RuleText-AD: Logical Anomaly Detection via Textual Rule Engines Generated by MLLMs

Leandro Souza da Silva, Mauricio Schiezaro, Diulhio Oliveira
Venturus - Innovation & Technology, Campinas, SP, Brazil, 13086-530

Abstract—**Visual anomaly detection is an important tool in industrial quality control that may impact production efficiency and product quality. Traditional supervised methods struggle due to the difficulties and cost to obtain labeled anomaly data, particularly logical anomalies that violate global constraints that yet appear normal samples. This study investigates multimodal large language models (MLLMs) in few-shot prompting configurations to address this issue. We introduce RuleText-AD, a lightweight anomaly detection framework based on textual rule engines automatically generated by MLLMs from minimal visual examples and concise textual schemas. RuleText-AD eliminates the need for external segmentation tools, extensive prompt engineering, or heavy visual processing backbones, enabling efficient anomaly detection via high-level attribute reasoning. Experiments are conducted on the MVTec-LOCO logical anomaly benchmark, demonstrating that Gemini 2.5 Flash provides the best balance between accuracy and computational cost, achieving competitive performance with significantly reduced deployment complexity. This work highlights the potential for scalable, interpretable, and cost-effective anomaly detection solutions suitable for diverse industrial environments. Our method targets logical (global, rule-violation) anomalies and does not address structural, pixel-level defects.**

I. Introduction

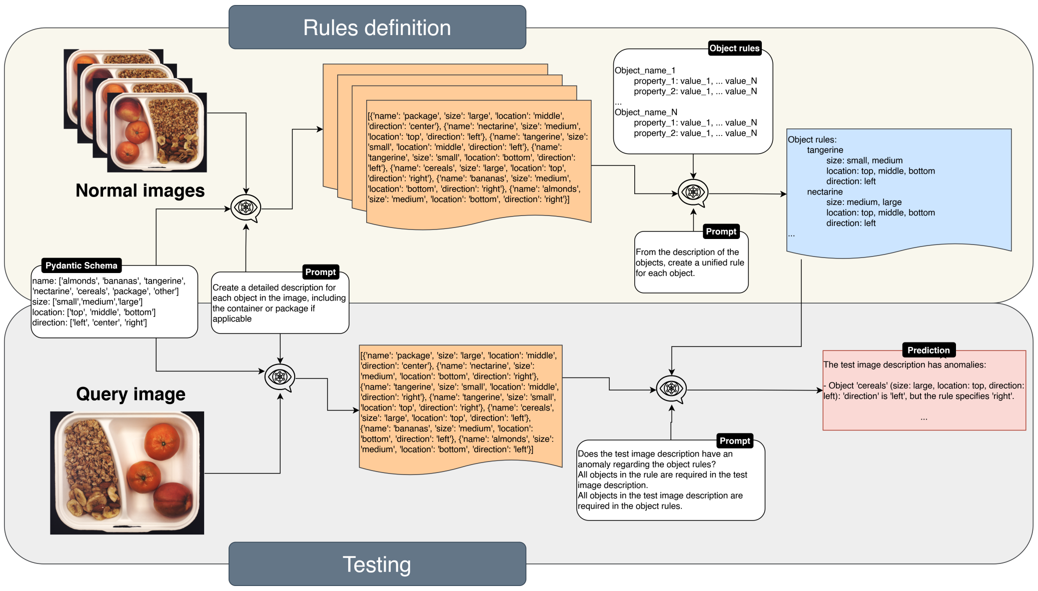
The visual anomaly detection plays an important role in industrial quality control systems, where early identification of defects or irregularities can significantly affect production efficiency, product reliability, and overall safety [1, 5, 6, 7, 13, 18]. Traditional automated inspection methods often rely on supervised learning models trained on large amounts of defect-annotated data [7, 13]. In real-world industrial environments, such labeled datasets may be rare, expensive to obtain, and often domain-specific [3, 4, 13]. In addition, the diversity of potential anomalies can make it impossible to anticipate and label every type of defect during training. As a result, there is growing interest in generalizable and scalable approaches capable of identifying a wide range of anomalies with minimal or no supervision [3, 4, 7, 13].

A useful way to categorize visual defects is to distinguish structural anomalies (SA), localized surface flaws such as scratches and dents, from logical anomalies (LA), which manifest when otherwise normal objects violate global constraints on count, position, orientation, or co-occurrence [4]. Because LAs often "look" normal when inspected patch-wise, they are harder to catch and can have a disproportionate impact on downstream packaging, assembly, or compliance checks. Addressing LAs, therefore, remains a high-value challenge for industry.

Recent advances in computer vision have introduced a variety of unsupervised and semi-supervised anomaly detection methods, including reconstruction-based or generative models [3, 5, 19], and embedding similarity methods [2, 16, 17]. Although effective in constrained settings, these models typically require domain adaptation and fine-tuning to achieve satisfactory performance, limiting their flexibility across diverse scenarios. In parallel, large multimodal models (LMMs), capable of reasoning over both visual and textual inputs, have demonstrated impressive generalization across a wide range of vision-language tasks. This opens new possibilities for deploying these models as zero-shot or few-shot anomaly detectors, enabling visual quality inspection with minimal data and no task-specific training.

The introduction of multimodal architectures has increasingly influenced anomaly detection research, particularly through the integration of powerful vision encoders — such as CLIP [15], GroudingDINO [14], and Segment Anything (SAM) [10] — with large language models (LLMs). These hybrid pipelines typically leverage the visual encoder to extract high-level representations of an image and then use the LLM to interpret or classify these features based on natural language instructions or queries. Some recent studies [8, 9, 11] have proposed using these combinations to detect visual irregularities by comparing embeddings, generating descriptions, or performing binary anomaly judgments in a zero-shot setting. Despite their promising results, these approaches often operate in a feature-matching or similarity-driven regime, underutilizing the reasoning capabilities that recent LMMs have. Also, they share three practical drawbacks: (i) a dependency on external region-of-interest extractors or segmenters (e.g., GroundingDINO, SAM), (ii) long or highly engineered prompts to steer large language models (LLMs), and/or (iii) manual segmentation annotations.

In this work, we propose a few-shot framework that reframes logical anomaly detection as a lightweight rule-induction problem. We restrict our scope to logical anomalies and do not aim to detect structural anomalies. Given only a handful of normal images and simple textual delimiters — object name, size category, coarse location, and direction — an off-the-shelf LLM (GPT-4o-mini or Gemini-2.5-Flash) automatically produces a compact, human-readable rule set describing permissible configurations. At inference time, the same LLM (with or without explicit reasoning traces) parses candidate images, checks each object against the learned rules, and emits a binary anomaly decision along with the violated



```
name: ['almonds', 'bananas', 'tangerine',
'nectarine', 'cereals', 'package', 'other']
size: ['small','medium','large']
location: ['top', 'middle', 'bottom']
direction: ['left', 'center', 'right']
```

Create a detailed description for each object in the image, including the container or package if applicable

```
[{'name': 'package', 'size': 'large', 'location': 'middle',
'direction': 'center'}, {'name': 'nectarine', 'size': 'medium',
'location': 'top', 'direction': 'left'}, {'name': 'tangerine', 'size':
'small', 'location': 'middle', 'direction': 'left'}, {'name':
'tangerine', 'size': 'small', 'location': 'bottom', 'direction':
'left'}, {'name': 'cereals', 'size': 'large', 'location': 'top',
'direction': 'right'}, {'name': 'bananas', 'size': 'medium',
'location': 'bottom', 'direction': 'right'}, {'name': 'almonds',
'size': 'medium', 'location': 'bottom', 'direction': 'right'}]
```

```
Object_name_1
    property_1: value_1, ... value_N
    property_2: value_1, ... value_N
...
Object_name_N
    property_1: value_1, ... value_N
    property_2: value_1, ... value_N
```

From the description of the objects, create a unified rule for each object.

```
Object rules:
    tangerine
        size: small, medium
        location: top, middle, bottom
        direction: left
    nectarine
        size: medium, large
        location: top, middle, bottom
        direction: left
...
```

```
[{'name': 'package', 'size': 'large', 'location': 'middle',
'direction': 'center'}, {'name': 'nectarine', 'size':
'medium', 'location': 'bottom', 'direction': 'right'},
{'name': 'tangerine', 'size': 'small', 'location': 'middle',
'direction': 'right'}, {'name': 'tangerine', 'size': 'small',
'location': 'top', 'direction': 'right'}, {'name': 'cereals',
'size': 'large', 'location': 'top', 'direction': 'left'},
{'name': 'bananas', 'size': 'medium', 'location':
'bottom', 'direction': 'left'}, {'name': 'almonds', 'size':
'medium', 'location': 'bottom', 'direction': 'left'}]
```

Does the test image description have an anomaly regarding the object rules?
All objects in the rule are required in the test image description.
All objects in the test image description are required in the object rules.

The test image description has anomalies:

- Object 'cereals' (size: large, location: top, direction: left): 'direction' is 'left', but the rule specifies 'right'.

...

Fig. 1: The complete framework pipeline, which is organized into an rule definition phase and a light-weight online testing phase. Both stages rely exclusively on a vision–language model (MLLM) — GPT and Gemini-Flash, or their reasoning-suppressed variants — without any external segmenter, handcrafted input segmentation annotation, or handcrafted prompt template.

constraints. By reducing LA detection to text-based reasoning over high-level attributes, the method dispenses with external segmenters, extensive prompt engineering, and heavy visual backbones, while remaining interpretable to non-technical personnel.

Our key contributions are:

- RuleText-AD framework: a two-phase pipeline (Figure 1 that learns object-level spatial rules from normal images and performs logical anomaly detection without external segmentation or fine-tuning.
- Reasoning vs. no-reasoning study: systematic comparison of LLM variants that expose or suppress reasoning, quantifying the cost-benefit of using these models.
- Comprehensive cost analysis: a thorough breakdown of per-image inference expenses, correlating the output token counts and real-world dollar costs for each model, to guide practitioners in balancing accuracy against budget constraints.
- MVTec-LOCO evaluation: extensive experiments on the canonical logical-anomaly benchmark, showing competitive F1-score while outperforming heavier baselines on classes where positional semantics dominate.
- Simplicity and Interpretability: the entire pipeline runs on CPU in real-time, produces editable text rules, and can be audited or modified by domain experts with no ML background.
- The complete implementation is available on GitHub [1].

II. Related work

Early research on industrial anomaly detection focused on structural anomalies (SA), that explore localized surface defects such as scratches, dents, or contaminations. Autoencoder (AE) variants [3, 5, 19] reconstruct normal appearances and flag high reconstruction error as abnormal, but struggle when the background is textured. [2, 16, 17] Embedding similarity methods [2, 16, 17] , such as PatchCore [16] store feature patches from a vision backbone and compare them via k-NN, while other methods [2, 17] compare embeddings between two models trained using the Student and Teacher paradigm. Although these techniques achieve high pixel-level AUROC on SA classes, they inspect each patch independently and thus miss violations of global layout, what characterizes the hallmark of logical anomalies.

Beyond the methods using multi modal models, LogicAD [9] combines GroundingDINO, SAM, and GPT-4o to extract region descriptions and perform first-order logic checking. WinCLIP [8] employs a compositional set of state words and prompt templates alongside multi-resolution window using CLIP feature extraction and aggregation to a zero-shot and few-shot anomaly classification and segmentation without any fine-tuning or pixel-level defect annotations. LogicQA [11] formulates the detection as a MLLM-generated question answering. These methods advance state-of-the-art performance on MVTec-LOCO logical anomalies using multimodal models, yet inherit substantial overhead: they rely on external ROI detectors, produce multi-stage pipelines, or demand elaborate prompt engineering and manual hyper-prompt tuning, factors that hinder deployment on resource-constrained factory lines.

III. Method

A. Problem definition

Let $\mathcal{I} = \{x_1, \ldots, x_K\}$ be a few-shot training set containing K normal images. Each image depicts a finite set of objects

[1]https://github.com/zeroshotfinder/ruletext-ad

whose semantic types are drawn from a closed vocabulary $\mathcal{V}$ (e.g. *bananas*, *almonds*, *cereals*).

For every object $o \in \mathcal{V}$ we assign an attribute vector $a(o) \in \mathcal{A}$, where $\mathcal{A}$ is a Pydantic model

$$\mathcal{A} = \langle a_1, \ldots, a_n \rangle,$$

and *each coordinate* a_j $(j = 1, \ldots, n)$ has a *field*, a pydantic *type*, and a *description*. The Table I shows an example of $\mathcal{A}$ Pydantic model.

TABLE I: Fields of the Pydantic schema for object attributes

Field	Type – Literal	Description
name	[bananas, almonds, cereals]	The name of the object
size	[small, medium, large]	The size of the object
location	[top, middle, bottom]	The location of the object
direction	[left, center, right]	The direction of the object

Given a previously unseen query image $\mathbf{x}_q$, the objective is to learn a decision function

$$f : \mathbf{x}_q \longrightarrow (y, \mathcal{E}),$$

where $y \in \{0, 1\}$ flags the presence of a *logical anomaly* and $\mathcal{E}$ is a human-readable explanation that enumerates each violated attribute. No pixel-wise annotations, bounding boxes, or anomaly labels are available during training or testing; the detector must rely only on the few normal samples and the attribute schema.

B. Overview of RuleText-AD

RuleText-AD employs the **Pydantic-AI framework** in conjunction with Pydantic schemas to facilitate execution agnostic to multiple language model providers and to structure and sanitize model outputs. All model outputs are validated against the Pydantic schema before use, invalid parses trigger a structured re-prompt, maximum 5 attempts. The numbers of attempts used on the experiments were enough to eliminate the parsing failures. This design enables compatibility with both advanced MLLMs capable of complex reasoning, such as o4-mini and Gemini-2.5-Flash, and with lower-latency models that lack inherent reasoning capabilities, including GPT-4.1, Gemini-2.0-Flash, and Llama-4-Maverick.

Figure 1 gives a high–level view of the proposed **RuleText-AD** framework pipeline. The pipeline is divided into distinct phases:

a) Rules definition: The MLLM receives the K normal images together with the attribute schema as delimiters. A short schema-guided prompt compels the model to output a structured description of every object in JSON form. A second prompt asks the MLLM to merge these descriptions into a compact *object-rule set* that enumerates the admissible attribute values for each class (see blue panel in Fig. 1). The result is a fully text-based rule engine that captures global layout regularities without external segmentation or heavy prompt engineering.

b) Testing: At inference time, the same MLLM is prompted to describe the objects in the query image using the identical schema, yielding a structured list of attribute tuples. A lightweight matcher compares each tuple against the stored rule set: any attribute that falls outside the permissible list triggers a violation. If at least one violation is detected, the image is flagged as anomalous and the attributes are collated into the explanation $\mathcal{E}$.

IV. Experiments

A. Experimental Setup

a) Multimodal Large Language Models (MLLMs): Multimodal Large Language Models are capable of processing and generating information across multiple modalities, such as text and images. We used a set of models using temperature 0.3 and top_p 0.95 for more deterministic responses. The Table II shows the usage price for each model. We used the following models in the experiments:

TABLE II: Token pricing per million tokens

Model	Provider	Input Price	Output Price	Context Window
G-2.0F	Google	$0.10	$0.40	1M tokens
G-2.5F	Google	$0.15	$3.50	1M tokens
GPT-4.1	Azure	$2.00	$8.00	1M tokens
o4-mini	Azure	$1.10	$4.40	200k tokens
L4-Mav	Groq	$0.20	$0.60	128k tokens

- **Gemini 2.0 Flash** — A multimodal model developed by Google, optimized for low-latency tasks with a 1 million token context window. It supports text, image, audio, and video inputs, making it suitable for real-time applications and agentic workflows.[2]
- **Gemini 2.5 Flash** — An advanced version of Google's Gemini series, featuring enhanced reasoning capabilities, improved coding performance, and support for complex scientific tasks. It maintains a 1 million token context window and introduces "thinking" capabilities for nuanced responses.[3]
- **GPT-4.1** — OpenAI's flagship large language model, designed for advanced instruction following, software engineering tasks, and long-context reasoning. It supports a 1 million token context window and excels in coding benchmarks, making it ideal for developing AI agents and enterprise applications.[4]
- **o4-mini** — A compact reasoning model from OpenAI's o-series, optimized for speed and cost-efficiency. It offers strong performance in coding, mathematics, and visual tasks, with support for tool use and a 200,000 token context window.[5]
- **Llama 4 Maverick 17B 128E** — Meta's high-capacity multimodal model utilizing a Mixture-of-Experts (MoE)

[2]Gemini 2.0 Flash Documentation
[3]Gemini 2.5 Flash Documentation
[4]GPT-4.1 Documentation
[5]OpenAI o4-mini Documentation

TABLE III: First block shows the average image-level F1, and the second block shows **cost per image** in millidollars (**mUSD**, where 1 mUSD = 10^{-3} USD) for each model and category. The model abbreviations, respectively, are: Gemini-2.0-Flash, Gemini-2.5-Flash, GPT-4.1, GPT o4-mini, and Llama-4-Maverick.

Category	**# images**	**F1-score**					**Cost per image (mUSD)**				
		G-2.0F	**G-2.5F**	**GPT-4.1**	**o4-mini**	**L4-Mav**	**G-2.0F**	**G-2.5F**	**GPT-4.1**	**o4-mini**	**L4-Mav**
Breakfast Box	185	0.79	**0.92**	0.73	**0.92**	0.76	0.22	20.00	4.65	7.62	0.86
Juice Bottle	236	0.67	**0.80**	0.78	0.68	0.67	0.30	4.53	3.47	5.68	0.85
Pushpins	229	**0.81**	0.71	0.67	0.39	0.16	0.35	21.14	9.34	24.32	1.00
Screw Bag	259	0.70	**0.71**	0.68	0.66	0.57	0.23	3.13	2.63	6.37	0.66
Splicing Connector	227	0.75	**0.84**	0.81	**0.84**	0.74	0.31	7.75	5.68	7.53	3.04
Mean	–	0.74	**0.80**	0.74	0.70	0.53	0.28	11.31	5.15	10.30	1.28

architecture with 128 experts and 17 billion active parameters. Suitable for complex reasoning and coding tasks. [6]

b) Generative AI Framework: To structure and validate the data generated by the language models, we employed **PydanticAI**[7], a Python framework designed to simplify the development of production-grade applications utilizing Generative AI. PydanticAI builds upon the capabilities of **Pydantic**[8], a widely-used data validation library in Python.

Pydantic leverages Python's type annotations to enforce data integrity and consistency, ensuring that inputs and outputs adhere to predefined schemas. This foundation allows developers to define clear data models and automatically validate data against them, reducing errors and improving code reliability.

Building on this, PydanticAI extends Pydantic's features to cater specifically to applications involving LLMs and MLLMs. It offers:

- **Structured data validation**: Ensures that data conforms to specified types and constraints, facilitating error detection and debugging.
- **Integration with LLMs** : Supports various LLM providers such as OpenAI, Groq, and Gemini, allowing for flexible integration into diverse AI workflows.

By integrating PydanticAI into our architecture, we established an abstraction layer that standardizes the handling of text, audio, and image files across various models from different providers, thereby streamlining our research experiments.

c) Dataset: The MVTec LOCO AD (Logical Constraints Anomaly Detection) dataset [4] is designed to evaluate unsupervised anomaly detection and localization algorithms, particularly in industrial inspection contexts. It comprises 3,644 high-resolution 220 images across five object categories: breakfast box, screw bag, pushpins, splicing connectors, and juice bottle. The dataset encompasses two distinct types of anomalies: (i) structural anomalies, these are visible defects such as scratches, dents, or contaminations in the products; (ii) Logical anomalies, these involve violations of semantic constraints, such as missing components, incorrect placements, or invalid combinations of parts.

[6]Llama 4 Maverick 17B-128E Documentation
[7]PydanticAI Documentation
[8]Pydantic Documentation

Each anomalous image in the test set is accompanied by pixel-precise ground truth annotations, facilitating detailed evaluation of detection and localization performance. The dataset is partitioned into 1,772 anomaly-free training images, 304 anomaly-free validation images, and 1,568 test images that include both normal and anomalous samples. This dataset split is defined by the original paper [4].

B. Results on different MLLMs

Table III compares the use of five MLLMs on the proposed framework on the MVTec-LOCO logical anomaly test, reporting per-class and mean F1-score together with the dollar cost (mUSD) per image processing. In these experiments, we used 10 randomly selected images for the rule definition phase, which is this shot number defined empirically. Using 10-shot examples accommodates the variability in object positions within the images, enabling the framework to capture all possible variations and establish more precise rules. Averaged in the five categories, Gemini 2.5 Flash (G-2.5F) showed the best mean F1 0.80, respectively, followed by Gemini 2.0 Flash (G-2.0F) and GPT-4.1 with F1 0.74, GPT o4-mini with F1 0.70 and Llama-4-Maverick (L4-Mav) with F1 0.53. The G-2.5F model incorporates a dedicated reasoning mechanism, which accounts for their consistently higher F1 score in four of the five categories. However, the results of o4-mini show that the reasoning mechanism is not enough to guarantee the best results. The trade-off is the amount of text that reasoning models emit, as the Figure 2 shows the output tokens are always higher for models that use reasoning, roughly increasing the budget compared to lighter baselines. This can represent a cost of $2\times$ higher compared to GPT-4.1, or more than $30\times$ higher compared to G-2.0F. In practice, G-2.0F is the cheapest option with competitive results, due to its lower number of output tokens and lower price (see Table II).

The results show that the models performance is not uniform along the categories. For pushpins category, F1 scores decrease across all models, most dramatically for o4-mini (0.39) and Llama-4 (0.16). Pushpins requires counting identical objects packed in a grid, a task current MLLMs still struggle with, large token counts with reasoning do not compensate for this limitation. We also observed that the MLLMs have difficulty distinguishing the size of objects, it is clear on categories such as screw bag, where the normal samples are characterized by

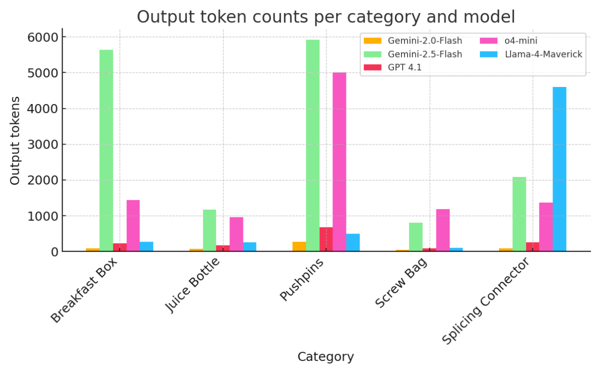


Fig. 2: Comparison of input and output token counts across models for each category.

the presence of two bolts of different size. However, spatial-logic classes such as splicing connectors and breakfast box benefit much more from explicit reasoning: G-2.5F and GPT o4-mini surpass, respectively, 0.84 and 0.92 F1, indicating that step-by-step layout verification is effective here.

Additional experiments are provided in the supplementary material, including qualitative analyses, ablations on the number of image shots, and per-class mean inference times.

C. *Literature comparison*

We conducted side-by-side experiments against three recent vision–language pipelines: LogicAD [9], LogicQA [11] and WinCLIP [8], to evaluate the advantages and limitations compared to methods with more steps. RuleText-AD works with nothing more than the raw MVTec-LOCO image and a concise Pydantic schema, because the spatial rules that drive reasoning are auto-generated from that schema. By contrast, WinCLIP relies on a compositional prompt ensemble of around 80 handcrafted templates to describe every normal/defective state; LogicQA first removes background with Back-Patch-Masking, then calls SAM for zero-shot segmentation before asking the MLLM its questions; and LogicAD introduces Guided Chain-of-Thought prompts together with ROI segmentation and a bespoke logic reasoner that must be authored class-by-class. Eliminating such external components makes our method far easier to reproduce and deploy.

On image-level F1 our schema-only approach reaches 0.80 mean, only 3 pp below LogicAD (0.83) and 7 pp behind the prompt-heavy LogicQA (0.87). The performance gaps remain modest in the categories where we did not outperform prior methods, pushpins (0.71 vs 0.97/0.98) and screw bag (0.71 vs 0.77), underscoring how external segmenters mitigate the current counting limitations of MLLMs. In breakfast box, we actually outperform every baseline (0.92 vs 0.91 LogicQA, 0.82 LogicAD and 0.63 WinCLIP), highlighting that position-aware rules combined with MLLM reasoning can surpass more elaborate pipelines when global layout cues dominate. Overall, despite its simplicity, RuleText-AD delivers competitive F1 while eliminating the engineering overhead endemic to recent MLLM-based detectors.

TABLE IV: Per–category Image-Level F1 comparison with literature techniques. Best score in each row is **bold**.

Category	RuleText-AD (Ours)	LogicAD [9]	LogicQA [11]	WinCLIP [8]
B. Box	**0.92**	0.82	0.91	0.63
Juice Btl.	0.80	0.83	**0.89**	0.58
Pushpins	0.71	**0.98**	0.97	0.57
Scr. Bag	0.71	**0.77**	0.64	0.59
S. Conn.	0.84	0.76	**0.91**	0.60
Mean	0.80	0.83	**0.87**	0.60

V. Costs and Environment

A. *Research Cost*

The total cost of our research during the three-month experimental period was **$496.05**, covering all reported, preliminary, and complementary experiments. This expenditure covers all usage of MLLMs sourced from three different providers: Google Cloud (Vertex AI), Microsoft Azure (Azure OpenAI Service), and Groq. A detailed breakdown of this cost is presented in Table V.

TABLE V: Cost Breakdown by Provider

Provider	Cost (USD)	Percentage of Total
Google Cloud	$326.75	65.9%
Microsoft Azure	$165.30	33.3%
Groq	$4.00	0.8%
Total	**$496.05**	**100%**

B. *CO_2 Emissions Summary*

During the entire research period, we conducted experiments using NVIDIA A100 PCIe 40/80GB GPUs (TDP of 250W) on both Google Cloud Platform (GCP) and Microsoft Azure. The total estimated carbon emissions amounted to **65.70 kgCO_2eq**, all of which were directly offset by the respective cloud providers. This emission corresponds to driving around 260 km in a typical gasoline car, or to one seat on a short-haul flight of roughly 400 to 450 km. Regarding Groq, due to the minimal computation time involved (4.22 hours) and the lack of publicly available carbon intensity data for their infrastructure, we considered the emissions negligible and thus not estimated. The summary of the carbon footprint can be seen in the Table VI.

TABLE VI: CO_2 Emissions by Cloud Provider

Provider	Region	Computation Time (h)	Emissions (kgCO_2eq)
Google Cloud Platform	us-central1	347.95	49.58
Microsoft Azure	eastus2	174.24	16.12
Groq	–	4.22	–
Total	–	**526.41**	**65.70**

We calculated these estimations using the Machine Learning Emissions Calculator proposed by [12].

VI. Conclusions

In this study, we presented RuleText-AD a minimalist multimodal framework that detects logical anomalies in MVTec-LOCO images using the raw frame and a concise, a Pydantic schema of expected object layout. By delegating domain knowledge to this declarative schema and leveraging a reasoning-capable LLM for constraint verification, the approach dispenses elaborate prompt ensembles and zero-shot segmenters required by recent baselines. We evaluated the method using four different MLLMs, showing the advantages of each one. Despite its simplicity, RuleText-AD achieved competitive results compared to baseline methods, outperforming the methods in the Breakfast Box category.

In future work, we will investigate the augment of RuleText-AD with zero-shot segmenters such as SAM to generate fine-grained instance masks before the reasoning step, thereby overcoming the current limitations in precise object counting. These masks can be used to obtain precise object counting, and then incorporate simple pixel-level cues, moving toward a unified detector that covers both logical and structural anomalies. Building on this richer visual input, we also plan to extend the framework into a unified detector that cover both logical anomalies and structural anomalies, allowing a single pass of the pipeline to handle heterogeneous defect types without sacrificing the interpretability and low engineering overhead that motivate our approach.

References

[1] A. Albanese, M. Nardello, G. Fiacco, and D. Brunelli. Tiny machine learning for high accuracy product quality inspection. *IEEE Sensors Journal*, 23(2):1575–1583, 2022.

[2] K. Batzner, L. Heckler, and R. König. Efficientad: Accurate visual anomaly detection at millisecond-level latencies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 128–138, 2024.

[3] P. Bergmann, M. Fauser, D. Sattlegger, and C. 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*, pages 9592–9600, 2019.

[4] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *International Journal of Computer Vision*, 130 (4):947–969, 2022.

[5] D. Candido de Oliveira, B. T. Nassu, and M. A. Wehrmeister. Image-based detection of modifications in assembled pcbs with deep convolutional autoencoders. *Sensors*, 23(3):1353, 2023.

[6] M. Canizo, I. Triguero, A. Conde, and E. Onieva. Multi-head cnn–rnn for multi-time series anomaly detection: An industrial case study. *Neurocomputing*, 363:246–260, 2019.

[7] Y. Cui, Z. Liu, and S. Lian. A survey on unsupervised anomaly detection algorithms for industrial images. *IEEE Access*, 11:55297–55315, 2023.

[8] J. Jeong, Y. Zou, T. Kim, D. Zhang, A. Ravichandran, and O. Dabeer. Winclip: Zero-/few-shot anomaly classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19606–19616, 2023.

[9] E. Jin, Q. Feng, Y. Mou, G. Lakemeyer, S. Decker, O. Simons, and J. Stegmaier. Logicad: Explainable anomaly detection via vlm-based text feature extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 4129–4137, 2025.

[10] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[11] Y. Kwon, D. Moon, Y. Oh, and H. Yoon. Logicqa: Logical anomaly detection with vision language model generated questions. *arXiv preprint arXiv:2503.20252*, 2025.

[12] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[13] J. Liu, G. Xie, J. Wang, S. Li, C. Wang, F. Zheng, and Y. Jin. Deep industrial image anomaly detection: A survey. *Machine Intelligence Research*, 21(1):104–135, 2024.

[14] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[15] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PmLR, 2021.

[16] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2022.

[17] M. Rudolph, T. Wehrbein, B. Rosenhahn, and B. Wandt. Asymmetric student-teacher networks for industrial anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2592–2602, 2023.

[18] M. Rusanovsky, O. Beeri, and G. Oren. An end-to-end computer vision methodology for quantitative metallography. *Scientific Reports*, 12(1):4776, 2022.

[19] Y. Shi, J. Yang, and Z. Qi. Unsupervised anomaly segmentation via deep feature reconstruction. *Neurocomputing*, 424:9–22, 2021.