DUAL-GRAINED MEMORY WITH SHAPLEY CONTEXT ATTRIBUTION FOR MULTIMODAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

External memory allows frozen multimodal models to bypass costly parameter updates by leveraging labeled training data at inference. Designing effective memory, however, requires balancing the specific visual and logical grounding of individual cases with the need for generalized, transferable procedures. Furthermore, jointly retrieving multiple memories complicates selection by frequently introducing redundant or contradictory context. To address these challenges, we introduce **DG-Mem**, an offline memory framework that extracts dual-grained representations—instance-grounded exemplars and category-level schemas—directly from a frozen model's training rollouts. Per-problem reflections are dynamically clustered by an incremental categorizer and synthesized into shared procedural advice. To optimize test-time ranking and resolve retrieval conflicts, a secondary offline pass evaluates subsets of these rules to assign Shapley-based utilities. Evaluated on held-out partitions of MathVista, MMMU, and MMMU-Pro, DG-Mem consistently improves accuracy over retrieval-free prompting across both open-weight and proprietary backbones. The framework maintains this advantage across repeated trials and outperforms tool-free XSkill under randomized splits. Applying DG-Mem to text-only mathematics yields similar gains: memory constructed from DAPO problems improves pooled AIME and HMMT accuracy, surpassing TFGRPO. Ablations confirm that the dual memory grains function complementarily, and Shapley utility weighting successfully provides further incremental gains. Ultimately, DG-Mem validates offline memory synthesis as a highly effective, practical adaptation strategy for frozen models, establishing a strong foundation for future advancements in rule-level context attribution.

1 INTRODUCTION

Multimodal large language models (MLLMs) can answer diverse vision-language questions (OpenAI, 2025; Google, 2026), but scientific and mathematical reasoning remain challenging (Lu et al., 2024; Yue et al., 2024; 2025). When a model repeatedly encounters such problems, its successful solutions and recurring mistakes provide information that can be reused. For models accessed through an API, or when fine-tuning is costly, that information can be supplied through the prompt. Retrieval-based in-context learning selects relevant training examples for this purpose (Liu et al., 2022); external memory also stores lessons extracted from the model's attempts to solve them.

Storing a solution trajectory preserves the setting in which a strategy succeeded or failed. Trajectory and case memories retain this context (Zhou et al., 2025a; Shinn et al., 2023; Chhikara et al., 2025; Zhong et al., 2024), whereas prescriptive memories condense experience into reusable guidance (Wang et al., 2025b; Zhao et al., 2024; Ouyang et al., 2026; Zhang et al., 2025b). For an ambiguous diagram, a similar solved example may clarify what to inspect. A rule for checking units or counting chart entries can instead remain useful across problems whose images look quite different. Several systems combine memory types, including logical and visual guidance in ViLoMem (Bo et al., 2025), task- and step-level skills in D2Skill (Tu et al., 2026), and experiences and skills in XSkill (Jiang et al., 2026). In our setting, both forms must be extracted from the same labeled corpus: problem-specific records preserve a strategy and its checks, while synthesis across related problems produces category-level advice.

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.

Once several memories enter the prompt together, their usefulness depends on the surrounding context. Two rules may repeat the same instruction; a third may introduce a check that neither supplies alone. Measuring these interactions requires evaluating combinations of rules. MemRL (Zhang et al., 2026b) uses outcome-based utilities to learn from past memory use, while Shapley valuation (Ghorbani & Zou, 2019) and context attribution (Cohen-Wang et al., 2024) examine contributions by varying the available information. We adopt rule-subset evaluations to estimate how each retrieved rule affects correctness in combination with others, then reuse the resulting utilities in retrieval.

The distinction between specific experiences and shared guidance also appears in the Complementary Learning Systems account of memory (McClelland et al., 1995; Kumaran et al., 2016). Together with work on exemplar-based generalization (Nosofsky, 1986), it offers a motivation for retaining individual cases alongside abstractions built from experience. The analogy informs the organization of our memory stores. We implement this organization in **DG-Mem** through two stores built from labeled training rollouts: an *exemplar memory* indexed by the problem's multimodal representation, and a *schema memory* containing category-level advice. Per-problem IF-THEN reflections provide the input to schema synthesis. An incremental categorizer organizes these reflections during the offline construction pass, avoiding the need to specify a taxonomy in advance.

To assign utilities, a second offline pass repeatedly solves training problems with different subsets of retrieved schema rules. Each rule receives its average marginal contribution over rule orderings, and these contributions are averaged across problems before being combined with similarity at test time. Attribution is restricted to the retrieved candidate set and uses a finite evaluation budget; missing coalition rewards and model stochasticity limit its precision. The model parameters and completed memory remain fixed during evaluation. DG-Mem therefore studies reuse of a labeled training corpus.

We evaluate this design on held-out partitions of MathVista, MMMU, and MMMU-Pro with open-weight and proprietary backbones. Component ablations and repeated evaluation examine the memory design, while comparisons with ViLoMem and tool-free XSkill provide complementary baselines. We also build memory from DAPO-Math-17k and evaluate it on AIME and HMMT against TFGRPO, extending the study to text-only mathematical reasoning. The results favor DG-Mem in many settings, but the effects depend on the backbone and benchmark; utility weighting provides a smaller incremental benefit than memory construction itself.

The paper makes three contributions:

- We develop a memory construction procedure that retains instance-grounded exemplars while synthesizing category-level advice from per-problem reflections. The MLLM remains frozen throughout construction and inference.
- For schema retrieval, Shapley context attribution supplies a correctness-based utility alongside similarity. We specify the bounded subset evaluation procedure and examine how often its budget covers the full coalition space.
- Experiments on multimodal and text-only reasoning test the approach against ViLoMem, XSkill, and TFGRPO. Memory ablations and repeated evaluations distinguish the benefits of combining grains from the smaller, less certain gains due to utility weighting.

2 Related Work

Recent works have treated the prompt context as a first-class object to be engineered, accumulated, and evolved (ang Gao et al., 2026; Huang et al., 2026; Hu et al., 2025). These methods can be organized along two axes: what is stored (granularity) and how it is constructed (regime).

Granularity. *Trajectory and case stores* preserve episodic experience and route through similarities or learned values. Memento (Zhou et al., 2025a) formulates case selection as a soft-Q policy over an episodic case bank. MemRL (Zhang et al., 2026b) attaches a TD-learned utility to each retrieved trace. Buffer-based methods (Shinn et al., 2023; Wang et al., 2024; Tang et al., 2025; Yang et al., 2024) and classical text-memory systems (Chhikara et al., 2025; Zhong et al., 2024) keep verbatim reasoning episodes or maintain unstructured notes. *Single-grain prescriptive playbooks* distill experience into a flat list of generic strategies (Wang et al., 2025b; Zhao et al., 2024; Ouyang et al., 2026; Zheng et al., 2025; Zhang et al., 2025b). ReasoningBank (Ouyang et al., 2026) and SkillWeaver (Zheng et al.,

Table 1: **Comparison of agentic memory methods across five design dimensions**: whether the method (i) handles *multimodal* inputs (vision+text), (ii) maintains *instance-grounded exemplars*, (iii) maintains *abstract schema rules*, (iv) performs *per-rule credit assignment* beyond similarity-only retrieval, and (v) is *gradient-free* (no model parameter updates). ✓ / × indicates whether the method satisfies a given dimension. See the surrounding text for column definitions.

Method	Multimodal	Exemplar	Schema	Credit Assn.	Grad-Free
Mem0 (Chhikara et al., 2025)	×	✓	×	×	✓
Memento (Zhou et al., 2025a)	×	✓	×	✓	✓
A-Mem (Xu et al., 2026)	×	✓	×	×	✓
ExpeL (Zhao et al., 2024)	×	×	✓	×	✓
AWM (Wang et al., 2025b)	×	×	✓	×	✓
ReasoningBank (Ouyang et al., 2026)	×	×	✓	×	✓
ACE (Zhang et al., 2025b)	×	×	✓	×	✓
ViLoMem (Bo et al., 2025)	✓	×	✓	×	✓
TFGRPO (Cai et al., 2025)	×	×	✓	✓	✓
D2Skill (Tu et al., 2026)	×	✓	✓	×	✓
Memory-R1 (Yan et al., 2025)	×	✓	×	✓	×
MemGen (Zhang et al., 2025a)	×	×	×	✓	×
DG-Mem (Ours)	✓	✓	✓	✓	✓

2025) extract heterogeneous episodes into a single bullet list. ACE (Zhang et al., 2025b) maintains an itemized playbook updated by a Generator-Reflector-Curator with grow-and-refine de-duplication. *Dual-grain or structured stores* factor memory into multiple types: ViLoMem (Bo et al., 2025) stores guidance about visual distractions and logical errors; D2Skill (Tu et al., 2026) stores task-level and step-level skill banks; structured stores such as A-Mem (Xu et al., 2026), Zep (Rasmussen et al., 2025), and G-Memory (Zhang et al., 2026a) organize memory as temporal or knowledge graphs. XSkill (Jiang et al., 2026) also combines two streams, pairing action-level experiences with task-level skills for multimodal agents; we include a tool-free comparison in Section 4.3. DG-Mem pairs problem-indexed exemplars with category-level advice synthesized from per-problem reflections. Table 1 compares agentic memory methods across modality, memory granularity, credit assignment, and parameter updates.

Construction Regime. Memory-R1 (Yan et al., 2025), Mem-α (Wang et al., 2025a), Memory-as-Action (Zhang et al., 2025d), and MEM1 (Zhou et al., 2025b) learn "add/update/delete" policies under outcome rewards. MemBuilder (Shen et al., 2026) trains a lightweight language model as the memory manager with attributed dense rewards; MemGen (Zhang et al., 2025a) fuses RL-generated latent memory tokens into the decoder; Mem-T (Yue et al., 2026) and LatentMem (Fu et al., 2026) apply GRPO over Memory-Operation Trees and learnable composers respectively. These approaches use learned memory-management policies or latent representations, whereas DG-Mem constructs explicit text memory with a frozen backbone. A complementary line distills memory offline from rollouts of a frozen backbone (Zhao et al., 2024; Wang et al., 2025b; Ouyang et al., 2026; Bo et al., 2025; Zhang et al., 2025b); DG-Mem follows this setting. Our concept categorizer grows the category space during the training pass. It indexes reasoning concepts, with STITCH's contextual-intent indexing (Yang et al., 2026) providing a related approach for dialogue. Our second offline pass estimates utilities for co-retrieved schema rules from their joint effect on correctness.

Attribution of data and context. Data Shapley (Ghorbani & Zou, 2019) uses average marginal contributions to value training examples. ContextCite (Cohen-Wang et al., 2024) attributes generated statements to parts of the supplied context. DG-Mem applies this general intervention-based perspective to a different target: the correctness of a frozen model under subsets of retrieved schema rules. The resulting scores guide retrieval on subsequent problems. This is an application of established attribution ideas to memory selection, with a bounded coalition budget and in-sample utility estimation.

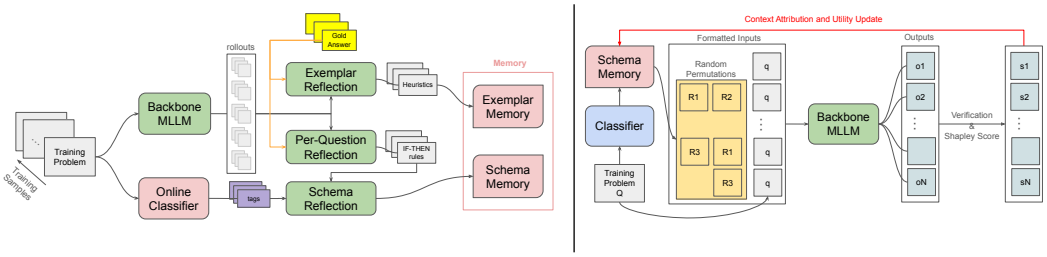

Figure 1: **Two-stage offline memory construction on a frozen MLLM.** Left: concept summaries and labeled training rollouts produce exemplars and reflections; reflections are grouped by category for schema synthesis. Right: rule-subset rollouts yield Shapley contributions, accumulated as utilities for test-time retrieval. The backbone receives no gradient updates. Sections 3.1 and 3.3 detail the two stages.

3 METHODS

Overview. DG-Mem augments a frozen MLLM with an external memory $\mathcal{M}$ constructed from labeled training problems. For a test problem q, a concept summarizer produces $\mathcal{K}(q)$; text and multimodal retrieval then select a context $\mathcal{M}_q$ from the schema and exemplar stores. The model answers using $\hat{a}_q = \mathrm{MLLM}(q, \mathcal{M}_q)$. Memory construction (Section 3.1), access (Section 3.2), and utility estimation (Section 3.3) involve no parameter updates.

3.1 MEMORY CONSTRUCTION

For each training problem q with gold answer $a_q^\star$, we sample $N = 5$ solution trajectories $\{\mathrm{MLLM}_i(q)\}_{i=1}^N$. The verification cascade Verify labels each trajectory correct or incorrect using a native benchmark matcher, symbolic parser, and LLM judge in succession. Gold answers provide the supervision for these labels.

A separate concept-summarizer call extracts $\mathcal{K}(q)$ from the problem. Once these concepts and the labeled rollouts are available, two reflectors produce problem-level summaries, while a categorizer assigns the concepts to the evolving registry Γ:

$$\rho(q) = \mathrm{Refl}_R\big(q, a_q^\star, \mathcal{K}(q), \{\mathrm{MLLM}_i(q)\}_{i=1}^N\big), \tag{1}$$

$$e(q) = \mathrm{Refl}_E\big(q, a_q^\star, \mathcal{K}(q), \{\mathrm{MLLM}_i(q)\}_{i=1}^N\big), \tag{2}$$

$$\kappa(q) = \mathrm{Cat}\big(\mathcal{K}(q), \Gamma\big). \tag{3}$$

Here $\rho(q)$ is a set of IF-THEN reflections, $e(q)$ is an exemplar summary, and $\kappa(q)$ is the category tag. The categorizer uses the concept summary and registry. The question-level reflector compares successful and unsuccessful trajectories to extract conditional guidance: the IF clause identifies when a lesson applies, and the THEN clause states an action or a check. Thus a training example contributes information about the model's reasoning behavior as well as its gold answer. The concept summary provides a separate description of the knowledge needed to solve the problem, which supports grouping examples even when their surface forms differ. The reflection store $\mathcal{M}_R = \{\rho(q)\}_q$ supplies the input to category-level synthesis. Its content reaches test-time inference through the synthesized schemas and the reflections associated with exemplar records.

Exemplar memory. The exemplar reflector Refl_E produces two fields: a *reasoning strategy* describing a procedure and a *verification check* describing an error to check for. The stored record associates these fields with the source problem, category tag, and reflections $\rho(q)$. It is indexed by a multimodal embedding $\mathrm{Emb}_M(I_q, t_q)$, where I_q is the image and t_q takes the form *"This problem tests $\mathcal{K}(q)$. Question: q."*

The reflector prompts ask for guidance that generalizes beyond the source problem and discourage copying its particular numbers or entities. The stored source questions retain their original details, while the generated guidance is prompted to abstract across instances. Multimodal indexing is intended to distinguish problems with similar text but different visual structure. For example, a full circle and a circular sector can share concept words while requiring different geometric treatment; a

grouped bar chart and a stacked bar chart likewise call for different readings. The joint representation retains visual cues for this selection.

Schema memory. After the training pass, reflections are grouped by category. The schema synthesizer produces advice from each group:

$$\sigma_\kappa = \mathrm{Synth}\left(\kappa, \mathrm{concepts}(\kappa), \bigcup_{q:\kappa(q)=\kappa} \rho(q)\right). \tag{4}$$

Here $\mathrm{concepts}(\kappa)$ is the category's representative concept set. The synthesizer is prompted to reconcile redundant reflections and return two to four actionable advice items, called *schema rules* below. These rules may be phrased as direct recommendations. A schema can therefore consolidate repeated observations into a shared procedure, such as enumerating all objects before applying attribute filters or checking every series before counting chart entries above a threshold. The synthesizer takes the grouped reflections as its source material.

Incremental categorization. A category determines which reflections are considered together during synthesis. Very broad groups can mix unrelated procedures, while a separate group for each concept signature leaves little opportunity to aggregate experience. Because this granularity is not known in advance, the registry is built incrementally from the training corpus. The categorizer compares $\mathcal{K}(q)$ with the existing registry and either assigns an existing category or creates a new identifier and representative concepts. Its prompt asks for assignment when conceptual overlap exceeds approximately 70%. We rely on the LLM's internal judgment for this threshold. Registry updates are serialized during traversal. A post-epoch embedding-based pass merges near-duplicate categories before the registry is frozen for Stage 2. Appendix C.3 reports sensitivity to training order.

3.2 Memory Access

At test time, $\mathcal{M} = \mathcal{M}_S \cup \mathcal{M}_E$ is frozen. Schema and exemplar retrieval provide complementary parts of the solver's context.

Schema retrieval. We first select categories relevant to $\mathcal{K}(q)$, then rank their individual rules. Text similarity is $\mathrm{sim}(r_i, q) = \cos\big(\mathrm{Emb}_T(r_i), \mathrm{Emb}_T(q)\big)$. The retrieval score combines similarity with the utility $U(r_i)$ estimated in Section 3.3:

$$s_i = \alpha\, \widehat{\mathrm{sim}}(r_i, q) + (1-\alpha)\, \widehat{U}(r_i). \tag{5}$$

The two quantities are independently min-max normalized over the candidate rules within each active category, preventing differences in their numerical scales from dominating the interpolation. Similarity measures the match to the current problem; utility summarizes a rule's observed contribution when retrieved during the offline attribution pass. We use $\alpha = 0.8$; $\alpha = 1$ gives similarity-only retrieval and $\alpha = 0$ gives utility-only ranking.

Exemplar retrieval. We rank exemplars by cosine similarity between the query embedding $\mathrm{Emb}_M(I_q, t_q)$ and each stored problem embedding. At most one exemplar is retained, subject to a 0.75 relevance gate; if none qualifies, no exemplar is injected. Exemplar ranking uses similarity alone.

Context assembly. Up to two rules from each of at most two categories are placed before the question as procedural guidance. The retrieved exemplar follows the question as a strategy and verification aid. Both components condition the same answer-generation call. We refer to the preceding schema context as *priming*: it supplies procedural guidance before the solver begins its response. Exemplar-based *verification* supplies a related strategy and checks that the solver can use while reasoning about the new instance. The solver receives both forms of guidance while generating its response. The concept-summarizer and answer-generation calls account for the two reported inference calls. Appendix A.2 lists retrieval thresholds.

3.3 Offline Utility Estimation

Stage 1 ranks synthesized rules by similarity. A rule may match the concepts of a query yet duplicate another retrieved rule, or recommend a procedure whose conditions are not satisfied. Rule-subset evaluation measures how correctness changes as individual rules enter or leave the context. Stage 2 keeps the category registry and schema text fixed, and evaluates rule subsets on training problems to estimate such a utility. Let $\mathcal{R}_q$ be the candidate set retrieved for problem q using similarity, before utility weighting. The same category and per-category caps give $K_q = |\mathcal{R}_q| \leq 4$; attribution is computed over this retrieved set.

Subset evaluations. With budget $B = 8$, we evaluate $\min(B, 2^{K_q})$ distinct subsets, including $\emptyset$ and $\mathcal{R}_q$. When the budget is smaller than the powerset, remaining subsets are sampled uniformly without replacement from the nonempty proper subsets. For each observed subset S, the model produces $o_S = \mathrm{MLLM}(q, S)$ and receives binary reward $v_q(S) = \mathrm{Verify}(o_S, a_q^\star)$. The problem and decoding configuration are held fixed to compare contexts, but residual generation non-determinism can still affect rewards. Define $\widetilde{v}_q(S) = v_q(S)$ for evaluated subsets and $\widetilde{v}_q(S) = 0$ otherwise. Zero-imputation completes the reward table at a fixed evaluation budget. This approximation can affect marginal contributions for partially covered problems.

Shapley attribution. For a retrieved rule r_i, we compute its average marginal contribution over rule orderings:

$$\Delta_q(r_i) = \frac{1}{K_q!} \sum_{\pi \in \mathfrak{S}_{K_q}} \left[\widetilde{v}_q(P_i^\pi \cup \{r_i\}) - \widetilde{v}_q(P_i^\pi)\right], \tag{6}$$

where P_i^π contains the rules preceding r_i under ordering π. A positive marginal contribution occurs when adding the rule changes an incorrect outcome to a correct one; a negative contribution records the reverse. Averaging over orderings includes the rule's effect both alone and in combination with other rules. For two rules, for example, each receives the mean of its contribution to the empty context and its contribution when the other rule is already present. This accounts for interactions within the candidate set. Full coverage requires 2^{K_q} distinct model evaluations, at most 16 under the retrieval cap. With full coverage, the value is exact for the observed reward table. Section 4.4 reports the fraction of fully covered problems.

For $n_i = |\{q : r_i \in \mathcal{R}_q\}|$, the rule utility is

$$U(r_i) = \frac{\sum_{q: r_i \in \mathcal{R}_q} \Delta_q(r_i)}{\max(n_i, 1)}. \tag{7}$$

Keeping the rule text fixed throughout this pass ensures that each accumulated mean refers to the same advice item. Averaging puts rules retrieved for different numbers of training problems on a common per-problem scale. These utilities enter Eq. (5) at test time. They re-rank the fixed schema rules. Both construction stages use the training partition, so schema utilities are in-sample estimates of usefulness. Subset evaluation uses schema rules, whose synthesis incorporates reflections from the same training partition. Exemplars remain in the similarity-ranked store for test-time use. Independent data for utility estimation would provide a stronger test of transfer.

4 Experiments

4.1 Main Results on Multimodal Benchmarks

Tasks and protocol. We evaluate MathVista (Lu et al., 2024), MMMU (Yue et al., 2024), and MMMU-Pro (Yue et al., 2025) using fixed 3:1 train/test splits (Appendix A.1). MathVista covers mathematical reasoning in visual contexts; MMMU and MMMU-Pro broaden the evaluation to multidisciplinary questions. The held-out partitions contain 250, 262, and 432 problems, respectively. Training gold answers supervise both offline construction stages. The resulting exemplars, schemas, and utilities remain fixed throughout testing. We report accuracy on held-out problems, using the same native-matcher $\rightarrow$ symbolic-parser $\rightarrow$ LLM-judge verification cascade as during construction.

Models and baselines. We use Qwen3.5-27B and Qwen3.5-122B-A10B (Team, 2026), GPT-5-Nano (OpenAI, 2025), and Gemini-3-Flash (Google, 2026). For each backbone, the rollout solver, concept summarizer, reflectors, categorizer, and schema synthesizer use that same model. Thus the construction pipeline extracts guidance from the evaluated model's own trajectories; only the judge is fixed to Qwen-3.5-Flash (Team, 2026). Text and multimodal retrieval use Qwen-3-Embedding (Zhang et al., 2025c) and Qwen-3-VL-Embedding (Li et al., 2026), respectively. No-memory uses the same backbone and decoding configuration with retrieval disabled. ViLoMem (Bo et al., 2025) builds logical/visual error memory: its offline variant freezes training-built memory, while its online variant continues updating on the test stream.

Main results. DG-Mem improves over No-memory in 11 of 12 settings (Table 2). The exception is Qwen3.5-27B on MMMU-Pro (73.6% versus 75.1%). DG-Mem also exceeds both ViLoMem variants on each benchmark for GPT-5-Nano, Gemini-3-Flash, and Qwen3.5-122B-A10B. Qwen3.5-27B shows a different pattern: online ViLoMem is slightly stronger on MathVista, and No-memory remains strongest on MMMU-Pro. Section 4.3 adds repeated evaluation and baseline comparisons.

Table 2: **Accuracy (%) on held-out multimodal reasoning splits.** Colored annotations give changes from the matching No-memory row in percentage points: green for gains, red for drops, gray for zero. *Avg Imp.* averages these changes across benchmarks. Asterisks denote held-out benchmark partitions; bold marks the best result per backbone and benchmark.

Backbone	Method	MathVista*	MMMU*	MMMU-Pro*	Avg Imp.
Qwen3.5-122B-A10B	No-memory	83.6	85.8	77.9	-
	ViLoMem (offline)	85.2 +1.6	86.4 +0.6	80.0 +2.1	+1.4
	ViLoMem (online)	88.4 +4.8	87.4 +1.6	80.3 +2.4	+2.9
	DG-Mem (Ours)	**88.7** +5.1	**89.2** +3.4	**86.1** +8.2	+5.6
Qwen3.5-27B	No-memory	82.9	83.4	**75.1**	-
	ViLoMem (offline)	86.4 +3.5	85.9 +2.5	74.6 -0.5	+1.8
	ViLoMem (online)	**87.2** +4.3	86.3 +2.9	74.8 -0.3	+2.3
	DG-Mem (Ours)	87.0 +4.1	**87.9** +4.5	73.6 -1.5	+2.4
GPT-5-Nano	No-memory	64.8	67.6	48.2	-
	ViLoMem (offline)	66.8 +2.0	56.9 -10.7	51.0 +2.8	-2.0
	ViLoMem (online)	70.4 +5.6	58.0 -9.6	52.5 +4.3	+0.1
	DG-Mem (Ours)	**74.4** +9.6	**79.4** +11.8	**64.4** +16.2	+12.5
Gemini-3-Flash	No-memory	87.2	82.4	69.0	-
	ViLoMem (offline)	80.4 -6.8	82.8 +0.4	73.8 +4.8	-0.5
	ViLoMem (online)	83.6 -3.6	82.4 0.0	71.1 +2.1	-0.5
	DG-Mem (Ours)	**91.6** +4.4	**84.7** +2.3	**77.1** +8.1	+4.9

4.2 Ablation Studies

Table 3 separates memory composition from utility weighting on Gemini-3-Flash. The three Stage-1 conditions compare schemas, exemplars, and their combination under similarity-only retrieval. Comparing the combined Stage-1 condition with the full method then measures the additional effect of Stage-2 utility. Appendix ablations extend the single-grain comparisons to GPT-5-Nano and Qwen3.5-122B-A10B.

Exemplar and schema memory. Under similarity-only retrieval, combining the grains improves on schema-only memory across all three benchmarks and on exemplar-only memory on MathVista and MMMU-Pro, while matching it on MMMU. On MathVista, the combined Stage-1 condition gains 2.6 points over schema-only and 3.4 over exemplar-only. On MMMU, it matches exemplar-only despite the weaker schema-only result. Thus the value of combining the stores varies with the task. Full DG-Mem also exceeds the Stage-1 exemplar-only and Stage-2 schema-only variants for GPT-5-Nano and Qwen3.5-122B-A10B across benchmarks (Appendix B).

Table 3: **Component ablations on Gemini-3-Flash.** The three Stage-1 variants use similarity-only retrieval; the full method adds Stage-2 utility weighting ($\alpha = 0.8$). Colored changes are percentage points relative to No-memory.

Datasets	MathVista*	MMMU*	MMMU-Pro*
Baseline	87.2	82.4	69.0
– w/o $\mathcal{M}_E$ (stage-1 schema only)	88.4 +1.2	82.1 -0.3	73.4 +4.4
– w/o $\mathcal{M}_S$ (stage-1 exemplar only)	87.6 +0.4	84.0 +1.6	73.4 +4.4
– w/o utility (α=1, stage-1 both memory)	91.0 +3.8	84.0 +1.6	75.0 +6.0
Full method	91.6 +4.4	84.7 +2.3	77.1 +8.1

Table 4: **Additional baselines.** (a) Four-split means: No-mem accuracy (%) and memory gains (percentage points). (b) Text-only mean@32 accuracy (%) with memory built from 100 DAPO problems; colored changes are gains/drops over No-memory (percentage points). n denotes test count. Pooled scores and changes are averaged over benchmarks before rounding.

(a) Multimodal memory effects across four splits

		Gemini-3-Flash			GPT-5-Nano		
Benchmark	n	No-mem	XSkill	DG-Mem	No-mem	XSkill	DG-Mem
MathVista	250	85.3	+1.5	+2.0	65.0	+4.5	+8.6
MMMU	262	82.4	+3.7	+4.4	66.0	+3.0	+8.1
MMMU-Pro	432	70.0	+6.8	+7.0	50.0	+5.4	+9.8

(b) DAPO-to-competition evaluation against TFGRPO

		GPT-5-Nano (mean@32)			Qwen3.5-27B (mean@32)		
Benchmark	n	No-mem	TFGRPO	DG-Mem	No-mem	TFGRPO	DG-Mem
AIME24	30	76.0	77.6 +1.6	78.4 +2.4	85.4	86.2 +0.8	87.6 +2.2
AIME25	30	77.0	78.4 +1.4	76.8 -0.2	84.7	85.5 +0.8	87.0 +2.3
HMMT	30	60.0	59.5 -0.5	60.9 +0.9	80.3	81.1 +0.8	81.7 +1.4
Pooled	90	71.0	71.8 +0.8	72.0 +1.0	83.5	84.3 +0.8	85.4 +2.0

Utility weighting. Adding utility weighting improves on the Stage-1 dual-memory condition in all three benchmarks, with the largest difference on MMMU-Pro. The increments are smaller than the overall memory gains. Section 5 discusses stochastic variation in the attribution estimates.

4.3 Repeated Evaluation and Additional Baselines

Repeated evaluation. On the original split, DG-Mem improves mean@8 accuracy in all 12 settings (Appendix C.1, Table 9), including Qwen3.5-27B on MMMU-Pro, where the single-run result is negative. These averages support a benefit across runs.

Comparison with XSkill across splits. We run XSkill's released code (Jiang et al., 2026) in tool-free, single-call `direct_cot` mode over four random splits, retaining the original held-out sample counts. DG-Mem's mean memory effect exceeds XSkill's in each evaluated setting (Table 4a). The separation is larger for GPT-5-Nano across all three benchmarks; on Gemini-3-Flash, the methods are much closer, particularly on MMMU-Pro.

Text-only mathematical reasoning. DG-Mem and our TFGRPO reimplementation (Cai et al., 2025) build memory from 100 random DAPO-Math-17k problems (Yu et al., 2025) and use frozen-memory, tool-free evaluation (TFGRPO's `direct` mode). On AIME24, AIME25, and HMMT (Zhang & Math-AI, 2024; 2025; HMMT), both methods improve pooled mean@32 accuracy over No-memory for both backbones (Table 4b). DG-Mem has the higher pooled score in each case, although the advantage is small for GPT-5-Nano. On AIME25, GPT-5-Nano with DG-Mem trails both baselines. These results extend the evaluation to memory constructed from text-only training problems and reused on competition benchmarks.

Appendices C.2–C.4 report subject-level results, categorization sensitivity, and grain disagreement.

4.4 Attribution Coverage and Computational Cost

Coalition coverage. With at most four retrieved rules, full enumeration requires at most 16 subset evaluations. The budget $B = 8$ covers the complete powerset whenever $K \leq 3$. Across the analyzed stores, 36–94% of training problems have full coverage. For Gemini-3-Flash, the rates are 53% on MathVista, 74% on MMMU, and 39% on MMMU-Pro. Four-rule problems use the sampled subsets and zero-imputation specified in Section 3.3; increasing the budget to 16 would cover their remaining coalitions.

Computational cost. Stage 1 uses nine LLM calls per training problem: five solution rollouts, concept summarization, categorization, and two reflections. Schema synthesis adds one call per category. Stage 2 averages 7.0–7.9 subset rollouts per problem across the analyzed settings. The completed stores are reused across test queries. Inference uses two LLM calls, for concept summarization and answer generation, versus one for No-memory, plus two embedding calls and cosine-similarity retrieval over cached vectors. Judge fallback adds evaluation-dependent calls; token usage and latency depend on context length and serving conditions.

4.5 Qualitative Results

Figure 2 in Appendix D shows corrections of visual enumeration, chart-threshold counting, and entity-grounded arithmetic errors. In the enumeration case, the retrieved guidance calls for an exhaustive object census before subtraction. In the chart case, the question asks how many objects are preferred by more than 90% of people in at least one category. The exemplar check calls attention to bars near a grid line, while the schema calls for evaluating each item. DG-Mem applies the strict threshold and returns zero, correcting the baseline's count of two. In the arithmetic case, the verification check emphasizes resolving the subjects' identities before using their birth years. Both single-grain variants also correct these cases; the figure supplies their retrieved exemplars and schema rules. Appendix C.4 analyzes cases where the grains disagree in correctness.

5 Limitations

Stochasticity of attribution. Non-deterministic GPU kernels, dynamic batching, and floating-point non-associativity (He & Lab, 2025) can change binary rewards across identical-context calls. With finite subset budget B, this variation can affect per-rule contributions and the resulting utilities. Repeated subset evaluations could improve these estimates at additional construction cost.

Frozen memory at test time. Memory remains fixed throughout evaluation, allowing test queries to reuse the offline store. Extending DG-Mem to learn from test-time experience would require a policy for updating exemplars, schemas, and their utilities. Combining offline consolidation with bounded online updates remains open.

6 Conclusions

DG-Mem combines instance-grounded exemplars with category-level advice constructed from a frozen model's training rollouts. Its offline attribution pass estimates how retrieved rules contribute to correctness and uses those estimates to refine retrieval.

The multimodal evaluation and DAPO-to-competition experiments indicate that this memory design can improve frozen-model reasoning without parameter updates. The evidence for utility weighting is more limited than the overall memory benefit, and comparisons depend on the backbone and task. Repeated attribution measurements, independent utility-estimation data, and bounded online updates are natural next steps.

AI USE STATEMENT

AI tools assisted with polishing and organizing the writing, literature checks and reviewing the consistency and interpretation of reported results, and preparing pseudocode from the described method. LLMs are also components of the experimental pipeline: they generate solution rollouts, summarize concepts, produce reflections and schemas, and assign categories. Qwen-3.5-Flash serves as the fallback judge in the answer-verification cascade during memory construction and evaluation. These experimental uses, including the backbone and embedding models, are described in Sections 3 and 4, with prompt templates in Appendix E. The authors take responsibility for the final manuscript, its claims, and the reported results.

REPRODUCIBILITY STATEMENT

Section 3 specifies memory construction, retrieval, and utility estimation. Appendix F, Algorithms 1–3, provides pseudocode for the two offline stages and frozen-memory inference. Section 4 identifies the evaluated backbones, embedding models, judge, baselines, and evaluation protocols. Appendices A.1 and A.2 give the dataset partitions and retrieval hyperparameters; Appendix E provides the solver and memory-construction prompt templates. Appendix C.1 reports repeated-evaluation summaries, and Section 4.4 reports coalition coverage and model-call costs. The effects of stochastic generation, incomplete coalition coverage, and in-sample utility estimation are discussed in Sections 3.3 and 5.

REFERENCES

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F. Survey Certification.

Weihao Bo, Shan Zhang, Yanpeng Sun, Jingjing Wu, Qunyi Xie, Xiao Tan, Kunbin Chen, Wei He, Xiaofan Li, Na Zhao, et al. Agentic learner with grow-and-refine multimodal semantic memory. *arXiv preprint arXiv:2511.21678*, 2025.

Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, et al. Training-free group relative policy optimization. *arXiv preprint arXiv:2510.08191*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. ContextCite: Attributing model generation to context. *arXiv preprint arXiv:2409.00729*, 2024. URL https://arxiv.org/abs/2409.00729.

Muxin Fu, Xiangyuan Xue, Yafu Li, Zefeng He, Siyuan Huang, Xiaoye Qu, Yu Cheng, and Yang Yang. Latentmem: Customizing latent memory for multi-agent systems. *arXiv preprint arXiv:2602.03036*, 2026.

Amirata Ghorbani and James Zou. Data Shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2242–2251. PMLR, 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.html.

Google. Gemini 3 flash, 2026. URL https://ai.google.dev/gemini-api/docs/gemini-3. Accessed: April 25, 2026.

Horace He and Thinking Machines Lab. Defeating nondeterminism in llm inference. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250910. https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

HMMT. The harvard-mit mathematics tournament (hmmt). URL https://www.hmmt.org.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, Shanglin Wu, Ruiyao Xu, Liangwei Yang, Rui Yang, Wooseong Yang, Chin-Yuan Yeh, Hanrong Zhang, Haozhen Zhang, Siqi Zhu, Henry Peng Zou, Wanjia Zhao, Song Wang, Wujiang Xu, Zixuan Ke, Zheng Hui, Dawei Li, Yaozu Wu, Langzhou He, Chen Wang, Xiongxiao Xu, Baixiang Huang, Juntao Tan, Shelby Heinecke, Huan Wang, Caiming Xiong, Ahmed A. Metwally, Jun Yan, Chen-Yu Lee, Hanqing Zeng, Yinglong Xia, Xiaokai Wei, Ali Payani, Yu Wang, Haitong Ma, Wenya Wang, Chenguang Wang, Yu Zhang, Xin Wang, Yongfeng Zhang, Jiaxuan You, Hanghang Tong, Xiao Luo, Xue Liu, Yizhou Sun, Wei Wang, Julian McAuley, James Zou, Jiawei Han, Philip S. Yu, and Kai Shu. Rethinking memory mechanisms of foundation agents in the second half: A survey, 2026. URL https://arxiv.org/abs/2602.06052.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. XSkill: Continual learning from experience and skills in multimodal agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=AjP1yvCyoG.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7): 512–534, 2016.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.deelio-1.10. URL https://aclanthology.org/2022.deelio-1.10/.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KUNzEQMWU7.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Robert M Nosofsky. Attention, similarity, and the identification–categorization relationship. *Journal of experimental psychology: General*, 115(1):39, 1986.

OpenAI. Openai gpt-5 system card, 2025. URL https://arxiv.org/abs/2601.03267.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Zhiyu Shen, Ziming Wu, Fuming Lai, Shaobing Lian, and Yanghui Rao. Membuilder: Reinforcing llms for long-term memory construction via attributed dense rewards. *arXiv preprint arXiv:2601.05488*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

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

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, and Dongbin Zhao. Dynamic dual-granularity skill bank for agentic rl. *arXiv preprint arXiv:2603.28716*, 2026.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025a.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1894–1907, 2024.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=NTAhi2JEEE.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=FiM0M8gcct.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544, 2024.

Ruozhen Yang, Yucheng Jiang, Yueqi Jiang, Priyanka Kargupta, Yunyi Zhang, and Jiawei Han. Grounding agent memory in contextual intent. *arXiv preprint arXiv:2601.10702*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15134–15186, 2025.

Yanwei Yue, Boci Peng, Xuanbo Fan, Jiaxin Guo, Qiankun Li, and Yan Zhang. Mem-t: Densifying rewards for long-horizon memory agents. *arXiv preprint arXiv:2601.23014*, 2026.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025a.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng YAN. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026a. URL https://openreview.net/forum?id=mmIAp3cVS0.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025b.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026b.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025c.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks. *arXiv preprint arXiv:2510.12635*, 2025d.

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, pp. 19632–19642, 2024.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. URL https://arxiv.org/abs/2504.07079.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025a.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025b.

APPENDIX

A ADDITIONAL EXPERIMENTAL DETAILS

A.1 DATASETS AND SPLITS

We evaluate on three multimodal reasoning benchmarks and use a fixed 3:1 train/test partition per benchmark. The gold answers of training problems provide the supervision for memory construction. Train problems drive Stage-1 memory construction and Stage-2 utility attribution; test problems are held out and evaluated with the memory frozen.

Table 5: Train/test partitions per benchmark. The same split file is used by every backbone and by every method (No-memory, ViLoMem, DG-Mem) so that comparisons are head-to-head.

Benchmark	Train (Stage 1 & 2)	Test (held out)	Total
MathVista (MINI)	750	250	1,000
MMMU (DEV+VAL)	788	262	1,050
MMMU-Pro	1,298	432	1,730

A.2 TEST-TIME RETRIEVAL

At test time $\mathcal{M} = \mathcal{M}_S \cup \mathcal{M}_E$ is frozen. For each query we run the concept summarizer $\mathcal{K}$ once, retrieve the top categories whose representative concepts clear the category-similarity gate, then retrieve up to $K_{\max}$ rules per active category ranked by Eq. (5). Exemplars are retrieved by multimodal cosine similarity against $\mathrm{Emb}_M(I_q, t_q)$ subject to a similarity-threshold gate; if no exemplar clears the gate, none is injected.

Table 6: Test-time retrieval hyperparameters.

Hyperparameter	Value	Description
α	0.8	Similarity / utility interpolation in Eq. (5)
Category similarity threshold	0.40	Min cosine similarity for category retrieval
Schema rule similarity threshold	0.35	Min cosine similarity for rule retrieval
$K_{\max}$ (rules per category)	2	Top-k rules retrieved per active category
Maximum active categories	2	At most four schema rules per query
Maximum exemplars	1	Verification context following the question
Exemplar relevance gate	0.75	Minimum relevance for exemplar injection

B ADDITIONAL ABLATION STUDIES

Tables 7 and 8 report single-grain variants for GPT-5-Nano and Qwen3.5-122B-A10B. These schema-only variants include Stage-2 utility weighting; the Gemini-3-Flash schema-only condition in Table 3 uses Stage 1. The full method exceeds both single-grain variants in each reported setting. Individual grains generally improve on No-memory, with Qwen3.5-122B-A10B's exemplar-only variant matching the baseline on MMMU. This pattern is consistent with partially complementary contributions.

C SUPPORTING ANALYSES OF MEMORY CONSTRUCTION AND USE

C.1 REPEATED EVALUATION ON THE ORIGINAL SPLIT

Table 9 reports DG-Mem's improvement over No-memory in mean accuracy across eight runs on the original held-out split. Gains are positive in all 12 settings. These summaries report means without run-level dispersion or confidence intervals.

Table 7: Component ablations on GPT-5-Nano. Exemplar-only uses Stage 1; schema-only includes Stage-2 utility weighting.

Datasets	MathVista*	MMMU*	MMMU-Pro*
Baseline	64.8	67.6	48.2
– w/o $\mathcal{M}_S$ (stage-1 exemplar only)	68.8 +4.0	72.5 +4.9	55.6 +7.4
– w/o $\mathcal{M}_E$ (stage-2 schema only)	67.2 +2.4	74.0 +6.4	58.1 +9.9
Full method	74.4 +9.6	79.4 +11.8	64.4 +16.2

Table 8: Component ablations on Qwen3.5-122B-A10B. Exemplar-only uses Stage 1; schema-only includes Stage-2 utility weighting.

Datasets	MathVista*	MMMU*	MMMU-Pro*
Baseline	83.6	85.8	77.9
– w/o $\mathcal{M}_S$ (stage-1 exemplar only)	87.2 +3.6	85.8 0.0	80.9 +3.0
– w/o $\mathcal{M}_E$ (stage-2 schema only)	86.4 +2.8	86.5 +0.7	81.3 +3.4
Full method	88.7 +5.1	89.2 +3.4	86.1 +8.2

C.2 Selected Subject-Level Improvements

Table 10 presents the ten largest subject-level improvements for GPT-5-Nano on MMMU-Pro. The selection highlights the largest observed gains in mathematics, sciences, and applied fields. These are ranked examples of subject-level improvement; Table 2 reports performance across the full held-out partition.

C.3 Sensitivity to Categorization Order

We repeat Stage-1 categorization over the same precomputed rollouts using three shuffled training orders, holding the other settings fixed. Table 11 compares each assignment to the original 68-category registry using adjusted Rand index (ARI) and normalized mutual information (NMI). The number of categories ranges from 58 to 85, while NMI against the original stays between 0.80 and 0.82. Across all pairs, including shuffle–shuffle comparisons, NMI ranges from 0.79 to 0.83 and ARI from 0.42 to 0.53. These measurements characterize the shared structure and order sensitivity of the analyzed store.

C.4 Outcomes When the Two Grains Disagree

We define disagreement as an exclusive-or in correctness: schema-only retrieval is correct and exemplar-only retrieval is incorrect, or vice versa, on the same test item. Table 12 reports how often the combined condition is correct on these items. The fractions range from 56% to 100%; conditional accuracy when both single-grain conditions are correct ranges from 93% to 100% across the 12 settings. The analysis compares correctness across three separately evaluated memory conditions. The smallest disagreement subsets contain nine or twelve items, making their percentages sensitive to individual outcomes.

D Qualitative Case Details

Figure 2 gives the retrieved exemplars and schema rules for the three cases discussed in the main text, preserving the question-level evidence alongside the aggregate disagreement analysis.

Table 9: **Repeated evaluation on the original split.** Difference between DG-Mem and No-memory mean@8 accuracy, in percentage points.

Backbone	MathVista	MMMU	MMMU-Pro
Gemini-3-Flash	+2.4	+2.9	+6.1
GPT-5-Nano	+7.6	+9.1	+8.8
Qwen3.5-122B-A10B	+4.2	+3.4	+7.3
Qwen3.5-27B	+3.4	+4.7	+2.9

Table 10: **Selected largest subject-level gains on MMMU-Pro with GPT-5-Nano.** Accuracy (%) and improvement (percentage points).

Subject	No-memory	DG-Mem	Improvement
Math	11.1	66.7	+55.6
Clinical	26.7	53.3	+26.7
Marketing	68.8	93.8	+25.0
Biology	52.4	76.2	+23.8
Chemistry	50.0	72.2	+22.2
Geography	35.0	55.0	+20.0
Manage	68.8	87.5	+18.8
Basic Medical Science	41.2	58.8	+17.6
Diagnostics	17.6	35.3	+17.6
Accounting	77.8	94.4	+16.7

Table 11: **Categorizer order sensitivity on the analyzed store (`CUSTOM`).** ARI and NMI compare each shuffle to the original assignments.

Ordering	Categories	ARI	NMI
Original	68	—	—
Shuffle 1	58	0.45	0.80
Shuffle 2	85	0.49	0.82
Shuffle 3	72	0.53	0.81

Table 12: **Combined-memory correctness conditional on single-grain disagreement.** Each cell gives correct/total and the rounded percentage.

Backbone	MathVista	MMMU	MMMU-Pro
Gemini-3-Flash	12/12 (100%)	10/13 (77%)	27/47 (57%)
GPT-5-Nano	20/31 (65%)	21/34 (62%)	41/70 (59%)
Qwen3.5-122B-A10B	14/17 (82%)	8/9 (89%)	24/32 (75%)
Qwen3.5-27B	14/22 (64%)	15/18 (83%)	40/71 (56%)

Test question	Retrieved exemplar ($\mathcal{M}_E$)	Retrieved schema rule ($\mathcal{M}_S$)	Baseline → Ours
MathVista #90. "Subtract all yellow matte blocks. Subtract all tiny brown cylinders. How many objects are left?"	Solved template: "Subtract all red things. Subtract all tiny matte balls. How many objects." *Strategy:* "Catalog every object's color, size, material, and shape, then translate the natural-language descriptors to those attribute categories before subtracting."	*Visual Counting and Attribute Analysis:* "Perform an exhaustive initial census of all objects, assign a count of zero to any missing categories, and calculate the cardinality of each subset *independently* before applying the subtraction."	**Gold:** 5 Baseline: 4 (skipped enumeration; "There are 4 objects left.") Ours: 5 (lists 7 objects, removes 2)
MathVista #13. "How many objects are preferred by more than 90% of people in at least one category?" (grouped bar chart)	Solved template: "How many algorithms have accuracy higher than 9 in at least one dataset?" *Verification check:* "Re-evaluate any bar that appears to end exactly on a midpoint or grid line to confirm whether it meets a 'strictly greater than' or 'strictly less than' condition."	*Data Analysis and Interpretation:* "When identifying threshold-based counts, *systematically* compute the metric for every distinct item rather than relying on visual estimation of distances or assumed ordering."	**Gold:** 0 Baseline: 2 (visually estimated bars near the 90% line) Ours: 0 (computes per-bar height, applies strict $>$)
MathVista #792. "What is the age gap between these two people?" (photo of P. Hammond and M. Kaljurand)	Solved template: same age-gap question structure on a different pair. *Verification check:* "Cross-verify the identity of the *less prominent* individual against the primary subject's known associates or event context, as misidentification of this figure is the most frequent source of error."	*Arithmetic: Age Problems and Data Extraction:* "Subtract the earlier birth year from the later birth year, ensuring the values belong to the *exact subjects specified* in the query."	**Gold:** 7 Baseline: 3 (used wrong birth year for Kaljurand: 1958) Ours: 7 (corrects to 1962, then $1962-1955$)

Figure 2: **Qualitative cases where DG-Mem corrects the No-memory backbone.** Each row is a held-out test instance for which both retrieved grains are populated and the same baseline error is corrected under *schema-only*, *exemplar-only*, and *both-memory* retrieval. The retrieved schema rule names the procedure the baseline omitted (an exhaustive census; per-item threshold computation; grounding the arithmetic in the correct entities), and the retrieved exemplar supplies a solved template whose verification check directly addresses the failure mode (boundary-bar evaluation; identity disambiguation of the less prominent subject).

E PROMPT TEMPLATES

This section reproduces the solver, concept summarizer, reflector, categorizer, and schema-synthesis prompt templates. It does not specify the complete runtime formatting or the judge and category-merge procedures. Each prompt is shown as a separate box; *system* prompts are highlighted in blue, *user* prompts in orange, and shared/utility templates in gray. Placeholders rendered as `{{name}}` are substituted at runtime with the corresponding field (e.g. `{{question}}` with the test problem text, `{{rollout_details}}` with the formatted solution trajectories and their correctness labels).

E.1 SOLVER PROMPT

The frozen backbone MLLM is invoked with a single, fixed system prompt that requests a step-by-step trace and a boxed final answer. The user turn carries the multimodal problem (image plus question text); at test time, schema guidance precedes the question and exemplar-based verification guidance follows it within the same answer-generation call.

Solver – System

```
Objective:
    Solve the given problem using a step by step process.

Expected Output Structure:
Step 1:
Step 2:
...
Step n: Final Answer: \boxed{answer}

Question:
```

E.2 CONCEPT SUMMARIZER ($\mathcal{K}$)

The concept summarizer produces the per-problem key-concept signature $\mathcal{K}(q)$ used both for online categorization during training and for memory retrieval at test time. It is a single user-turn prompt; no system prompt is used, and the model is instructed not to attempt a solution.

Problem Analysis – User

```
Objective:
    Analyze the following problem to identify its subject area and
    ↪ the
    key concepts, principles, formulas, or laws required for its
    solution. This analysis will be used to retrieve relevant guiding
    principles from a knowledge base.

Instructions:
    - Do not solve the problem.
    - First, identify the primary subject (e.g., Physics, Chemistry,
      Biology, Mathematics).
    - Then, list the core concepts or principles involved (e.g.,
    ↪ Newton's
      Second Law, Conservation of Energy, Stoichiometry, Pythagorean
      theorem).
    - Keep the analysis concise and focused.

Problem:
{{question}}

Output Format:
Subject: <The primary subject>
Key Concepts: <A brief list of key concepts>
```

E.3 Question-Level Reflector (Refl_R, writes to $\mathcal{M}_R$)

The question-level reflector condenses the labeled rollout group into IF-THEN rules that populate the transient reflection store $\mathcal{M}_R$. The abstraction constraints (no problem-specific numbers, options, or entities) are what make these rules safe to consolidate into the schema memory $\mathcal{M}_S$ in the post-epoch step.

Question-Level Reflector – System

```
You are an Expert Multimodal Problem-Solving Reflector. Your task is
to extract concise, generalizable IF-THEN rules from multiple
solution attempts for a single scientific or mathematical problem.
You must analyze both visual interpretation errors (e.g., misreading
diagrams, missing chart labels) and logical reasoning errors. These
rules will be clustered by concept to serve as foundational strategic
heuristics for future AI agents.
```

Question-Level Reflector – User

```
Given the following multimodal problem, its key concepts, and
↪  multiple
solution attempts with their verification results, extract
generalizable IF-THEN rules.

<problem>
{{question}}
</problem>

<gold_answer>
{{gold_answer}}
</gold_answer>

<key_concepts>
{{key_concepts}}
</key_concepts>

<rollouts>
{{rollout_details}}
</rollouts>

Instructions:
1. Analyze the rollouts to identify the critical decision points,
   visual interpretations, or logical steps that determined success
   or failure.
2. Formulate rules as actionable IF-THEN statements.
   - The "IF" part should describe a specific problem context, visual
     pattern, or conceptual trigger (derived from the
     ↪  <key_concepts>).
   - The "THEN" part should provide a specific visual or logical
     intervention (e.g., "THEN explicitly verify the axis scale",
     "THEN apply the Pythagorean theorem").
3. Rule Categories (Try to extract at least one of each if
↪  applicable):
   - Positive Heuristics: "IF <context>, THEN <strategy>" -- what
     specifically worked in the correct attempts.
   - Visual/Perceptual Traps: "IF <visual context>, THEN AVOID
     <perceptual pitfall>" -- visual elements the agent
     misinterpreted in failed attempts.
   - Logical/Calculation Traps: "IF <logical context>, THEN AVOID
     <reasoning pitfall>" -- logical fallacies or formula errors
```

```
     observed in failed attempts.
4. Constraints for Generalization:
   - Rules MUST be tightly connected to the provided <key_concepts>
     so they can be accurately grouped with similar problems later.
   - Keep rules to a single, concise sentence.
5. Return 2-5 rules total. Quality and transferability over quantity.

CRITICAL CONSTRAINT -- Abstraction:
Rules MUST NOT contain problem-specific numbers, option letters
(A/B/C/D), yes/no answers, variable names (unless standard formulas),
or specific entity names. If a rule mentions any value from the
gold_answer or rollouts, rephrase it abstractly.

GOOD: "IF measuring length with a ruler, THEN align the start of the
       object with the zero mark and read the nearest whole unit at
       the endpoint."
BAD:  "IF measuring length, THEN the endpoint is at the 8 cm mark."
       (contains problem-specific value)
GOOD: "IF comparing counts of object categories, THEN systematically
       enumerate each category before comparing."
BAD:  "IF counting brown bikes, THEN there are 0 brown tandem bikes."
       (contains problem-specific entity and count)

Output Format:
Return a JSON object with a single key "rules" whose value is a list
of 2-5 IF-THEN rule strings. Each rule must start with "IF " and
contain "THEN ". If no meaningful rules can be extracted, return
{"rules": []}.
```

E.4 Exemplar Reflector (Refl_E, writes to $\mathcal{M}_E$)

The exemplar reflector produces the `reasoning_strategy` and `verification_check` fields associated with each source problem in $\mathcal{M}_E$. Both reflectors receive the labeled rollouts; the schema synthesizer receives the question-level reflections rather than the exemplar payload.

Compact Reflector – System

```
You are an Expert Multimodal Problem Analyst. Your task is to extract
a generalizable reasoning strategy and verification check from
multiple solution attempts. Your output will guide a different AI
agent solving a DIFFERENT problem of the same type -- it must be
abstract enough to transfer, not specific to this instance.
```

Compact Reflector – User

```
Given a multimodal problem, its key concepts, and multiple solution
attempts, extract a transferable reasoning strategy.

<problem>
{{question}}
</problem>

<gold_answer>
{{gold_answer}}
</gold_answer>

<key_concepts>
{{key_concepts}}
```

```
</key_concepts>

<rollouts>
{{rollout_details}}
</rollouts>

Instructions:
1. Analyze where correct and incorrect approaches diverge.
2. Produce a JSON object with these fields:
   - "reasoning_strategy": A step-by-step reasoning procedure that
     would work for ANY problem of this type. Describe WHAT to do
     and HOW to verify, not what the answer is.
   - "verification_check": A specific check the solver should perform
     to catch the most common error pattern (or null if all attempts
     were correct).

CRITICAL CONSTRAINTS:
- NEVER include specific numerical values, option letters (A/B/C/D),
  yes/no answers, or entity names from this problem.
- The output must work for a DIFFERENT problem with DIFFERENT numbers
  and images.
- Frame positively: describe what TO DO, not what to avoid.

GOOD examples:
- reasoning_strategy: "Align the object's starting edge with the zero
  mark, then read the position of the ending edge to the nearest
  whole unit."
- verification_check: "Re-read the ending position independently --
  ensure it is not confused with an adjacent marking."

BAD examples (contain problem-specific values -- DO NOT do this):
- reasoning_strategy: "Count 8 individual cubes and add to 30 to get
  ↪  38"
- verification_check: "The endpoint is at the 8 cm mark, not the 7 cm
  ↪  mark"

Output Format (return ONLY this JSON, no other text):
```json
{
  "reasoning_strategy": "<generalizable step-by-step reasoning
  ↪  procedure>",
  "verification_check": "<specific verification check or null>"
}
```
```

E.5 Online Concept Categorizer (Cat)

The categorizer is the operational implementation of the *online* taxonomy growth described in §3.1. It receives the running registry of categories and decides whether to assign the new problem to an existing one or to spawn a new category, returning a structured action that the registry consumes verbatim.

Online Categorizer – System

```
You are an Expert Problem Categorizer for multimodal scientific and
mathematical problems. Your task is to assign each problem to a
concept category, either selecting an existing category or creating
a new one when no existing category is a good fit.
```

Online Categorizer – User

```
Given a problem's key concepts and the current set of existing
categories, decide whether to assign the problem to an existing
category or create a new one.

<problem_concepts>
{{key_concepts}}
</problem_concepts>

<existing_categories>
{{existing_categories}}
</existing_categories>

Instructions:
1. Compare the problem's key concepts against each existing
   category's name and id.
2. If a category is a strong conceptual match (>70% overlap in domain
   and principles), assign to it.
3. If no category is a good fit, create a new one with a descriptive
   name and representative concepts.
4. When assigning to an existing category, you may update its
   representative_concepts to better reflect the expanded set of
   problems.

Output Format (return ONLY this JSON, no other text):
```json
{
  "action": "assign" or "create",
  "category_id": "<existing category_id if assign, or a new unique
  ↪  slug if create>",
  "category_name": "<human-readable category name>",
  "representative_concepts": ["concept1", "concept2", ...],
  "reasoning": "<1-sentence justification>"
}
```
```

E.6 Schema Synthesizer (Synth, writes to $\mathcal{M}_S$)

The schema synthesizer is the post-epoch consolidation step. It reads only the IF-THEN rules in $\mathcal{M}_R$ within a category (never the exemplar payload) and emits the prescriptive advice list that constitutes the category's entry in $\mathcal{M}_S$.

Category-Level Reflector – System

```
You are an Expert Multimodal Knowledge Synthesizer. Your task is to
analyze a cluster of question-level IF-THEN rules within a specific
scientific or mathematical concept category. You must synthesize
these isolated experiences into coherent, highly generalizable advice
to guide future AI agents in solving problems within this category.
```

Category-Level Reflector – User

```
You are given a concept category, its representative concepts, and a
collection of IF-THEN rules extracted from solving multimodal
problems in this category. Synthesize them into coherent,
non-redundant, and actionable advice.

<category>
```

```
{{category_name}}
</category>

<representative_concepts>
{{representative_concepts}}
</representative_concepts>

<question_level_rules>
{{rules_text}}
</question_level_rules>

Instructions:
1. Analyze and group the provided rules, eliminating redundancies and
   resolving contradictions. Elevate problem-specific rules into
   general principles applicable to the entire <category>.
2. Produce a single flat list of advice items covering:
   - Visual/perceptual strategies (interpreting images, charts,
     diagrams, spatial relationships).
   - Logical/reasoning strategies (workflows, theorems, formulas,
     step-by-step approaches).
3. Each advice item must be:
   - A single, self-contained, and actionable sentence starting with
     ↪  "- ".
   - **Prescriptive** -- tell the solver WHAT TO DO, not just what to
     avoid. Use "Do X when you see Y" rather than "Never assume X".
   - **Specific** -- reference concrete visual patterns, mathematical
     relationships, or domain-specific checks. Avoid abstract
     ↪  warnings.
   - Broadly generalizable across any problem that tests the
     <representative_concepts>.
4. Return 2-4 items total. Focus on the 2-4 most critical and
   specific insights.

IMPORTANT -- quality criteria for each rule:
- BAD (too vague): "Never assume post-event birth years from visual
  cues." -- This tells the solver to not do something but gives no
  actionable alternative.
- GOOD (prescriptive): "When estimating ages from photos to compare
  against a historical date, assess each person independently and
  count anyone who appears under 80 as likely born after 1945." --
  This gives a concrete decision procedure.
- BAD (too generic): "Always verify exact values against labeled axis
  ticks." -- This applies to any chart problem and adds no insight.
- GOOD (specific): "For scatter plots with overlapping points, trace
  each data point's y-value to the nearest labeled tick mark before
  comparing against the threshold." -- This addresses a specific
  visual challenge.

5. If no meaningful advice can be extracted, return {"advice": []}.

Output Format:
Return a JSON object with a single key "advice" whose value is a list
of 2-4 advice strings. Each string must be a single, self-contained
sentence.
```

F DG-MEM ALGORITHMS

Algorithms 1–3 specify the construction, attribution, and inference procedures described in Section 3 and illustrated by the two-stage pipeline in Figure 1. All language-model parameters remain frozen. The gold answers in $\mathcal{D}_{\text{train}}$ are used only in the offline stages; Algorithm 3 receives no gold answer and does not update memory.

Notation and shared operations. $\mathcal{L}_q$ contains sampled trajectories paired with their correctness labels. Verify denotes the native-matcher, symbolic-parser, and LLM-judge cascade. Γ is the category registry, and $\kappa(q)$ is the assignment of problem q. Category finalization denotes the post-epoch merge described in Section 3.1; the resulting assignments determine the reflection groups used for synthesis. The reflector and synthesizer calls use the prompt templates in Appendix E.

Algorithm 1 Stage 1: construct exemplar and schema memory

Require: Labeled training partition $\mathcal{D}_{\text{train}}$; frozen backbone MLLM;
concept summarizer $\mathcal{K}$; reflectors $\text{Refl}_R, \text{Refl}_E$; categorizer Cat;
synthesizer Synth; embedders $\text{Emb}_T, \text{Emb}_M$; rollout count $N = 5$
Ensure: Exemplar memory $\mathcal{M}_E$; schema memory $\mathcal{M}_S$; category registry Γ

Initialize Γ, reflection store $\mathcal{M}_R$, and exemplar memory $\mathcal{M}_E$ as empty
for each $(q, a_q^\star) \in \mathcal{D}_{\text{train}}$ in training order **do**
  $k_q \leftarrow \mathcal{K}(q)$; initialize $\mathcal{L}_q \leftarrow \emptyset$
  for $i = 1, \ldots, N$ **do**
    $o_i \leftarrow \text{MLLM}(q, \emptyset)$
    $y_i \leftarrow \text{Verify}(o_i, a_q^\star)$; append (o_i, y_i) to $\mathcal{L}_q$
  end for
  $\rho(q) \leftarrow \text{Refl}_R(q, a_q^\star, k_q, \mathcal{L}_q)$
  $e(q) \leftarrow \text{Refl}_E(q, a_q^\star, k_q, \mathcal{L}_q)$
  $\kappa(q) \leftarrow \text{Cat}(k_q, \Gamma)$; update Γ if a new category is created
  Store $(q, \kappa(q), \rho(q))$ in $\mathcal{M}_R$
  Form t_q from the problem text and concept summary k_q
  Store $(q, e(q), \rho(q), \kappa(q), \text{Emb}_M(I_q, t_q))$ in $\mathcal{M}_E$
end for
Finalize Γ and update category tags in $\mathcal{M}_R$ and $\mathcal{M}_E$
Initialize $\mathcal{M}_S \leftarrow \emptyset$
for each category $\kappa \in \Gamma$ **do**
  $\mathcal{P}_\kappa \leftarrow \bigcup_{q:\kappa(q)=\kappa} \rho(q)$
  $\sigma_\kappa \leftarrow \text{Synth}(\kappa, \text{concepts}(\kappa), \mathcal{P}_\kappa)$
  Store the advice items in σ_κ with category tags and text embeddings in $\mathcal{M}_S$
end for
return $(\mathcal{M}_E, \mathcal{M}_S, \Gamma)$

Schema retrieval. The shared operation RETRIEVESCHEMAS$(q, k_q, \Gamma, \mathcal{M}_S, U, \alpha)$ first selects up to two categories by concept similarity, subject to the 0.40 threshold. Within each category, it filters rules by the 0.35 similarity threshold and returns up to two rules ranked by Eq. (5). Similarity and utility are normalized independently within each active category. Stage 2 uses similarity-only ranking ($\alpha = 1$), before utilities have been estimated; inference uses $\alpha = 0.8$. These caps bound the returned set by four rules. Thresholds and caps are listed in Table 6.

Algorithm 2 Stage 2: estimate schema utilities from rule-subset evaluations

Require: $\mathcal{D}_{\text{train}}$; frozen MLLM; $\mathcal{M}_S, \Gamma$ from Algorithm 1; $B = 8$
Ensure: Utility $U(r)$ for every schema rule $r \in \mathcal{M}_S$

Freeze category assignments and schema text; exclude exemplar memory from this stage
Initialize $A(r) \leftarrow 0$, $n(r) \leftarrow 0$, and $U_0(r) \leftarrow 0$ for every $r \in \mathcal{M}_S$
for each $(q, a_q^\star) \in \mathcal{D}_{\text{train}}$ **do**
    $k_q \leftarrow \mathcal{K}(q)$ ▷ The Stage-1 concept summary may be reused
    $\mathcal{R}_q \leftarrow \textsc{RetrieveSchemas}(q, k_q, \Gamma, \mathcal{M}_S, U_0, 1)$
    $K_q \leftarrow |\mathcal{R}_q|$; $b_q \leftarrow \min(B, 2^{K_q})$
    if $b_q = 2^{K_q}$ **then**
        $\mathcal{C}_q \leftarrow 2^{\mathcal{R}_q}$ ▷ Enumerate the full powerset
    else
        $\mathcal{C}_q \leftarrow \{\emptyset, \mathcal{R}_q\}$
        Sample $b_q - 2$ distinct nonempty proper subsets uniformly without replacement
        Add the sampled subsets to $\mathcal{C}_q$
    end if
    Initialize $\widetilde{v}_q(S) \leftarrow 0$ for every $S \subseteq \mathcal{R}_q$
    for each $S \in \mathcal{C}_q$ **do**
        $o_S \leftarrow \text{MLLM}(q, S)$ ▷ Keep problem and decoding settings fixed
        $\widetilde{v}_q(S) \leftarrow \text{Verify}(o_S, a_q^\star)$
    end for
    for each $r \in \mathcal{R}_q$ **do**
        Compute $\Delta_q(r)$ from $\widetilde{v}_q$ using Eq. (6)
        $A(r) \leftarrow A(r) + \Delta_q(r)$; $n(r) \leftarrow n(r) + 1$
    end for
end for
$U(r) \leftarrow A(r)/\max(n(r), 1)$ for every $r \in \mathcal{M}_S$
return U

Algorithm 3 Frozen-memory inference on a held-out problem

Require: Test problem q; frozen MLLM, $\mathcal{M}_E, \mathcal{M}_S, \Gamma, U$; $\alpha = 0.8$; $\tau_E = 0.75$
Ensure: Predicted answer $\hat{a}_q$

$k_q \leftarrow \mathcal{K}(q)$
$\mathcal{R}_q \leftarrow \textsc{RetrieveSchemas}(q, k_q, \Gamma, \mathcal{M}_S, U, \alpha)$
Form t_q from q and k_q; compute $z_q \leftarrow \text{Emb}_M(I_q, t_q)$
Initialize exemplar context $E_q \leftarrow \emptyset$
if $\mathcal{M}_E \neq \emptyset$ **then**
    Find the exemplar $e^\star$ with maximum cosine similarity to z_q
    if its similarity is at least τ_E **then**
        $E_q \leftarrow \{e^\star\}$
    end if
end if
Place schema guidance $\mathcal{R}_q$ before q and exemplar guidance E_q after q
$\hat{a}_q \leftarrow \text{MLLM}(q, \mathcal{R}_q \cup E_q)$ ▷ One answer-generation call
return $\hat{a}_q$ ▷ All memory stores and utilities remain unchanged

The empty candidate set is permitted: it has one subset, $\emptyset$, and contributes no rule updates. For partially observed powersets, Algorithm 2 uses the same zero-imputation as Section 3.3. Even with full coverage, attribution is exact only for the observed reward table; stochastic generation can change that table across runs.