereas on VisA the teacher still leads (75.18% vs 70.39%). All four balanced accuracies are recomputed directly from the confusion-matrix counts of the four evaluation JSONs. The selected examples illustrate the mechanism behind those numbers: on DS-MVTec the student’s careful re-inspection fixes several of the teacher’s hasty errors, while on VisA the teacher’s conservative “No” bias both rescues normal images and silently misses genuine defects.

G.1. DS-MVTec: Student Corrects the Teacher

The first case is a DS-MVTec good image that the teacher *over-rejects* (a false positive) but that the student correctly accepts after an explicit, region-by-region inspection. The second is the mirror image: an anomalous capsule that the teacher *under-rejects* (a false negative) and the student correctly rejects.



bottle, ground truth: normal
no annotation, the part is clean

Teacher, Gemini 2.5 Flash

<answer> yes, <type> discoloration/residue, <location> around the inner rim of the bottle, particularly visible in the upper right and lower left quadrants.

There are several areas around the inner rim of the bottle that show discoloration and what appears to be residue or uneven coating, indicating a defect in the finish.

Student, SFT+GRPO

<answer> no

I am inspecting the interior of a dark-colored bottle, viewed from above. The central opening is circular and appears to be free of any blockages or irregularities. I'm scanning the inner walls for any signs of contamination, such as discoloration or foreign particles, but the surface looks clean. The outer rim of the bottle shows a consistent thickness and a smooth finish without any visible nicks or chips. There are no cracks or stress marks evident on the material. The overall symmetry of the part is maintained, indicating no warping during the manufacturing process.

Based on these observations, the component meets all visual quality standards.

Figure G.1: DS-MVTec bottle/good/005. **GT: No** (normal). **Teacher verdict: Yes** (false positive). **Student verdict: No** (correct). The teacher hallucinates “residue / uneven coating” on the inner rim from glare. The student inspects opening, inner walls and rim in turn and finds the part clean.



capsule, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

Teacher, Gemini 2.5 Flash

<answer> no, <type> minor surface imperfections, <location> near the "0" in "500" and on the black part of the capsule

The capsule shows some minor surface imperfections, specifically a small scratch or mark near the "0" in "500" and a slight discoloration or scuff on the black part of the capsule. However, these are very minor and do not appear to be significant defects that would affect the product's integrity or function.

Student, SFT+GRPO

<answer> yes, <type> scratch, <location> bottom-center

I am inspecting a black and orange capsule pill. The pill has "500" printed on its orange half. I'll start by checking the overall shape and surface. The black cap looks smooth and properly sealed. Moving to the orange body, the printing is clear.

However, upon closer look at the bottom-center of the orange section, there is a small, irregular indentation or mark that looks like a shallow gouge or scratch in the plastic. This blemish breaks the otherwise smooth, matte finish of the pill's surface. It doesn't appear to be a simple shadow or reflection, but rather a physical mark on the material itself. Such surface damage could indicate poor handling during packaging or storage. Based on this visual evidence, the pill is rejected.

Figure G.2: DS-MVTec capsule/faulty_imprint/016. **GT: Yes** (anomalous). **Teacher verdict: No** (false negative). **Student verdict: Yes** (correct). The teacher sees a scratch but dismisses it as “very minor” and answers No. The student treats the same surface gouge as a rejecting defect.

G.2. DS-MVTec: Teacher and Student Agree

When the defect is unambiguous, both models reach the right verdict. The difference is one of *style*: the teacher emits a one-line justification, the student narrates a full inspection. Both are counted as correct (and both emit the full schema), so neither has an accuracy edge here. This is the common ground that underlies the close 82.73 vs 81.52 DS-MVTec scores.



hazelnut, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

Teacher, Gemini 2.5 Flash

<answer> yes, <type> cracked shell,
<location> center of the hazelnut

The hazelnut has a clear crack or split in its shell, which is a defect.

Student, SFT+GRPO

<answer> yes, <type> scratch,
<location> center

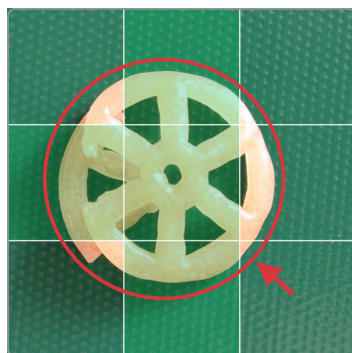
I am inspecting a single hazelnut. The overall shape of the nut is rounded and typical, with a brownish-tan color and visible fibrous texture on the top. I am scanning the surface for any signs of damage or contamination.

In the center of the nut, I see a distinct horizontal indentation or score mark. This feature appears as a dark, narrow trough that has breached the otherwise smooth, convex surface of the shell. It does not look like a natural variation in the wood grain but rather a mechanical scratch or scoring from handling. Such a defect could indicate poor surface quality or damage during processing. Given this visual evidence, the part is rejected.

Figure G.3: DS-MVTec hazelnut/cut/007. **GT: Yes** (anomalous). **Teacher verdict: Yes** (correct). **Student verdict: Yes** (correct). Both flag the central surface defect. The teacher calls it a “cracked shell”, the student a “scratch”, and both localise it to the centre.

G.3. VisA: Where the Teacher's Bias Cuts Both Ways

On VisA the teacher's conservative “No” tendency is double-edged. The first case (anomalous) shows the cost: the teacher misses a genuine defect (a false negative, the kind of recall-conservative error that gives the teacher 475 false negatives on VisA), while the student catches it. The second case (normal) shows the benefit: the teacher's blunt “No” is correct, whereas the student's eager inspection invents a defect and over-rejects.



fryum, ground truth: defective
shaded cell: annotated location
arrow: the defect itself

Teacher, Gemini 2.5 Flash

<answer> no

The fryum appears to be a complete and well-formed wheel shape, with no breaks, missing pieces, or deformities visible.

Student, SFT+GRPO

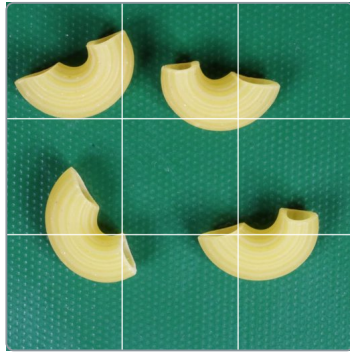
<answer> yes, <type> missing parts,
<location> middle-left

I am inspecting a small, circular orange component with a central hub and radiating spokes. I'll scan the outer rim and the internal geometry. The overall structure appears to be made of a molded plastic or composite material. As I examine the left side of the outer ring, I find a clear structural defect.

A large section of the outer rim has been broken off or was never properly formed, leaving a flat, irregular surface where there should be a rounded edge.

This missing piece significantly alters the part's profile and would likely affect its ability to fit into a larger assembly. The rest of the component, including the central hub and the remaining spokes, seems intact, but this localized damage on the left edge is a major failure.

Figure G.4: VisA fryum/bad/046. **GT: Yes** (anomalous). **Teacher verdict: No** (false negative). **Student verdict: Yes** (correct). The teacher declares the wheel “complete and well-formed” with no missing pieces. The student spots that a section of the outer rim is broken off.



macaroni2, ground truth: normal
no annotation, the part is clean

Teacher, Gemini 2.5 Flash

<answer> no

(bare <answer> tag, no reasoning)

Student, SFT+GRPO

<answer> yes, <type> missing parts,
<location> bottom-left

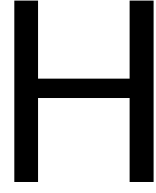
I am inspecting four yellow plastic components, each shaped like a quarter-circle or a bow tie. I'll start by checking the overall geometry and surface texture. The three components at the top and bottom appear to have smooth, continuous edges and consistent material density.

However, when I look closely at the bottom-left component, I see a clear structural defect.

There is a significant chunk missing from the left edge of the bow-tie shape, creating a jagged, irregular boundary instead of a smooth curve. This missing material breaks the symmetry of the part and represents a clear instance of damage or incomplete molding. The rest of the part shows some minor surface texture but no other major flaws. Based on this visual evidence, the part is rejected.

Figure G.5: VisA macaroni2/good/0016. **GT: No** (normal). **Teacher verdict: No** (correct). **Student verdict: Yes** (false positive). The teacher's bare "No" happens to be right. The student mistakes the natural shape of the bottom-left piece for a "missing part" and over-rejects, the type of error behind the teacher's VisA lead.

Across these five matched images the pattern is consistent with the headline numbers. On DS-MVTec the student's longer, region-by-region reasoning resolves borderline cases in its favour and lets it edge past the teacher. On VisA the same eagerness to find and name a defect produces false positives that the teacher's terse, conservative replies avoid.



Training Curves

This appendix collects the training-time diagnostics for the headline supervised fine-tuning (SFT) and GRPO runs of Chapters 4 and 5. All balanced-accuracy (BA) values are recomputed from the per-checkpoint evaluation JSONs using $BA = \frac{1}{2} \left(\frac{tp}{tp+fn} + \frac{tn}{tn+fp} \right)$ on the $\{tp, tn, fp, fn\}$ confusion counts. The GRPO traces are read directly from the `log_history` of the run's `trainer_state.json`.

H.1. SFT Balanced Accuracy vs. Epoch

Figure H.1 tracks BA on DS-MVtec and VisA (both cross-domain MMAD benchmarks, since the model is trained only on Real-IAD traces) across the four epochs of the two frozen-vision SFT runs: the 6K-trace headline run (`sft_qwen25v1_7b_zeroshot_6k_frozen`, checkpoints 188/376/564/752) and the larger 15K-trace run (`sft_qwen25v1_7b_15k_frozen`, checkpoints 453/906/1359/1812). Both runs use the full-schema train prompt at evaluation. The 6K run only retained a VisA evaluation at epoch 3, shown as a single marker.

Two patterns are visible. First, BA on DS-MVtec is non-monotone in epoch and peaks early (the 6K run is best at epoch 3 with BA = 80.16%, and is already at 79.24% after a single epoch), so additional epochs do not help and can hurt. Second, the larger 15K dataset does *not* translate into higher BA: across every epoch the 15K run sits several points below the 6K run on DS-MVtec (e.g. epoch 3: 71.66% vs. 80.16%). On VisA the 15K run stays in the 64 to 67% band across epochs with no clear upward trend, and the 6K run's single VisA evaluation (epoch 3, 64.78%) sits in the same band, underscoring the cross-dataset generalisation gap discussed in Chapter 6.

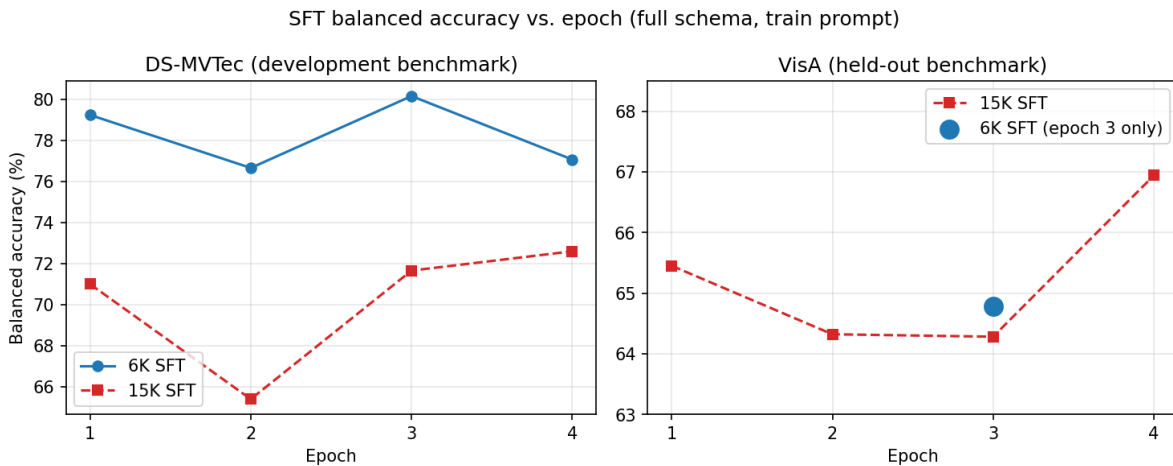


Figure H.1: Balanced accuracy vs. epoch for the two frozen-vision SFT runs, evaluated with the full-schema train prompt. *Left:* DS-MVtec, peaking at epoch 3 for the 6K run (80.16%). *Right:* VisA. The 15K run stays in the 64 to 67% band, and the 6K run's single retained VisA checkpoint (epoch 3, 64.78%) sits on top of the 15K curve. On DS-MVtec more traces (15K) do not beat the 6K run, and more epochs do not monotonically help.

H.2. GRPO Training Dynamics

Figures H.2 and H.3 show the GRPO dynamics for the SFT+GRPO run (grpo_qwen25v1_7b_6k_frozen_ep3_full_run2, checkpoint-1060, 1060 logged steps spanning 2 epochs), initialised from the strong 6K-frozen SFT checkpoint. As established in Chapter 5 and Appendix E, the optimised reward is the *unweighted* sum of two reward functions, $\text{reward} = \text{accuracy_reward} + \text{consistency_reward}$, with a theoretical maximum of 3.0. The accuracy reward is itself a sum of a binary verdict term (max 1.0), a Nomic-similarity type term (binned, max 0.5), and a 3×3 region-match location term (max 0.5), for a sub-maximum of 2.0. The consistency reward is a binary full-schema match (max 1.0). We verified that the logged reward equals $\text{accuracy_reward} + \text{consistency_reward}$ at every one of the 1060 steps.

The smoothed mean reward (25-step moving average) rises modestly from ≈ 1.7 to ≈ 2.0 over training and then plateaus, never approaching the 3.0 ceiling. The per-step reward (grey) is noisy, ranging from 0.56 to 2.83. Both components contribute to the rise: the accuracy reward climbs from ≈ 0.77 towards ≈ 1.1 (smoothed) and the consistency reward from ≈ 0.72 towards ≈ 0.85 . Read alongside the SFT curves above and the benchmark results in Chapter 6, this is the central observation: *the GRPO reward rises much faster than held-out balanced accuracy*. Run 2 gains +2.57 pp on DS-MVTEC (Table 6.4) while the reward climbs by roughly a third, and on the stronger Arm-C initialisation of \$6.8 the same reward rise buys no accuracy at all. This indicates that the gains are concentrated in schema-formatting and the soft type/location sub-signals rather than in the binary verdict that drives BA.

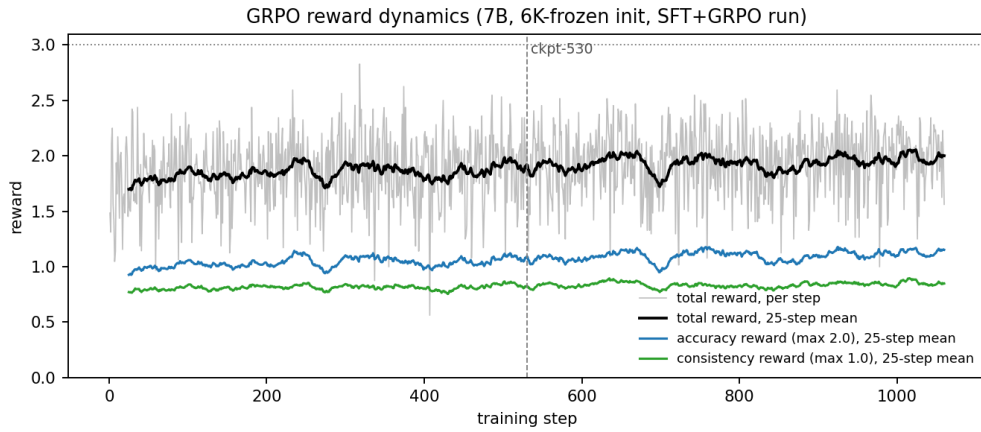


Figure H.2: GRPO reward vs. training step for the SFT+GRPO run (6K-frozen SFT init). Grey: raw per-step total reward. Black: 25-step moving average rising from ≈ 1.7 to ≈ 2.0 . Blue/green: smoothed accuracy reward (verdict+type+location, max 2.0) and consistency reward (full-schema match, max 1.0). The red dotted line marks the unweighted maximum reward of 3.0, which is never approached.

Figure H.3 reports the two remaining logged diagnostics. The KL divergence to the frozen reference policy grows from 0 and stabilises around 0.10 to 0.20 (range 0.0 to 0.197), confirming the policy stays close to the SFT init throughout. The mean completion length stays essentially flat in a narrow band (153 to 186 tokens, 176 at step 1, 163 at step 1060), i.e. GRPO does not induce reasoning-length inflation here.

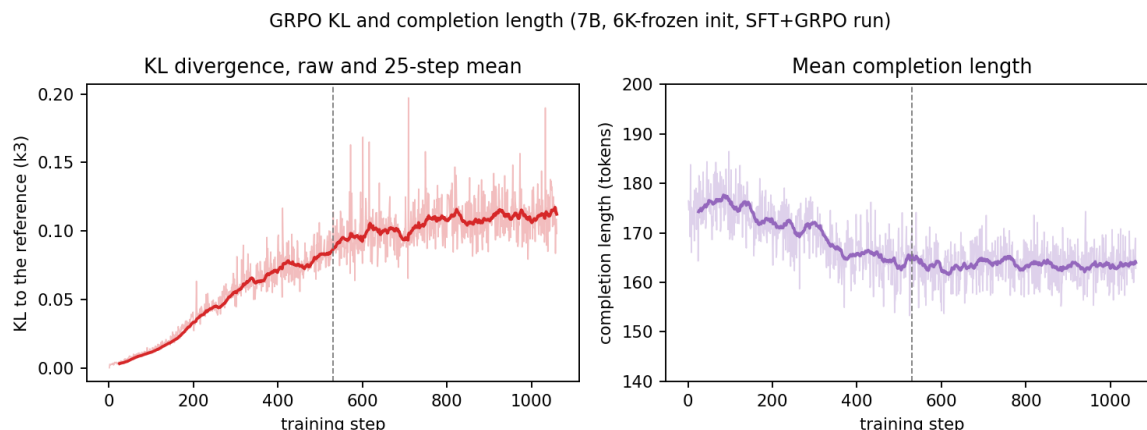


Figure H.3: GRPO secondary diagnostics for the SFT+GRPO run. *Left:* KL divergence to the reference policy, rising from 0 and stabilising in the 0.10 to 0.20 range (raw faint, 25-step average bold). *Right:* mean completion length, flat at 153 to 186 tokens across training. The policy stays close to its 6K-frozen SFT init and does not lengthen its reasoning.

H.3. Per-Item Loss of the 6K Pool under the GRPO Checkpoint

§6.7 attributes the regression of the post-GRPO refinement run to a loss asymmetry between the kept and the patched items of the 6K pool. This section is the measurement behind that sentence. The SFT+GRPO checkpoint (ckpt-530) was run once over all 6,000 items of the pool with the training chat template, the training image budget of 262,144 pixels and teacher forcing, and the cross-entropy was averaged over the assistant tokens of each item, exactly the tokens the SFT loss of Eq. 4.1 sees. Items are classed by the routing of §5.8: kept (the model's own rollout), corrected (teacher-corrected after a failed rollout) or rewritten (teacher-rewritten for weak grounding). Table H.1 gives the result, and Listing H.1 the log of the run. The script is `measure_iter2_pool_loss.py` in the repository.

Table H.1: Cross-entropy of ckpt-530 on the assistant tokens of the 6K pool, by item class. “Share of loss” is the class's summed cross-entropy divided by the total over all 6,000 items. The anomalous half of the pool (3,000 items, 52.3% of the assistant tokens) produces 67.0% of the summed loss.

Class	Items	Anomalous	Assistant tokens	Nats per token	Share of loss
Kept	3,557	653	533,068	0.44	36.3%
Corrected	812	717	130,037	1.47	29.6%
Rewritten	1,631	1,630	257,240	0.85	34.1%
Patched (corrected + rewritten)	2,443	2,347	387,277	1.06	63.7%

Listing H.1: Log of the measurement run (10 September 2026, one RTX A6000, 24 minutes) and the first record of the per-item file.

```
breakdown Counter({'kept': 3557, 'rewritten': 1631, 'corrected': 812})
100/6000 24s
1000/6000 231s
3000/6000 694s
6000/6000 1438s
{"kept": {"n_items": 3557, "n_anomalous": 653, "assistant_tokens": 533068,
  "mean_ce_per_token": 0.4396, "share_of_total_summed_ce": 0.3633},
 "corrected": {"n_items": 812, "n_anomalous": 717, "assistant_tokens": 130037,
  "mean_ce_per_token": 1.4695, "share_of_total_summed_ce": 0.2962},
 "rewritten": {"n_items": 1631, "n_anomalous": 1630, "assistant_tokens": 257240,
  "mean_ce_per_token": 0.8541, "share_of_total_summed_ce": 0.3406}}
{"idx": 0, "kind": "kept", "image": ".../vcpill/OK/S0078/vcpill_0078_OK_C1_20230920154321.jpg",
 "anomalous": false, "n_assistant_tokens": 150, "mean_ce": 0.3903, "sum_ce": 58.54}
```

Two readings follow. The kept items are not loss-free: a rollout sampled at temperature 0.7 still carries 0.44 nats per token under the policy that wrote it, so the kept half of the pool supplies a third of the gradient rather

than none of it. And the asymmetry is about two to one, not the near total one that the item counts of the patched set alone would suggest. Two thirds of the first epoch's loss comes from anomalous items on a corpus that is balanced by count, which is enough to move the operating point towards anomaly recall (Figure 6.4).



Datasheet for the AnomalyThink Dataset

This appendix documents the **AnomalyThink** dataset following the “Datasheets for Datasets” template of Gebru et al. [17] (the seven standard sections: *Motivation*, *Composition*, *Collection Process*, *Preprocessing/Cleaning/Labeling*, *Uses*, *Distribution* and *Maintenance*), in the recommended question-and-answer style. The intent is to make the dataset’s design choices, biases and intended uses explicit and reusable. All quantitative claims here are consistent with the split-composition table in Appendix A (Table A.3). The full generation prompt is reproduced verbatim in Appendix B.1.

I.1. Motivation

For what purpose was the dataset created? **AnomalyThink** was created to supply structured, machine-parseable chain-of-thought *reasoning traces* for training and post-training a Qwen2.5-VL industrial-anomaly-detection model that not only *detects* a defect but also *localises* and *explains* it in natural language. Each trace pairs a single Real-IAD test image with a six-phase reasoning rationale and an XML body carrying a `<think>` block and, for anomalous items, `<location>`, `<type>` and `<answer>` tags. This four-tag schema is what allows the dataset to be consumed directly by the four-sub-signal GRPO reward of Chapter 5 (format, verdict, type, location). No openly-released reasoning-trace dataset built on Real-IAD and purpose-designed for structured GRPO consumption existed prior to this work.

Who created the dataset and on whose behalf? The dataset was assembled by the thesis author as part of an MSc thesis at TU Delft. The underlying images are sourced from the public Real-IAD dataset. The reasoning traces were generated by Gemini 2.5-Flash. No external funding was tied to the dataset specifically.

What gap does it fill relative to existing datasets? It complements **Anomaly-Instruct-125K** [75], which is larger and openly released but is conversation-format and draws its industrial subset from MVTec and VisA (exactly the benchmarks held out for evaluation here), so a single conversation does not expose machine-parseable `<location>/<type>/<answer>` tags. It contrasts with **Expert-AD** [39], the design-closest dataset, which is *not* publicly distributed (only the trained IAD-R1 checkpoints are released). **AnomalyThink** is, to the best of our knowledge, the first openly-released reasoning-trace dataset that is both built on Real-IAD and structured for direct GRPO consumption.

I.2. Composition

What do the instances represent? Each instance is a single-image industrial inspection example: one top-down Real-IAD photograph of a manufactured product, paired with a structured reasoning trace and a gold verdict label (anomaly / normal). There is exactly one test image per instance. Auxiliary images (overlay, reference) exist only at generation time (see Collection Process) and are *not* part of any instance consumed by the model at training or inference.

How many instances are there in total? The dataset comprises three product- and class-stratified splits, totalling 14,472 instances ($\approx 15K$). Counts are by the *true* `<answer>` verdict label, not by the source acquisition

folder:

Table I.1: Per-split composition of **AnomalyThink**, counted by true `<answer>` verdict. Identical to Appendix A, Table A.3.

Split	Products	Anomaly	Normal	Total
SFT pool	30	3,000	3,000	6,000
GRPO pool	23	2,118	2,118	4,236
Held-out Real-IAD	23	2,129	2,107	4,236
Total	30	7,247	7,225	14,472

Every split is balanced to $\approx 50/50$ anomaly/normal by true verdict. The SFT pool spans all 30 Real-IAD products. The GRPO pool and the held-out split span the same 23 products (the per-product coverage table is Appendix A, Table A.2).

Does the dataset contain all possible instances or a sample? It is a *sample* of Real-IAD. Real-IAD provides 150K+ images of 30 products from five camera viewpoints (C1 to C5). **AnomalyThink** is restricted to the single top-down `_C1_` camera angle and is class- and product-stratified to roughly 100 anomaly + 100 normal per product in the SFT pool (≈ 92 per class for the GRPO and held-out splits). It is therefore not a complete or uniform-random sample of Real-IAD.

What data does each instance consist of? A single processed RGB image path, a natural-language reasoning trace (six-phase template, target 120 to 200 words, observed mean ≈ 137 words), and structured XML tags: a `<think>` block plus `<location>`, `<type>` and `<answer>` for anomalous items, or only `<answer>` for normal items. Defect types are drawn from a controlled vocabulary of 26 categories (organised as Surface / Structural / Completeness / Other) and locations from a 9-cell 3×3 grid. Both vocabularies are reproduced in Appendix B.1.

Is there a label or target associated with each instance? Yes, the gold verdict (anomaly / normal) carried in the `<answer>` tag, plus, for anomalies, a defect type and a grid location. These labels are derived from Real-IAD ground truth (see Labeling, below), not annotated from scratch.

Is any information missing from individual instances? Anomalous instances do not retain the binary defect mask or the red overlay used during generation. These privileged inputs are discarded by design, so that the trained student sees only the test image. Normal instances never had a mask. Real-IAD’s internal two-letter defect codes (e.g. BX, HS, ZW) are translated into the 26-type vocabulary and the original codes are not stored in the trace.

Are relationships between instances made explicit, and are there recommended splits? The three splits (SFT / GRPO / held-out) are explicit and overlap-prevented by image basename exclusion. Important caveat: the held-out split is *sample-level*, not *product-disjoint*: specific image IDs are withheld, but its 23 products overlap with the products seen in the SFT and GRPO pools. The held-out split is therefore an *in-domain, out-of-sample* check, not a product-transfer test. Cross-domain generalisation is measured separately on the MMAD DS-MVTec and VisA subsets (Chapter 6).

Are there errors, sources of noise, or redundancies? Two kinds are known and intentional. (i) The reasoning traces are model-generated by Gemini 2.5-Flash and, although filtered (see Collection Process), are not human-verified line-by-line. Residual hallucination or over-confident phrasing is possible. (ii) The verdict labels can disagree with the Real-IAD acquisition folder: a Real-IAD NG (No-Good) capture is labelled *normal* in this dataset when the defect is not visible at the `_C1_` camera angle (no appearing anomaly / no usable mask). The folder name is thus not the label. This is the explicit reason the composition is counted by true `<answer>` verdict rather than by NG/OK folder.

Does the dataset contain sensitive, personal, or offensive data? No. All images depict manufactured industrial parts. There are no human subjects, faces, or personally identifying information.

I.3. Collection Process

How was the data associated with each instance acquired, and how were the traces generated? Source images come from the public Real-IAD dataset (acquired by its original authors with calibrated multi-view rigs, and we did not re-photograph anything). For each retained sample we built a prompt and called Gemini 2.5-Flash at temperature 1.0, using the 489-line inspector system prompt reproduced in Appendix B.1 (the six-phase reasoning template, the two controlled vocabularies, the six auto-reject rules, and a silent self-scoring acceptance filter). Samples were processed in batches of 5 to 10 to amortise API latency.

What was the input asymmetry at generation time? This is a deliberate *privileged-input asymmetry* and is documented honestly. At *generation* time only: each *anomalous* sample was shown three images: (1) the original, (2) the same image with the defect region overlaid in red from the Real-IAD binary mask, and (3) a normal reference image of the same product and camera view, whereas each *normal* sample was shown only the single original image. The overlay is the only privileged channel: it tells the generator *where* the defect is, so the teacher can focus on producing a plausible six-phase explanation rather than first having to localise the defect. The overlay and reference are then discarded. The student model at training and inference time sees only the single test image and never the mask. A consequence of this asymmetry is that anomalous and normal traces are not generated under identical evidence conditions.

What mechanisms or procedures were used to collect/filter the data, and how were they validated? Quality control is performed *within* the generation prompt rather than by a second reviewer model. The generator is instructed to score each trace silently against an eight-category rubric (0 to 2 each, 16 maximum) and to return it only if (a) no auto-reject rule fires and (b) the total is at least 13/16, with specified categories (visual grounding, evidence-before-conclusion, single-image independence, output-structure correctness, and decision quality) required to score the maximum, and otherwise to rewrite it before returning. These self-scores are not returned or logged, so acceptance is a prompted behaviour rather than a verified gate. The only enforced check is the parse-level validation below. The six auto-reject rules cover batch/sequence leakage, metadata leakage, naming the defect before describing its evidence, wrong output structure, unsupported defect claims, and overstated causality. Malformed or schema-violating JSON triggers regeneration. No separate human or second-model verification stage was run.

Who was involved and over what timeframe? Generation was performed programmatically by the thesis author via the Gemini API as part of the thesis work. No crowdworkers were involved. The Real-IAD source images predate this work.

Were ethical review processes conducted? No formal IRB review was required, as the data contains no human subjects. The use of Real-IAD follows that dataset's terms.

I.4. Preprocessing / Cleaning / Labeling

Was any preprocessing/cleaning/labeling done? Yes. (i) *Camera-angle filtering*: only top-down `_C1_` images were retained. (ii) *Mask-presence filtering*: anomalous (NG) candidates without a defect mask at the `_C1_` angle have no visible defect from that view and were relabelled *normal* (so verdict labels are recomputed from visible evidence, not inherited from the folder). (iii) *Vocabulary normalisation*: Real-IAD two-letter defect codes were translated to the 26-type controlled vocabulary, and locations were quantised to the 3×3 grid, enabling zero-shot transfer of defect terminology to MVTec AD/VisA at evaluation. (iv) *Schema validation and auto-reject*: a batch whose response was not valid JSON was regenerated. The auto-reject rules are applied by the generator itself inside the call (§3.4), which rewrites a failing trace before returning it, and no second parser checks them afterwards. (v) *Class re-balancing*: each split was stratified to $\approx 50/50$ by true verdict. (vi) *Overlap prevention*: image basenames were excluded across splits.

Was the “raw” data saved? The source Real-IAD images, masks and overlays remain available from Real-IAD. Within this project the privileged overlay/reference inputs are not retained inside the released instances (they were used only to condition the teacher).

Is the preprocessing software available? The generation prompt is disclosed verbatim (Appendix B.1) and the split logic, vocabulary mapping, and quality controls are described in Chapter 3. This is sufficient to reconstruct the pipeline given Real-IAD and a comparable generator.

I.5. Uses

What tasks has the dataset been used for, and what could it be used for? It has been used (i) as the SFT cold-start dataset for Qwen2.5-VL industrial anomaly detection, (ii) as the preference-balanced training pool for GRPO post-training, and (iii) as an in-domain held-out evaluation set. Beyond this thesis it could support research on structured-output chain-of-thought distillation, reasoning-faithfulness studies, and reward-design ablations for verifiable multimodal RL.

What should users be aware of that might affect future uses? Three points. (i) *Privileged-input asymmetry*: the teacher saw masks/overlays for anomalies but not for normals. Downstream users should not interpret the traces as unaided human reasoning, and analyses of reasoning faithfulness must account for this. (ii) *Product overlap*: the held-out split is sample-disjoint but *product-overlapping*, so reporting it as a product-transfer or fully out-of-distribution benchmark would overstate generalisation. Cross-domain claims should use DS-MVTec/VisA. (iii) *Teacher-model bias*: traces inherit any stylistic or systematic biases of Gemini 2.5-Flash, and the silent acceptance filter is self-applied rather than independently verified.

Are there tasks for which the dataset should not be used? It should not be used as a clean human-annotated reasoning benchmark, nor as evidence of human-level inspection rationale, given the model-generated and privileged-input nature of the traces. It should not be used to claim product-disjoint generalisation via its own held-out split. It is restricted to top-down _C1_ views and should not be assumed representative of other viewpoints.

I.6. Distribution

Will the dataset be distributed, and how? Yes. **AnomalyThink** is openly released alongside the thesis as the first openly-available Real-IAD-based reasoning-trace dataset structured for direct GRPO consumption. It is distributed as JSON instances (image path, reasoning trace, structured XML tags) with the generation prompt and split logic documented.

How does it relate to existing released datasets? It *complements* Anomaly-Instruct-125K [75] (open, conversation-format, MVTec/VisA-sourced industrial subset) by providing Real-IAD-based, tag-structured traces, and it *contrasts* with the closed Expert-AD [39], which is not publicly distributed.

Under what license, and with what IP/terms? The traces are released under the Apache-2.0 licence. The dataset contains no images, only Real-IAD image identifiers, and the images themselves remain under Real-IAD's own licence, which users must obtain and comply with separately. The added value of **AnomalyThink** (the generated reasoning traces, vocabulary normalisation and split metadata) is released for research use.

Are there export controls or regulatory restrictions? None known. The data depicts generic manufactured parts with no sensitive content.

I.7. Maintenance

Who maintains the dataset and how can they be contacted? The dataset is maintained by the thesis author. Contact is via the thesis repository/release channel accompanying this work.

Will the dataset be updated, and how will updates be communicated? Updates (e.g. corrected traces, additional camera angles, or extension to Anomaly-Instruct-125K's medical/in-the-wild splits behind the same six-phase template) are anticipated as future work (Chapter 8) and would be versioned in the public release. Consumers should pin a version for reproducibility.

Will older versions continue to be supported? The version used in this thesis (the splits of Table I.1) is the reference version and is preserved as-is to keep the reported results reproducible.

Can others extend/build on the dataset, and is there a mechanism to contribute? Yes. Because the full generation prompt, controlled vocabularies, quality-control rubric and split logic are disclosed (Appendix B.1, Chapter 3), others can regenerate or extend the dataset to additional products or camera angles. Contributions can be proposed through the public release channel.



Evaluation Harness and Prompt-Mode Robustness

This appendix documents the evaluation harness used throughout the thesis: the five prompt modes it can run, their decoding settings, and the fixed evaluation sets. It then quantifies how sensitive a trained checkpoint is to the choice of prompt mode, using the released IAD-R1 checkpoint as a probe. A converged checkpoint is essentially prompt-robust. That conclusion licenses reporting a single train-prompt number for every row of the state-of-the-art (SOTA) table in §6.12, including the released IAD-R1 checkpoint, which is not disadvantaged by being scored under our prompt rather than its own.

J.1. Prompt Modes

A single test sample (one image and the product name) can be wrapped into a model query in several ways. The harness exposes five prompt modes, each recorded by a distinct suffix in the evaluation-result filename so that no two runs can be silently confused.

train-prompt (suffix `_trainprompt`). The SFT training format re-used at evaluation time: a minimal one-line system message (“Please answer by yes or no”) followed by the product-conditioned user turn that the model saw during supervised fine-tuning. This is the headline mode for the SFT and Arm-C results of Chapter 6. The exact text is reproduced in Appendix B.2.

GRPO-prompt (suffix `_grprompt`). The single fixed GRPO question (“Are there any defects in the query image?”, with the expert preamble), used with no system message. This matches the prompt the policy was optimised against during GRPO and is reported only where explicitly marked. The exact text is in Appendix B.3.

bare-prompt (suffix `_bareprompt`). A product-agnostic baseline phrasing that drops the per-sample product name, used to test whether a checkpoint leans on the product-conditioned wording.

yes-no-user (suffix `_yesnouser`). The train-prompt wording exactly, with the one-line instruction “Please answer by yes or no” carried in the user turn instead of the system turn. It exists for backbones that do not attend to the system slot, LLaVA-OneVision in particular, whose base checkpoint emits no parseable verdict under the system-turn form (§6.9). The wording is identical, so a score under this mode is comparable to a train-prompt score.

IAD-R1-native (suffix `_iadrlnative`). The released IAD-R1 checkpoint’s own native prompt, included so that the released baseline is evaluated under the exact conditions its authors intended, rather than being penalised for a foreign prompt.

In every mode the verdict is read from the `<answer>` tag, with a fallback that reads a bare yes or no from a tag-less response. The fallback matches substrings, so a tag-less free-text answer that contains “no” inside another word (“anomaly”) is read as no. This is what the LLaVA-OneVision base row under the system-turn prompt shows (§6.2). A response that yields no verdict either way is counted as wrong.

The bare-prompt and IAD-R1-native modes exist primarily for the robustness check in §J.4: they let a single checkpoint be re-scored under three different wordings on the same images, isolating prompt sensitivity from model quality.

J.2. Decoding and Image Settings

All evaluation is deterministic: decoding is **greedy** (no sampling), so a checkpoint produces the same answer on a given image every time and re-evaluating a checkpoint reproduces its numbers exactly. This says nothing about the variance across training runs or across test samples, which §7.7 discusses. Every checkpoint, supervised or GRPO, is evaluated with the same image cap and token budget. What differs between the two regimes is the prompt.

Table J.1: Evaluation settings of the harness, by regime. Decoding and image settings are identical, only the prompt differs.

Setting	SFT eval	GRPO eval
Decoding	greedy	greedy
Max new tokens	1024	1024
image_max_pixels	262,144	262,144
Prompt	train-prompt (system turn)	GRPO-prompt (no system turn)

The generation budget (1024 new tokens, image_max_pixels = 262,144) matches the training resolution cap used during supervised fine-tuning (Appendix B.2) and is applied to every checkpoint, including the GRPO ones, whose training used a larger image cap and a shorter completion limit (Appendix A.4). The GRPO checkpoints are scored under the GRPO training prompt. The headline SFT+GRPO checkpoint (ckpt-530) scores 82.73% and 70.39% under that prompt and 80.74% and 70.50% under the supervised train-prompt (files eval_*_full_trainprompt_default.json), so the prompt moves the DS-MVTec number by two points and the VisA number by none. Under the vLLM path with the train-prompt it scores 81.83% and 70.79%. One decoding detail separates the two generation paths of §6.9: the HuggingFace path applies the repetition penalty of 1.05 carried in the Qwen2.5-VL release’s generation config, whereas the vLLM path decodes with none, which is the likely reason the two paths agree less on the Qwen2.5-VL checkpoints than on LLaVA-OneVision. Because decoding is greedy, all reported balanced-accuracy numbers are point estimates with no sampling noise.

J.3. Fixed Evaluation Sets and Headline Metric

Every model in the thesis is scored on the same three fixed evaluation sets, so that all numbers are directly comparable.

Table J.2: The three fixed evaluation sets. DS-MVTec and VisA are the cross-domain MMAD sub-benchmarks. The held-out Real-IAD split is an in-domain out-of-sample check (Appendix L).

Evaluation set	Role	Samples
DS-MVTec (MMAD)	Cross-domain	1,670
VisA (MMAD)	Cross-domain	2,141
Held-out Real-IAD	In-domain out-of-sample	4,236

The **headline metric** is balanced accuracy (BA), which corrects for the class imbalance that arises within individual products. From the confusion-matrix counts { tp , tn , fp , fn } stored in each result file,

$$BA = \frac{1}{2} \left(\frac{tp}{tp + fn} + \frac{tn}{tn + fp} \right),$$

i.e. the unweighted mean of the anomaly-class recall (sensitivity) and the normal-class recall (specificity). Accuracy and F_1 are also recorded but are secondary.

J.4. Prompt-Mode Robustness

To check whether a converged checkpoint’s headline number depends on which prompt mode is used, the released IAD-R1 checkpoint was re-scored on the two cross-domain sets under three prompt modes (train-prompt, bare-prompt, IAD-R1-native), holding the images and decoding fixed. Each BA in Table J.3 was re-computed with strict scoring from the per-sample results of the corresponding result file, balanced accuracy being the mean of sensitivity and specificity over all samples. The spread is the largest minus the smallest BA across the three modes.

Table J.3: Balanced accuracy (%) of the released IAD-R1 checkpoint across three prompt modes, computed with strict scoring from the per-sample results in the `iad_r1_qwen_recanon` result files, so the one IAD-R1-native DS-MVTec generation that yields no verdict counts as wrong. “Spread” is max – min across the three modes.

Set	train-prompt	bare-prompt	IAD-R1-native	Spread (pp)
DS-MVTec	81.92	81.45	81.09	0.83
VisA	71.34	71.56	71.87	0.53

The underlying confusion-matrix counts behind each cell are listed in Table J.4 for full reproducibility.

Table J.4: Confusion-matrix counts behind Table J.3, read directly from the result files. N is the number of scored samples.

Set	Mode	N	TP	TN	FP	FN
DS-MVTec	train-prompt	1,670	965	378	66	261
DS-MVTec	bare-prompt	1,670	934	385	59	292
DS-MVTec	IAD-R1-native	1,670	964	371	72	262 (+1 unparsed)
VisA	train-prompt	2,141	606	869	75	591
VisA	bare-prompt	2,141	591	885	59	606
VisA	IAD-R1-native	2,141	644	849	95	553

Conclusion. Across the three prompt modes the released IAD-R1 checkpoint varies by 0.83 pp on DS-MVTec and 0.53 pp on VisA, in both cases below the 1.08-point difference between two same-seed GRPO launches (§7.7). For that checkpoint the prompt does not decide the comparison, which is what licenses scoring it under the train prompt in the SOTA table of §6.12. This is a statement about the released IAD-R1 checkpoint, not a general one: our own SFT+GRPO checkpoint moves by 1.99 pp on DS-MVTec between the GRPO prompt and the supervised prompt (§6.1), which is why that row is marked in the table. The released IAD-R1 checkpoint is not disadvantaged by being scored under our prompt rather than its own, because the prompt mode contributes less than a percentage point to BA.

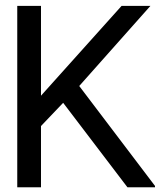
J.5. Type-Label Transfer to MMAD

The `<type>` tag is scored in §6.11 on the 137 shared images. This section scores it on the full benchmarks with the same rule, for the six checkpoints of Table 6.14. A prediction scores 1 when it matches the MMAD label or contains it, and otherwise the cosine similarity of the two strings under `nomic-embed-text-v1.5`. The MMAD label comes from the benchmark’s own defect-type question, available for 94.8% of DS-MVTec and 99.3% of VisA anomalies, with the folder name as fallback. Only each model’s detected anomalies are scored.

Table J.5: Mean type similarity to the MMAD label and share of exact or substring matches, among each model’s detected anomalies on the full DS-MVTec and VisA sets. [†] The DS-MVTec cells on this backbone carry the contamination caveat of §6.1.

Stage	DS-MVTec		VisA	
	mean sim.	match	mean sim.	match
Qwen2.5-VL-7B				
SFT	0.56	9.4 %	0.54	8.3 %
SFT+GRPO	0.56	10.5 %	0.53	4.1 %
KCR	0.55	8.4 %	0.54	5.9 %
LLaVA-OneVision-7B [†]				
SFT	0.58	12.8 %	0.57	12.8 %
SFT+GRPO	0.55	10.3 %	0.55	6.5 %
KCR	0.56	10.2 %	0.55	7.5 %

Unlike localisation (Table 6.14), no stage learns the label. Mean similarity sits between 0.53 and 0.58 for every checkpoint and the match rate below 13 %, and the reinforcement-learning stage lowers the VisA match rate on both backbones. The cause is a vocabulary mismatch rather than a perception failure. The models name the eight Real-IAD classes they were trained on, 18 distinct strings for the Qwen2.5-VL KCR checkpoint, dominated by missing parts, contamination, pit and scratch. MMAD labels defects in product-specific terms it never showed them, 435 distinct on DS-MVTec and 218 on VisA, and its most frequent VisA label, discoloration, is not a Real-IAD class at all. The most common confusions are near-synonyms across the two vocabularies, which the rule scores as misses: discoloration called contamination (78 times on VisA), discoloration called pit (43), chip called missing parts (27), and crack called missing parts (19 on DS-MVTec). Where the vocabularies overlap the models are right, and scratch against scratch is the most frequent pair on DS-MVTec. The type tag therefore transfers as a defect family and not as a label, which is the zero-shot transfer gap that §3.2 anticipated when it chose a generic vocabulary over the Real-IAD codes. A benchmark-specific label set, or a mapping from the generic vocabulary to each benchmark’s taxonomy, would be needed before the type tag could be scored as a classification.



Explainability Scoring: Executed Protocol and Proposed Extension

This appendix has two parts. §K.1 records the explanation-quality protocol that was executed for §6.11, so that its numbers can be reproduced. The sections after it describe a proposed extension to a six-axis rubric with reference-grounded binary axes and a human-validation plan. That extension has *not* been run, and every explanation number in this thesis comes from the executed protocol of §K.1 only. The thresholds, weights and agreement bands of the extension are design proposals, not measured values.

K.1. The Executed Protocol

Table 6.15 and Figure 6.11 score traces with a Gemini-3-Flash judge (gemini-3-flash-preview, Vertex AI, thinking budget 512, JSON output). The judge receives the original image, an overlay with the ground-truth mask in red, the ground-truth defect category as context, and the model's reasoning trace, and scores five axes from 0 to 2: visual grounding, defect faithfulness, evidence before conclusion, coherence and conciseness. The overall score is their sum out of 10. Each trace is scored three times and the median of the three overall scores is kept. The image set is the shared true-positive set of §6.11: 90 DS-MVTec anomalies, six per product, and 47 VisA anomalies, four per product where available, drawn with a fixed seed from the images that every compared model detects. The <location> and <type> scores in the same table are computed mechanically, not by the judge. The script is explainability_judge_shared.py. The raw outputs and the image list are in the repository folder results/explainability_shared/. The judge model is a newer generation than the Gemini 2.5-Flash trace teacher, but the same model family corrected and rewrote the KCR corpus (§5.8), which is the dependence noted in §6.11.

The system instruction given to the judge, verbatim except that the JSON schema line is wrapped for print:

```
You are an expert evaluator of INDUSTRIAL-DEFECT INSPECTION EXPLANATIONS. A vision-language model inspected a product and wrote a reasoning trace ending in a defect verdict. Your job is to score the QUALITY OF ITS REASONING/EXPLANATION - how well the written reasoning is grounded in and faithful to the actual visual evidence - NOT whether the final yes/no is right (it is already known to be correct here), and NOT the exact <type>/<location> wording (scored separately).
```

```
You are given: (1) the ORIGINAL product image; (2) an OVERLAY with the TRUE defect region in RED; (3) the ground-truth defect category (context); (4) the model's reasoning trace. Judge the trace against WHAT IS ACTUALLY VISIBLE - never reward fluent, confident, or long writing that is not supported by the image. A convincing description of a defect that is NOT in the red region must score low on faithfulness.
```

```
Score each axis 0, 1, or 2 (2=fully, 1=partial, 0=no), writing a one-line justification FIRST:
```

- A. VISUAL GROUNDING - claims point to specific, actually-visible regions/cues (vs generic statements that could fit any image).
- B. DEFECT FAITHFULNESS - the anomaly the trace describes corresponds to the REAL defect shown in the red region (same kind of flaw, same area); it does not hallucinate, invent, or describe a different/absent defect.
- C. EVIDENCE-BEFORE-CONCLUSION - it describes the visual cues first and then concludes, rather than asserting the defect up front and back-filling.
- D. COHERENCE - the steps are internally consistent and logically support the conclusion (no contradictions, no non-sequiturs).
- E. CONCISENESS / NO-FILLER - compact and on-point; no padded, repetitive, or template scan narration. (Length is NEVER a positive signal.)

```
Return STRICT JSON only:
```

```
{"justification":"<=2 sentences citing the image",
```

```
"visual_grounding":0-2, "defect_faithfulness":0-2,
"evidence_before_conclusion":0-2, "coherence":0-2,
"conciseness":0-2, "overall":0-10}
where overall is the sum of the five axes. Judge on the image, not on eloquence.
```

K.2. Proposed Extension

The protocol is grounded in recent best practice for LLM/VLM-as-judge explainability evaluation. The decomposed-rubric design (analytic rather than holistic criteria, binary MET/UNMET for objective axes, narrow ordinal scales with behavioural anchors, a clamped normalised weighted sum, and ensemble judging with per-criterion reliability tracked via Cohen's kappa) follows the rubric-evaluation synthesis of Autorubric [56]. The separation of *faithfulness* from *plausibility*, the warning that VL judges reward surface-level alignment and can be fooled by fluent post-hoc rationalisation, and the use of a human reader study with Kendall/Spearman correlation inside the inter-annotator range as the validation target follow the medical-VLM faithfulness study [46].

K.3. Scope and Design Principles

The protocol scores the explanation, not the verdict. It applies to the *anomalous* test samples of the cross-domain evaluation sets, because those are the samples that carry a ground-truth defect mask and a ground-truth defect type against which the explanation can be checked. In the thesis evaluation sets this is the DS-MVTec full split (1,670 samples, of which 1,226 are anomalous and carry a GT mask+type, 444 normal) and the VisA full split (2,141 samples, of which 1,197 are anomalous). The in-domain held-out Real-IAD split adds a further 2,129 anomalous samples. Normal samples carry no mask, so for them the localisation axis is omitted and the rubric scores only grounding, restraint, ordering, and conciseness.

Five design principles, each traceable to the literature above, govern the rubric:

1. **Reference-guided, pointwise.** The judge scores one trace at a time against an injected reference (GT mask region, GT type, and the test image). It does not rank or compare traces head-to-head, which avoids position bias and keeps each score interpretable [56].
2. **Decomposed (analytic), not holistic.** Quality is split into independent axes scored separately. This prevents the halo effect (a fluent trace inflating an unrelated axis), makes each axis individually reliability-checkable, and exposes *which* dimension a model fails.
3. **Faithfulness over plausibility, do not reward length.** A long, fluent, plausible-sounding rationalisation that is not anchored in the actual defect must not outscore a short, correct one. Conciseness is an explicit scored axis and length is never a positive signal [46].
4. **Inject perception as a numeric fact.** VL judges are weak perceivers and unreliable at reading masks or measuring overlap from pixels. The localisation axis is therefore driven by a *pre-computed* Intersection-over-Union (IoU) between the model's claimed region and the GT mask, computed outside the judge and passed into the prompt as a number. The judge only applies the rubric band. It never estimates overlap itself.
5. **Sample and aggregate.** The judge is stochastic, so each trace is scored ≥ 3 times and the per-axis *median* is taken, with inter-sample spread tracked as a self-consistency diagnostic.

K.4. Decomposed Multi-Axis Rubric

The rubric has six axes. Objective axes (those checkable against the injected reference or a deterministic rule) are scored binary MET/UNMET, because binary criteria yield the highest inter-rater reliability. Subjective-quality axes use a narrow ordinal 0 to 3 scale with anchored descriptions at the extremes. Intermediate levels interpolate. Narrow scales with behavioural anchors are preferred over broad (0 to 10) scales, which suffer central-tendency bias [56].

Table K.1: Proposed explanation-quality rubric. Objective axes use binary MET/UNMET. Subjective axes use anchored ordinal 0 to 3. “Reference fact” is the information injected into the judge prompt for that axis. The localisation axis consumes a *pre-computed* IoU. The judge never measures overlap itself.

Axis	Type	Reference fact injected	Extreme anchors
Visual Grounding	0 to 3	test image	0: claims generic, fit any image. 3: every claim tied to a named visible region/cue
Localisation Faithfulness	MET/ UNMET	pre-computed IoU(claimed region, GT mask), GT region	MET iff $\text{IoU} \geq \tau_{\text{IoU}}$ and the trace’s <code><location></code> words name that region, else UNMET
Type Correctness	MET/ UNMET	GT defect type	MET iff <code><type></code> is the GT type or an accepted synonym in the controlled vocabulary, else UNMET
Evidence-before-Conclusion	0 to 3	none (structural)	0: names the defect before any visual evidence. 3: regions and cues described first, verdict follows
Causal Restraint	0 to 3	none (structural)	0: asserts an unseen manufacturing cause as fact. 3: any cause is tentative and secondary to visible evidence
Conciseness / No-filler	0 to 3	token count (diagnostic only)	0: padded, repetitive, generic scan narration. 3: compact, no filler, every sentence load-bearing

Two axes deserve emphasis. **Localisation Faithfulness** is the axis most prone to judge error if left to the model, which is exactly why its decision is reduced to a threshold on an externally computed IoU. The judge is told, e.g., $\text{IoU}=0.62$, and applies the band τ_{IoU} (proposed default 0.30), a lenient grounding threshold chosen because the model emits a coarse 3×3 -grid `<location>` rather than a pixel mask. The mask-vs-region overlap can also be coarsened to the same 3×3 region grid the model uses, so that MET means “named the correct grid cell”. **Conciseness** receives the token count only as a *diagnostic* input. The anchor is qualitative (filler vs. load-bearing) so that length itself is never rewarded and a short faithful trace can score 3.

K.5. Separating Faithfulness from Plausibility

A trace can be *plausible* (reads like a competent inspection) while being *unfaithful* (the stated evidence is not what actually drove, or does not match, the verdict). Rewarding plausibility alone reproduces the failure documented for medical VLMs, where proprietary models scored higher on fluency yet still fabricated or misattributed findings [46]. The rubric enforces the separation structurally:

- Plausibility-correlated signals (fluency, length, confident tone) are *never* positive axes. Conciseness penalises filler. Length is at most neutral.
- Faithfulness is checked against external references the judge cannot talk its way around: the injected GT type (Type Correctness) and the injected IoU (Localisation Faithfulness). A confident, well-written trace that points to the wrong region scores UNMET on localisation regardless of prose quality.
- The Evidence-before-Conclusion and Causal Restraint axes target the two most common rationalisation patterns (naming the defect first and then back-filling, and asserting an unseen cause as fact), both of which a purely plausibility-driven judge would miss.

K.6. Aggregation

Per axis, the judge is sampled $J \geq 3$ times and the *median* score is taken (ordinal axes: median of $\{0, 1, 2, 3\}$. Binary axes: majority vote of MET/UNMET, ties broken to UNMET). Median over the judge samples is robust to the occasional outlier sample and exposes judge self-consistency as the inter-sample range.

Binary axes are mapped $\text{MET} \rightarrow 1$, $\text{UNMET} \rightarrow 0$. Ordinal axes are normalised to $[0, 1]$ by dividing by 3. The

trace score is a clamped normalised weighted sum, following the Autorubric aggregation [56]:

$$S = \max\left(0, \min\left(1, \frac{\sum_i w_i v_i}{\sum_{i: w_i > 0} w_i}\right)\right), \quad (\text{K.1})$$

where $v_i \in [0, 1]$ is the normalised median score of axis i and w_i its weight. Negative weights (penalty axes) are excluded from the denominator so a perfect trace scores exactly 1, and the outer clamp prevents any penalty from driving S below 0. Proposed default weights put the two externally-verified faithfulness axes first: Localisation Faithfulness and Type Correctness $w=1.0$ each, Visual Grounding and Evidence-before-Conclusion $w=0.75$ each, Causal Restraint $w=0.5$, Conciseness $w=0.25$. For normal samples the localisation axis is dropped from both numerator and denominator. These weights are tunable and should be frozen *before* the validation study (§K.9) to avoid fitting them to the human golden set.

This explanation-quality score S is reported *alongside*, never blended into, the decision metrics of Chapter 6. It is also kept distinct from the GRPO training reward described in Chapter 5, an unweighted sum of the accuracy and consistency reward functions (max 3.0, with sub-signals format/verdict each max 1 and type and location each max $\frac{1}{2}$). The GRPO reward is an automatic *training* signal, whereas S is an offline *evaluation* judgment of explanation quality.

K.7. Judge Sampling

The judge is google/gemini-3-flash-preview (Gemini-3-Flash). Each (trace, axis-set) is scored $J = 3$ times at temperature ≈ 0.4 with a fixed master seed offset per sample so reruns are reproducible. The judge returns strict JSON. Non-parseable returns are re-requested up to a fixed retry budget and otherwise dropped from that sample's median. Per-axis inter-sample variance is logged. An axis whose three samples disagree across the full range is flagged as low-confidence for that trace. Using Gemini-3-Flash as the judge while Qwen2.5-VL is the evaluated model keeps the judge architecturally distinct from the policy, reducing self-preference bias. (Note that the AnomalyThink *traces* were generated by Gemini 2.5-Flash, a different and earlier model than the proposed Gemini-3-Flash judge. The judge must not be told the traces share a model family.)

K.8. Example Judge Prompt

The block below is a compact judge prompt the thesis could use with Gemini-3-Flash. The test image is attached as an image part. The GT region, GT type, and the pre-computed IoU are injected as text facts. ASCII only.

```
SYSTEM:
You are a strict industrial-inspection EXPLANATION judge. You score the
QUALITY of a model's reasoning trace, NOT whether its yes/no verdict is
correct. Judge faithfulness, not fluency. Never reward length. A long,
confident, well-written trace that points at the WRONG region or names
the WRONG defect type must score badly. Use ONLY the facts given to you;
do NOT estimate region overlap yourself -- an IoU number is provided.

USER:
[ATTACHED IMAGE: the test product image]

GROUND-TRUTH FACTS (trust these; do not re-derive):
gt_defect_type : Scratch
gt_region_3x3 : middle-right
precomputed_iou : 0.62 # IoU(model_claimed_region, gt_mask)
iou_met_threshold: 0.30 # MET iff precomputed_iou >= this value

MODEL TRACE TO JUDGE (verbatim, between fences):
<<<TRACE
<think> ... model reasoning ... </think>
<location>middle-right</location>
<type>Scratch</type>
<answer>Yes</answer>
TRACE>>>

Score each axis. Objective axes are MET or UNMET. Subjective axes are an
integer 0..3 (0 = worst anchor, 3 = best anchor). Anchors:
visual_grounding 0..3:
  0 = generic claims that fit any image; 3 = every claim tied to a
    named visible region/cue in THIS image.
localisation_faithfulness MET/UNMET:
  MET iff precomputed_iou >= iou_met_threshold AND the <location>
    words name that region; else UNMET. (Use the IoU number only.)
```

```

type_correctness MET/UNMET:
  MET iff <type> equals gt_defect_type or an accepted synonym; else
    UNMET.
evidence_before_conclusion 0..3:
  0 = defect named before any visual evidence; 3 = regions and visible
    cues described first, verdict follows from them.
causal_restraint 0..3:
  0 = asserts an unseen manufacturing cause as fact; 3 = any cause is
    tentative and secondary to visible evidence.
conciseness_no_filler 0..3:
  0 = padded, repetitive, generic scan narration; 3 = compact, every
    sentence load-bearing. Length is NEVER a positive signal.

Return STRICT JSON only, no prose, no markdown fences:
{
  "visual_grounding": 0,
  "localisation_faithfulness": "UNMET",
  "type_correctness": "UNMET",
  "evidence_before_conclusion": 0,
  "causal_restraint": 0,
  "conciseness_no_filler": 0,
  "one_line_reason_per_axis": {
    "visual_grounding": "string",
    "localisation_faithfulness": "string",
    "type_correctness": "string",
    "evidence_before_conclusion": "string",
    "causal_restraint": "string",
    "conciseness_no_filler": "string"
  }
}

```

K.9. Validation Against a Human Golden Set

An automatic judge is only trustworthy once it has been validated against humans on the same task [46]. The proposed validation builds a *golden set* of ≈ 150 to 200 anomalous traces stratified across DS-MVTec and VisA products and across the four trained variants of Chapter 6 (so the set spans good and bad explanations). Two or more annotators familiar with industrial inspection score each trace on the rubric of Table K.1. Their per-axis agreement establishes the human ceiling.

- **Judge-human agreement.** Per binary axis, report Cohen's κ between the Gemini-3-Flash median and the human consensus. Per ordinal axis and for the aggregate score S , report Spearman ρ . The judge is considered usable when agreement lands in the *moderate-to-substantial* band, a proposed target of $\kappa, \rho \in [0.5, 0.75]$, and, critically, when judge-human agreement is no worse than human-human agreement on the same axis. (Values below 0.5 mean the axis anchors are ambiguous and must be rewritten. Values approaching 1.0 on a subjective axis would be suspicious and suggest the axis is trivially binary.)
- **Synthetic-degradation probes.** Programmatically corrupt known-good traces and verify the judge's score drops on the right axis: swap <location> to a wrong grid cell (must break Localisation Faithfulness), swap <type> to a wrong vocabulary entry (must break Type Correctness), reorder so the defect is named first (must lower Evidence-before-Conclusion), insert filler sentences (must lower Conciseness, and must *not* raise any other axis), and inject an asserted unseen cause (must lower Causal Restraint). A judge that does not respond monotonically to these single-axis edits is not yet calibrated.
- **Adversarial probes.** Stress-test for the plausibility trap: feed fluent, confident, verbose traces that are factually wrong (wrong region, fabricated cue) and confirm they do *not* outscore short correct traces. Feed a trace whose verdict is right but whose <think> block is unrelated boilerplate and confirm grounding scores low. Passing these probes is the operational definition of “faithfulness over plausibility” for this protocol.

Only after the judge clears the agreement target *and* both probe suites should the aggregate score S be reported as an explanation-quality metric in any follow-up study. Until then, the protocol of this appendix remains a design, not a result.



Held-Out Real-IAD Evaluation

A persistent concern in industrial deployment is whether models trained on Real-IAD “leak” through the trace generator into the test distribution. We address this with a *sample-level* held-out split: each model is evaluated on the same 4,236-image subset of Real-IAD whose individual image IDs entered no SFT or GRPO training set. Traces exist for these images, but no model in this table was trained on them. The 23 product categories present in this evaluation are also present in the training dataset, so the test measures generalisation to unseen images and unseen production batches of *seen* product categories, not generalisation to fully unseen product categories. This held-out split therefore probes in-domain generalisation to unseen images rather than cross-domain transfer, which is measured separately on DS-MVTec and VisA. A product-disjoint held-out split is a stronger experiment that we leave for future work (§8.4).

Table L.1: Balanced accuracy on the 4,236-image held-out Real-IAD split, strict scoring. The three Qwen2.5-VL rows are HF-path evaluations, the LLaVA-OneVision row is a vLLM-path evaluation (§6.9). IAD-R1 is the released checkpoint under its own prompt.

Model	BA	Acc.	Prec.	Recall	TNR	F1
Qwen2.5-VL-7B SFT+GRPO (ckpt-530)	80.87	80.81	91.28	68.34	93.40	78.16
Qwen2.5-VL-7B KCR (ckpt-376)	79.32	79.25	90.12	65.95	92.69	76.16
LLaVA-OneVision-7B KCR (ckpt-376, corrected corpus)	84.03	83.97	93.21	73.46	94.59	82.16
IAD-R1 released checkpoint	79.58	79.53	86.69	70.03	89.13	77.47

The two Qwen2.5-VL rows sit within 2 to 3.5 pp of their cross-domain DS-MVTec numbers (82.73% and 82.80%), so neither model shows a sign of overfitting to the Real-IAD trace-generation distribution it was trained on. Both show the same profile, a TNR above 92% with recall in the high sixties: the system rarely flags a normal held-out image, which is the false-positive concern that motivated balanced accuracy in the first place. Two further observations. First, the Qwen2.5-VL ranking on held-out Real-IAD is the reverse of the MMAD ranking: SFT+GRPO is 1.55 pp above KCR here, while KCR is ahead on DS-MVTec and VisA. The corrective corpus is therefore not a uniform improvement, it trades a little in-domain recall for cross-domain gains. Second, the LLaVA-OneVision KCR checkpoint of §6.9 is the strongest row on this split by more than 3 pp, with the best recall and the best TNR at once, and it is also 4.45 pp above the released IAD-R1 checkpoint scored under its own prompt. Because this is a sample-level held-out, every row speaks to robustness against the natural production-line variation captured by different acquisition batches of the same product family at the same top-down _C1_ viewpoint, not against fully new product categories, nor against unseen viewpoints.



The Corrected LLaVA-OneVision Corpus

§6.9 reports a replication of the Keep-Correct-Revise loop on LLaVA-OneVision-7B-SI and flags that the corpus behind its winning row was not built the way the Qwen2.5-VL KCR corpus was. This appendix documents the difference, because it changes what the replication is a replication of, and reports the rerun on a corpus that matches.

M.1. What Differed

The Qwen2.5-VL KCR corpus is drawn from the 6,000-image SFT split and balanced on the verdict the trace itself teaches. The first LLaVA-OneVision corpus was drawn from the whole 10,236-image rollout pool and balanced on whether the file path contains an anomalous folder name. Two consequences follow, and both were measured directly from the two corpus files.

Table M.1: Composition of the two LLaVA-OneVision KCR corpora against the Qwen2.5-VL one, measured from the training files. “Inside the SFT split” counts images shared with the 6,000-image supervised split, matched on the path from `Real-1AD/` onwards. The anomalous share is measured from the `<answer>` tag of each stored trace.

Corpus	Items	Inside the SFT split	Outside it	Anomalous share
Qwen2.5-VL KCR	6,000	6,000	0	50.0 %
LLaVA-OneVision KCR, first build	6,000	3,516	2,484	45.0 %
LLaVA-OneVision KCR, corrected build	6,000	6,000	0	50.0 %

The first consequence is that 2,484 of the 6,000 images in the first build come from the GRPO split rather than the SFT split. The comparison in §6.9 therefore varied the image set as well as the trace author, which is one factor more than the design intends. The second is that balancing on the folder name is not the same as balancing on the label. Anomalous-folder captures without a defect mask carry a correct negative trace, and counting those as positives leaves the first build at 45.0 % anomalous rather than the intended even split.

M.2. The Corrected Build

The corrected corpus applies exactly two changes and keeps everything else, including the keep, correct and rewrite precedence, the record schema, the prompt template, the caps and the random seed. The candidate pool is restricted to the 6,000-image SFT split, and the balance is computed from the `<answer>` tag of the trace rather than from the file path. The result is 6,000 items, all inside the SFT split, at exactly 3,000 anomalous and 3,000 normal. It is trained with the identical recipe used for every LLaVA-OneVision row in §6.9, from base, frozen vision tower, four epochs, learning rate 10^{-5} , effective batch 32, DeepSpeed ZeRO-3 with optimizer offload, and it is evaluated at every epoch on the full DS-MVTec and VisA subsets under vLLM.

Table M.2: Per-epoch balanced accuracy of the corrected build, against the first build and against the SFT+GRPO model on the same backbone.

Corpus	Checkpoint (epoch)	DS-MVTec	VisA
LLaVA-OneVision SFT+GRPO, reference	ckpt-530	87.66 %	72.38 %
LLaVA-OneVision KCR, first build (corpus defect of \$M)	ckpt-748 (ep4)	88.45 %	74.25 %
LLaVA-OneVision KCR, corrected build	ckpt-188 (ep1)	84.35 %	71.36 %
LLaVA-OneVision KCR, corrected build	ckpt-376 (ep2)	87.32 %	72.65 %
LLaVA-OneVision KCR, corrected build	ckpt-564 (ep3)	86.96 %	73.57 %
LLaVA-OneVision KCR, corrected build	ckpt-748 (ep4)	86.60 %	74.29 %

The corrected build ends where the first one did on VisA, 74.29% against 74.25% at epoch 4, and 1.85 points lower on DS-MVTec. The path differs more than the end point. The corrected run peaks on DS-MVTec at epoch 2 and climbs on VisA at every epoch, while the first build peaked on both at epoch 4. The 2,484 out-of-split images therefore bought the first build its DS-MVTec margin and nothing on VisA. The value of the rerun is that it makes the second-backbone comparison a genuine one-factor comparison. The Qwen2.5-VL result of \$6.6 is unaffected either way, because that corpus was correctly constructed from the start, as the first row of Table M.1 shows.

N

GRPO with a Frozen Vision Encoder

The SFT+GRPO run of Chapter 5 updates every parameter of the model, including the vision encoder, whereas the supervised stages keep the encoder frozen. This appendix records one additional run, made after the main experiments, in which the same GRPO recipe is applied with the vision encoder frozen. It is included for reference only.

N.1. Setup

The run starts from the same SFT checkpoint (6K, frozen encoder, ckpt-564) and uses the same reward, group size $G=4$, learning rate, prompt set of 4,236 items and two epochs as the SFT+GRPO run, with checkpoints at every half epoch. The only change is a flag that sets `requires_grad` to false on the 385 vision-tower tensors and wraps the tower’s forward pass in `no_grad`. Because the merger (projector) sits inside that forward pass, its five tensors receive no gradient either, so the run trains the language model only. Both facts were verified on the saved weights: every vision-tower and merger tensor of the frozen run is bit-identical to the SFT initialisation, while 322 of the 385 vision-tower tensors of the SFT+GRPO checkpoint differ from it. The checkpoints are scored with the same harness as the SFT+GRPO row, under the GRPO prompt.

N.2. Result

Table N.1: Balanced accuracy of the frozen-encoder GRPO run at each half-epoch checkpoint, strict scoring, against its SFT initialisation and the SFT+GRPO run that trained the encoder. Sensitivity and specificity in parentheses.

Model	Checkpoint	DS-MVTec	VisA
SFT initialisation	ckpt-564	80.16	64.78
GRPO, encoder and projector frozen	265 (epoch 0.5)	80.57 (77.8 / 83.3)	67.25 (67.8 / 66.7)
	530 (epoch 1)	82.15 (77.8 / 86.5)	68.95 (64.5 / 73.4)
	795 (epoch 1.5)	82.08 (79.9 / 84.2)	69.81 (65.2 / 74.4)
	1060 (epoch 2)	81.94 (78.3 / 85.6)	69.71 (62.8 / 76.6)
SFT+GRPO, encoder trained	ckpt-530	82.73	70.39

Under the DS-MVTec selection rule the frozen run reports checkpoint 530 at 82.15 % and 68.95 %, which is 0.58 and 1.44 points below the run that trained the encoder. On DS-MVTec the gap is smaller than the 1.08 points that separate two same-seed launches of the SFT+GRPO run (§7.7). The trajectory has the same shape as the trainable run, with most of the gain reached by the end of the first epoch and a plateau afterwards, and the KL to the reference policy ends at 0.116 against 0.085 at step 530 of the trainable run. The gain of GRPO over its SFT initialisation therefore does not depend on updating the vision encoder. It was a single seed, and the run freezes the projector as well as the encoder, so it is a language-model-only variant rather than a minimal change to the SFT recipe.