EvoInspect: A Unified Self-Evolving Multi-Agent Framework for Industrial Hardware Inspection

Zhuyun Yuan[1] **Jun Yin**[2,*] **Jiming Zhang**[1] **Haoyuan Sun**[3]
Qing Wang[4] **Jian Kang**[4] **Jie Li**[4] **Yongxiang Li**[4]

[1]Beihang University, China
[2]Tsinghua University, China
[3]Nanyang Technological University, Singapore
[4]Xingchen AGI Lab, China Telecom Artificial Intelligence Technology Co., Ltd., China

First Author: 20241064@buaa.edu.cn
[*]Corresponding Author: yinj24@mails.tsinghua.edu.cn

Abstract

Human experts can accumulate experience from open-ended and complex inspection tasks, thereby continually improving their inspection capabilities. Actually, industrial hardware inspection faces challenges including open-set defect categories, complex defects (tiny, high-gloss, low-light, or blurred), high inference cost, and difficulty in accumulating experience post-deployment. Existing end-to-end inspection methods typically couple image restoration with defect recognition in a single forward pass, limiting generalization to unseen components or composite defects; cascade "restore-then-detect" approaches often rely on static preprocessing, hindering adaptive adjustment based on scenario, defect morphology, and historical failures. To address this, we propose EvoInspect, the first memory-enhanced multi-modal multi-agent framework for industrial defect object detection. It leverages a multi-modal agentic memory mechanism to distill self-evolving scenario-specific Defect Detail Restoration decisions and dynamically grounded trajectories, enabling "accumulating experience like human experts." Specifically, (1) we introduce a dynamic Defect Detail Restoration strategy, including adaptive slicing for tiny defects, highlight suppression, low-light enhancement, super-resolution, and binarization, combined with a coarse-to-fine token-efficient grounding mechanism for efficient defect capture; (2) we design a dual-track distillation mechanism unifying restoration and reasoning experience, incorporating both successful and failed offline feedback trajectories for training-free self-evolution of the Skill Bank. Experiments on bearing surfaces, PCBs, solar electroluminescence (solar EL), and magnetic tile datasets show that EvoInspect consistently outperforms strong baselines in defect localization recall, inference accuracy, while demonstrating cross-scenario adaptability and offline continual improvement. Project page: https://github.com/yuan-zhuyun/EvoInspect.

1 Introduction

Visual industrial defect detection is central to quality assurance in domains such as PCBs, bearings, photovoltaic components, and metal surfaces. In practice, however, inspection systems must cope with open-set defects, tiny targets, image degradations, and limited opportunities to accumulate experience after deployment.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Existing approaches address only part of this problem. End-to-end detectors can perform well on fixed label sets but often generalize poorly to unseen or composite defects (YOLO [1], YOLO-World [2], SAHI [3]). Restore-then-detect pipelines provide more control, yet their restoration strategies are usually static and therefore brittle across scene conditions and defect morphologies (AnomalyGPT [4], Grounding DINO [5], Anomaly-OV [6]). Recent self-evolving methods can accumulate experience over time, but they are mostly single-modal and have not been systematically integrated into industrial hardware inspection (ReMe [7], Memento-Skills [8], SkillRL [9]).

These limitations suggest that industrial inspection should be framed as a coordinated process spanning perception, localization, defect-detail restoration, structured reasoning, and experience accumulation.

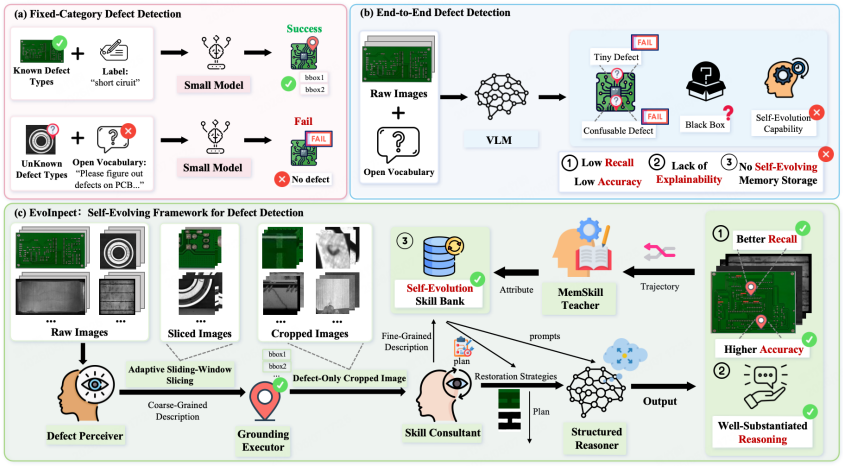

Figure 1: **Comparison of Defect Detection Paradigms and the Proposed EvoInspect Framework**

To address this gap, we propose **EvoInspect** (Figure 1), a unified self-evolving framework for industrial hardware inspection. EvoInspect organizes inspection as a coarse-to-fine pipeline with explicit agent roles and an external memory mechanism, enabling cross-scenario adaptation and post-deployment continual improvement. Our main contributions are:

- **A unified inspection framework with agentic memory:** EvoInspect connects coarse perception, open-vocabulary localization, local restoration planning, structured adjudication, and dual-track memory updates into a single pipeline. On bearing surfaces, PCBs, solar EL, and magnetic tile datasets, it achieves strong performance compared with representative large models (Qwen3.5 [10], MiMo-VL [11]) and is competitive with specialized small-object detectors (YOLO26 [1]), suggesting potential for broader industrial hardware defect inspection scenarios.
- **A coarse-to-fine detection and restoration design:** We combine adaptive slicing for tiny defects with staged defect-detail restoration (including low-light enhancement, highlight suppression, super-resolution, and morphology-aware binarization) to improve the quality of evidence passed to the reasoner.
- **A dual-track skill-evolution mechanism:** We maintain a cross-modal dual-track memory architecture with reusable restoration plans, jointly leveraging successful and failed trajectories to evolve strategy memory and reasoning memory, enabling learning from both successes and failures without updating deployed model weights.

2 Related Work

Prior work relevant to industrial hardware inspection falls into three groups. First, restoration methods improve defect visibility under tiny scales and degraded imaging conditions, including adaptive slicing [3], highlight suppression and low-light enhancement [12–14], and super-resolution or denoising-based reconstruction [15–17]. Second, localization and reasoning methods have progressed from conventional CNN/Transformer detectors [18–20] to open-vocabulary vision-language models [21, 5, 2], making it possible to localize and interpret previously unseen anomalies. Third,

self-evolving agent systems use trajectory distillation, reflection, and experience banks to improve over time [22–24, 7–9, 25].

Despite this progress, these strands are rarely integrated into a single industrial inspection pipeline. Existing systems typically optimize restoration, localization, reasoning, or memory in isolation, and few support post-deployment improvement grounded in both successful and failed industrial trajectories. EvoInspect addresses this gap by coupling adaptive restoration, grounded localization, structured reasoning, and dual-track memory write-back in one framework. A broader review of related work on image restoration, defect localization, multimodal reasoning, and self-evolving memory is provided in Appendix A.

3 Method

3.1 Motivation

Industrial defect inspection must operate under heterogeneous imaging conditions, mixed degradations, and deployment constraints that make failures costly and model refresh expensive. In this setting, a single end-to-end model is hard to diagnose, difficult to update in a controlled way, and brittle when transferred across production lines or hardware variants.

We therefore adopt a modular coarse-to-fine pipeline with explicit interfaces between global perception, open-vocabulary localization, defect-detail restoration, and structured adjudication. In parallel, an external **Skill Bank** is refined through offline self-review of recorded trajectories with outcome feedback. This design supports controlled post-deployment improvement, clearer failure analysis, and less reliance on repeated backbone fine-tuning.

3.2 System Overview

EvoInspect is a self-evolving multi-agent framework for industrial hardware inspection. It consists of five components: the **Defect Perceiver** for scene understanding, the **Grounding Executor** for open-vocabulary localization, the **Skill Consultant** for defect-detail restoration planning, the **Structured Reasoner** for defect adjudication, and the **MemSkill Analyzer** role for memory refinement (Figure 2). Together, these agents pursue high recall, accurate defect discrimination, and continual skill updates.

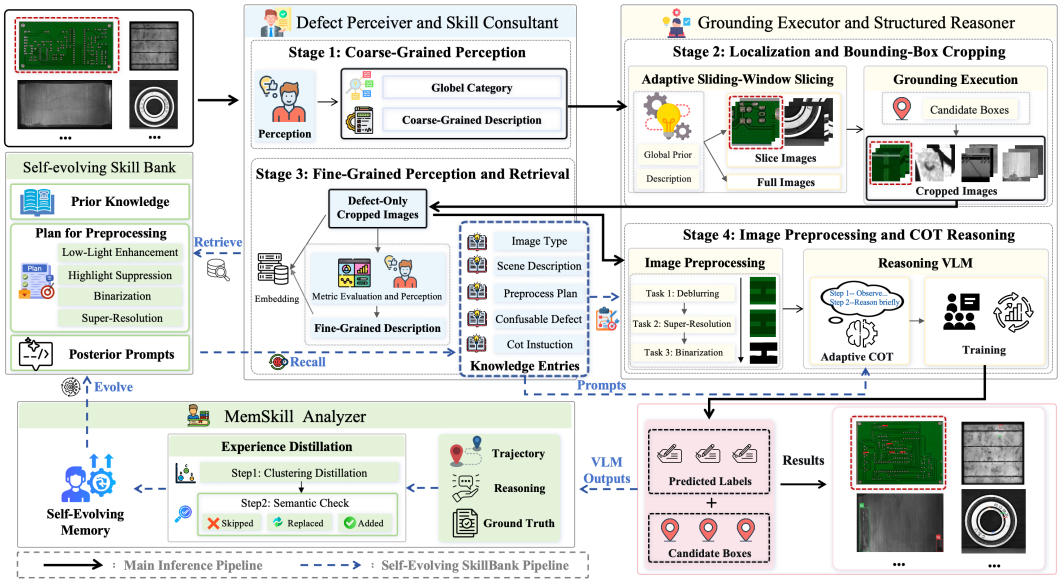

Figure 2: **System Overview of EvoInspect**

Before inference, we warm-start an external **Skill Bank** with two tracks: **Mem(Strategies)** stores reusable defect-detail restoration plans, and **Skill(CoT)** stores structured reasoning scaffolds distilled from prior trajectories.

During inference, the **Defect Perceiver** first produces a coarse scene description and decides whether adaptive slicing is needed. The **Grounding Executor** then proposes high-recall candidate boxes, which are cropped and passed to the **Skill Consultant** for fine-grained description and restoration planning. If the consultant retrieves a high-confidence plan from the skill bank, that plan is executed directly; otherwise, the consultant falls back to adaptive planning guided by no-reference IQA signals. The restored crop is finally passed to the **Structured Reasoner**, which predicts the defect type and produces an interpretable reasoning trace.

After deployment, the **MemSkill Analyzer** periodically refreshes the external **Skill Bank**: successful trajectories consolidate reliable restoration plans, while failed trajectories revise the reasoning scaffolds and clarification rules used in later inference.

3.3 Coarse-to-Fine Adaptive Restoration

EvoInspect performs defect-detail restoration through three tightly scoped modules: the **Defect Perceiver** for global context, the **Grounding Executor** for localization and cropping, and the **Skill Consultant** for restoration planning. The compact interface of each module facilitates systematic analysis, verification, and ablation.

Defect Perceiver. Given an input image I of size $H_I \times W_I$, let $M_I = \max(H_I, W_I)$. If $M_I \leq 1024$, we set the slice size to $s = M_I$ and use a single tile, i.e., $n_{\text{slice}} = 1$. If $M_I > 1024$, the slice size is selected adaptively:

$$s = \begin{cases} 256, & 1024 < M_I < 2048, \\ 512, & 2048 \leq M_I < 4096, \\ 1024, & M_I \geq 4096. \end{cases} \tag{1}$$

Given an overlap ratio $\delta \in (0, 1)$, the image is partitioned into $\{I_j\}_{j=1}^{n_{\text{slice}}}$ using a standard sliding-window grid with stride $(1-\delta)s$ along each axis. Edge padding or cropping is applied when necessary. The perceiver $\mathrm{P}_R(\cdot)$ produces a coarse description together with the tiling metadata generated during preprocessing:

$$(D_I, s, n_{\text{slice}}) = \mathrm{P}_R(I). \tag{2}$$

Grounding Executor. Let $\mathcal{I} = \{I_j\}_{j=1}^{n_{\text{slice}}}$ for a tiled input and $\mathcal{I} = \{I\}$ otherwise. Given the coarse description D_I and the image set $\mathcal{I}$, the localization model $\mathrm{G}_E(\cdot)$ produces candidate bounding boxes and their corresponding crops:

$$(\{\text{bbox}_i\}_{i=1}^{n}, \{I'_i\}_{i=1}^{n}) = \mathrm{G}_E(D_I, \mathcal{I}). \tag{3}$$

Skill Consultant. For each candidate crop I'_i, the Skill Consultant first derives an internal fine-grained description D'_i. Using reusable strategies from its memory bank as priors, it dynamically determines the operator order and parameters from a finite restoration library $\mathcal{O}$ according to the semantics and morphology of the current defect. The strategy for crop i is represented as

$$\mathcal{S}_i = \mathrm{S}_{\mathrm{C}}\left(I'_i, D'_i; \mathcal{M}, \mathcal{O}\right) = \left(o_{i,1}(\theta_{i,1}), \ldots, o_{i,L_i}(\theta_{i,L_i})\right), \tag{4}$$

where $\mathcal{M}$ denotes the available, possibly empty, strategy memory, L_i is the number of selected operations, and $\theta_{i,l}$ denotes the adaptive parameters of the l-th operation. The restored crop is obtained through sequential operator composition:

$$I^*_i = \left(o_{i,L_i}(\theta_{i,L_i}) \circ \cdots \circ o_{i,1}(\theta_{i,1})\right)(I'_i). \tag{5}$$

When multiple admissible strategies or restored candidates are considered, the Skill Consultant selects the result by jointly considering defect semantics, morphology, and auxiliary quality diagnostics. Applying this procedure to all candidate crops produces

$$\mathcal{R} = \{(I^*_i, \mathcal{S}_i)\}_{i=1}^{n}, \tag{6}$$

which serves as the localized visual input for subsequent structured fine-grained reasoning. EvoInspect thus replaces a fixed restoration pipeline with defect-adaptive operator composition while retaining explicit and traceable strategy records.

3.4 Self-Reviewed Dual-Track Skill Evolution

EvoInspect introduces an outcome-driven dual-track skill evolution mechanism that unifies defect-detail restoration plans and structured adjudication trajectories within a shared experience loop. During offline self-review, the **MemSkill Analyzer** consolidates successful trajectories into restoration skills in **Mem(Strategies)** and distills failed trajectories into corrective reasoning rules in **Skill(CoT)**. This enables non-parametric evolution of external skills while keeping the underlying model parameters frozen.

For a correctly predicted trajectory, the **MemSkill Analyzer** abstracts the image, defect description, restoration sequence, and adjudication outcome into a candidate restoration skill:

$$v_i^s = A_\theta(x_i^s) = (p_i, \mathcal{S}_i, e_i)\,, \qquad \hat{z}_i = z_i, \tag{7}$$

where $p_i = (t_i, D_i, G_i)$ characterizes the applicability boundary in terms of the industrial scenario, defect semantics, and degradation profile; $\mathcal{S}_i$ is the reusable restoration operator sequence, and e_i is the multimodal retrieval representation.

The system retrieves the existing skill with the most similar applicability conditions and evaluates the task-level gain of the candidate on relevant historical experience:

$$\begin{aligned} j^\star &= \arg\max_j \mathrm{Sim}_s\left(p_i, e_i;\, p_j, e_j\right), \\ s_i &= \mathrm{Sim}_s\left(p_i, e_i;\, p_{j^\star}, e_{j^\star}\right), \\ \Delta_i^s &= \mathcal{V}_s\left(\mathcal{S}_i; \mathcal{B}_i^s\right) - \mathcal{V}_s\left(\mathcal{S}_{j^\star}; \mathcal{B}_i^s\right). \end{aligned} \tag{8}$$

Here, Sim_s jointly measures consistency in scenario, semantics, degradation characteristics, and visual representations; $\mathcal{V}_s$ evaluates the contribution of a restoration strategy to downstream defect adjudication; and $\mathcal{B}_i^s$ denotes historical experience relevant to the candidate skill. Based on the matching score and task-level gain, **Mem(Strategies)** is structurally updated as

$$\mathrm{Op}_s = \begin{cases} \mathrm{Add}(v_i^s), & s_i < \eta_s, \\ \mathrm{Replace}(v_{j^\star}^s, v_i^s), & s_i \geq \eta_s \wedge \Delta_i^s > \epsilon_s, \\ \mathrm{Delete}(v_j^s), & \mathrm{Redundant}\left(v_j^s; \mathcal{M}_s\right) = 1. \end{cases} \tag{9}$$

ADD extends skill coverage to new scenarios and degradation patterns, REPLACE incorporates restoration experience with higher task utility, and DELETE preserves a compact and semantically consistent skill bank according to accumulated feedback and skill coverage. A successful trajectory matching an existing skill in both applicability and operator sequence only updates its empirical support and usage statistics.

For a failed trajectory, the **MemSkill Analyzer** identifies the critical reasoning deviation, generates a corrected trajectory, and abstracts it into a candidate reasoning rule with explicit triggering conditions:

$$\left(b_i, a_i, y_i^+\right) = A_\theta\left(I_i, q, m_i, y_i, z_i\right), \qquad r_i^{\mathrm{CoT}} = C(a_i, y_i^+) = \left(p_i^f, \ell_i, c_i^+\right). \tag{10}$$

Here, b_i is the retention indicator, a_i is the structured error attribution, and y_i^+ is the corrected reasoning trajectory. The term p_i^f describes the failure pattern and its triggering conditions, ℓ_i specifies the corrective rule, and c_i^+ provides additional discriminative evidence.

The candidate rule is either incorporated into **Skill(CoT)** or used to revise an existing rule. To jointly account for error correction and behavioral preservation, the candidate update is validated on relevant failed trajectories $\mathcal{B}_i^-$ and historical successful trajectories $\mathcal{B}_i^+$:

$$\Delta_i^\pm = \mathcal{V}_c\left(U\left(\mathcal{M}_{\mathrm{old}}^{\mathrm{CoT}}, r_i^{\mathrm{CoT}}\right); \mathcal{B}_i^\pm\right) - \mathcal{V}_c\left(\mathcal{M}_{\mathrm{old}}^{\mathrm{CoT}}; \mathcal{B}_i^\pm\right), \tag{11}$$

where Δ_i^- denotes the correction gain on related failures, Δ_i^+ measures the preservation of previously correct behavior, and $U(\cdot)$ denotes external reasoning-memory editing. The **Skill(CoT)** memory is finally updated as

$$\mathcal{M}_{\text{new}}^{\text{CoT}} = \begin{cases} U\left(\mathcal{M}_{\text{old}}^{\text{CoT}}, r_i^{\text{CoT}}\right), & b_i = 1 \wedge \Delta_i^- > \epsilon_f \wedge \Delta_i^+ \geq -\epsilon_p, \\ \mathcal{M}_{\text{old}}^{\text{CoT}}, & \text{otherwise.} \end{cases} \tag{12}$$

Through this closed loop of trajectory review, experience abstraction, retrieval-aware matching, and validated write-back, **Mem(Strategies)** consolidates scenario-adaptive restoration experience from successful trajectories, while **Skill(CoT)** transforms failed trajectories into reusable corrective reasoning rules. Together, the two tracks enable EvoInspect to continually accumulate stable, retrievable, and transferable industrial inspection skills without updating the model parameters.

4 Experiments

4.1 Experimental Setup

We evaluate EvoInspect across four industrial domains: Bearing, PCB, Solar EL, and Magnetic Tile.

Coarse localization. The **Grounding Executor** is built upon Grounding DINO and optimized on four-domain data with open-vocabulary textual prompts. The model is trained for 15 epochs using AdamW with a learning rate of 1×10^{-4}. Regions corresponding to the predicted boxes are then cropped and passed to the defect-detail restoration stage.

Defect-detail restoration. The **Skill Consultant** applies scenario-adaptive restoration pipelines to the cropped patches. We report PSNR when reference images are available and use TOPIQ-NR, MUSIQ, and CLIPIQA to evaluate no-reference image quality.

Fine-grained reasoning. Under the MS-Swift framework, we LoRA-fine-tune Qwen3.5-9B as the **Structured Reasoner** on a mixed four-domain dataset, with rank 8 and α=32. The model is trained for 5 epochs on four NVIDIA A100 GPUs using the corresponding cropping and prompting strategies.

Skill self-evolution. Qwen3.5-9B also serves as the **MemSkill Analyzer**, which reviews execution trajectories and proposes candidate updates to the external Skill Bank.

Evaluation. We report localization performance on full-image and tiny-defect subsets, patch-level restoration quality, fine-grained recognition accuracy on recalled regions, and gains from skill updates under fixed inference settings. We further evaluate Skill Bank scalability in terms of retrieval, merging, and token overhead, and analyze end-to-end latency and the extensibility of the modular framework to compatible components.

4.2 Coarse-Grained Localization

We evaluate instance-level localization using R@0.5, P@0.5, and mAP@0.5 (IoU$=0.5$) to quantify **Grounding Executor** recall under challenging scales. Adaptive slicing (+Adp) is used only for Bearing and PCB when the longer side exceeds 1024 pixels; Solar EL and Magnetic Tile are evaluated on full-resolution inputs, hence +Adp is omitted ("–" in Table 1). For instances with relative box area $\leq 0.01\%$, we additionally report tR/tP at IoU$=0.3$ on Bearing and PCB (Table 2).

Table 1: Overall localization results.

Method	Bearing			PCB			Solar EL			Magnetic		
	R	P	mAP@0.5	R	P	mAP@0.5	R	P	mAP@0.5	R	P	mAP@0.5
EvoInspect	*0.7644*	*0.8499*	*0.7521*	0.6103	*0.9672*	0.6047	**0.7370**	0.8826	**0.7243**	*0.8571*	**0.8639**	**0.8518**
+Adp	**0.9088**	**0.9391**	**0.9315**	**0.9320**	0.9697	0.9304	–	–	–	–	–	–
YOLO26	0.865	0.891	0.927	0.9	**0.973**	**0.947**	*0.656*	**0.909**	0.714	**0.891**	0.839	0.8
Qwen3.5-9B	0.7176	0.7472	0.6151	*0.7864*	0.7941	*0.6457*	0.7035	*0.7841*	*0.4627*	*0.7966*	*0.8332*	*0.6905*
MiMo-VL-7B	0.6005	0.6214	0.5469	0.6019	0.6078	0.4592	0.4796	0.5785	0.3109	0.6429	0.6637	0.5334
DeepSeek-VL-7B-base	0.1009	0.1048	0.0342	0.3689	0.3725	0.2103	0.1861	0.1981	0.0685	0.1677	0.1890	0.1325
InternVL-3.5-8B	0.5596	0.4761	0.4738	0.7282	0.7353	0.6044	0.6369	0.7111	0.4226	0.7418	0.7778	0.5707
Rex-Thinker-7B	0.6274	0.6915	0.5745	0.6117	0.6176	0.4682	0.4236	0.5634	0.3739	0.6606	0.6525	0.6198

Quantitative analysis. Table 1 compares EvoInspect with vision-language models and a dedicated detector (YOLO26) under the same localization annotations. EvoInspect is competitive among the vision-language methods, and adaptive slicing (+Adp) raises mAP@0.5 from 0.7521 to 0.9315 on

Bearing and from 0.6047 to 0.9304 on PCB. This brings its localization results close to the dedicated detector on these two domains.

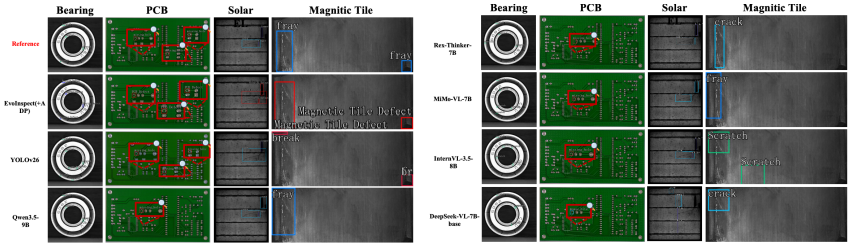

Figure 3: Overall visual comparisons.(Please zoom in for details).

Qualitative analysis. Figure 3 shows that EvoInspect usually covers subtle defects with tight predicted boxes. YOLO26 more often introduces spurious boxes on non-defect regions, while open VLMs more frequently miss small targets or fire on clutter.

Table 2: Tiny-defect localization results on Bearing and PCB.

Method	Bearing			PCB		
	tR	tP	mAP@0.3	tR	tP	mAP@0.3
EvoInspect	*0.6957*	*0.7914*	0.5872	0.6167	*0.9196*	*0.5571*
+Adp	0.8125	**0.8776**	0.8206	**0.9029**	0.9394	0.8778
YOLO26	**0.833**	0.871	**0.855**	0.896	**0.958**	**0.928**
Qwen3.5-9B	0.6545	0.7472	*0.5917*	0.6505	0.6509	0.4965
MiMo-VL-7B	0.5176	0.5951	0.4245	0.4689	0.4725	0.3358
DeepSeek-VL-7B-base	0.0741	0.0878	0.0391	0.1845	0.1863	0.0634
InternVL-3.5-8B	0.5065	0.4081	0.3934	*0.6602*	0.6667	0.5356
Rex-Thinker-7B	0.5296	0.6172	0.4775	0.4854	0.4902	0.3519

Quantitative comparison. On the tiny-defect subset (relative box area $\leq 0.01\%$), Table 2 shows that +Adp improves EvoInspect's mAP@0.3 by 39.8% on Bearing and 57.6% on PCB relative to the unsliced configuration. EvoInspect (+Adp) leads the compared vision-language models and approaches YOLO26, which has the highest mAP@0.3 in both domains.

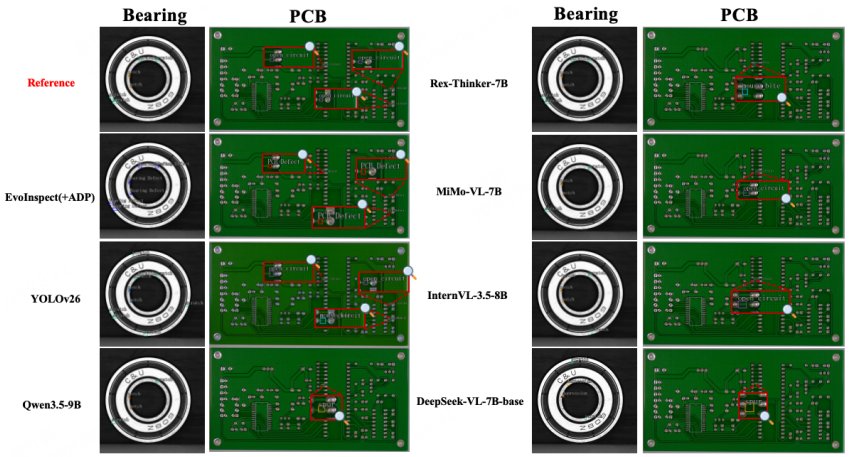

Figure 4: Tiny-defect visual comparisons.

Qualitative analysis. Figure 4 compares models on tiny defects. EvoInspect recalls the defect regions with few false positives even when the boundaries are blurred, whereas end-to-end multimodal models exhibit frequent misses. This supports the robustness advantage of stage-wise decoupling.

Overall, these results show that the **Grounding Executor**, combined with adaptive slicing, recovers fine localization signal and improves recall without changing the annotation protocol.

4.3 Defect Detail Restoration

We evaluate dynamic defect-detail restoration on cropped defect regions by measuring changes in local visual evidence before downstream reasoning. We report no-reference quality metrics for all restored outputs and PSNR when aligned references are available.

Table 3: Per-domain restoration stacks on defect crops.

Domain	Restoration Strategies	PSNR↑	TOPIQ-NR↑	MUSIQ↑	CLIPIQA↑
Bearing	✗	–	0.4916	24.52	0.4964
	✓	23.35	**0.5461**	**32.54**	**0.5397**
PCB	✗	–	0.4547	26.78	0.4333
	✓	20.93	**0.5474**	**37.91**	**0.5425**
Solar EL	✗	–	0.2093	21.39	0.2802
	✓	19.06	**0.5892**	**39.34**	**0.4771**
Magnetic Tile	✗	–	0.4410	25.77	0.4301
	✓	19.32	**0.6291**	**34.83**	**0.5323**

Quantitative comparison. Table 3 shows that dynamic defect-detail restoration consistently improves perceptual quality metrics, including TOPIQ-NR, MUSIQ, and CLIPIQA, across all four industrial domains. This demonstrates its ability to enhance local defect evidence under diverse degradation conditions and provide higher-quality inputs for downstream fine-grained reasoning.

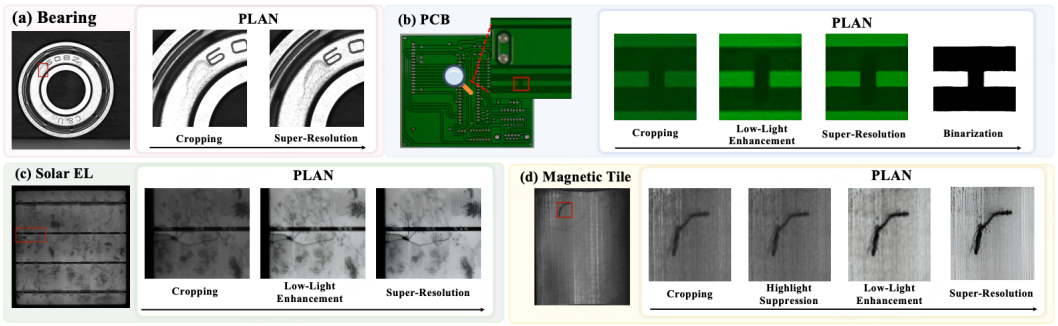

Figure 5: Representative dynamic restoration strategies across four industrial domains.

Qualitative analysis. Figure 5 visualizes representative restoration trajectories selected from the most frequently invoked strategies in each dataset. These examples illustrate how adaptive operator compositions enhance defect-relevant details under heterogeneous degradation conditions.

4.4 Fine-grained Reasoning

We evaluate the complete inspection pipeline under identical data splits and a unified end-to-end criterion. Starting from EvoInspect, we progressively enable **Mem(Strategies)** and **Skill(CoT)**, and compare against YOLO26 and VLM baselines.

Quantitative comparison. Table 4: +Mem and +Skill consistently improve the end-to-end accuracy of EvoInspect. EvoInspect (+*Skill(CoT)*) is competitive with fully supervised YOLO26 and exceeds all listed VLMs in every column.

Table 4: End-to-end fine-grained inference accuracy.

Method	Bearing Acc	PCB Acc	Solar EL Acc	Magnetic Acc
EvoInspect	0.7079	0.7651	0.5417	0.6537
+*Mem (Strategies)*	*0.7656*	*0.7952*	0.6497	*0.6832*
+*Skill (CoT)*	**0.8215**	0.8596	**0.6929**	0.7411
YOLO26	0.7897	**0.8668**	*0.6093*	**0.7925**
Qwen3.5-9B	0.5734	0.6158	0.5072	0.5527
MiMo-VL-7B	0.4423	0.4057	0.3349	0.3834
DeepSeek-VL-7B-base	0.0631	0.2115	0.1097	0.0899
InternVL-3.5-8B	0.4273	0.5600	0.4588	0.4798
Rex-Thinker-7B	0.4894	0.4552	0.2983	0.4150

Generalization. We further evaluate fine-grained reasoning on test samples from different product models within the same defect categories (Table 5). With the Skill Bank, EvoInspect achieves the high-

est accuracy across all four industrial domains, demonstrating the effective transfer of defect-centric knowledge across variations in appearance and hardware configurations. In contrast, as a closed-set supervised detector, YOLO26 relies more strongly on the training distribution; product-level shifts in component geometry, surface texture, and imaging characteristics can therefore substantially degrade its recognition performance. Without retraining on the target product models, these results further highlight EvoInspect's adaptability to open and heterogeneous industrial scenarios.

Table 5: Generalization accuracy with and without Skill Bank augmentation.

Method	Skill Bank	Bearing	PCB	Solar EL	Magnetic
EvoInspect	✗	0.3412	0.4398	0.3788	0.2813
	✓	**0.4592**	**0.5455**	**0.4884**	**0.3978**
YOLO26	✗	0.192	0.260	0.215	0.204
	✓	–	–	–	–
Qwen3.5-9B	✗	0.3328	0.3810	0.3444	0.1967
	✓	0.3992	0.4255	0.3943	0.3256
MiMo-VL-7B	✗	0.3149	0.2542	0.1256	0.1441
	✓	0.3512	0.2913	0.2768	0.2375
InternVL-3.5-8B	✗	0.3477	0.1071	0.2103	0.1762
	✓	0.4535	0.1942	0.2955	0.2638
Rex-Thinker-7B	✗	0.3203	0.2233	0.1019	0.1595
	✓	0.3764	0.2381	0.2386	0.2767

†YOLO26 is evaluated in its native detector configuration, as its architecture does not expose an interface for external Skill Bank injection.

4.5 Scalable Experience Consolidation and Efficient Modular Inference

EvoInspect further evaluates the scalability of its dual-track Skill Bank and the inference efficiency of defect-centric routing. With the detector and reasoner parameters frozen, the **MemSkill Analyzer** performs offline self-review of deployment trajectories: successful experience is consolidated into restoration skills in **Mem(Strategies)**, while failed experience is used to revise corrective reasoning rules in **Skill(CoT)**, forming a non-parametric experience-evolution loop.

Table 6 evaluates the consolidation and retrieval efficiency of the dual-track Skill Bank at different scales. As the Skill Bank expands, **Mem(Strategies)** and **Skill(CoT)** continuously integrate new experience into the existing knowledge structure while maintaining stable retrieval latency and controlled context overhead, demonstrating strong scalability.

Figure 6 further shows that defect-centric cropping substantially reduces visual-token overhead, focusing computation on locally discriminative defect evidence. Combined with lightweight skill retrieval, EvoInspect supports continual experience accumulation while preserving efficient multimodal reasoning.

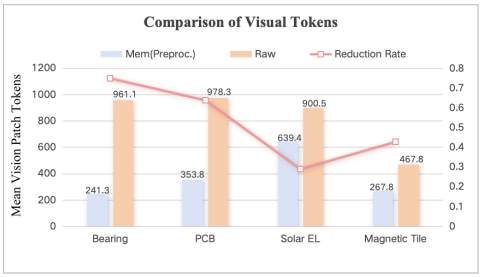

Figure 6: Mean vision patch tokens for raw full-image and crop-based inputs.

Table 6: Scalability of dual-track Skill Bank consolidation and retrieval.

Scale	Consolidation Rate M/C (%)	Search p50/p95	Tok.
25%	78.7/15.1	0.63/0.83	791
50%	87.3/25.8	1.12/1.34	803
75%	93.5/35.6	1.55/1.73	842
100%	97.4/47.8	1.91/2.20	838

M/C: Mem/CoT merge rate; search: p50/p95 in ms; Tok.: injected tokens.

End-to-end inference efficiency. Figure 7 shows that reusing high-confidence restoration plans reduces redundant planning and substantially lowers both **Skill Consultant** and overall inference overhead. The **MemSkill Analyzer** performs experience review and skill write-back offline, keeping these operations outside the critical path and enabling efficient execution of the complete multi-stage

inspection pipeline. Although the complete pipeline incurs slightly higher end-to-end latency than a single-pass model, the preceding results demonstrate that EvoInspect achieves a more competitive overall trade-off among computational efficiency, defect recall, and fine-grained recognition accuracy.

Modular extensibility. EvoInspect's explicit module interfaces support the plug-and-play integration of efficient components without restructuring localization, restoration, or Skill Bank updates. After incorporating VisionZip [26], the framework maintains stable recognition performance as visual-token retention decreases (Table 7), demonstrating strong compatibility with computational optimization modules.

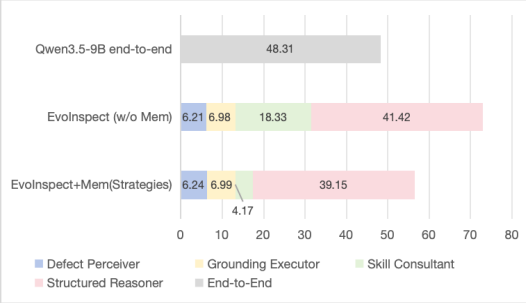

Figure 7: End-to-end latency (s/sample).

Table 7: VisionZip retention.

Retain	Acc
100%	0.7553
75%	0.7468
50%	0.7215
25%	0.6796

Together, training-free Skill Bank revision and defect-centric routing reduce the need for retraining to incorporate feedback and the visual-token load of full-image VLM calls. On-demand skill retrieval and modular interfaces also let compatible optimizations such as VisionZip be plugged into the inspection pipeline without redesigning its core logic.

5 Conclusions

We presented EvoInspect, a self-evolving multi-agent framework for industrial hardware inspection. EvoInspect organizes global perception, open-vocabulary localization, defect-detail restoration, structured adjudication, and experience write-back into a unified collaborative pipeline, while connecting restoration and reasoning experience through a dual-track Skill Bank. Experiments across Bearing, PCB, Solar EL, and Magnetic Tile demonstrate strong performance in overall and tiny-defect localization, local evidence enhancement, fine-grained defect recognition, and cross-model generalization. EvoInspect also produces structured and interpretable adjudication traces grounded in cropped defect evidence.

More importantly, EvoInspect consolidates successful and failed trajectories into reusable restoration skills and corrective reasoning rules while keeping the detector and reasoner parameters frozen, enabling non-parametric continual evolution after deployment. The dual-track Skill Bank maintains efficient retrieval and controlled context overhead as it scales, while defect-centric routing and high-confidence plan reuse reduce visual-token consumption and redundant planning. Combined with modular interfaces that support compatible components, EvoInspect provides a unified industrial inspection paradigm with self-evolution capability, token-efficient inference, and strong system extensibility.

6 Limitations

While EvoInspect offers improved recall and interpretability, its multi-stage pipeline, tiling strategy, and large-model adjudication increase computational load and memory usage, and the overall system remains relatively complex; nonetheless, the design provides flexibility for adapting to different inspection scenarios and experimental configurations.

Acknowledgments and Disclosure of Funding

Funding and Acknowledgments. This work received no direct financial funding. We gratefully acknowledge Xingchen AGI Lab, China Telecom Artificial Intelligence Technology Co., Ltd., for providing the computing platform and computational resources that supported this research.

Competing interests. The authors declare no competing interests.

References

[1] R Sapkota and M Karkee. Ultralytics yolo evolution: an overview of yolo26. *YOLO11, YOLOv8 and YOLOv5 object detectors for computer vision and pattern recognition*, 26:1–16, 2025.

[2] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16901–16911, 2024.

[3] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing aided hyper inference and fine-tuning for small object detection. In *2022 IEEE international conference on image processing (ICIP)*, pages 966–970. IEEE, 2022.

[4] Zhaopeng Gu, Bingke Zhu, Guibo Zhu, Yingying Chen, Ming Tang, and Jinqiao Wang. Anomalygpt: Detecting industrial anomalies using large vision-language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 1932–1940, 2024.

[5] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[6] 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 Computer Vision and Pattern Recognition Conference*, pages 20370–20382, 2025.

[7] Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. *arXiv preprint arXiv:2512.10696*, 2025.

[8] Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

[9] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026. *URL https://arxiv. org/abs/2602.08234*.

[10] Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

[11] Jiaze Li, Jingyang Chen, Yuxun Qu, Shijie Xu, Zhenru Lin, Junyou Zhu, Boshen Xu, Wenhui Tan, Pei Fu, Jianzhong Ju, et al. Xiaomi mimo-vl-miloco technical report. *arXiv preprint arXiv:2512.17436*, 2025.

[12] Gang Fu, Qing Zhang, Lei Zhu, Ping Li, and Chunxia Xiao. A multi-task network for joint specular highlight detection and removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7752–7761, 2021.

[13] Zhaoyangfan Huang, Kun Hu, and Xingjun Wang. M2-net: multi-stages specular highlight detection and removal in multi-scenes. *arXiv preprint arXiv:2207.09965*, 2022.

[14] Yuanhao Cai, Hao Bian, Jing Lin, Haoqian Wang, Radu Timofte, and Yulun Zhang. Retinexformer: One-stage retinex-based transformer for low-light image enhancement. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12504–12513, 2023.

[15] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1905–1914, 2021.

[16] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1844, 2021.

[17] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739, 2022.

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6): 1137–1149, 2016.

[20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[22] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[23] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[24] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

[25] Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026.

[26] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19792–19802. IEEE, 2025.

[27] 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*, pages 9592–9600, 2019.

[28] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In *European conference on computer vision*, pages 392–408. Springer, 2022.

[29] T Yang, L Chang, J Yan, J Li, Z Wang, and K Zhang. A survey on foundation-model-based industrial defect detection. arxiv 2025. *arXiv preprint arXiv:2502.19106*.

[30] Simon Baeuerle, Pratik Khanna, Nils Friederich, Angelo Jovin Yamachui Sitcheu, Damir Shakirov, Andreas Steimer, and Ralf Mikut. Are foundation models ready for industrial defect recognition? a reality check on real-world data. *arXiv preprint arXiv:2509.20479*, 2025.

[31] Yuqi Cheng, Yunkang Cao, Haiming Yao, Wei Luo, Cheng Jiang, Hui Zhang, and Weiming Shen. A comprehensive survey for real-world industrial surface defect detection: Challenges, approaches, and prospects. *Journal of Manufacturing Systems*, 84:152–172, 2026.

[32] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1780–1789, 2020.

[33] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12):4695–4708, 2012.

[34] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012.

[35] Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5148–5157, 2021.

[36] Chaofeng Chen, Jiadi Mo, Jingwen Hou, Haoning Wu, Liang Liao, Wenxiu Sun, Qiong Yan, and Weisi Lin. Topiq: A top-down approach from semantics to distortions for image quality assessment. *IEEE Transactions on Image Processing*, 33:2404–2418, 2024.

[37] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 2555–2563, 2023.

[38] Chaofeng Chen and Jiadi Mo. Iqa-pytorch: Pytorch toolbox for image quality assessment. *Github, published*, 2022.

[39] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[40] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International conference on pattern recognition*, pages 475–489. Springer, 2021.

[41] Jongheon Jeong, Yang Zou, Taewan Kim, Dongqing Zhang, Avinash Ravichandran, and Onkar Dabeer. Winclip: Zero-/few-shot anomaly classification and segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19606–19616, 2023.

[42] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10965–10975, 2022.

[43] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *European conference on computer vision*, pages 728–755. Springer, 2022.

[44] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[45] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4818–4829, 2024.

[46] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198, 2024.

[47] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[48] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[49] Qing Jiang, Xingyu Chen, Zhaoyang Zeng, Junzhi Yu, and Lei Zhang. Rex-thinker: Grounded object referring via chain-of-thought reasoning. *arXiv preprint arXiv:2506.04034*, 2025.

[50] 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. *arXiv preprint arXiv:2410.09453*, 2024.

[51] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

[52] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

[53] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[54] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

[55] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

[56] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[57] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

[58] Zhehao Zhang, Ryan A Rossi, Tong Yu, Franck Dernoncourt, Ruiyi Zhang, Jiuxiang Gu, Sungchul Kim, Xiang Chen, Zichao Wang, and Nedim Lipka. Vipact: Visual-perception enhancement via specialized vlm agent collaboration and tool-use. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36536–36546, 2026.

[59] Jiale Liu, Huan Wang, Yue Zhang, Xiaoyu Luo, Jiaxiang Hu, Zhiliang Liu, and Min Xie. Insightx agent: An lmm-based agentic framework with integrated tools for reliable x-ray ndt analysis. *IEEE Transactions on Reliability*, 2026.

[60] 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.

[61] Yongshi Ye, Hui Jiang, Feihu Jiang, Tian Lan, Yichao Du, Biao Fu, Xiaodong Shi, Qianghuai Jia, Longyue Wang, and Weihua Luo. Umem: Unified memory extraction and management framework for generalizable memory. *arXiv preprint arXiv:2602.10652*, 2026.

[62] Junwen Miao, Penghui Du, Yi Liu, Yu Wang, and Yan Wang. Agentiad: Tool-augmented single-agent for industrial anomaly detection. *arXiv preprint arXiv:2512.13671*, 2025.

A Related Work

A.1 Restoration of Degraded Images

In industrial imaging, extremely small defects often coexist with specular highlights, shadows, noise, and defocus or underexposure, which can easily overwhelm effective textures. Representative benchmarks [27, 28] and surveys [29–31] consistently reveal the domain gap between **controlled laboratory conditions** and **real production lines**. To improve pixel-level separability, **adaptive slicing and high-resolution inference** (the SAHI sliding-window overlapping-fusion paradigm) is widely adopted to mitigate global down-sampling loss for tiny instances [3], and is complementary to ROI cropping and re-magnification pipelines. For specular reflection, the path from joint highlight datasets and joint detection-removal [12] to the multi-scene multi-stage M^2-Net [13] provides a coarse-to-fine suppression trajectory. Underexposed and back-lit scenes benefit from zero-reference curve estimation [32] and one-stage Retinex-based Transformers for low-light enhancement [14]. **Super-resolution and denoising-style reconstruction** convert sensor and optical bandwidth constraints into more interpretable fine-grained textures; representative works include Real-ESRGAN [15], SwinIR [16] and Restormer [17]. **No-reference/learned quality metrics**, often used alongside PSNR/SSIM for automatic routing or comparison of pre-processing gains, include BRISQUE, NIQE [33, 34], as well as deep metrics such as MUSIQ [35], TOPIQ [36], CLIPIQA [37], together with PyTorch ecosystem toolkits that facilitate batch evaluation [38].

A.2 Localization–Reasoning Paradigm for Object Detection

Traditional industrial detection has long evolved along **specialised CNN/Transformer detection backbones**: FPN [18], Faster/Mask R-CNN [19] laid the groundwork, while DETR [20] and EfficientDet [39] further pushed the trade-off between speed and accuracy. After the advent of CLIP, engineering backbones such as PADiM [40], WinCLIP [41] and YOLO [1] moved windowed representations and end-to-end bounding-box paradigms toward production lines. **Large-scale vision–language alignment** has brought open-vocabulary detection: following CLIP [21], GLIP [42], OWL-ViT [43], Grounding DINO [5], YOLO-World [2] and others extend phrase-region binding from fixed categories to free text. **Multimodal large models** further advance the landscape with SAM [44], Florence-2 [45], InternVL [46], Qwen2-VL [47], Qwen3.5 [10], and detection-segmentation assembly pipelines such as Grounded SAM [48]—all of which can **simultaneously provide localisation hints and semantic explanations**. The **"localise candidates first, then perform discriminative reasoning on them"** has already been explicitly applied in general scenarios: for example, Rex-Thinker introduces a plan-act-summarise chain of inference and leverages box hints from open-vocabulary detection to reduce hallucinations and support rejection [49]. For **zero-shot/unified anomaly understanding in industrial contexts**, methods such as Anomaly-OV [6] and benchmarks like MMAD [50] have also appeared. Nevertheless, **deep coupling with general BBO (box-based optimisation) pipelines** and **evolvability under multi-domain strategy and non-stationary defect semantics** remain in early stages [29]; this is precisely the research space that this paper aims to extend.

A.3 LLM/MLLM Agents

The LLM agent ecosystem has systematically structured the **central–perception–action** architecture and single/multi-agent division of labour [51, 52]. Tool augmentation, trial-and-error reflection, and external environment interaction have led to paradigms such as ReAct [53], Toolformer [54], HuggingGPT [55], Generative Agents [56], and AutoGen [57]. Multi-tool agents that are close to the visual domain have also emerged, e.g., VipAct [58].**In industrial inspection and quality assurance**, recent work has gone further: InsightX Agent coordinates an LMM with dedicated X-ray NDT detectors and evidence-reflection tools [59]; AnomalyR1 explores GRPO-driven end-to-end MLLMs [60]. Although several industrial multi-agent detection frameworks already exist, most rely on hand-crafted fixed rules or static prompt templates and lack systematic distillation and reuse of successes and failures. EvoInspect is the first to introduce a dual-track experience write-back mechanism into industrial defect detection, enabling true post-deployment continual evolution.

A.4 Self-Evolving Skill Banks

The paradigm of evolving **without parameter fine-tuning** but through **contextual memory/skill recitation** has been demonstrated in Voyager [22], Reflexion [23] and ExpeL [24] via trajectory and rule distillation. More recent studies extend this direction with **multi-faceted distillation, procedural memory [7], semantic neighbourhoods and GRPO [61], reusable skill cards [8], hierarchical skill libraries [9], and skillification of memory operations [25]**, pushing toward more fine-grained **adaptive memory governance**. On the **industrial quality assurance side**, tool-aware rewards or explainable inspection trajectories have also started to be explicitly adopted [62]. Nonetheless, existing self-evolution methods still focus mainly on general text tasks or weakly-structured tool calls; no prior work has deeply coupled a self-evolving skill bank with an industrial defect detection pipeline. EvoInspect introduces a skill-bank mechanism into the industrial inspection scenario, achieving a complete closed loop that distills strategy plans from successful trajectories, attributes correction clues from failure cases, and guides subsequent reasoning using structured chain-of-thought.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:The abstract and introduction clearly state the paper's main contributions: a five-stage multi-agent framework with coarse-to-fine localization-reasoning, a dual-track Skill Bank for self-evolution, and empirical validation on four industrial hardware domains. These claims are fully supported by the experimental results presented in Sections 4 and the ablation studies.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Section 6 and in the "Efficiency and computational cost", "Cross-domain capability and transferability", and "Skill banks and human governance" paragraphs, covering system complexity, latency, domain transfer risks, memory management, and the need for human oversight.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not contain theoretical results such as theorems or lemmas; it presents an empirical study of a multi-agent framework with experimental validation.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide comprehensive experimental details in Section 4, including dataset splits, hyperparameters (learning rate, epochs, LoRA rank/alpha), hardware specifications (RTX 4090, A100), and preprocessing steps. The framework uses open-source models (Grounding DINO, Qwen3.5) with specified fine-tuning protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have released the source code for the perception, execution, and self-evolution modules in a GitHub repository. Detailed instructions for reproducing the main experiments are provided in the README file of the repository.All hyperparameters and experimental settings are fully described in Section 4 of the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup is described in Section 4, including dataset splits, evaluation protocols, and implementation details. Hyperparameters (learning rate, number of epochs, LoRA rank/alpha, optimizer) are provided in the supplementary material (Appendix). All information necessary to reproduce the results is thus fully disclosed.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: Due to computational constraints (multiple large-model fine-tuning and inference runs), we did not repeat experiments multiple times to compute error bars or statistical significance tests. However, the main conclusions are based on clear performance margins across multiple domains and baselines, which are consistent and substantial enough to be reliable without formal significance testing.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 (Experimental Setup) specifies the hardware used.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper does not involve human subjects, crowdsourcing, or personally identifiable data. It uses only publicly available industrial defect datasets (bearing, PCB, solar EL, magnetic tile) and open-source models. The work has no direct societal harm, dual-use risk, or potential for discrimination, and it does not aim to increase lethality of weapons systems. Therefore, it fully adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The positive societal impact is improving industrial quality control efficiency and reducing manual inspection errors. The negative impact is briefly mentioned in the limitations: the system relies on human oversight to prevent misuse, with no direct malicious use or serious societal risk.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release any models or data with a high risk of misuse. All models used are publicly available open-source models, and all datasets are publicly available industrial defect detection datasets, involving no web-scraped or sensitive content.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: While we do not list the license names individually in the paper, we have complied with their non-commercial research use terms. For further details, please refer to the official release pages of each asset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces new source code for the proposed framework (EvoInspect), which is provided in an anonymized GitHub repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor any research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or any research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: This paper uses multimodal large language models for reasoning as part of its core methodology, and the description of the experiments follows the LLM policy.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.