SYNTHETIC MULTIMODAL IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the central questions in multimodal in-context learning concerns how to construct effective multimodal contexts that provide appropriate task and reasoning guidance for multimodal large language models (MLLMs). Existing methods largely build such contexts by selecting demonstrations from a predefined corpus, limiting their ability to provide task- and reasoning-aligned context when suitable examples are unavailable. To overcome this limitation, we introduce synthetic multimodal in-context learning, a paradigm that shifts multimodal context construction from fixed-corpus demonstration selection toward query-adaptive demonstration synthesis at inference time. To instantiate it, we propose SynMIC, a self-evolving agentic framework that synthesizes demonstrations aligned with the task and reasoning schema required for in-context inference on each target query. Specifically, it abstracts the target query into a synthesis specification capturing its content, task, and reasoning schemas, and uses the specification to plan diverse demonstrations by specifying new visual scenarios, queries, pre-committed answers, and answer-supporting visual constraints. These plans are realized through tool-mediated synthesis by adaptively assembling and executing toolchains over code-based rendering, model-based image generation/editing, and auxiliary perception. Meanwhile, a verification-guided self-evolution loop verifies synthesized demonstrations, resynthesizes invalid cases, and distills failure experience into an evolving skill that assists subsequent synthesis. Experiments across diverse benchmarks and MLLMs demonstrate consistent improvements over existing baselines, highlighting the promise of shifting multimodal context construction from corpus-bounded retrieval toward adaptive synthesis. Code will be publicly available.

1 INTRODUCTION

Multimodal large language models (MLLMs) can adapt to unseen tasks at inference time through multimodal in-context learning (MM-ICL), conditioning on a few image-text demonstrations without updating model parameters (Alayrac et al., 2022). These demonstrations act as implicit task specifications: they convey task intent, ground language instructions in visual evidence, and suggest the reasoning pattern the model should follow (Xie et al., 2021; Xu et al., 2025; Wang & Li, 2026). Therefore, MM-ICL performance depends not only on model capability, but also on whether the constructed context provides appropriate task semantics and reasoning cues (Chen et al., 2025d). To construct such contexts, existing approaches typically select demonstrations from a predefined corpus, using random sampling, similarity-based retrieval, or agentic strategies that iteratively refine the candidate examples (Zong et al., 2025; Chen et al., 2025c; Fu et al., 2026). Despite yielding promising improvements in downstream performance, these approaches still operate under a fixed-corpus limitation: the context can be assembled only from demonstrations available in the corpus.

This limitation could not be fully addressed by improving demonstration selection alone (Xiong et al., 2026; Wang et al., 2025; Lee et al., 2026). When the corpus lacks demonstrations sufficiently aligned with the target query in task format or reasoning requirements, even sophisticated strategies, such as agentic selection, organization, or refinement, cannot construct a sufficiently informative context (Fu et al., 2026). As illustrated in Figure 1, for example, the target query requires assessing bicycle security from visible fastening cues, while the most relevant available demonstrations may instead concern component states or orientations, which provide insufficient contextual guidance for answering the query. The bottleneck therefore lies not merely in how demonstrations are selected and curated, but more fundamentally in whether the demonstration space contains sufficiently well-aligned

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

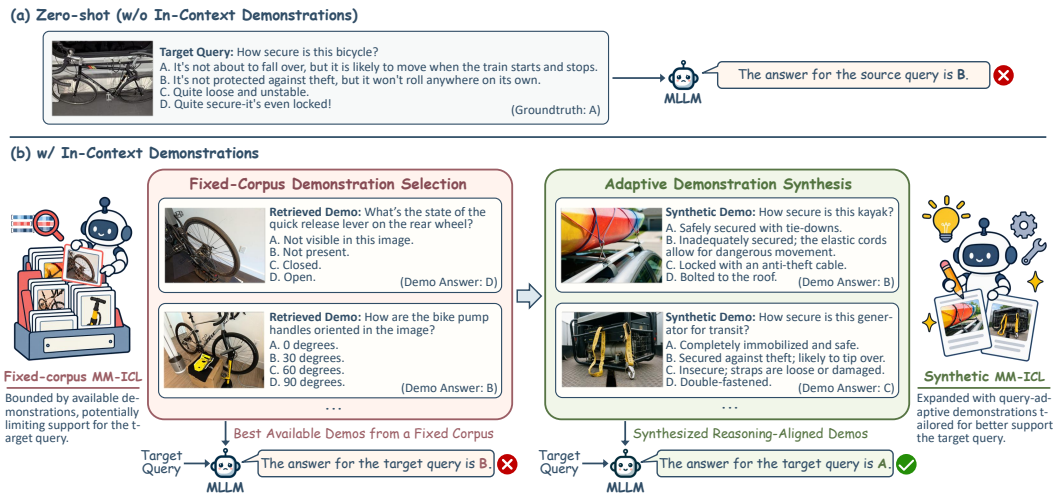

Figure 1: **Motivation for synthetic multimodal in-context learning.** When a predefined corpus lacks reasoning-aligned demonstrations, even the most relevant available examples may provide insufficient contextual guidance. Active demonstration synthesis could overcome this corpus-bounded limitation by constructing query-adaptive demonstrations.

examples in the first place. This naturally raises a question: *How can MM-ICL move beyond selecting from a fixed corpus toward operating over a more adaptive multimodal demonstration space?*

Driven by this question, we introduce *synthetic multimodal in-context learning*, a paradigm that shifts general-purpose multimodal context construction from fixed-corpus demonstration selection toward query-adaptive demonstration synthesis at inference time. However, this paradigm also faces a challenge that is salient in MM-ICL: prior studies (Li, 2025; Li et al., 2025b; Wang et al., 2026) have suggested that the effectiveness of MM-ICL is sensitive to the reasoning patterns elicited by the provided demonstrations. Beyond the general data-synthesis requirement of generating visually and semantically valid samples, synthetic demonstrations should therefore further ensure that the constructed context induces a reasoning schema aligned with the target query and task. To instantiate this paradigm while satisfying such reasoning requirement, we propose **SynMIC**, a self-evolving agentic framework for **Syn**thetic **M**ultimodal **I**n-**C**ontext Learning. Rather than generating demonstrations that are merely visually plausible or semantically related to the target, SynMIC seeks to preserve the underlying task and reasoning schema of the target multimodal query.

Specifically, SynMIC first abstracts the target multimodal query into a synthesis specification spanning its content, task, and reasoning schemas. Based on this specification, it constructs diverse demonstration plans, each specifying a new scenario with a query, a pre-committed answer, and related visual constraints. Guided by these plans, SynMIC synthesizes the demonstrations by assembling toolchains over code-based rendering, model-based image generation/editing, and auxiliary perception. The resulting demonstrations are then verified against their plans and the target schemas, retaining valid demonstrations while using verification feedback to refine invalid ones. Moreover, SynMIC distills failure-derived experience into an evolving synthesis skill that assists subsequent demonstration planning and tool execution across cases. This planning, synthesis, verification, and self-evolution agent loop constructs query-adaptive contexts that are diverse in scenario while remaining faithful to the target task and reasoning schema. Our contributions are summarized as follows:

- We propose SynMIC, to our knowledge the first agentic framework that instantiates synthetic multimodal in-context learning by actively synthesizing query-adaptive multimodal demonstrations at inference time, moving MM-ICL beyond fixed-corpus demonstration selection.
- SynMIC develops a self-evolving, reasoning-faithful synthesis framework that explicitly models the target's semantic, task, and reasoning structure to guide demonstration planning and active synthesis, and distills verification feedback from failed attempts into an evolving synthesis skill that enables cross-case experience reuse, thereby improving overall synthesis quality and efficiency.
- Extensive experiments demonstrate that SynMIC consistently outperforms state-of-the-art baselines across diverse benchmarks and MLLMs, highlighting the effectiveness and broader promise of synthetic context construction beyond fixed-corpus selection for downstream MM-ICL.

2 Related Work

Multimodal In-Context Learning (MM-ICL). MM-ICL refers to the ability of MLLMs to adapt to new multimodal tasks from a few in-context demonstrations without parameter updates. Since its emergence in models such as Flamingo (Alayrac et al., 2022) and KOSMOS-1 (Huang et al., 2023), it has become a core capability in contemporary MLLMs (Awadalla et al., 2023; Zhao et al., 2024; Li et al., 2025a). A widely recognized theoretical perspective explains ICL through implicit optimization in Transformers, where attention over in-context examples induces hidden-state updates analogous to gradient descent (Akyürek et al., 2023; Von Oswald et al., 2023; Cheng et al., 2024). This view casts demonstrations as inference-time optimization signals, making their design crucial to MM-ICL performance and motivating a growing body of research on constructing more effective contexts. Early approaches mainly rely on in-domain random sampling (Jiang et al., 2024; Zong et al., 2025) or retrieval-based demonstration selection (Chen et al., 2025c; Luo et al., 2024; Yi-Ge et al., 2025). Recent agentic methods (Fu et al., 2026; Wang et al., 2026) further improve context quality by iteratively searching, reorganizing, and refining demonstrations. Nevertheless, as discussed in Sec. 1, existing MM-ICL methods remain constrained by the fixed corpus, motivating SynMIC to shift context construction from bounded demo-corpus organization toward open-ended synthesis.

Synthetic Multimodal Data for Optimizing MLLMs. Recent progress in MLLM training suggests that synthetic multimodal data can provide effective supervision signals for model optimization (Chen et al., 2024a; Yu et al., 2025). In earlier stages, multimodal data synthesis mainly focuses on generating task-desired linguistic supervision from images, such as dense captions and visual question-answering pairs (Sugiura et al., 2026; Zhang et al., 2025; Zhou et al., 2025). With the rapid advancement of generative models, another line synthesizes controllable images from textual instructions using text-to-image (T2I) models for alignment and generation training (Chen et al., 2025b; Ye et al., 2025). Meanwhile, code-based synthesis can efficiently render structured visual scenes, such as charts, diagrams, and other visual content, with precise textual information (Liu et al., 2024; Zhang et al., 2024; Yang et al., 2025; Chen et al., 2025e). Beyond these specific pipelines, frontier MLLMs such as the Qwen and InternVL series also systematically introduce large-scale synthetic multimodal supervision to strengthen the training recipes (Bai et al., 2025; Team et al., 2026b; Chen et al., 2024c). Together, these practices indicate that synthetic multimodal data can serve as effective supervision for optimizing MLLMs, lending empirical grounding to the idea that synthesized examples may also serve as in-context optimization signals under the implicit-gradient view of MM-ICL.

3 SynMIC: Synthetic Multimodal In-Context Learning

SynMIC is an agentic MM-ICL framework that goes beyond retrieving demonstrations from a fixed example pool by *adaptively synthesizing multimodal in-context demonstrations* for a target multimodal query. Starting from this query, its central objective is to construct a set of synthetic demonstrations that preserve the underlying task and reasoning structure needed for in-context inference, where each demonstration comprises a synthesized image, a query, and a pre-committed answer. As illustrated in Figure 2, SynMIC realizes this objective through three components: (1) **Synthetic Demonstration Planning** abstracts the target query into a synthesis specification capturing its content, task, and reasoning structure, and uses this specification to plan a diverse set of demonstration candidates; (2) **Adaptive Tool-mediated Synthesis** operationalizes each demonstration plan through adaptive tool execution, constructing an appropriate toolchain and corresponding tool-compatible inputs to realize the planned demonstration; (3) **Verification & Self-evolution** determine whether each synthesized demonstration conforms to its plan and remains consistent with the target task and reasoning schemas. Feedback and associated context from invalid cases are distilled into an evolving synthesis skill. This skill guides resynthesis of invalid cases and transfers reusable experience to subsequent demonstration planning and tool execution across cases. Together, these components form a closed loop of planning, synthesis, verification, and self-evolution that constructs the available demonstration space on demand while maintaining task and reasoning faithfulness.

3.1 Synthetic Demonstration Planning

This module plans synthetic demonstrations for a target multimodal query. It first analyzes what task the query requires and how to reason from the relevant evidence to the answer. It then uses this

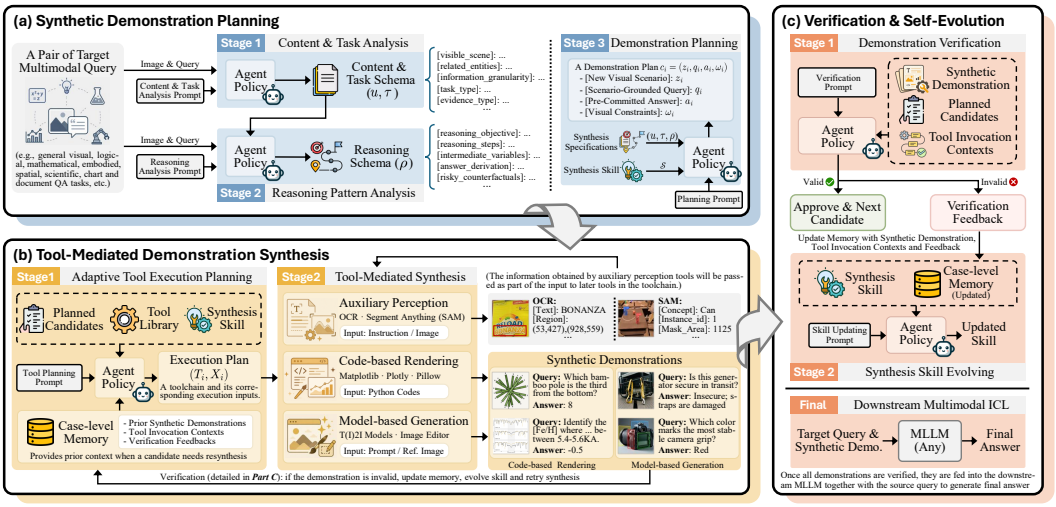

Figure 2: **Overview of SynMIC.** The framework consists of three components: **(a) Synthetic demonstration planning**, which derives a synthesis specification spanning content, task, and reasoning levels from the target query to plan diverse, task- and reasoning-faithful demonstration candidates; **(b) adaptive tool-mediated synthesis**, which agentically assembles toolchains and tool-compatible inputs to realize the plans as multimodal demonstrations; and **(c) verification & self-evolution**, which verifies synthesized demonstrations, resynthesizes invalid cases, and distills failure feedback into an evolving synthesis skill for cross-case reuse. The verified demonstrations then form a query-adaptive context for downstream multimodal in-context inference.

analysis to design new scenarios, queries, answers, and the visual evidence needed to support them. The resulting specifications guide the subsequent tool-mediated synthesis.

Content and Task Analysis. Given a target multimodal query x, SynMIC uses a content and task analysis prompt P_{task} to derive a task schema (u, τ):

$$(u, \tau) = \pi_\theta(x; P_{\text{task}}), \tag{1}$$

where the prompt guides the MLLM policy π_θ to convert the query into a content-grounded representation u and a structured task representation τ. Specifically, u describes the content instantiated in the image, covering the scene, entities, information granularity, salient visual cues, etc. Complementarily, τ specifies the underlying task structure, including the task type, evidence type, answer space, etc.

Reasoning Pattern Analysis. SynMIC further uses a reasoning pattern analysis prompt P_{r} to infer a plausible and generalizable reasoning schema ρ from the target query and its derived task schema, with the latter grounding the analysis in the underlying task structure:

$$\rho = \pi_\theta(x, u, \tau; P_{\text{r}}). \tag{2}$$

The schema ρ captures the reasoning objective, potential key steps and intermediate variables, the expected answer derivation, and counterfactual variations that may affect reasoning consistency across the task space, etc. We define $\xi = (u, \tau, \rho)$ as the synthesis specification, which characterizes the target query at the content, task, and reasoning levels. Rather than prescribing a particular surface realization, it provides structured guidance on the query-relevant evidence, underlying task, and reasoning process that subsequent demonstrations should instantiate.

Demonstration Planning. Based on the synthesis specification ξ and a planning prompt P_{plan}, SynMIC plans a set of synthetic demonstrations. To prevent the demonstrations from overfitting to the target query, P_{plan} imposes set-level diversity constraints on their scenarios, queries and answers. For each demonstration plan c_i, it specifies a new visual scenario z_i, a scenario-grounded query q_i, a pre-committed answer a_i, and the visual constraints ω_i required to support that answer:

$$\mathcal{C} = \{c_i\}_{i=1}^{K} = \pi_\theta(\xi; P_{\text{plan}}, \mathcal{S}), \qquad c_i = (z_i, q_i, a_i, \omega_i). \tag{3}$$

This process is also assisted with a self-evolving synthesis skill $\mathcal{S}$, which distills failure feedback into reusable cross-case experience to improve synthesis quality and efficiency, as detailed in Sec. 3.3. The resulting demonstration plans can be diverse in scenarios, visual content, query formulations, and answers while faithfully preserving the task and reasoning structure of the target query.

3.2 Adaptive Tool-Mediated Synthesis

This module operationalizes demonstration plans through agentic tool execution. Given a demonstration plan, the policy adaptively constructs an appropriate toolchain, formulates tool-compatible inputs, and executes related tools to generate the required image(s).

Synthesis Tool Library. SynMIC is equipped with a synthesis tool library $\mathcal{T}$ that contains tools for different forms of visual controllability, including code-based rendering, text-to-image generation, and image editing, as well as auxiliary perception like OCR and concept segmentation. Each tool takes a policy-generated input, such as coding requirement, generation prompt, or perception instruction, and produces either a synthesized image or auxiliary information that facilitates other synthesis steps.

Adaptive Tool Execution. Given the tool library $\mathcal{T}$ and a demonstration plan c_i, the policy is guided by a tool planning prompt P_{tool} and the synthesis skill $\mathcal{S}$ to adaptively assemble an execution plan:

$$(T_i, X_i) = \pi_\theta(c_i, \mathcal{M}_i, \mathcal{T}; P_{\text{tool}}, \mathcal{S}). \tag{4}$$

Here, $T_i = (T_{i,1}, T_{i,2}, \ldots, T_{i,L_i})$ denotes an ordered toolchain, where each $T_{i,k} \in \mathcal{T}$ and $L_i \geq 1$. The corresponding input $X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,L_i})$ translates the planned scenario, query, pre-committed answer, and visual constraints in c_i into stage-wise, tool-compatible inputs. When multiple tools are involved, X_i may include predefined slots for intermediate auxiliary evidence. During execution, outputs from perception tools, such as segmentation masks or recognized text and its region, could be filled into these slots to condition subsequent synthesis steps without refining T_i or X_i. Executing T_i with X_i produces an image $I_i = T_i(X_i)$ and yields a synthetic demonstration $d_i = (I_i, q_i, a_i)$ for subsequent verification, as detailed in Sec. 3.3. When resynthesis is required following verification, the case-level working memory $\mathcal{M}_i$ records previous demonstrations, toolchains, inputs, and verification feedback, providing context for refining the execution plan.

3.3 Verification and Self-Evolution

This module verifies whether each generated demonstration aligns with the planned content, supports the pre-committed answer, and maintains task and reasoning consistency. For invalid cases, it distills verification feedback and the associated context into the evolving synthesis skill $\mathcal{S}$, supporting resynthesis for the invalid cases while transferring reusable experience to the planning of demonstrations and tool execution across subsequent cases, thereby enabling continual self-evolution.

Verification. Given a synthesized demonstration at synthesis round $t \geq 1$, the working memory is updated as $\mathcal{M}_i^{(t)} = \mathcal{M}_i^{(t-1)} \cup \{d_i^{(t)}, T_i^{(t)}, X_i^{(t)}\}$, where $\mathcal{M}_i^{(0)} = \varnothing$. Based on the demonstration plan c_i, task specification τ and reasoning schema ρ, the policy evaluates the synthesized demonstration:

$$v_i^{(t)} = \pi_\theta(\mathcal{M}_i^{(t)}, c_i^{(t)}, \tau, \rho; P_{\text{ver}}), \tag{5}$$

where v_i contains a validity decision and diagnostic feedback on alignment with the demonstration plan and target query's task and reasoning schema, together with failure analysis for invalid cases. The result is then stored in the working memory $\mathcal{M}_i^{(t)} = \mathcal{M}_i^{(t)} \cup v_i{}^{(t)}$. Valid demonstrations are retained, whereas the feedback of invalid ones is used for skill-updating and resynthesis. If tool execution fails before verification stage, $v_i^{(t)}$ is instead represented by the execution-error log.

Self-Evolution and Resynthesis. For an invalid demonstration, SynMIC uses a skill-evolution prompt P_{skill} to distill verification feedback into reusable cross-case guidance, thereby updating the synthesis skill and enabling failure-driven self-evolution:

$$\mathcal{S}^{(t+1)} = \pi_\theta(\mathcal{S}^{(t)}, \mathcal{M}_i^{(t)}; P_{\text{skill}}). \tag{6}$$

Conditioned on the updated skill, SynMIC then refines the failed synthesis process:

$$\left(c_i^{(t+1)}, T_i^{(t+1)}, X_i^{(t+1)}\right) = \pi_\theta\left(c_i^{(t)}, \mathcal{M}_i^{(t)}, \mathcal{T}; P_{\text{refine}}, \mathcal{S}^{(t+1)}\right). \tag{7}$$

The refinement is selective rather than exhaustive: depending on the identified failure, the policy updates only the necessary subset of $\{c_i^{(t)}, T_i^{(t)}, X_i^{(t)}\}$. For failures that can be resolved through tool-level correction, the demonstration plan remains unchanged, while the toolchain and its compatible input are revised for resynthesis. If repeated failures indicate that the plan itself is infeasible, the policy

may additionally revise its scenario, query, pre-committed answer, or visual constraints. Together, the refinement process supports case-specific correction, while the evolving skill transfers failure-derived experience across cases to improve subsequent demonstration planning and tool execution.

Downstream Multimodal In-Context Learning. Once all K synthetic demonstrations are obtained, they are provided together with the original target query x to an arbitrary downstream MLLM f to produce the final prediction $\hat{y} = f\left(\{d_i\}_{i=1}^{K}, x\right)$. In this way, SynMIC uses synthesized demonstrations to construct query-adaptive contexts for downstream multimodal in-context inference.

4 Experiment

4.1 Dataset and Implementation

Dataset. We first annotate the difficulty of queries from recent multimodal benchmarks commonly used to evaluate frontier MLLMs, including CharXiv (Wang et al., 2024b), MathVision (Wang et al., 2024a), MMStar (Chen et al., 2024b), and ERQA (Gemini et al., 2025). Following the evaluation protocol of ContextNav (Fu et al., 2026), we determine query difficulty based on the performance of the evaluated models: a query is labeled as easy if more than half of the models answer it correctly, and as hard otherwise. We then randomly sample queries at an easy-to-hard ratio of 3:7, resulting in 150 test instances from each of CharXiv, MathVision and MMStar, and 100 from ERQA due to its smaller pool of available queries. The remaining instances are used as the demonstration pool for other baselines. We also include single-task datasets such as CLEVR (Johnson et al., 2017) and TextOCR (Singh et al., 2021), adopting the test and support splits specified in VL-ICL Bench (Zong et al., 2025), where each test split contains 200 instances. Collectively, these benchmarks, comprising 950 instances in total, span diverse multimodal tasks (Chen et al., 2025a), including general visual perception, chart and diagram understanding, mathematical problem solving, spatial and embodied reasoning, counting, attribute recognition, and text recognition.

Implementation. We use Gemini-3-Flash (Comanici et al., 2025) as the default MLLM policy, which also serves as the backbone for code-based rendering. We deploy all open-source models, including the MLLMs Gemma-4-26B-A4B (Team et al., 2026a) and Qwen3.6-35B-A3B (Qwen, 2026), on H100 80GB GPUs. For tool-mediated synthesis, we use Qwen-Image and Qwen-Image-Edit (Wu et al., 2025) as the image-generation and editing backbones, respectively, along with HunyuanOCR-1.5 (Li et al., 2026) and Segment Anything Model 3 (Carion et al., 2026) for auxiliary perception. Closed-source models, including the Gemini series (Comanici et al., 2025) and GPT-5 (Singh et al., 2025), are accessed via APIs. Unless otherwise specified, we report experimental results as the median over five trials under the standard 8-shot setting used in ContextNav, and we conduct ablation studies on MMStar with GPT-5. Appendix E introduces the initialization of self-evolving skills, Appendix F documents the prompts used in the agentic system and experiments, while Appendix G defines the tool library. In addition, Appendix H presents studies of both successful and failed cases.

4.2 Comparison against baselines

We evaluate SynMIC against zero-shot inference and representative fixed-corpus multimodal context construction methods, including ManyICL (Jiang et al., 2024), MMICES (Chen et al., 2025c), and ContextNav (Fu et al., 2026). Following its standard configuration, ManyICL randomly draws demonstrations from the candidate pool until the available context window is exhausted. For MMICES and ContextNav, we reproduce their methods using the standard configurations reported in their respective papers, with MMICES using 32 shots and ContextNav using 8 shots. For a fair comparison with SynMIC, we further update the agent backbone of ContextNav to Gemini-3-Flash.

As summarized in Table 1, SynMIC delivers consistently competitive or superior performance across almost all evaluated benchmarks and MLLMs. Its advantage is particularly pronounced on CharXiv, MathVision, ERQA and MMStar, where the tasks exhibit substantial heterogeneity in problem structures and formats. For these benchmarks, the demonstration pool may provide relatively few examples closely aligned with the structure and reasoning requirements of an individual query. Under such conditions, SynMIC yields substantial gains. Moreover, its effectiveness is not limited to scenarios where highly matched context examples are scarce. SynMIC also performs strongly on CLEVR and TextOCR from VL-ICL Bench, where the task distributions are more concentrated and the demonstration pools contain more structurally homogeneous examples. This suggests that

Table 1: **Comparison of downstream accuracies across MLLMs and benchmarks.** Higher values indicate better performance. The bold numbers represent the best results within each block.

Methods	CharXiv	MathVision	MMStar	ERQA	CLEVR	TextOCR	Average
Gemma-4-26B-A4B							
Zero-Shot	0.293	0.340	0.307	0.330	0.670	0.475	0.403
Random	0.247	0.233	0.273	0.240	0.645	0.460	0.350
ManyICL	0.307	0.347	0.320	0.310	0.695	0.485	0.411
MMICES	0.320	0.340	0.327	0.320	0.680	0.475	0.410
ContextNav	0.340	0.360	0.347	0.340	0.700	0.495	0.430
SynMIC	**0.373**	**0.400**	**0.367**	**0.380**	**0.705**	**0.505**	**0.455**
Gemini-2.5-flash							
Zero-Shot	0.300	0.360	0.327	0.310	0.840	0.465	0.434
Random	0.260	0.273	0.300	0.230	0.820	0.450	0.389
ManyICL	0.320	0.380	0.333	0.320	0.855	0.470	0.446
MMICES	0.327	0.353	0.347	0.290	0.845	0.455	0.436
ContextNav	0.347	0.400	0.353	0.340	0.855	0.480	0.463
SynMIC	**0.380**	**0.440**	**0.373**	**0.370**	**0.865**	**0.485**	**0.486**
Qwen3.6-35B-A3B							
Zero-Shot	0.493	0.447	0.407	0.360	0.680	0.410	0.466
Random	0.447	0.387	0.367	0.300	0.650	0.400	0.425
ManyICL	0.507	0.460	0.427	0.380	0.700	0.415	0.482
MMICES	0.507	0.447	0.420	0.350	0.685	0.410	0.470
ContextNav	0.527	0.480	0.440	0.370	0.710	0.430	0.493
SynMIC	**0.567**	**0.527**	**0.467**	**0.410**	**0.715**	**0.435**	**0.520**
GPT-5							
Zero-Shot	0.460	0.480	0.387	0.410	0.795	0.655	0.531
Random	0.427	0.447	0.380	0.320	0.805	0.640	0.503
ManyICL	0.487	0.493	0.413	0.410	0.825	0.660	0.548
MMICES	0.493	0.487	0.413	0.400	0.810	0.645	0.541
ContextNav	0.507	0.520	0.427	0.430	**0.835**	0.670	0.565
SynMIC	**0.540**	**0.553**	**0.460**	**0.470**	**0.835**	**0.680**	**0.590**
Gemini-3.5-flash							
Zero-Shot	0.513	0.500	0.427	0.400	0.945	0.485	0.545
Random	0.460	0.420	0.393	0.330	0.930	0.470	0.501
ManyICL	0.533	0.520	0.440	0.410	**0.965**	0.495	0.561
MMICES	0.527	0.513	0.440	0.410	0.950	0.490	0.555
ContextNav	0.547	0.533	0.453	0.420	0.960	0.500	0.569
SynMIC	**0.580**	**0.567**	**0.480**	**0.450**	**0.965**	**0.510**	**0.592**

even when closely matched demonstrations are abundant in the existing pool, synthesizing examples explicitly aligned with the reasoning demands of each query can further strengthen downstream MM-ICL performance. Overall, SynMIC achieves an average ICL gain of 11.2% across models and 12.5% across datasets, substantially exceeding the previous SOTA (6.0% and 6.4%, respectively). Together, these results demonstrate the effectiveness of SynMIC in improving downstream MM-ICL across a broad range of task settings. Appendix A further compares SynMIC and the baselines with different numbers of in-context demonstrations.

4.3 Ablations

We conduct ablation studies to examine the contributions of SynMIC's functional components in Table 2 and its scaling behavior with increasing context size in Figure 3. Appendix B further investigates the sources of ICL gains by examining the contribution of multimodal demonstration content, controlling for additional agentic reasoning under matched inference budgets, and comparing structured task- and reasoning-guided synthesis with straightforward demonstration generation.

Demonstration Planning. The results show that effective MM-ICL demonstrations rely on both task and reasoning fidelity. The task schema mainly constrains what problem a demonstration should instantiate, whereas the reasoning schema specifies how its answer should be derived. Although distinct, the two schemas partially overlap. The task schema implicitly encodes a coarse reasoning structure, while the reasoning schema also reflects the underlying task objective. Consequently, removing either explicit constraint does not preclude effective synthesis, but weakens the corresponding guidance and yields suboptimal demonstrations. Since the reasoning schema is inferred conditioned on the task schema, the "w/o Task Schema" variant still constructs the task schema for reasoning-schema induction, but does not use it as an explicit constraint in any subsequent stage.

Iterative Refinement and Self-Evolution. Memory, the evolving skill, and verification play complementary roles in case-level refinement and agent self-evolution. Removing memory discards the synthesis trajectory of current case, making resynthesis less targeted and less efficient. It also weakens

Table 2: **Ablation studies on the functional components of SynMIC.** 'Gain' denotes the relative improvement in downstream accuracy over zero-shot inference. Valid Demonstration Rate (VDR) is the fraction of synthesis attempts retained as ICL demonstrations. Reasoning Faithfulness Score (RFS) and Task Faithfulness Score (TFS) measure faithfulness to the source reasoning structure and task, respectively, while Diversity measures how well synthesized demonstrations avoid overfitting to the source query's surface patterns. RFS, TFS, and Diversity are average GPT-5.6 Sol ratings on a five-point scale, using the evaluation prompts in Appendix F. '-' denotes not applicable.

Method	Gain (%) ↑	RFS ↑	TFS ↑	VDR ↑	Diversity
Full (default)	19.0	4.038	4.137	0.906	3.852
Demonstration Planning					
w/o Reasoning Schema	10.3	3.298	3.847	0.940	3.665
w/o Task Schema	5.2	3.612	3.162	0.929	3.885
w/o Diversity Constraint	12.1	4.085	4.166	0.922	2.671
Iterative Refinement and Self-Evolution					
w/o Case-level Memory	13.8	3.794	3.892	0.829	3.597
w/o Self-evolving Skill	10.3	3.558	3.738	0.842	3.462
w/o Verification	3.4	3.236	3.398	-	4.067
MLLM Policy					
Qwen-3.5-9B	12.1	3.676	3.803	0.883	3.538
Qwen3.6-35B-A3B	17.2	3.958	4.160	0.901	3.826
GPT-5.5	19.0	4.069	4.143	0.915	3.901
Model-based Generation Backbone					
SD3-Medium	6.9	3.442	3.616	0.833	3.567
FLUX.2-klein-4B	15.5	3.890	3.978	0.888	3.826
Code-based Rendering Backbone					
Qwen-3.5-9B	13.8	3.783	3.904	0.785	3.803
GPT-5.5	22.4	4.159	4.251	0.922	3.837
Auxiliary Perception					
w/o OCR	17.2	3.986	4.051	0.884	3.891
w/o SAM	19.0	4.017	4.106	0.889	3.872

the context available for skill evolution, reducing the quality of reusable guidance for subsequent planning and synthesis. Removing the self-evolving skill preserves case-specific refinement but prevents failure experience from transferring across cases, degrading subsequent planning and synthesis. As a result, previously encountered failure modes may recur in new cases, requiring additional refinement rounds and reducing synthesis efficiency. Removing verification causes the largest degradation, as it eliminates both the quality-control signal for resynthesis and feedback required for refinement and skill evolution, reducing iterative correction toward one-shot synthesis and disabling self-evolution. Overall, memory supports case-specific refinement, the evolving skill enables cross-case experience transfer, and verification sustains both through feedback and quality control.

MLLM Policy. Scaling the policy backbone generally improves reasoning and task faithfulness, but with diminishing returns as model capacity increases. This suggests that relatively lightweight MLLMs already capture much of the task and reasoning structure needed to guide demonstration synthesis. Given these diminishing returns, we adopt Gemini-3-Flash as the default policy backbone.

Generation and Coding Backbone. Stronger synthesis backbones generally produce more task- and reasoning-faithful demonstrations, leading to better downstream gains. They also require fewer refinement rounds to obtain valid demonstrations, while weaker backbones can still recover effective ones through multi-round resynthesis. These results underscore the importance of synthesis quality while showing that verification and self-evolution can compensate for limited synthesis capability.

Auxiliary Perception. Auxiliary perception tools provide more precise grounded visual cues, increasing the likelihood of valid demonstrations. Their impact on downstream ICL performance remains limited, however, because verification-guided resynthesis and self-evolution can correct many synthesis failures. Thus, auxiliary perception mainly improves synthesis efficiency and robustness by reducing failed attempts and refinement overhead, rather than being essential to final ICL gains.

Scaling. Figure 3 shows that SynMIC achieves smooth gains as context size increases, whereas removing the reasoning schema causes greater fluctuations, especially in the low-shot regime. Under the implicit-optimization view of ICL, this suggests that reasoning-consistent demonstrations provide more compatible update signals, allowing the induced optimization process to accumulate more stably. By contrast, fixed-corpus methods are constrained by the coverage of the available demonstration space. Once highly aligned demonstrations are exhausted, additional demonstrations may become less

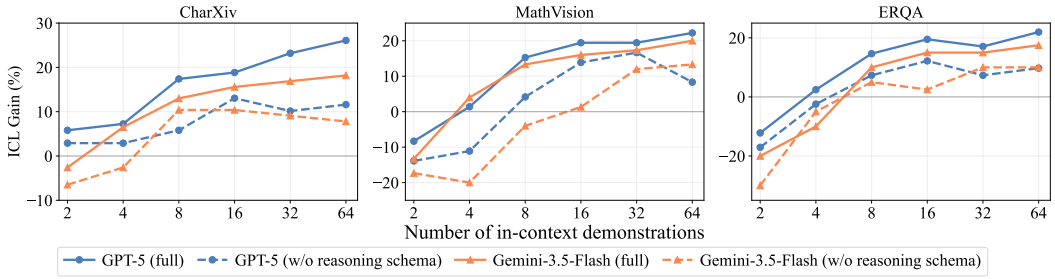

Figure 3: **Scaling with context size.** We compare SynMIC and its variant without the reasoning schema across downstream models to assess context-size scaling and reasoning-schema effects.

Table 3: **Effect of corpus augmentation via synthetic demonstrations on retrieval-based MM-ICL** 'Util. Rate' denotes the proportion of synthetic demonstrations among the retrieved examples utilized for in-context inference, with higher values indicating greater reliance on synthetic data. 'Gain' denotes the relative improvement in accuracy over zero-shot inference. '-' denotes not applicable.

Methods	Corpus	CharXiv		MathVision		ERQA	
		Util. Rate (%)	Gain (%) ↑	Util. Rate (%)	Gain (%) ↑	Util. Rate (%)	Gain (%) ↑
ManyICL	Original	-	5.8	-	2.8	-	0
	+ Augmentation	59.9	8.7	26.6	4.2	76.1	2.4
ContextNav	Original	-	10.1	-	8.3	-	4.9
	+ Augmentation	70.7	14.5	34.3	11.1	85.9	9.8

aligned, leading to earlier saturation or less reliable gains. Still, the gains diminish with context size, indicating decreasing marginal utility once the relevant reasoning structure is sufficiently established.

4.4 Discussion

Synthetic Demonstrations as Corpus Augmentation. To further examine the fixed-corpus limitation, we augment the demonstration pools of retrieval-based baselines with SynMIC demonstrations while keeping their retrieval strategies unchanged. As shown in Table 3, this augmentation consistently improves downstream performance, while the added synthetic demonstrations are actively selected by the retrievers (see the high util. rates). These gains and selection patterns further suggest that fixed-corpus MM-ICL is constrained by the coverage of its available demonstration space.

Limitation. As shown in Table 2, SynMIC remains sensitive to synthesis-backbone capability: weaker image-generation or code-rendering models can reduce demonstration faithfulness and require more refinement rounds. Nevertheless, they still yield positive downstream gains, suggesting that verification-guided resynthesis can partially compensate for limited synthesis capability. The synthesis also incurs non-negligible computational overhead from tool invocation, verification, and iterative refinement, as reported in Appendix C. As shown in Table 3 and Table 9, this overhead can be partially amortized by retaining verified demonstrations as reusable context resources, while improving synthesis efficiency remains an important direction for future work.

Further Discussion. Appendix D provides additional analyses that disentangle synthetic ICL gains from capability differences between the policy and downstream MLLM, examine how verified synthetic demonstrations can be accumulated and reused to amortize synthesis overhead on unseen queries, and validate the reliability of our LLM-as-a-judge evaluation against human judgments.

5 Conclusion

We present SynMIC, a self-evolving agentic framework for synthetic multimodal in-context learning. SynMIC treats the target multimodal query's task semantics and reasoning structure as explicit constraints for demonstration planning. It further combines tool-mediated synthesis with verification-guided correction and synthesis skill evolution to generate effective multimodal contexts. Across diverse benchmarks and MLLMs, SynMIC consistently achieves competitive performance against SOTA baselines. Moreover, augmenting predefined corpus with SynMIC-generated demonstrations also benefits retrieval-based baselines. Together, these results highlight the promise of constructing query-adaptive demonstrations at inference time to advance MM-ICL beyond fixed-corpus selection.

AI USE STATEMENT

In this work, we used generative AI tools to assist with coding, including code completion, debugging, and code editing, and to polish the manuscript for grammar, clarity, and readability.

ETHICS STATEMENT

To the best of our knowledge, this work does not raise any ethical concerns.

REPRODUCIBILITY STATEMENT

We provide detailed experimental configurations and implementation settings in Sec. 4.1, including the evaluated models, synthesis backbones, auxiliary tools, inference settings, and evaluation protocol. Additional details on the initialization of the synthesis process are provided in Appendix E. To facilitate faithful reproduction of the agentic pipeline, Appendix F includes the complete prompts used for demonstration planning, tool execution, verification, self-evolution, and evaluation, while Appendix G specifies the synthesis tool library, including the functionality, input/output interfaces, and invocation requirements of each tool. Together, these materials document the key components required to reproduce SynMIC and the reported experiments. Our code will be publicly available to further support reproducibility.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *The Eleventh International Conference on Learning Representations*, 2023.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, et al. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. Allava: Harnessing gpt4v-synthesized data for lite vision-language models. *arXiv preprint arXiv:2402.11684*, 2024a.

Jiacheng Chen, Tianhao Liang, Sherman Siu, Zhengqing Wang, Kai Wang, Yubo Wang, Yuansheng Ni, Ziyan Jiang, Wang Zhu, Bohan Lyu, et al. Mega-bench: Scaling multimodal evaluation to over 500 real-world tasks. In *International Conference on Learning Representations*, volume 2025, pp. 28384–28494, 2025a.

Junying Chen, Zhenyang Cai, Pengcheng Chen, Shunian Chen, Ke Ji, Xidong Wang, Yunjin Yang, and Benyou Wang. Sharegpt-4o-image: Aligning multimodal models with gpt-4o-level image generation. *arXiv preprint arXiv:2506.18095*, 2025b.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 2024b.

Shuo Chen, Zhen Han, Bailan He, Jianzhe Liu, Mark Buckley, Yao Qin, Philip Torr, Volker Tresp, and Jindong Gu. Can multimodal large language models truly perform multimodal in-context learning? In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6000–6010. IEEE, 2025c.

Shuo Chen, Jianzhe Liu, Zhen Han, Yan Xia, Daniel Cremers, Philip Torr, Volker Tresp, and Jindong Gu. True multimodal in-context learning needs attention to the visual context. In *Second Conference on Language Modeling*, 2025d.

Xiaohui Chen, Satya Narayan Shukla, Mahmoud Azab, Aashu Singh, Qifan Wang, David Yang, ShengYun Peng, Hanchao Yu, Shen Yan, Xuewen Zhang, et al. Compcap: Improving multimodal large language models with composite captions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23582–23592, 2025e.

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*, pp. 24185–24198, 2024c.

Xiang Cheng, Yuxin Chen, and Suvrit Sra. Transformers implement functional gradient descent to learn non-linear functions in context. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 8002–8037, 2024.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Honghao Fu, Yuan Ouyang, Kai-Wei Chang, Yiwei Wang, Zi Huang, and Yujun Cai. Contextnav: Towards agentic multimodal in-context learning. In *International Conference on Learning Representations*, volume 2026, pp. 74934–74959, 2026.

Robotics Gemini, Saminda Abeyruwan Team, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, et al. Language is not all you need: Aligning perception with language models. *Advances in Neural Information Processing Systems*, 36: 72096–72109, 2023.

Yixing Jiang, Jeremy Irvin, Ji Hun Wang, Muhammad Ahmed Chaudhry, Jonathan H Chen, and Andrew Y Ng. Many-shot in-context learning in multimodal foundation models. *arXiv preprint arXiv:2405.09798*, 2024.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Eugene Lee, Yu-Chi Lin, and Jiajie Diao. Learning to select visual in-context demonstrations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9455–9465, June 2026.

Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Joshua Adrian Cahyono, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025a.

Gengluo Li, Xingyu Wan, Shangpin Peng, Weinong Wang, Hao Feng, Yongkun Du, Binghong Wu, Zheng Ruan, Zhiqiong Lu, Liang Wu, et al. Hunyuanocr-1.5: Making lightweight ocr vlms faster and better. *arXiv preprint arXiv:2607.04884*, 2026.

Yanshu Li. Unveiling and mitigating shortcuts in multimodal in-context learning. In *ICLR Workshop on Spurious Correlation and Shortcut Learning: Foundations and Solutions*, 2025.

Yanshu Li, Jianjiang Yang, Tian Yun, Pinyuan Feng, Jinfa Huang, and Ruixiang Tang. Taco: Enhancing multimodal in-context learning via task mapping-guided sequence configuration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 736–763, 2025b.

Mengchen Liu, Qixiu Li, Dongdong Chen, Dong Chen, Jianmin Bao, and Yunsheng Li. Synchart: Synthesizing charts from language models. *arXiv preprint arXiv:2409.16517*, 2024.

Yang Luo, Zangwei Zheng, Zirui Zhu, and Yang You. How does the textual information affect the retrieval of multimodal in-context learning? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5321–5335, 2024.

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Amanpreet Singh, Guan Pang, Mandy Toh, Jing Huang, Wojciech Galuba, and Tal Hassner. Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8802–8812, 2021.

Issa Sugiura, Keito Sasagawa, Keisuke Nakao, Koki Maeda, Ziqi Yin, Zhishen Yang, Shuhei Kurita, Yusuke Oda, Ryoko Tokuhisa, Daisuke Kawahara, et al. Jagle: Building a large-scale japanese multimodal post-training dataset for vision-language models. *arXiv preprint arXiv:2604.02048*, 2026.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026a.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026b.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Haoyu Wang, Haonan Wang, Yuyan Chen, Jun Chen, Gang Liu, Qian Wang, Jiahong Yan, and Yanghua Xiao. Enhancing multimodal in-context learning via inductive-deductive reasoning. *arXiv preprint arXiv:2605.02378*, 2026.

Jinpeng Wang, Tianci Luo, Yaohua Zha, Yan Feng, Ruisheng Luo, Bin Chen, Tao Dai, Long Chen, Yaowei Wang, and Shu-Tao Xia. Embracing collaboration over competition: Condensing multiple prompts for visual in-context learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25156–25165. IEEE, 2025.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 2024a.

Yu Wang and Sharon Li. Why multimodal in-context learning lags behind? unveiling the inner mechanisms and bottlenecks. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13670–13685, 2026.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 2024b.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Guangzhi Xiong, Sanchit Sinha, Zhenghao He, and Aidong Zhang. Retrieving counterfactuals improves visual in-context learning. *arXiv preprint arXiv:2603.16737*, 2026.

Nan Xu, Fei Wang, Sheng Zhang, Hoifung Poon, and Muhao Chen. From introspection to best practices: Principled analysis of demonstrations in multimodal in-context learning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3299–3324, 2025.

Yue Yang, Ajay Patel, Matt Deitke, Tanmay Gupta, Luca Weihs, Andrew Head, Mark Yatskar, Chris Callison-Burch, Ranjay Krishna, Aniruddha Kembhavi, et al. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17486–17505, 2025.

Junyan Ye, Dongzhi Jiang, Zihao Wang, Leqi Zhu, Zhenghao Hu, Zilong Huang, Jun He, Zhiyuan Yan, Jinghua Yu, Hongsheng Li, et al. Echo-4o: Harnessing the power of gpt-4o synthetic images for improved image generation. *arXiv preprint arXiv:2508.09987*, 2025.

Ellen Yi-Ge, Jiechao Gao, Wei Han, and Wei Zhu. Drum: Learning demonstration retriever for large multi-modal models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 1051–1063, 2025.

Xiaomin Yu, Pengxiang Ding, Wenjie Zhang, Siteng Huang, Songyang Gao, Chengwei Qin, Kejian Wu, Zhaoxin Fan, Ziyue Qiao, and Donglin Wang. Unicorn: Text-only data synthesis for vision language model training. *arXiv preprint arXiv:2503.22655*, 2025.

Haotian Zhang, Mingfei Gao, Zhe Gan, Philipp Dufter, Nina Wenzel, Forrest Huang, Dhruti Shah, Xianzhi Du, Bowen Zhang, Yanghao Li, et al. Mm1. 5: Methods, analysis & insights from multimodal llm fine-tuning. In *International Conference on Learning Representations*, volume 2025, pp. 64669–64714, 2025.

Wenqi Zhang, Zhenglin Cheng, Yuanyu He, Mengna Wang, Yongliang Shen, Zeqi Tan, Guiyang Hou, Mingqian He, Yanna Ma, Weiming Lu, et al. Multimodal self-instruct: Synthetic abstract image and visual reasoning instruction using language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19228–19252, 2024.

Haozhe Zhao, Zefan Cai, Shuzheng Si, Xiaojian Ma, Kaikai An, Liang Chen, Zixuan Liu, Sheng Wang, Wenjuan Han, and Baobao Chang. Mmicl: Empowering vision-language model with multi-modal in-context learning. In *International Conference on Learning Representations*, volume 2024, pp. 14942–14980, 2024.

Shijie Zhou, Ruiyi Zhang, Yufan Zhou, and Changyou Chen. A high-quality text-rich image instruction tuning dataset via hybrid instruction generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 10091–10110, 2025.

Yongshuo Zong, Ondrej Bohdal, and Timothy Hospedales. Vl-icl bench: The devil in the details of multimodal in-context learning. In *International Conference on Learning Representations*, volume 2025, pp. 100058–100100, 2025.

A SUPPLEMENTARY COMPARISON: SCALING WITH CONTEXT SIZE

Table 4: **Comparison of downstream accuracies across different numbers of in-context demonstrations with GPT-5.** We report the downstream accuracy. Higher values indicate better performance. '-' suggests that the method would not construct the context under the current benchmark due to an insufficient number of valid samples being selected from the corpus.

Methods	Number of In-Context Demonstrations					
	2	4	8	16	32	64
CharXiv						
MMICES	0.420	0.407	0.453	0.480	0.493	0.473
ContextNav	0.440	0.473	0.507	0.520	-	-
SynMIC	0.487	0.493	0.540	0.547	0.567	0.580
MathVision						
MMICES	0.420	0.427	0.453	0.467	0.487	0.467
ContextNav	0.433	0.480	0.520	0.547	-	-
SynMIC	0.440	0.487	0.553	0.573	0.573	0.587
ERQA						
MMICES	0.360	0.350	0.380	0.400	0.400	0.380
ContextNav	0.370	0.410	0.430	-	-	-
SynMIC	0.360	0.420	0.470	0.490	0.480	0.500

To further examine how context size affects SynMIC, we evaluate its downstream performance with 2 to 64 in-context demonstrations. As shown in Table 4, SynMIC generally improves with more demonstrations, albeit with diminishing returns as the context grows. By contrast, fixed-corpus baselines tend to saturate earlier and sometimes degrade at larger context sizes, potentially because, once the most relevant corpus examples are exhausted, additional demonstrations become increasingly less aligned with the target query. Overall, these results suggest that SynMIC remains more effective across a broad range of context sizes.

B SUPPLEMENTARY ABLATIONS

B.1 DISENTANGLING SYNTHETIC ICL GAINS FROM ADDITIONAL AGENTIC REASONING

Table 5: **Comparison of demonstration-content ablations and alternative inference strategies.** We report downstream accuracy with GPT-5. Higher values indicate better performance, and bold numbers denote the best results.

Method	CharXiv	MathVision	MMStar	ERQA
SynMIC (Full)	**0.540**	**0.553**	**0.460**	**0.470**
w/o Demonstration Images	0.407	0.420	0.353	0.330
w/ Textual Scene Descriptions	0.433	0.447	0.373	0.360
SynMIC (w/o Auxiliary Perception)	**0.533**	**0.553**	**0.453**	**0.450**
Repeated Inference + Voting	0.473	0.500	0.393	0.420
Schema-guided Direct Solving	0.480	0.493	0.407	0.430

Table 5 compares SynMIC with demonstration-content ablations and matched-budget direct-inference baselines using GPT-5. We first investigate the role of visual content in in-context demonstrations. For the demonstration-content ablations in the upper block of the table, the two indented variants reuse SynMIC's synthesized demonstrations while modifying only the final context. w/o Demonstration Images removes the demonstration images, isolating their contribution beyond the associated query-answer pairs. w/ Textual Scene Descriptions instead replaces the images with textual descriptions derived from the scene content in the task schema and the visual constraints in the final synthesis plans, allowing us to compare visual and textual representations of the same demonstration content. The results show that visual content is critical for effective in-context reasoning: removing the images or replacing them with textual descriptions substantially weakens the downstream MLLM's ability to exploit the reasoning patterns conveyed by the context, and can even introduce misleading contextual signals.

In the lower block of the table, we compare synthetic MM-ICL with direct target-query solving under the same upstream policy-inference budget. The two direct-inference baselines reallocate SynMIC's policy computation to solving the target query without MM-ICL, with the reasoning-token budget

capped by the policy tokens consumed by SynMIC for the corresponding demonstration planning and tool-invocation reasoning. To ensure comparable perceptual access, auxiliary perception tools are excluded from SynMIC in this comparison; importantly, the downstream MLLM still receives only the target multimodal query and the final demonstrations, without access to intermediate policy analysis, synthesis plans, or verification feedback. Repeated Inference + Voting uses the matched budget to sample independent solutions to the target query and selects the most frequent normalized answer, without demonstrations or synthesis specifications. Schema-guided Direct Solving first derives SynMIC's content, task, and reasoning specification, then provides this specification directly as textual guidance and allocates the remaining budget to the same repeated-reasoning-and-voting procedure. Thus, this comparison tests whether the same policy computation is better used for direct target-query reasoning or for constructing synthetic multimodal demonstrations that support MM-ICL. The results show that, under the matched reasoning budget, using the policy computation to construct multimodal demonstrations yields more effective downstream inference than allocating it directly to target-query solving.

B.2 Straightforward Synthetic

To examine whether SynMIC's gains can be explained simply by synthesizing query-adaptive demonstrations, we construct a straightforward synthesis baseline using one-step direct prompting. Given a target image-query pair, the policy is instructed to generate diverse yet similar multimodal QA cases that can serve as useful in-context demonstrations, without explicit content/task abstraction or reasoning-schema induction. The prompt is shown below. For each case, the policy directly specifies a set of demonstration plans, each comprising a new visual scenario, a scenario-grounded query, an answer, and the visual content needed to support it. These plans are then realized using the same synthesis tools and backbones as SynMIC. We further equip this baseline with a validation-and-resynthesis procedure the same as SynMIC's: each synthesized demonstration is checked for standalone validity, and invalid cases trigger a fresh one-step synthesis attempt.

Straightforward Synthesis Prompt

The target image and query are attached.

Directly create K new multimodal QA demonstrations that are similar and useful for solving the target query, while using different visual scenarios and surface content.

For each demonstration, provide:

- a new visual scenario,

- a query grounded in that scenario,

- the answer,

- the visual content required to support the answer.

Table 6: **Comparison with straightforward synthetic demonstration construction.** We compare SynMIC with straightforward demonstration synthesis and its iterative-refinement variant. Straightforward Synthetic directly constructs multimodal demonstrations from the target image-query pair, while the validation variant repeatedly verifies the synthesized demonstrations and resynthesizes invalid cases with fresh synthesis attempts, for up to three rounds to avoid infinite rollouts. We report downstream accuracy together with the average inference cost per valid demonstration on MathVision and MMStar.

Method	MathVision	MMStar	Policy Cost		Code-based Rendering		Image Generation
			Input Token	Output Token	Input Token	Output Token	Latency (min)
Zero-Shot	0.480	0.387	-	-	-	-	-
Straightforward Synthetic	0.460	0.373	0.8K	0.5K	3.6K	13.3K	0.55
+ Validation	0.507	0.393	22.1K	2.2K	11.7K	38.7K	1.36
SynMIC	0.553	0.460	20.7K	9.8K	21.8K	18.4K	0.60

As shown in Table 6, straightforward synthesis alone provides insufficient guidance for downstream MM-ICL. Repeated validation improves performance by generating and retaining more useful synthetic demonstrations, but at a substantially higher synthesis cost. In contrast, SynMIC combines

finer-grained task and reasoning analysis, structured demonstration planning, and targeted refinement to produce higher-quality demonstrations. Consequently, it achieves substantially stronger downstream gains while incurring costs comparable to, or lower than, the validation-based rollout along several major dimensions. These results suggest that straightforward generation and repeated validation alone are insufficient for effective synthetic MM-ICL; explicitly structuring demonstration synthesis around the target task and reasoning requirements is critical for producing more informative and cost-effective in-context demonstrations.

C Inference Costs

Table 7: **Average inference cost per valid synthetic image.** We report the average policy and code-based rendering token costs, together with the average image-generation latency on CharXiv, MathVision, MMStar, and ERQA.

Policy Cost		Code-based Rendering		Image Generation
Input Token	**Output Token**	**Input Token**	**Output Token**	**Latency (min)**
25.3K	6.6K	15.2K	15.3K	0.63

As shown in Table 7, SynMIC incurs a manageable synthesis overhead. On average, generating a valid synthetic image requires 25.3K/6.6K policy input/output tokens for demonstration planning, tool invocation, verification, and the self-evolution loop. For code-based rendering, the rendering tool consumes an additional 15.2K/15.3K input/output tokens per valid synthetic image. Since policy inference and code-based rendering calls can run concurrently through APIs, their contribution to overall wall-clock latency is relatively limited. In addition, model-based image generation introduces 0.63 minutes of latency per valid synthetic image. Importantly, these costs are not strictly one-off. Once verified, synthetic demonstrations can be retained as reusable context assets. As shown by the corpus-augmentation results in Table 3 and the reuse process in Table 9, previously synthesized demonstrations can either be incorporated into the demonstration corpus to improve retrieval-based methods or be directly reused for subsequent queries, allowing the synthesis overhead to be progressively amortized over time.

D Supplementary Discussions

D.1 Disentangling Synthetic ICL Gains from Backbone Understanding

To disentangle whether SynMIC's gains arise from using a more capable synthesis policy than the downstream MLLM or from constructing synthetic multimodal demonstrations for in-context inference, we evaluate a self-improvement setting in which the same MLLM serves as the SynMIC policy, the code-based rendering backbone, and the downstream model. As shown in Table 8, SynMIC consistently improves over zero-shot inference across all three backbones and four benchmarks. These results show that the gains do not depend on a capability gap between the synthesis policy and the downstream MLLM. Instead, the same backbone can improve its downstream inference by externalizing its task and reasoning analysis into verified multimodal demonstrations that are subsequently reused as in-context guidance.

Table 8: **Evaluating self-improvement with SynMIC.** Each model serves as the SynMIC policy for demonstration synthesis, backbone for code-based rendering and the downstream MLLM for in-context inference. We report downstream accuracy, consistent with Table 1.

Model	Setting	CharXiv	MathVision	MMStar	ERQA
Gemma-4-26B-A4B	Zero-Shot	0.293	0.340	0.307	0.330
	w/ SynMIC	0.353	0.373	0.367	0.360
Gemini-2.5-flash	Zero-Shot	0.300	0.360	0.327	0.310
	w/ SynMIC	0.360	0.407	0.353	0.350
Qwen3.6-35B-A3B	Zero-Shot	0.493	0.447	0.407	0.360
	w/ SynMIC	0.567	0.513	0.460	0.390

D.2 Amortizing Synthesis through Reusable Synthetic Demonstrations

Synthetic demonstrations generated by SynMIC are not limited to serving as one-off context. Once verified, they can be retained as reusable context resources and added to the demonstration corpus, allowing subsequent queries to reuse them rather than synthesizing new demonstrations from scratch. While Table 3 shows that augmenting the corpus with synthetic demonstrations can amortize synthesis costs while benefiting retrieval-based MM-ICL, we further examine whether previously synthesized demonstrations can be reused for unseen queries. Specifically, for each benchmark, we collect the verified synthetic demonstrations generated for the target queries in our main experiments and freeze them as a synthetic demonstration corpus. We then randomly sample an equal-sized, non-overlapping set of queries from the remaining benchmark instances for evaluation. For each unseen query, we use ContextNav with its original retrieval and selection procedure unchanged to retrieve demonstrations from the frozen synthetic corpus for an 8-shot context. We report the Reuse Rate, defined as the average fraction of the eight context slots that can be filled by previously synthesized demonstrations across unseen queries.

Table 9: **Reuse of previously synthesized demonstrations on unseen queries.** For each benchmark, we construct a fixed corpus from verified synthetic demonstrations generated for the target queries in the main experiments, and evaluate reuse on an equal-sized, non-overlapping set of unseen queries. 'Reuse Rate' denotes the average proportion of the 8-shot context that can be filled by previously synthesized demonstrations.

Benchmark	**CharXiv**	**MathVision**	**ERQA**
Reuse Rate (%) ↑	53.2	40.8	36.9

As shown in Table 9, previously synthesized demonstrations can be meaningfully reused for unseen queries, with Reuse Rates of 53.2%, 40.8%, and 36.9% on CharXiv, MathVision, and ERQA, respectively. This indicates that a non-trivial portion of the context can be retrieved from previously synthesized demonstrations without further synthesis, supporting their use as reusable context resources for amortizing synthesis cost over subsequent queries.

D.3 Agreement Between Human Evaluation and LLM-as-a-Judge Evaluation

Table 10: **Human evaluation of LLM-as-a-judge agreement.** We evaluate the GPT-5.6 Sol model against the human consensus. Adjacent Match Accuracy (± 1) measures the proportion of samples with a score deviation ≤ 1; Pearson Correlation assesses the linear dependence between predictions and consensus; and Cohen's Kappa utilizes linear weighting to evaluate inter-rater reliability for ordinal ratings.

Metric	**RFS**	**TFS**	**Diversity**
Adjacent Match Accuracy (± 1)	0.922	0.946	0.892
Pearson Correlation	0.850	0.867	0.833
Weighted Cohen's Kappa	0.722	0.725	0.704

To assess the reliability of our LLM-as-a-Judge evaluation, we randomly sample 500 synthetic demonstrations from MMStar and compare the GPT-5.6 Sol ratings against human consensus. As shown in Table 10, the LLM-based evaluation exhibits strong agreement with human judgments across RFS, TFS, and Diversity: the adjacent match accuracy is consistently above 0.89, Pearson correlations exceed 0.83, and weighted Cohen's Kappa remains above 0.70. These results indicate that the LLM-as-a-Judge scores are broadly aligned with human judgments, providing empirical support for their use in our automatic evaluation.

E Skill Initialization

At the start of each experimental trial, the synthesis skill is reset to a blank template for each dataset and synthesis configuration. Target queries are processed exactly once in a randomized order to ensure statistical validity. Within each run, the skill is updated online based on verification feedback and execution errors observed on synthetic demonstrations, and the updated skill is reused for subsequent

synthesis calls, including resynthesis of the current candidate. No evolved skill is carried over across datasets, trials, or synthesis configurations. The case-level memory is likewise initialized as empty for each demonstration candidate.

Blank Skill

```
{ "DemonstrationPlanningSkill": "", "ToolExecutionSkill": "" }
```

F PROMPTS

Content and Task Analysis Prompt

Analyze the target image and query to guide synthetic demonstration planning.

The target image is attached.

original query: {original_query}

Describe the content relevant to the query and identify the underlying task. Separate the particular scene and entities from the task structure that new demonstrations should preserve. Ground the analysis in the available visual evidence.

Return a JSON object with:

- content abstraction: the query-relevant image content, with example fields such as:
  - visible scene: string
  - related entities: [string]
  - information granularity: string
  - salient visual cues: [string]
- task schema: the underlying task structure, with example fields such as:
  - task type: string
  - evidence type: string
  - answer space: string

You may add other relevant fields beyond these examples when appropriate for the task.

Reasoning Pattern Analysis Prompt

Derive a plausible and generalizable reasoning schema for the target multimodal query.

The target image is attached.

original query: {original_query}

content abstraction: {content_abstraction}

task schema: {task_schema}

Describe how the relevant visual evidence can lead to an answer. Capture the reasoning pattern that new demonstrations should preserve while allowing their scenarios and concrete details to vary.

Return a JSON object with:

- reasoning schema: the generalizable reasoning pattern, with example fields such as:
  - reasoning objective: string
  - reasoning steps: [string]
  - intermediate variables: [string]
  - answer derivation: string
  - risky counterfactuals: [string]

You may add other relevant fields beyond these examples when appropriate for the task.

Demonstration Planning Prompt

Plan a set of synthetic demonstrations using the synthesis specification and synthesis skill.

synthesis specification: {synthesis_specification}

synthesis skill: {synthesis_skill}

num demonstrations: {num_demonstrations}

Create the requested number of demonstration plans. Preserve the target task and reasoning schema while varying scenarios, visual content, queries, and answers across the set to avoid overfitting to the target instance.

For each plan, write a query grounded in the new scenario and commit its answer before image synthesis. Specify the visual evidence needed to support that answer and the intended reasoning. Use relevant synthesis skill guidance when planning the demonstrations.

Return a JSON object with:

- demonstration plans: the planned demonstrations, each described through the following four parts:
  - new visual scenario: the visual scenario to be synthesized.
  - scenario grounded query: the demonstration query appropriate for that scenario.
  - pre committed answer: the answer that the synthesized image should support.
  - visual constraints: the visual content and conditions needed to support the answer while preserving the task and reasoning schema.

Tool Planning Prompt (Step 1)

Plan a toolchain to synthesize the image required by a single demonstration plan.

A single demonstration plan: {demonstration_plan}

case memory: {case_memory}

tool library: {tool_library}

synthesis skill: {synthesis_skill}

Select suitable tools from the provided library according to the planned scenario, query, pre-committed answer, and visual constraints. Use the synthesis skill and prior attempt feedback where relevant.

Select at least one tool and arrange the selected tools in execution order. Include auxiliary perception when its results may be useful for subsequent synthesis.

Return a JSON object with:

- toolchain: the selected tools in execution order.

Tool Planning Prompt (Step 2)

Generate the inputs for the planned toolchain.

A single demonstration plan: {demonstration_plan}

toolchain: {toolchain}

case memory: {case_memory}

tool library: {tool_library}

synthesis skill: {synthesis_skill}

Prepare the input for each tool in the provided toolchain so that their execution realizes the plan. Use the synthesis skill and prior attempt feedback where relevant.

When a tool requires an earlier tool's output, leave a corresponding slot in its input.

Return a JSON object with:

- tool inputs: the input for each tool in execution order, following its required format as specified in the tool library.

Verification Prompt

Verify the current synthetic demonstration against its plan and the target task and reasoning schemas.

The current synthesized image is attached.

A single demonstration plan: {demonstration_plan}

task schema: {task_schema}

reasoning schema: {reasoning_schema}

case memory: {case_memory}

Check whether the image matches the planned content, supports the pre-committed answer to the demonstration query, and preserves task and reasoning consistency. Judge the actual visual evidence rather than assuming that the intended content was successfully synthesized.

For an invalid demonstration, explain the mismatch and provide feedback that can guide resynthesis and synthesis skill updating. Use the tool invocation context and earlier feedback where relevant to the failure analysis.

Return a JSON object with:

- validity decision: whether the current demonstration is valid under these checks.

- verification feedback: the assessment of plan alignment, answer support, and task and reasoning consistency, including failure analysis when the demonstration is invalid, and any other information you consider relevant to the verification.

Skill Evolution Prompt

Update the synthesis skill using feedback and context from failed synthesis attempts.

synthesis skill: {synthesis_skill}

case memory: {case_memory}

Distill the failure experience into concise, reusable guidance for demonstration planning and tool execution. Consider the attempted demonstrations, tool invocation contexts, and verification feedback or execution errors.

Preserve useful existing guidance and incorporate relevant lessons from the failures. Keep the guidance applicable to similar cases without making it overly specific to an individual example.

If the synthesis skill is empty, initialize it with a numbered item starting from "1. ...". Otherwise, revise the existing items as needed or append new guidance as additional numbered items.

Return a JSON object with:

- synthesis skill: the updated natural-language guidance for subsequent demonstration planning, tool execution, and resynthesis.

Refinement Planning Prompt

Determine which parts of a failed synthesis attempt need refinement.

A single demonstration plan: {demonstration_plan}

toolchain: {toolchain}

tool inputs: {tool_inputs}

case memory: {case_memory}

synthesis skill: {synthesis_skill}

Use the failure feedback and updated synthesis skill to identify the necessary changes. For failures that can be corrected at the tool level, keep the demonstration plan unchanged. Consider revising the demonstration plan when repeated failures indicate that it is infeasible.

Select the stages needed to address the failure and keep their outputs consistent. If the toolchain changes, include tool input refinement. If the demonstration plan changes, consider whether the toolchain or inputs also need adjustment.

Return a JSON object with:

- refinement stages: the stages to run, selected from demonstration plan, toolchain, and tool inputs, in this order where applicable.

- refinement guidance: what needs to change at the selected stages and why.

Demonstration Plan Refinement Prompt

Refine a single demonstration plan according to the refinement guidance.

A single demonstration plan: {demonstration_plan}

synthesis specification: {synthesis_specification}

refinement guidance: {refinement_guidance}

case memory: {case_memory}

synthesis skill: {synthesis_skill}

Revise the scenario, query, pre-committed answer, or visual constraints as needed to address the failure while preserving the target task and reasoning schema. Retain the parts that remain suitable. Any revised answer is committed before resynthesis, and the resulting demonstration must be verified again.

Return a JSON object with:

- demonstration plan: the revised plan in the same format as the input demonstration plan.

Toolchain Refinement Prompt

Refine the toolchain according to the current demonstration plan and refinement guidance.

A single demonstration plan: {demonstration_plan}

toolchain: {toolchain}

refinement guidance: {refinement_guidance}

case memory: {case_memory}

tool library: {tool_library}

synthesis skill: {synthesis_skill}

Use the feedback and synthesis skill to adjust the tools or their execution order where needed. Select at least one tool from the provided library that can help realize the current plan. Include auxiliary perception when its results may be useful for subsequent synthesis. Focus on the toolchain; its inputs will be refined separately.

Return a JSON object with:

- toolchain: the revised tools in execution order.

Tool Input Refinement Prompt

Refine the tool inputs for the current demonstration plan and target toolchain.

A single demonstration plan: {demonstration_plan}

target toolchain: {target_toolchain}

previous toolchain: {previous_toolchain}

previous tool inputs: {previous_tool_inputs}

refinement guidance: {refinement_guidance}

case memory: {case_memory}

tool library: {tool_library}

synthesis skill: {synthesis_skill}

Using the previous toolchain and previous tool inputs as reference, revise or prepare the input for each tool in the target toolchain to address the failure and realize the current plan. Use the feedback and synthesis skill while retaining inputs that remain suitable. When a tool requires an earlier tool's output, leave a corresponding slot in its input.

Return a JSON object with:

- tool inputs: the revised input for each tool in the target toolchain, in execution order and following its required format as specified in the tool library.

RFS Rating Prompt

Please rate the Reasoning Faithfulness Score (RFS) of the synthetic demonstration relative to the source, using the images, queries, answers, and source reference. Assess whether it preserves the key reasoning operations, evidence dependencies, and answer derivation, regardless of surface changes.

Five-point anchors: 1 = The essential reasoning operation is absent, unrelated, contradictory, or completely bypassed. 2 = Some reasoning cues overlap, but a major operation/dependency is missing or replaced; it largely teaches a different computation. 3 = The main operation is shared, but a material simplification, evidence/dependency gap, or derivation mismatch remains. 4 = The essential operations and dependencies are preserved and grounded; only a minor mismatch or limited ambiguity remains. 5 = The demonstration instantiates an essential evidence-to-answer structure, with usable evidence and a supported derivation, despite allowed surface changes.

Return only JSON with an integer score and a brief reason: {"score": ..., "reason": "..."}

TFS Rating Prompt

Please rate the Task Faithfulness Score (TFS) of the synthetic demonstration relative to the source, using the images, queries, answers, and source reference. Assess whether it preserves the task objective, required visual evidence, answer type, and essential constraints, allowing scenario-specific changes.

Five-point anchors: 1 = An unrelated task, or no usable instantiation of the source task. 2 = Only a broad task family overlaps; a major objective, evidence requirement, or answer-space constraint changes. 3 = The core task is recognizable, but a material task-defining constraint, evidence granularity, or answer type is only partly preserved. 4 = The same specific task and principal constraints are instantiated, with only a minor mismatch or ambiguity. 5 = The same specific task, necessary visual evidence roles, answer type, and essential constraints are preserved under scenario-appropriate changes.

Return only JSON with an integer score and a brief reason: {"score": ..., "reason": "..."}

Diversity Rating Prompt

Please rate the Diversity of ONE synthetic demonstration relative to its source image-query pair. Assess meaningful changes in the scenario, visual content, query formulation, and answer content that go beyond copying the source's surface patterns. Evaluate this source-relative variation independently of task/reasoning faithfulness. Shared task-required structure and naturally repeated answer labels are not evidence of copying.

Five-point anchors: 1 = An almost exact copy of the source, with negligible meaningful changes to its concrete content or presentation. 2 = Limited variation, mostly cosmetic substitutions of names, values, colors, or wording within the source's specific template. 3 = Meaningful variation in some applicable aspects, but substantial dependence on the source's concrete scenario, visual arrangement, or query template remains. 4 = Substantial, meaningful variation across several applicable aspects, with clear departure from source-specific content and only limited surface reuse. 5 = A distinct surface realization with rich, meaningful changes in applicable scenario, visual, or linguistic aspects; it avoids unnecessary reuse of source-specific patterns and goes well beyond cosmetic substitutions.

Return only JSON with an integer score and a brief reason: {"score": ..., "reason": "..."}

G TOOL LIBRARY

The tool library provides descriptions and invocation requirements for code-based rendering, model-based image generation and editing, and auxiliary perception. Each entry specifies the tool's function, inputs, outputs, and calling requirements. These descriptions guide toolchain planning and the preparation of tool-compatible inputs. During execution, outputs from earlier tools may be passed to subsequent tools as specified by the execution plan.

Image inputs and outputs denote image data supplied or returned by the execution environment. Code-based tools mainly receive coding requirements, generate and execute Python code internally, and return both the resulting image and the corresponding executed code. Image generation and editing tools receive the corresponding text instructions and, when required, a reference image. Perception tools return information associated with the input image, such as recognized text and regions or instance masks.

Tool Library


```
{
  "matplotlib_python": {
    "description": "Generate and execute Python code to render charts and programmable diagrams using Matplotlib.",
    "inputs": {"coding_requirements": "string"},
    "outputs": {"image": "image", "code": "string"},
    "call_requirements": ["Describe the rendering requirements, including relevant data and visual constraints."]
  },
  "plotly_python": {
    "description": "Generate and execute Python code to render data visualizations and scientific figures using Plotly.",
    "inputs": {"coding_requirements": "string"},
    "outputs": {"image": "image", "code": "string"},
    "call_requirements": ["Describe the rendering requirements, including relevant data and visual constraints."]
  },
  "pillow_python": {
    "description": "Generate and execute Python code to render text or perform image operations using Pillow.",
    "inputs": {"coding_requirements": "string", "input_image": "image (optional)"},
    "outputs": {"image": "image", "code": "string"},
    "call_requirements": ["Describe the text-rendering requirements or image operations to perform.", "Supply input_image when operating on an existing image."]
  },
  "image_generator": {
    "description": "Synthesize an image from a text prompt.",
    "inputs": {"prompt": "string", "negative_prompt": "string (optional)"},
    "outputs": {"image": "image"},
    "call_requirements": ["Provide a text prompt describing the image to synthesize.", "Use negative_prompt only when additional content exclusions are needed."]
  },
  "image_editor": {
    "description": "Edit a reference image according to a text instruction.",
    "inputs": {"reference_image": "image", "edit_prompt": "string", "negative_prompt": "string (optional)"},
    "outputs": {"image": "image"},
    "call_requirements": ["Supply the reference image and an instruction describing the requested changes.", "Use negative_prompt only when additional content exclusions are needed."]
  },
  "ocr": {
    "description": "Recognize text in an image and locate the corresponding text regions.",
    "inputs": {"image": "image", "instruction": "string"},
    "outputs": {"text": "recognized text", "regions": "corresponding text regions"},
```

```
    "call_requirements": ["Supply the image and an instruction specifying the text information to extract.",
"Keep each text result associated with its region and identify the coordinate convention relative to the
input image."]
  },
  "concept_segmentation": {
    "description": "Segment instances of a specified visual concept in an image.",
    "inputs": {"image": "image", "concept": "string"},
    "outputs": {"masks": "instance masks"},
    "call_requirements": ["Supply the image and the concept to segment.", "Keep the masks aligned with
the input image."]
  }
}
```


H Case Study

H.1 Comparison against other baselines

As shown in Figure 4–Figure 7, demonstrations retrieved from a fixed corpus may be visually or semantically similar to the target query yet offer limited support for the required reasoning. In contrast, SynMIC provides more targeted contextual guidance and addresses gaps in fixed-corpus coverage, thereby better supporting downstream reasoning.

H.2 Failure cases

As shown in Figure 8 and Figure 9, SynMIC may offer limited benefits when decision boundaries are ambiguous or reasoning requirements are highly complex. Ambiguous boundaries can cause synthesized demonstrations to misrepresent the distinctions required by the target query, resulting in mismatched contextual guidance. Meanwhile, even well-aligned demonstrations may be insufficient for downstream models to resolve intricate logical constraints.

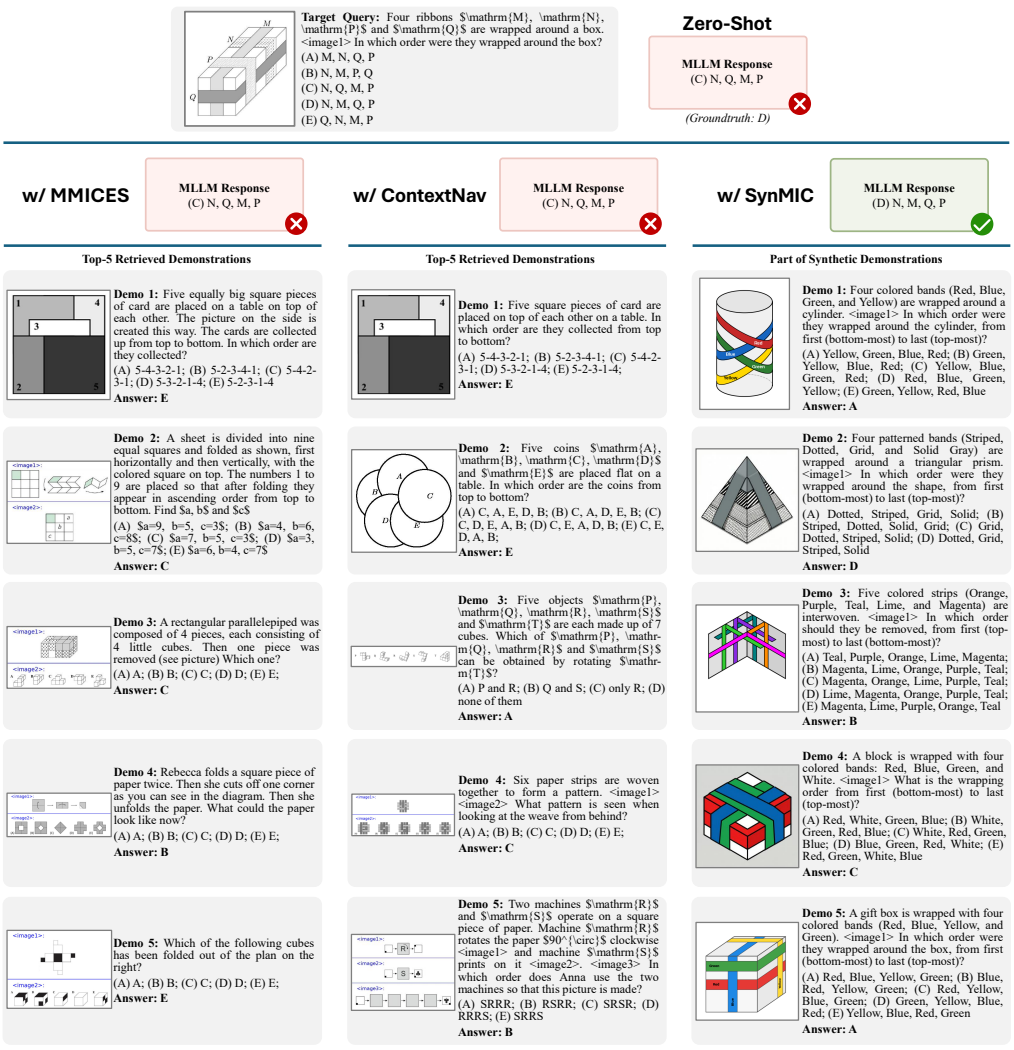

Figure 4: Visualized case comparison with other baselines on the MathVision benchmark.

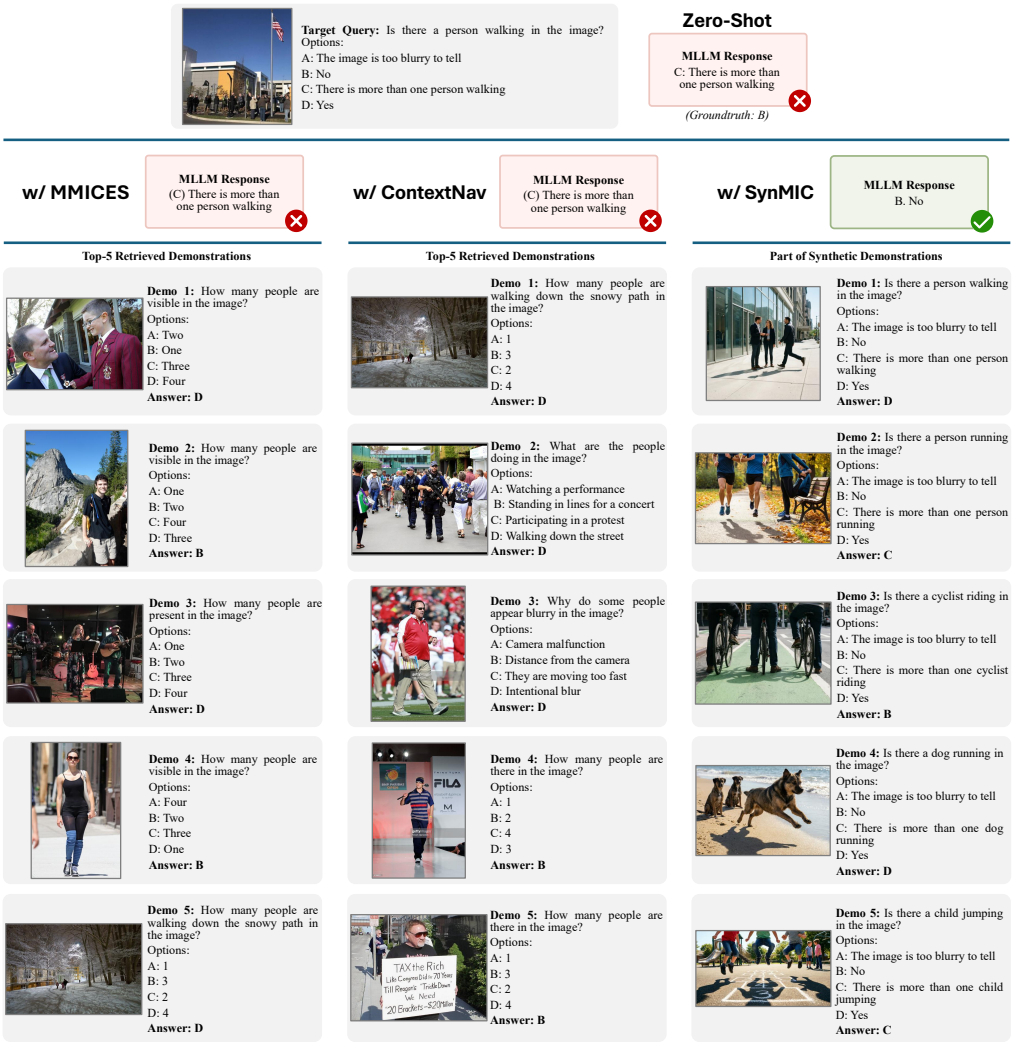

Figure 5: Visualized case comparison with other baselines on the MMStar benchmark.

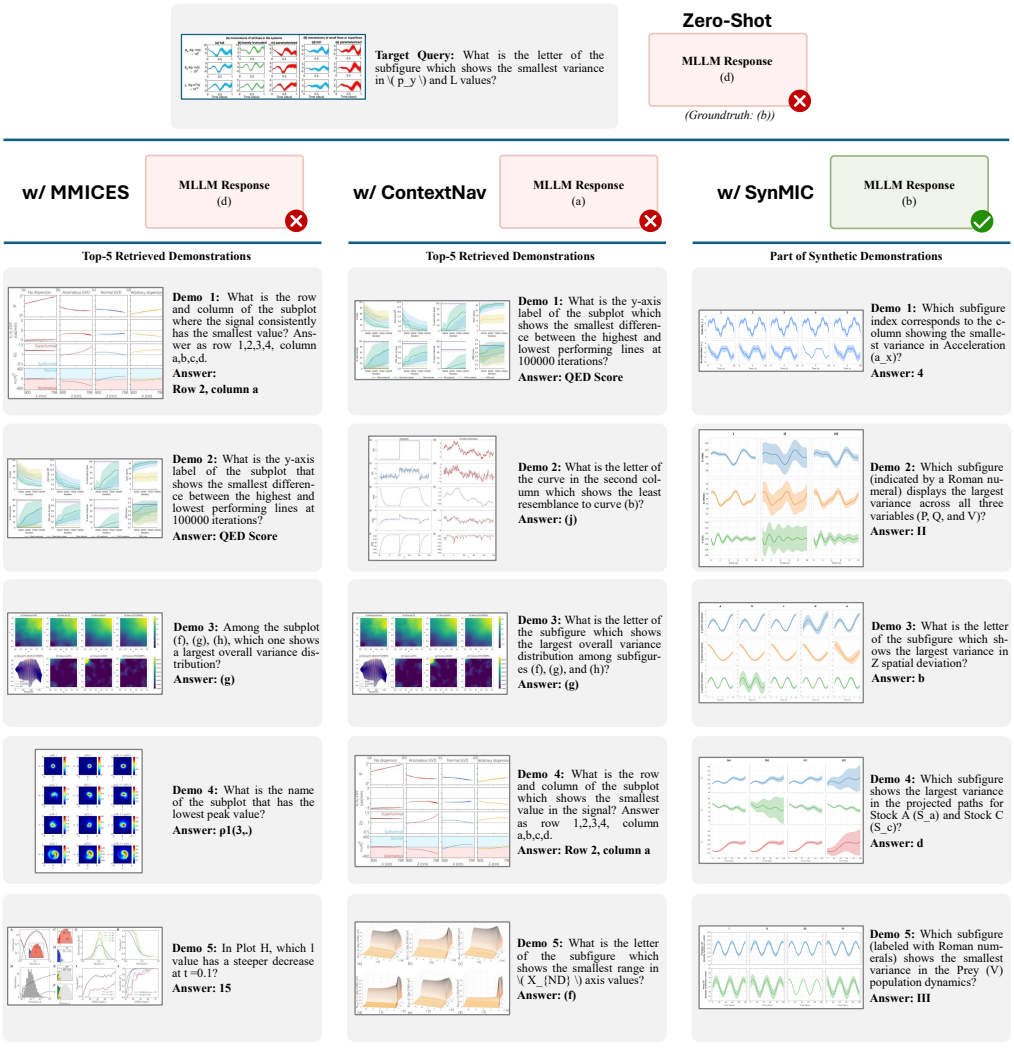

Figure 6: Visualized case comparison with other baselines on the CharXiv benchmark.

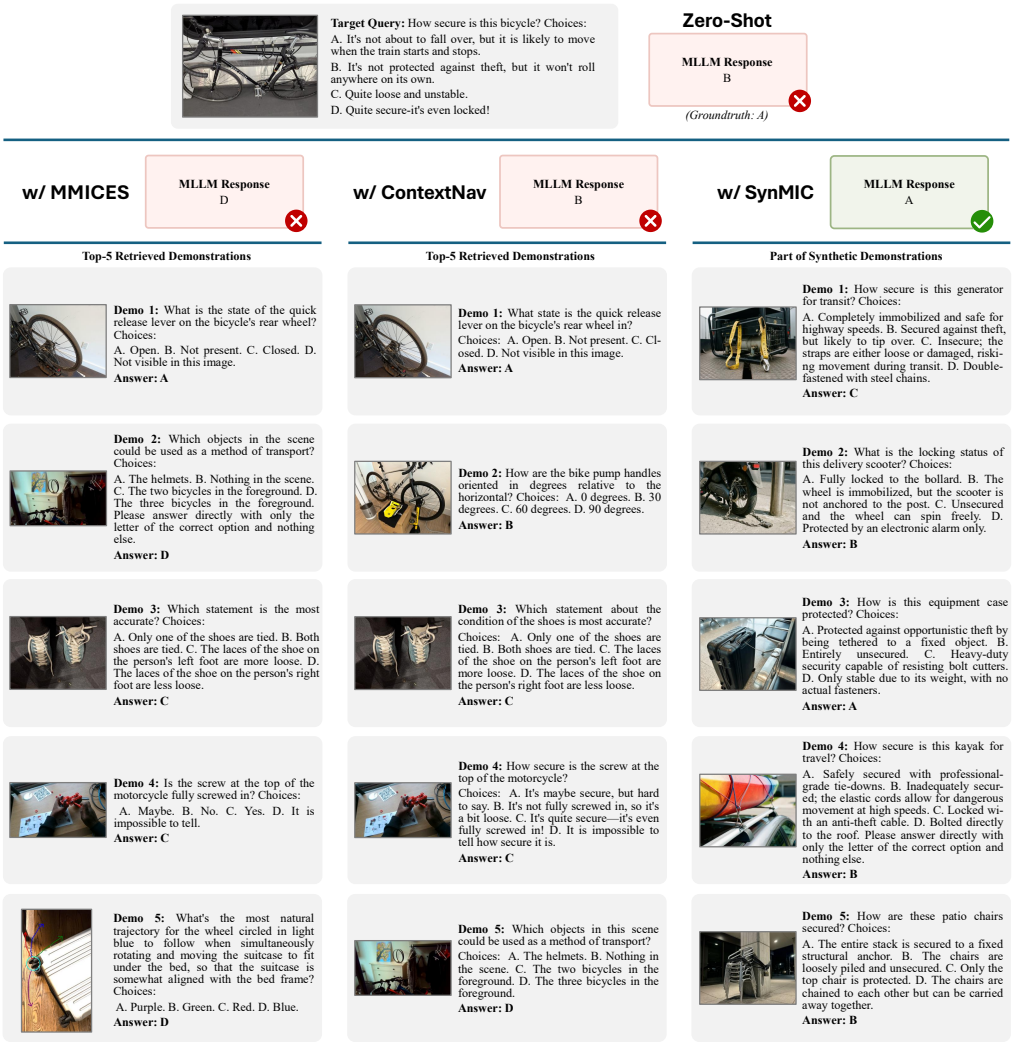

Figure 7: Visualized case comparison with other baselines on the ERQA benchmark.

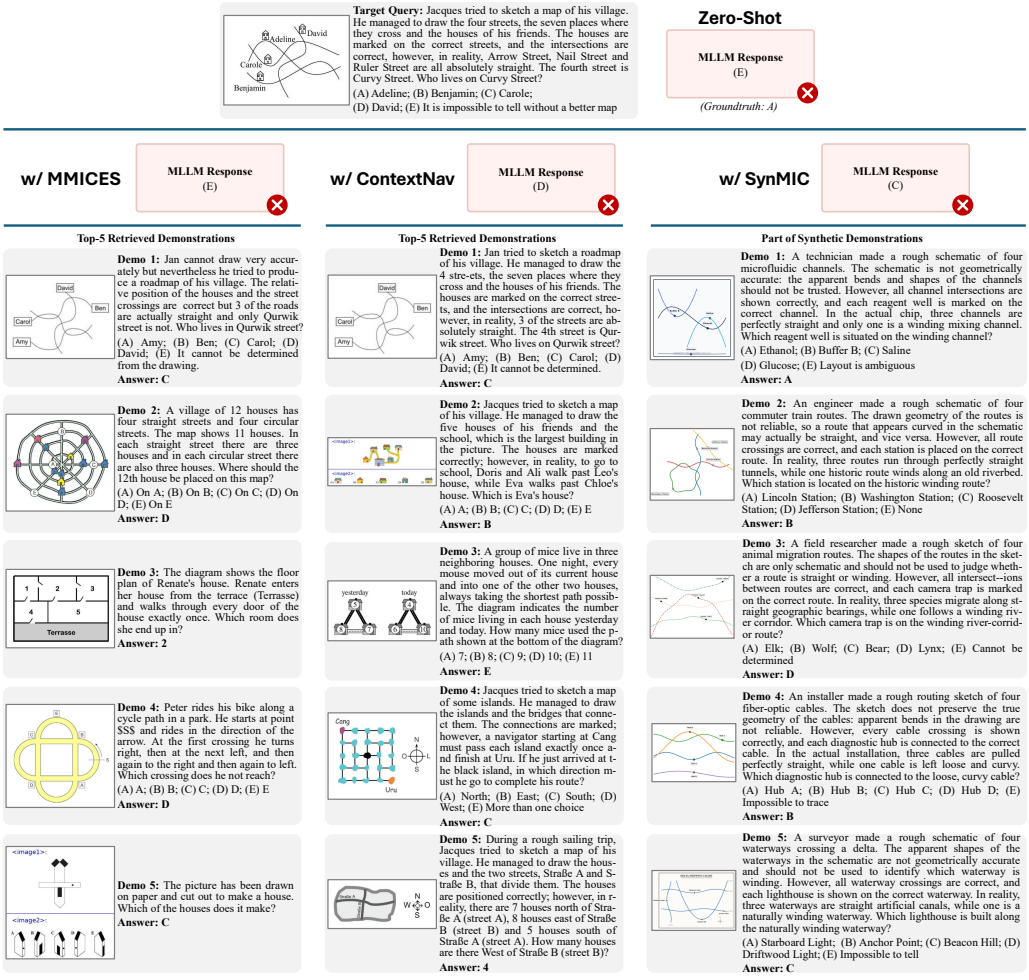

Figure 8: Failure case of SynMIC about code-based rendering.

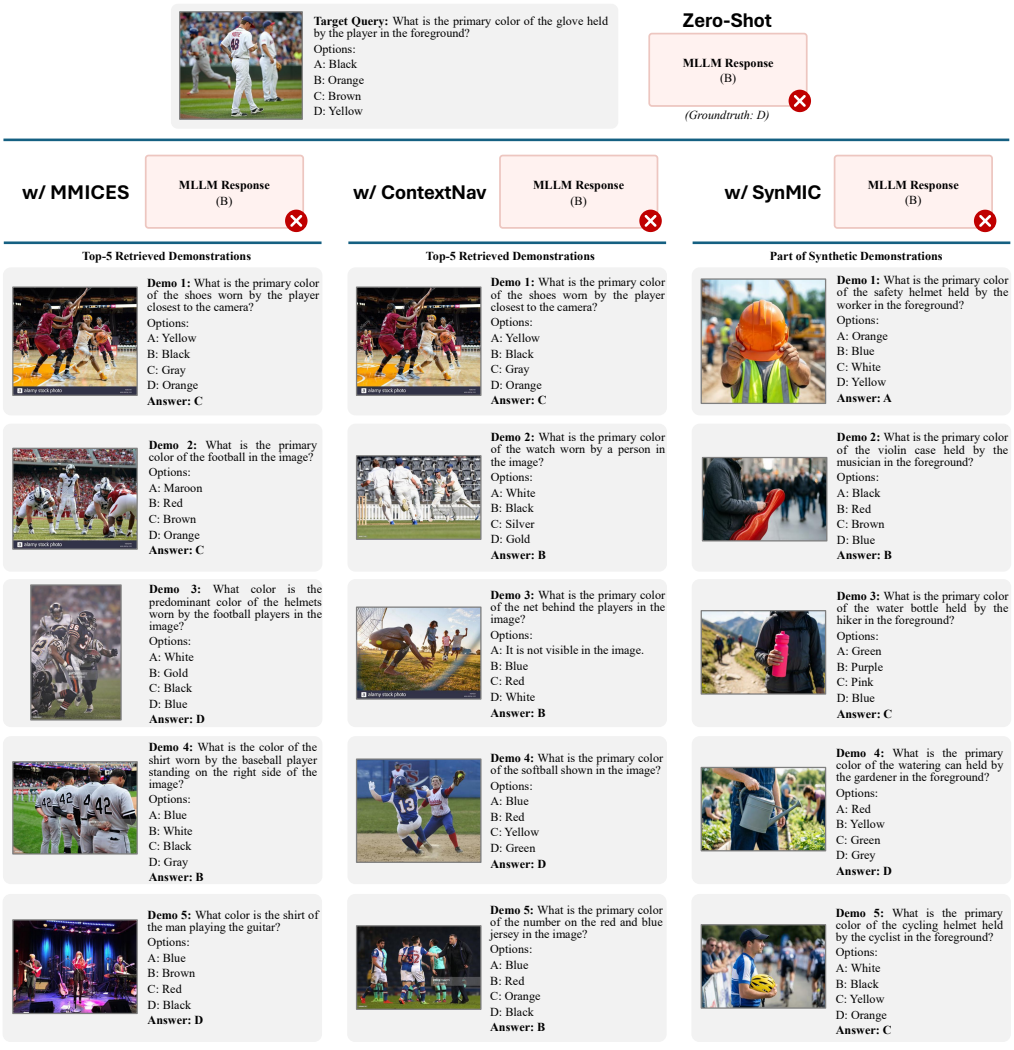

Figure 9: Failure case of SynMIC about model-based image generation.