FROZEN MODELS, EVOLVING EXPERTISE: MODEL-AGNOSTIC LEARNING FROM DEPLOYMENT EXPERIENCE FOR MULTIMODAL MEDICAL AI

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) and vision-language models (VLMs) are usually frozen after deployment, so they do not learn from the cases they solve. This is especially concerning in medicine, where new clinical evidence, updated guidelines, and new therapies can change established practice. Fine-tuning can update the model, but it requires access to model weights and additional training. Parameter-free methods avoid training, but they may overfit a fixed validation set, lack reliable domain knowledge, or lose visual details by saving experience only as text. To address these limitations, we present a model-agnostic framework that allows frozen LLMs and VLMs to learn from deployment experience through three forms of external expertise: a *Skill* that guides reasoning and tool use, a *Knowledge Memory* that stores reliable facts supported by earlier cases or trusted external evidence, and a *Multimodal Knowledge Base* that keeps visual examples and guides the model to relate each retrieved case to the current image. Instead of relying on a fixed validation set, a validation strategy keeps an update only if it helps on new cases without degrading performance on earlier ones. Across six benchmarks covering clinical diagnosis, clinical workflows, medical reasoning, and medical and non-medical visual reasoning, and with four open-weight and closed-source base models, our framework improves performance during online deployment by up to 34.2% over the base model on medical tasks, generalizes to unseen cases, transfers to other models without further optimization, and works in non-medical domains.

1 INTRODUCTION

Large language models (LLMs) and vision-language models (VLMs) can now solve complex problems, use external tools, and reason over both text and images (Yao et al., 2023; Liu et al., 2023). However, once these models are deployed, their parameters are usually frozen, so their internal knowledge and capabilities do not automatically improve with new experience (Zhao et al., 2024; Majumder et al., 2023). This limitation is especially problematic in medicine, where new clinical evidence, updated guidelines, and new therapies and drugs can change established clinical practice (Shojania et al., 2007). During deployment, a model can generate experience: it solves cases, searches for evidence, makes mistakes, and receives feedback. A verified clinical case may reveal a better diagnostic procedure, a reliable medical fact, or a useful visual pattern. If a model cannot reuse this experience, it has to start from scratch whenever it sees a similar case and may make the same mistakes again (Shinn et al., 2023; Zhao et al., 2024). This raises an important question: *how can a frozen model learn from deployment experience and develop stronger capabilities in a specific domain or task like medicine?*

Recent work has explored how models can learn from interactions and feedback, either by saving past episodes and reusable procedures outside the model (Shinn et al., 2023; Zhao et al., 2024; Majumder et al., 2023; Alzubi et al., 2026; Yang et al., 2026; Zhang et al., 2026) or by updating model parameters through fine-tuning or reinforcement learning on interaction trajectories (Song et al., 2024; Yuan et al., 2025).

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.

However, improving a deployed model in these ways has practical limitations. Training requires access to the model weights, which may be impossible for closed-source models (Zhao et al., 2024). Training also incurs additional costs and depends on the quality of training data (Shinn et al., 2023; Song et al., 2024; Yuan et al., 2025). Parameter-free methods such as EvoSkill and SkillOpt avoid retraining, but they use a fixed validation set to select the updates to their skills document (Alzubi et al., 2026; Yang et al., 2026). Repeatedly selecting updates on the same validation cases can lead to overfitting and hinder generalization to new cases (Dwork et al., 2016). Moreover, experience derived only from interaction trajectories may lack the reliable domain knowledge needed for new cases. Finally, methods such as Reflexion, ExpeL, and ACE save learned experience as text (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2026), which can omit useful visual details.

To address these limitations, we develop an experience-driven learning framework for medical AI. It improves frozen LLMs and VLMs by accumulating and updating external expertise. The framework maintains three components. The *Skill* provides reusable guidance for reasoning and tool use. It is periodically revised based on which reasoning and tool-use steps led to high or low task scores on earlier cases. A *Knowledge Memory* stores reliable facts supported by earlier cases or trusted external evidence. For multimodal tasks, a *Multimodal Knowledge Base (MMKB)* saves earlier visual cases so that the original visual evidence remains available to later reasoning. These three components address general needs in medicine: reusing better diagnostic procedures, reliable medical facts, and useful visual patterns. To avoid overfitting a fixed validation set, we also design a validation strategy that tests each update to the learned expertise on new and earlier cases to keep only updates that generalize to new cases without degrading performance on earlier ones. In this way, the system can improve during deployment while the model parameters remain unchanged.

We evaluate the framework on six benchmarks covering interactive clinical diagnosis, clinical workflows, text-based medical reasoning, and medical and non-medical visual reasoning (Schmidgall et al., 2025; Liu et al., 2025; Zhou et al., 2025; Yao et al., 2026; Xu et al., 2025; Cao et al., 2026). Our framework improves online performance over the base model by up to 34.2% on medical tasks, generalizes to unseen cases, and transfers to other models without further optimization. These results show that deployment experience can produce reusable expertise.

Our main contributions are as follows:

- **Continual improvement through external expertise.** We develop a framework that enables frozen LLMs and VLMs to continually improve during deployment without training by turning experience into reusable external expertise. Because it does not require access to model weights, it applies to both open-weight and closed-source models.
- **Persistent multimodal expertise.** We introduce MMKB, which preserves images, source answers, and available explanations from reference data and completed cases. It guides the model to examine the evidence behind each reference answer and to compare relevant observations and conditions with the current case, so that visual patterns learned from previous cases can improve reasoning on new images. Used together with the Skill and Knowledge Memory, MMKB further improves performance on visual tasks.
- **Reusable expertise across models and domains.** Expertise learned with one model can be reused by other models without further optimization, so it does not need to be relearned when the base model changes. The framework also improves non-medical visual reasoning tasks, showing that it is not limited to medicine.

2 Related Work

External memory and reusable procedures. Parameter-free methods preserve expertise outside the base model. Many of them accumulate reusable experience from earlier trajectories, such as reflections, strategies, guidance, or executable skills, and add it directly (Shinn et al., 2023; Zhao et al., 2024; Majumder et al., 2023; Suzgun et al., 2026; Ouyang et al., 2026; Li et al., 2025a; Fu et al., 2024; Chen et al., 2024; Wang et al., 2024; 2023; Zheng et al., 2025; Ni et al., 2026; Zhang et al., 2026). Others, such as GEPA, EvoSkill, and SkillOpt, generate candidate updates and select them by their performance on a held-out validation set (Agrawal et al., 2026; Alzubi et al., 2026; Yang et al., 2026). However, experience added without checking its effect may not help new cases, repeatedly optimizing and validating on the same fixed set of cases can overfit that set, and memories

and procedures derived only from interaction trajectories may lack reliable domain knowledge. Our framework instead tests each candidate update on new cases and a representative buffer of past cases, and Knowledge Memory adds supported facts from verified cases or trusted external sources.

Fine-tuning and soft prompting. Many methods update model parameters on interaction trajectories through supervised fine-tuning (Chen et al., 2023; Zeng et al., 2023; Song et al., 2024) or reinforcement learning (Yuan et al., 2025; Luo et al., 2025). These methods require access to model weights, incur substantial training costs, typically rely on many reliable trajectories, and must be repeated for each base model. Gare et al. (2026) keep the VLM backbone frozen and train a few continuous prompt tokens, which can transfer to a newer model version, but training them still requires gradients through the model weights, so the method applies only to open-weight models. Our framework instead updates external expertise from deployment experience without gradient access, so it also applies to closed-source models, and the learned expertise can transfer to other models without further optimization.

Multimodal memory and retrieval. Most parameter-free experience learning methods return learned experience to the model as text. MIRIX and M3-Agent instead retain multimodal observations for long-term recall (Wang & Chen, 2025; Long et al., 2026), while Med-RwR, RULE, and MMed-RAG retrieve textual or visual evidence for medical VLMs (Wang et al., 2025; Xia et al., 2024; 2025). However, these systems mainly provide the retrieved content as context, without explicit instructions to analyze it or compare it with the current case. MMKB instead guides the model to examine how each retrieved visual case, with its known answer, relates to the current image.

3 METHOD

3.1 FRAMEWORK OVERVIEW

Our goal is to let a frozen model improve at a task by learning from verified deployment experience. As shown in Figure 1, the framework maintains a *Skill* for reasoning and tool-use procedures, a *Knowledge Memory* for reusable facts, and a *MMKB* for visual examples. Together, they provide guidance on how to solve a task and evidence that may be missing from the model's own knowledge.

Problem setting. A frozen base model processes a stream of cases in batches $F_1, F_2, \ldots$, where F_t denotes the t-th batch. Let $\mathcal{E}_t = (S_t, K_t, V_t)$ denote the Skill, Knowledge Memory, and MMKB used for batch F_t, and let $s_i(\mathcal{E}_t)$ denote the task score of case i when the base model uses $\mathcal{E}_t$. The model answers every case in F_t before the task scores and source answers of these cases are released, so this feedback can affect only later batches. The source answer of a case is the reference answer or report provided by its dataset, and we call a case verified once its source answer is released. S_1 and K_1 are empty, and for visual tasks V_1 contains the initial reference data. The goal is to update $\mathcal{E}_t$ so that later cases receive higher scores, while the model parameters remain fixed.

The base model receives the Skill S_t in its task instructions, queries Knowledge Memory K_t when needed, and retrieves MMKB references from V_t for visual tasks. After each batch, an LLM optimizer proposes a new Skill and knowledge additions from scored trajectories, and these updates are validated before deployment (Section 3.5). The MMKB V_t grows from completed visual cases with source answers.

3.2 LEARNING REUSABLE PROCEDURES

The Skill is a natural-language document that describes useful steps, their conditions of use, and when additional actions are unnecessary. For example, a diagnostic Skill may suggest checking evidence that distinguishes competing diagnoses before ordering another test. The complete document is available from the start of each case, and the model applies it according to current evidence and task requirements.

After each batch, the optimizer reviews the current Skill and Memory, the scored trajectories, and the comparison results of the previous candidate (Section 3.5). Based on successful and unsuccessful actions, it rewrites the whole Skill document: it adds missing steps, revises overly broad rules, removes unhelpful guidance, and keeps the Skill within a length limit. The optimizer rewrites instead

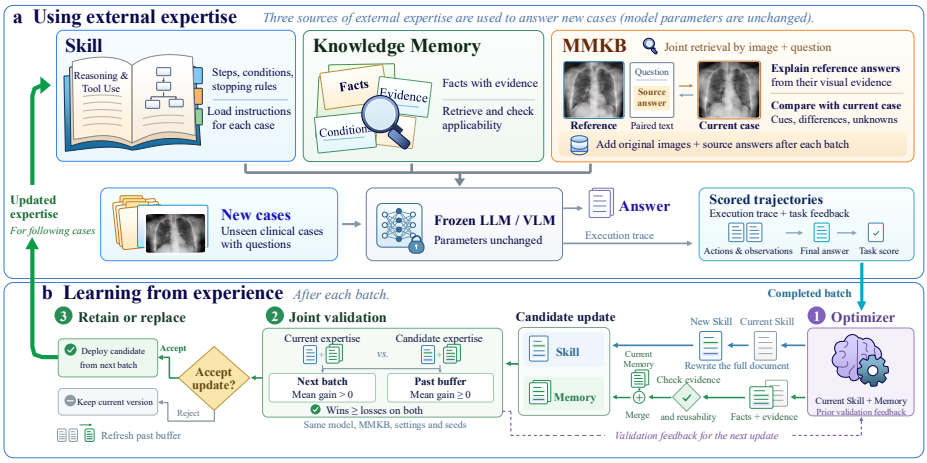

Figure 1: **Framework overview.** (a) A frozen model uses Skill, Knowledge Memory, and MMKB to answer new cases. MMKB supports comparison with visual references and their source answers. (b) The optimizer proposes new Skills and supported knowledge additions from scored trajectories. The candidate update is tested on new and past cases before deployment, and validation feedback guides the next update.

of appending because appending makes the Skill, and thus the model's context, longer after every batch and can leave conflicting rules; both can degrade performance. For example, if one batch suggests "order a chest X-ray for patients with cough and fever" and a later batch shows that the X-ray is often unnecessary when vital signs and lung examination are normal, rewriting merges them into one rule: "order a chest X-ray for patients with cough and fever when vital signs or lung examination are abnormal." The optimizer is instructed not to include case identifiers or answer keys, and the proposed Skill is validated before it is used on subsequent cases.

3.3 Acquiring and Reusing Knowledge

Even with a better procedure, the model may lack facts needed for a decision. Knowledge Memory stores facts and relations with supporting evidence and conditions of use. When the model asks a knowledge question through a tool call, text retrieval finds potentially relevant entries, and a separate model checks whether they directly answer the question and apply to the case. If none is suitable, the system searches trusted external sources, such as PubMed.

Knowledge candidates come from these searches and from the optimizer's analysis of completed cases. Case-based proposals must cite supporting trajectories and quote their evidence. The model checks that the evidence exists, supports the fact, and justifies reuse beyond the original case. Multiple supporting cases are normally required; a single case is sufficient only with direct support from a trusted external source. Patient-specific observations and individual answer keys are excluded. Supported facts are merged with the current Memory to form a candidate update. Factual support alone does not establish that an addition will improve decisions, so we test the updated expertise before deployment.

3.4 Learning from Visual References

Textual lessons may omit visual details. MMKB retains original images, their questions or context, source answers, and explanations when available. It is initialized from reference cases outside the evaluated stream (Table 13) and grows through completed deployment cases.

A fixed multimodal encoder jointly represents each case's images and available question text. Cosine similarity retrieves relevant references.

MMKB guides how retrieved references are used, rather than only which references are retrieved. For each reference, the model examines how its images and context support the source answer,

then identifies which observations and conditions also apply to the current case. It must consider differences, missing evidence, and contradictions before answering. This comparison keeps the current decision grounded in the current input while drawing on experience from earlier cases.

After batch F_t, all cases in F_t that have images and source answers are added to V_t with their source answers and existing explanations, not the model's predictions, to form V_{t+1}. Even an incorrectly answered case can therefore become a useful reference. These references are retained even if proposed updates to Skill and Memory are rejected or earlier versions are restored. New references are available from the following batch. Retrieval also excludes the current case, cases from its source group, and duplicate images (Appendix B).

3.5 Validating Updates During Deployment

To test whether an update helps beyond the cases that produced it, we compare the current expertise $\mathcal{E}_t$ with a candidate version $\widetilde{\mathcal{E}}_t = (\widetilde{S}_t, \widetilde{K}_t, V_t)$ that includes Skill and Memory updates proposed after batch F_{t-1}, where $\widetilde{S}_t$ and $\widetilde{K}_t$ denote the candidate Skill and Knowledge Memory. Both versions are tested on F_t and a representative buffer of past cases B_t. The buffer is initialized with cases from F_1, so the first comparison takes place at $t = 2$. The base model first answers F_t using the current expertise, and these answers are final; the candidate's answers are used only for the comparison.

Both versions are evaluated under the same base model, MMKB V_t, task settings, and random seed for each case. When a past case from B_t is replayed, its own MMKB entry, including its images and source answer, is excluded from retrieval. For $D \in \{F_t, B_t\}$, define

$$\overline{\Delta}_D = \frac{1}{|D|} \sum_{i \in D} \left[s_i(\widetilde{\mathcal{E}}_t) - s_i(\mathcal{E}_t) \right], \tag{1}$$

where s_i is the task score. Let W_D and L_D count cases on which the candidate scores higher and lower. We accept the update only if

$$\overline{\Delta}_{F_t} > 0, \quad W_{F_t} \geq L_{F_t}, \qquad\qquad \overline{\Delta}_{B_t} \geq 0, \quad W_{B_t} \geq L_{B_t}. \tag{2}$$

New cases must show improvement, while past cases may tie but not decline on average. For tasks with binary scores, the win–loss condition follows from the mean condition. For tasks with continuous scores, it additionally rejects a candidate whose mean gain comes from large improvements on a few cases while more cases decline. If the update is accepted, $\mathcal{E}_{t+1} = (\widetilde{S}_t, \widetilde{K}_t, V_{t+1})$.

After each comparison, an LLM refreshes the buffer using only task goals and question texts, not answers or scores. It replaces some buffer cases with cases from F_t so that the buffer covers the range of questions seen so far. Periodically, the current version is also compared on the buffer with an earlier retained version, and the earlier Skill and Memory are restored if the current version performs worse. These checks reduce the risk of overfitting to reused validation cases.

The optimizer then uses trajectories generated with the selected version on the completed batch to propose the next update. Appendices B and C give the key settings and prompt templates.

4 Experiments

We test whether deployment experience improves a frozen model, whether each expertise component contributes to that improvement, and whether the learned expertise remains useful on unseen cases or with a different base model. These comparisons distinguish reusable expertise from corrections that help only the cases used for learning.

4.1 Experimental Setup

Benchmarks and models. We evaluate six benchmarks. AgentClinic-MedQA (Schmidgall et al., 2025), MedChain (Liu et al., 2025), and MedThink-Bench (Zhou et al., 2025) cover interactive diagnosis, multi-stage clinical workflows, and text-based medical reasoning. MedThinkVQA (Yao et al., 2026) adds multi-image diagnosis. VisuLogic (Xu et al., 2025) and QCalEval (Cao et al., 2026) test visual logic and quantum-calibration reasoning beyond medicine. We use Qwen3.8-27B, Nemotron-3-Nano-Omni-30B, GPT-5.6-Sol, and GPT-4o-mini as frozen base models, with GPT-5.6-Sol as the optimizer. MMKB is enabled only for image tasks.

Comparisons. Base retains each benchmark's native task interface and external knowledge searches, without Skill, Knowledge Memory, or MMKB. We also compare with ACE (Zhang et al., 2026), which maintains an evolving context, and SkillOpt (Yang et al., 2026), which learns reusable Skills. ACE is included in the online and frozen comparisons; SkillOpt is included in the frozen comparison. Task inputs and scoring rules are matched within each comparison.

Online protocol. Each benchmark follows a fixed case order with seed 42, and our framework updates after batches of 32 cases. All predictions are recorded before their feedback is used for learning. Appendix B gives dataset sizes, retrieval settings, and input limits.

Evaluation. We report diagnosis accuracy (%) for AgentClinic-MedQA; answer accuracy (%) for MedThink-Bench, MedThinkVQA, and VisuLogic; the official overall score for QCalEval; and MedChain's Average on a 0–1 scale. MedChain's Average covers six metrics across its five tasks (Liu et al., 2025); Appendix B gives the definition. The online score uses only the deployed model's original predictions; candidate evaluations and replay do not contribute. Frozen and transfer evaluations retain local expertise retrieval but disable optimization, writes, and new external searches. Tables 1 and 4 report these scores and their relative change from Base.

4.2 Online Improvement During Deployment

This comparison tests whether earlier experience improves later decisions. Each method starts with its own initial state and follows the same complete stream. Expertise is fixed within a batch, and the full-stream score includes the initial period before any update has been accepted. Figure 2 shows the scores of Base and Ours with GPT-5.6-Sol as more cases are processed.

Ours improves every base model on every benchmark. Ours scores above Base in all 24 benchmark–model pairs and achieves the best score in 22 of them (Table 1). Its average gain over Base is 17.6%, compared with 5.6% for ACE, which falls below Base in 7 pairs; where ACE scores higher, the margin is small (0.404 vs. 0.400 and 26.4 vs. 26.1). We attribute this stability to the validation strategy: each update is kept only if it helps on new cases without degrading performance on earlier ones, whereas ACE adds new experience without such a check.

Gains are largest where learned expertise can be reused. Gains reach 34.2% on AgentClinic and 81.9% on VisuLogic, and exceed 16% on QCalEval for every base model. These benchmarks involve multi-step consultations or recurring question types, where a learned procedure or visual reference is likely to help many later cases. Gains are smaller on MedThink-Bench and MedThinkVQA (0.4%–5.0%), whose independent questions likely depend more on the model's own medical knowledge. Figure 2 shows that the gap between Ours and Base usually opens after the first few batches and then persists.

The improvement is robust to case order. Because the framework learns from a stream, its gains could depend on the order of cases. We therefore repeat the online experiment with GPT-5.6-Sol on AgentClinic and QCalEval under five random case orders (Table 2). Ours improves on Base by 5.7% and 16.5% on average, and both gaps are at least five times the largest standard deviation. The fixed-order results in Table 1 also lie within one standard deviation of these means.

4.3 Ablation Study

To isolate the role of each component, we compare the full framework with variants that remove Skill, Knowledge Memory, or MMKB. Each variant learns independently from the start of the same stream using one selected base model; it is not formed by removing a component after training. Removing Memory disables stored knowledge but keeps external knowledge searches, while removing MMKB disables visual retrieval and growth. Text-only tasks need no separate MMKB ablation. We also test a variant that accepts every update without validation (Section 3.5). We use GPT-5.6-Sol as the base model, and Table 3 reports the results.

Each component accounts for a different part of the gain. Among the three components, removing the Skill causes the largest drops on text-based and multi-step tasks, 12.1% on MedChain

Benchmark	Nemotron-3-Nano-Omni-30B			GPT-4o-mini			Qwen3.8-27B			GPT-5.6-Sol		
	Base	ACE	Ours	Base	ACE	Ours	Base	ACE	Ours	Base	ACE	Ours
AgentClinic	42.1	43.9 +4.3%	**56.5** +34.2%	42.5	43.5 +2.4%	**54.2** +27.5%	67.3	66.8 −0.7%	**73.4** +9.1%	79.0	73.4 −7.1%	**83.6** +5.8%
MedChain	0.334	0.341 +2.1%	**0.375** +12.3%	0.353	**0.404** +14.4%	0.400 +13.3%	0.413	0.443 +7.3%	**0.460** +11.4%	0.435	0.497 +14.3%	**0.513** +17.9%
MedThink-Bench	42.2	42.2 0.0%	**42.8** +1.4%	37.8	38.7 +2.4%	**39.0** +3.2%	64.0	59.4 −7.2%	**64.6** +0.9%	68.4	70.2 +2.6%	**71.8** +5.0%
MedThink VQA	25.4	**26.4** +3.9%	26.1 +2.8%	23.9	23.7 −0.8%	**24.5** +2.5%	48.3	47.4 −1.9%	**48.5** +0.4%	68.3	65.9 −3.5%	**68.9** +0.9%
VisuLogic	14.4	21.4 +48.6%	**23.9** +66.0%	26.6	23.6 −11.3%	**29.3** +10.2%	21.5	29.9 +39.1%	**39.1** +81.9%	45.8	47.3 +3.3%	**50.0** +9.2%
QCalEval	47.3	48.6 +2.7%	**55.5** +17.3%	40.9	46.0 +12.5%	**58.5** +43.0%	62.9	63.4 +0.8%	**82.3** +30.8%	72.3	76.9 +6.4%	**84.2** +16.5%

Table 1: Online performance on the fixed stream. Higher is better for all metrics. The number below each score gives its relative change from Base (green: gain; red: decline). The best score for each base model is shaded.

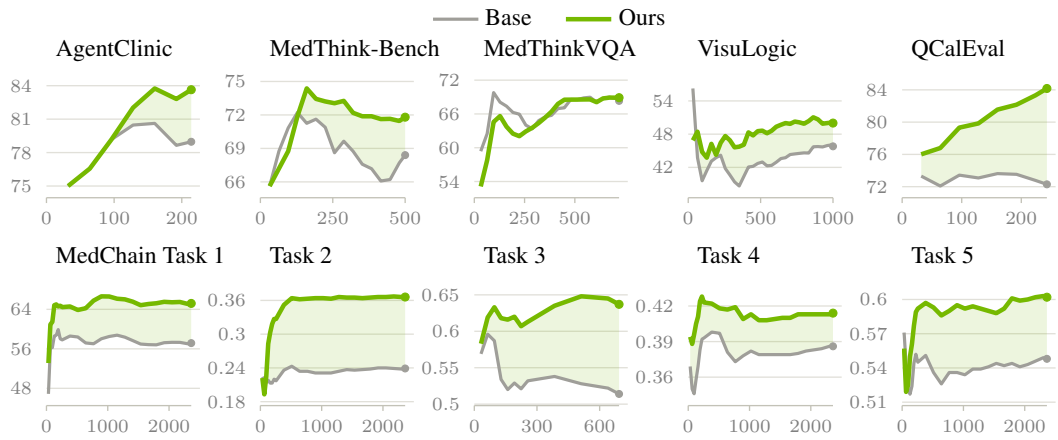

Figure 2: Online scores of Base and Ours with GPT-5.6-Sol along the fixed stream. The x-axis counts processed cases, and each point is the score over all of them. The top row uses the metrics in Section 4.1; the bottom row shows the five MedChain tasks, whose metrics are listed in Table 5. The last points equal the scores in Tables 1 and 5. The shaded area marks where Ours is above Base.

and 5.5% on AgentClinic, where the score returns to Base (Table 3). Removing MMKB causes the largest drops on visual tasks, 14.0% on QCalEval and 8.2% on VisuLogic, and brings both back to their Base scores (72.4 vs. 72.3 and 45.9 vs. 45.8), so nearly all of the visual gain comes from MMKB. Removing Knowledge Memory has a smaller effect, largest on MedThink-Bench (2.5%), which tests medical knowledge. Within MedChain, removing MMKB changes only Task 3, the task with images, whereas removing the Skill lowers all five tasks (Table 6). The three components are therefore complementary rather than redundant.

Validation is necessary for the gains. Accepting every update without validation lowers the score on all six benchmarks by 21.8% to 37.8% relative to Full and leaves every score below Base (Table 3). Updates that hurt performance are then kept and used on later cases, so validation is needed for the learned expertise to help rather than harm.

MMKB versus standard multimodal retrieval. We also test whether MMKB's reference analysis adds value beyond standard multimodal retrieval. On image-based tasks, both conditions use GPT-5.6-Sol under the online protocol, keep all other components and task settings the same, and receive identical retrieved references with images, context, and source answers. The baseline uses

Benchmark	Base	Ours
AgentClinic	79.3 ± 0.7	**83.8 ± 0.9**
QCalEval	71.3 ± 1.8	**83.1 ± 1.5**

Table 2: Online results of GPT-5.6-Sol under five random case orders (mean ± standard deviation).

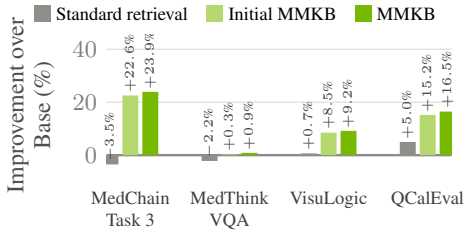

Figure 3: Standard multimodal retrieval versus MMKB with GPT-5.6-Sol. Initial MMKB does not add deployment cases. Absolute scores are in Table 7.

Benchmark	Base	Full	−Skill	−Memory	−MMKB	−Validation
AgentClinic	79.0	83.6	79.0 −5.5%	83.2 −0.5%	= Full	53.5 −36.0%
MedChain	0.435	0.513	0.451 −12.1%	0.505 −1.6%	0.494 −3.7%	0.401 −21.8%
MedThink-Bench	68.4	71.8	69.4 −3.3%	70.0 −2.5%	= Full	51.2 −28.7%
MedThink VQA	68.3	68.9	68.9 0.0%	68.9 0.0%	68.3 −0.9%	52.1 −24.4%
VisuLogic	45.8	50.0	49.6 −0.8%	50.0 0.0%	45.9 −8.2%	38.1 −23.8%
QCalEval	72.3	84.2	82.8 −1.7%	84.2 0.0%	72.4 −14.0%	52.4 −37.8%

Table 3: Online component ablation with GPT-5.6-Sol. Full is the complete framework; −Skill, −Memory, and −MMKB remove the Skill, the stored Knowledge Memory (external searches remain), and MMKB, respectively; −Validation accepts every update without validation. The number below each ablated score gives its relative change from Full.

these references as context and can still reason before answering, but it is not asked to analyze each reference or compare it with the current case. We also test Initial MMKB, which uses only the initial references (Table 13) and does not add deployment cases. Figure 3 reports the results.

MMKB gains come from how references are used. With identical references, standard retrieval changes the score by -3.5% to $+5.0\%$ relative to Base, whereas MMKB improves all four image-based tasks, by up to 23.9% on MedChain Task 3 (Figure 3). Standard retrieval even lowers the score on MedChain Task 3 and MedThinkVQA, suggesting that references given only as context can distract the model when their relation to the current image is not analyzed. MMKB turns the same references into gains by guiding the model to examine each reference and compare it with the current case. Initial MMKB still improves all four tasks, by up to 22.6% on MedChain Task 3, and adding deployment cases increases the gain further on every task.

4.4 Frozen Generalization to Unseen Cases

To separate generalization from continued adaptation, each method learns from the first approximately 70% of the fixed stream and is then evaluated on the remaining cases with learning disabled. Cases sharing a patient, source, or experiment identity stay in the same split. We freeze the final Skill, Knowledge Memory, and MMKB, including only visual additions from the learning prefix. Test feedback is not available to the learner.

The learned expertise generalizes to unseen cases. After being frozen, Ours scores above Base on the test split in 21 of 24 pairs and achieves the best score in 19 (Table 4). Its average gain over Base is 20.7%, compared with 8.3% for SkillOpt and 1.9% for ACE. This gain is comparable to the online gain in Section 4.2, suggesting that the expertise captures reusable procedures and knowledge rather than corrections to the learning cases. The baselines are less stable on unseen cases: SkillOpt falls below Base in 7 pairs and ACE in 12, whereas Ours falls below Base only once (-3.6% on MedThinkVQA with Nemotron-3-Nano-Omni-30B). This matches the motivation of our validation strategy. SkillOpt repeatedly selects updates on a fixed validation set, and ACE adds experience without validation, so their updates can fit the learning cases without carrying over to new ones.

4.5 Zero-Step Cross-Model Transfer

This experiment tests whether expertise learned with one model can improve a different model without further learning. We use two source models. When GPT-5.6-Sol is the source, it serves as both the base model answering cases and the optimizer; when Qwen3.8-27B is the source, it

Benchmark	Nemotron-3-Nano-Omni-30B				GPT-4o-mini				Qwen3.8-27B				GPT-5.6-Sol			
	Base	SkillOpt	ACE	Ours	Base	SkillOpt	ACE	Ours	Base	SkillOpt	ACE	Ours	Base	SkillOpt	ACE	Ours
AgentClinic	40.0	39.4 −1.5%	40.2 +0.5%	**58.5** +46.3%	38.5	**53.8** +39.7%	41.5 +7.8%	52.3 +35.8%	67.7	67.7 0.0%	64.6 −4.6%	**73.8** +9.0%	76.9	75.4 −2.0%	72.3 −6.0%	**80.0** +4.0%
MedChain	0.336	0.367 +9.2%	0.331 −1.5%	**0.377** +12.2%	0.356	0.379 +6.5%	0.330 −7.3%	**0.389** +9.3%	0.431	0.450 +4.4%	0.433 +0.5%	**0.456** +5.8%	0.442	**0.526** +19.0%	0.497 +12.4%	0.520 +17.6%
MedThink-Bench	42.7	**44.7** +4.7%	43.7 +2.3%	42.7 0.0%	34.7	36.7 +5.8%	38.0 +9.5%	**39.3** +13.3%	64.7	57.7 −10.8%	60.4 −6.6%	**66.0** +2.0%	72.0	72.4 +0.6%	71.2 −1.1%	**76.0** +5.6%
MedThink VQA	25.0	**25.0** 0.0%	21.3 −14.8%	24.1 −3.6%	23.0	21.9 −4.8%	21.3 −7.4%	**23.0** 0.0%	51.4	**53.7** +4.5%	51.6 +0.4%	53.2 +3.5%	69.0	69.4 +0.6%	67.7 −1.9%	**69.4** +0.6%
VisuLogic	14.3	23.1 +61.5%	15.3 +7.0%	**23.7** +65.7%	27.0	22.7 −15.9%	21.7 −19.6%	**27.3** +1.1%	21.7	27.8 +28.1%	30.2 +39.2%	**41.7** +92.2%	49.3	48.0 −2.6%	46.3 −6.1%	**50.7** +2.8%
QCalEval	43.2	42.7 −1.2%	41.0 −5.1%	**54.8** +26.9%	36.8	53.4 +45.1%	52.1 +41.6%	**61.1** +66.0%	58.8	62.4 +6.1%	59.2 +0.7%	**87.9** +49.5%	69.0	70.8 +2.6%	72.2 +4.6%	**89.7** +30.0%

Table 4: Frozen generalization after learning on the first approximately 70% of the fixed stream and evaluating on the remaining cases. Source groups are kept intact. Higher is better for all metrics. The number below each score gives its relative change from Base (green: gain; red: decline). The best score for each base model is shaded.

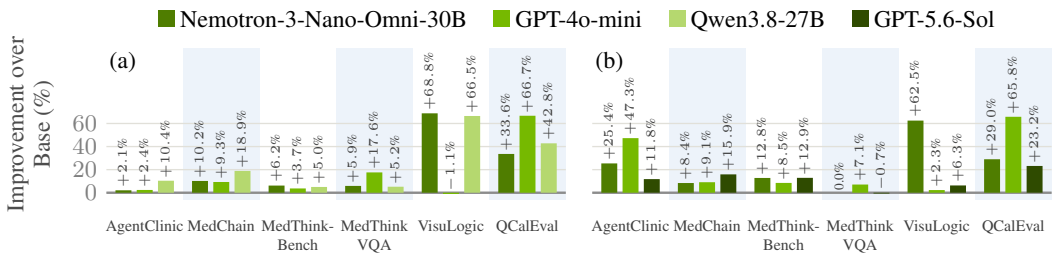

Figure 4: Zero-step transfer with (a) GPT-5.6-Sol and (b) Qwen3.8-27B as the source. Bars show the relative change from each target's Base (the zero line); absolute scores are in Tables 9 and 10.

answers cases and GPT-5.6-Sol serves as the optimizer. For each source, we freeze the final Skill, Knowledge Memory, and MMKB from its complete online run, then transfer them unchanged to the other three models.

Each target is evaluated on the complete benchmark without further learning. References matching the current case or detected as duplicates remain excluded. We compare against the same target model's full-stream Base result under matching evaluation conditions. Figure 4 reports the results for both source models.

Expertise transfers across models without further learning. Transferred expertise improves the target model in 33 of the 36 transfer results, with an average gain of 20.0% over the target's Base (Figure 4). Expertise learned with the open-weight Qwen3.8-27B also improves the closed-source GPT-5.6-Sol, by 11.8% on AgentClinic and 23.2% on QCalEval, and gives the weaker GPT-4o-mini a 47.3% gain on AgentClinic. Because the expertise is kept outside the model, a new or upgraded base model can use it immediately instead of learning it again.

5 Conclusion

We presented a framework that improves frozen LLMs and VLMs by accumulating external expertise during deployment. It maintains a Skill for reasoning and tool use, a Knowledge Memory for reliable facts, and an MMKB for visual cases with known answers, and it tests updates to the Skill and Knowledge Memory on new and past cases before deployment. Across six benchmarks, the framework improved performance during online deployment, generalized to unseen cases, transferred to other models without further optimization, and also worked in non-medical domains.

AI USE STATEMENT

We used generative AI tools to clean and reformat datasets, create and edit software code, create and modify scientific figures and images, and improve the readability of the manuscript. The authors take responsibility for the final content and all AI-assisted contributions.

ETHICS STATEMENT

Our evaluation uses existing benchmarks and reference materials; it does not establish clinical safety or effectiveness. The system may reproduce errors and biases in models or reference data. Any clinical use would require independent validation, privacy safeguards, and qualified human oversight.

REPRODUCIBILITY STATEMENT

Appendix B provides the implementation details, including data splits, reference sources, input limits, generation settings, and the safeguards against answer leakage in MMKB. Appendix C provides the core prompt templates and message construction. We will open-source the code upon acceptance.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning, 2026. URL https://arxiv.org/abs/2507.19457.

Hany Ali, Jorge Marques, Ophelia Crawford, Joonas Majaniemi, Marc Serra-Peralta, David Byfield, Boris Varbanov, Barbara M. Terhal, Leonardo DiCarlo, and Earl T. Campbell. Reducing the error rate of a superconducting logical qubit using analog readout information. *Physical Review Applied*, 22(4), October 2024. ISSN 2331-7019. doi: 10.1103/physrevapplied.22.044031. URL http://dx.doi.org/10.1103/PhysRevApplied.22.044031.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.02766.

Shuxiang Cao, Zijian Zhang, Abhishek Agarwal, Grace Bratrud, Niyaz R. Beysengulov, Daniel C. Cole, Alejandro Gómez Frieiro, Elena O. Glen, Hao Hsu, Gang Huang, Raymond Jow, Greshma Shaji, Tom Lubowe, Ligeng Zhu, Luis Mantilla Calderón, Nicola Pancotti, Joel Pendleton, Brandon Severin, Charles Etienne Staub, Sara Sussman, Antti Vepsäläinen, Neel Rajeshbhai Vora, Yilun Xu, Varinia Bernales, Daniel Bowring, Elica Kyoseva, Ivan Rungger, Giulia Semeghini, Sam Stanwyck, Timothy Costa, Alán Aspuru-Guzik, and Krysta Svore. Qcaleval: Benchmarking vision-language models for quantum calibration plot understanding, 2026. URL https://arxiv.org/abs/2604.25884.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning, 2023. URL https://arxiv.org/abs/2310.05915.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning, 2024. URL https://arxiv.org/abs/2405.16247.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Preserving statistical validity in adaptive data analysis, 2016. URL https://arxiv.org/abs/1411.2664.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents, 2024. URL https://arxiv.org/abs/2403.08978.

Gautam Rajendrakumar Gare, Siyi Li, Hewei Wang, Cesar Daniel Hernandez, Wei Zhao, Wolfgang M. Pauli, John Galeotti, and Deva Ramanan. Your model already knows don't teach it, learn to ask it: Soft prompting for few-shot adaptation of vision-language models, 2026. URL https://arxiv.org/abs/2609.11310.

Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, and Yang Liu. Agent hospital: A simulacrum of hospital with evolvable medical agents, 2025a. URL https://arxiv.org/abs/2405.02957.

Xiao Li, Jia-Yi Hou, Jia-Chao Wang, Guang-Wei Wang, Xiao-Dong He, Feng Zhou, Yi-Bo Wang, Min Liu, Jin Wang, Peng Xu, and Ming-Sheng Zhan. A fiber array architecture for atom quantum computing. *Nature Communications*, 16(1), November 2025b. ISSN 2041-1723. doi: 10.1038/s41467-025-64738-8. URL http://dx.doi.org/10.1038/s41467-025-64738-8.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Jie Liu, Wenxuan Wang, Zizhan Ma, Guolin Huang, Yihang SU, Kao-Jung Chang, Wenting Chen, Haoliang Li, Linlin Shen, and Michael Lyu. Medchain: Bridging the gap between llm agents and clinical practice with interactive sequence, 2025. URL https://arxiv.org/abs/2412.01605.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. In *International Conference on Learning Representations*, volume 2026, pp. 146197–146246, 2026.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning, 2025. URL https://arxiv.org/abs/2508.03680.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. Clin: A continually learning language agent for rapid task adaptation and generalization, 2023. URL https://arxiv.org/abs/2310.10134.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills, 2026. URL https://arxiv.org/abs/2603.25158.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. 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, 2026. URL https://arxiv.org/abs/2509.25140.

J. E. Padilla-Castillo, S. Hofsäss, L. Palánki, J. Cai, C. J. H. Rich, R. Thomas, S. Kray, G. Meijer, S. C. Wright, and S. Truppe. A large magneto-optical trap of cadmium atoms loaded from a cryogenic buffer gas beam, 2025. URL https://arxiv.org/abs/2506.01180.

Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor. Agentclinic: a multimodal agent benchmark to evaluate ai in simulated clinical environments, 2025. URL https://arxiv.org/abs/2405.07960.

Aruku Senoo, Alexander Baumgärtner, Joanna W. Lis, Gaurav M. Vaidya, Zhongda Zeng, Giuliano Giudici, Hannes Pichler, and Adam M. Kaufman. High-fidelity entanglement and coherent multiqubit mapping in an atom array, 2025. URL https://arxiv.org/abs/2506.13632.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Kaveh G Shojania, Margaret Sampson, Mohammed T Ansari, Jun Ji, Steve Doucette, and David Moher. How quickly do systematic reviews go out of date? a survival analysis. *Annals of internal medicine*, 147(4):224–233, 2007.

Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Agentbank: Towards generalized llm agents via fine-tuning on 50000+ interaction trajectories. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2124–2141, 2024.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Lehan Wang, Yi Qin, Honglong Yang, and Xiaomeng Li. Proactive reasoning-with-retrieval framework for medical multimodal large language models, 2025. URL https://arxiv.org/abs/2510.18303.

Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents, 2025. URL https://arxiv.org/abs/2507.07957.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024. URL https://arxiv.org/abs/2409.07429.

Peng Xia, Kangyu Zhu, Haoran Li, Hongtu Zhu, Yun Li, Gang Li, Linjun Zhang, and Huaxiu Yao. Rule: Reliable multimodal rag for factuality in medical vision language models, 2024. URL https://arxiv.org/abs/2407.05131.

Peng Xia, Kangyu Zhu, Haoran Li, Tianze Wang, Weijia Shi, Sheng Wang, Linjun Zhang, James Zou, and Huaxiu Yao. Mmed-rag: Versatile multimodal rag system for medical vision language models, 2025. URL https://arxiv.org/abs/2410.13085.

Weiye Xu, Jiahao Wang, Weiyun Wang, Zhe Chen, Wengang Zhou, Aijun Yang, Lewei Lu, Houqiang Li, Xiaohua Wang, Xizhou Zhu, Wenhai Wang, Jifeng Dai, and Jinguo Zhu. Visulogic: A benchmark for evaluating visual reasoning in multi-modal large language models, 2025. URL https://arxiv.org/abs/2504.15279.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills, 2026. URL https://arxiv.org/abs/2605.23904.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Zonghai Yao, Benlu Wang, Yifan Zhang, Junda Wang, Iris Xia, Zhipeng Tang, Shuo Han, Feiyun Ouyang, Zhichao Yang, Arman Cohan, and Hong Yu. Medical thinking with multiple images, 2026. URL https://arxiv.org/abs/2604.16506.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training, 2025. URL https://arxiv.org/abs/2501.11425.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms, 2023. URL https://arxiv.org/abs/2310.12823.

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. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL https://arxiv.org/abs/2308.10144.

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.

Shuang Zhou, Wenya Xie, Jiaxi Li, Zaifu Zhan, Meijia Song, Han Yang, Cheyenna Espinoza, Lindsay Welton, Xinnie Mai, Yanwei Jin, Zidu Xu, Yuen-Hei Chung, Yiyun Xing, Meng-Han Tsai, Emma Schaffer, Yucheng Shi, Ninghao Liu, Zirui Liu, and Rui Zhang. Automating expert-level medical reasoning evaluation of large language models, 2025. URL https://arxiv.org/abs/2507.07988.

A Detailed Experimental Results

MedChain	Nemotron-3-Nano-Omni-30B		GPT-4o-mini		Qwen3.8-27B		GPT-5.6-Sol	
	Base	Ours	Base	Ours	Base	Ours	Base	Ours
Task 1	53.8 / 0.243	58.0 / 0.305	59.1 / 0.304	59.9 / 0.366	58.9 / 0.360	63.0 / 0.393	57.2 / 0.351	65.2 / 0.406
Task 2	0.225	0.289	0.174	0.323	0.241	0.340	0.239	0.366
Task 3	0.238	0.244	0.190	0.203	0.410	0.416	0.514	0.637
Task 4	0.353	0.360	0.362	0.363	0.374	0.414	0.386	0.414
Task 5	0.406	0.473	0.500	0.543	0.501	0.564	0.548	0.602

Table 5: Per-task MedChain results during online deployment (Section 4.2). Task 1 reports first-level referral accuracy (%) / second-level referral IoU, Task 2 history-taking examination-item IoU, Task 3 macro claim recall, Task 4 normalized diagnosis score, and Task 5 treatment IoU. The mean of these six metrics on a 0–1 scale gives the MedChain Average in Table 1.

MedChain	Base	Full	−Skill	−Memory	−MMKB	−Validation
Task 1	57.2 / 0.351	65.2 / 0.406	57.3 / 0.361	64.9 / 0.402	= Full	54.4 / 0.330
Task 2	0.239	0.366	0.241	0.364	= Full	0.218
Task 3	0.514	0.637	0.604	0.613	0.523	0.488
Task 4	0.386	0.414	0.387	0.397	= Full	0.327
Task 5	0.548	0.602	0.539	0.604	= Full	0.501
Average	0.435	0.513	0.451	0.505	0.494	0.401

Table 6: Per-task MedChain results of the online component ablation with GPT-5.6-Sol; the other benchmarks are reported in Table 3. Task 1 reports first-level referral accuracy (%) / second-level referral IoU, Task 2 history-taking examination-item IoU, Task 3 macro claim recall, Task 4 normalized diagnosis score, and Task 5 treatment IoU. Average is the mean of these six metrics on a 0–1 scale (Appendix B). −MMKB equals Full on tasks that do not use MMKB.

Benchmark	Base	Standard retrieval	Initial MMKB	MMKB
MedChain Task 3	0.514	0.496	0.630	0.637
MedThinkVQA	68.3	66.8	68.5	68.9
VisuLogic	45.8	46.1	49.7	50.0
QCalEval	72.3	75.9	83.3	84.2

Table 7: Absolute scores for the comparison in Figure 3 with GPT-5.6-Sol. MedChain Task 3 reports macro claim recall, and the other benchmarks use the metrics in Section 4.1. Standard retrieval and MMKB receive identical retrieved references; Initial MMKB uses only the initial references (Table 13) and does not add deployment cases.

MedChain	Nemotron-3-Nano-Omni-30B		GPT-4o-mini		Qwen3.8-27B		GPT-5.6-Sol	
	Base	Ours	Base	Ours	Base	Ours	Base	Ours
Task 1	52.1 / 0.237	57.5 / 0.292	58.4 / 0.308	58.4 / 0.369	60.8 / 0.375	58.5 / 0.361	57.8 / 0.358	65.5 / 0.424
Task 2	0.219	0.278	0.179	0.325	0.239	0.345	0.246	0.366
Task 3	0.281	0.266	0.188	0.192	0.431	0.434	0.496	0.635
Task 4	0.358	0.383	0.362	0.317	0.407	0.421	0.409	0.423
Task 5	0.400	0.465	0.512	0.545	0.529	0.592	0.562	0.617

Table 8: Per-task MedChain results on unseen cases (Section 4.4). Expertise is learned on the learning split and frozen during evaluation on the test split. Metrics follow Table 5, and their mean gives the MedChain Average in Table 4.

Benchmark	Nemotron-3-Nano-Omni-30B		GPT-4o-mini		Qwen3.8-27B	
	Base	Transfer	Base	Transfer	Base	Transfer
AgentClinic	42.1	43.0	42.5	43.5	67.3	74.3
MedChain	0.334	0.368	0.353	0.386	0.413	0.491
MedThink-Bench	42.2	44.8	37.8	39.2	64.0	67.2
MedThinkVQA	25.4	26.9	23.9	28.1	48.3	50.8
VisuLogic	14.4	24.3	26.6	26.3	21.5	35.8
QCalEval	47.3	63.2	40.9	68.2	62.9	89.8
MedChain Task 1	53.8 / 0.243	61.1 / 0.314	59.1 / 0.304	61.6 / 0.347	58.9 / 0.360	67.5 / 0.436
MedChain Task 2	0.225	0.233	0.174	0.244	0.241	0.369
MedChain Task 3	0.238	0.261	0.190	0.218	0.410	0.463
MedChain Task 4	0.353	0.346	0.362	0.368	0.374	0.397
MedChain Task 5	0.406	0.445	0.500	0.524	0.501	0.608

Table 9: Absolute scores for zero-step transfer with GPT-5.6-Sol as the source (Figure 4a). Base is each target model's full-stream Base result from Table 1; Transfer uses the frozen expertise learned with GPT-5.6-Sol without further learning. Each benchmark uses the metric in Section 4.1, and MedChain Tasks 1–5 use the metrics in Table 5.

Benchmark	Nemotron-3-Nano-Omni-30B		GPT-4o-mini		GPT-5.6-Sol	
	Base	Transfer	Base	Transfer	Base	Transfer
AgentClinic	42.1	52.8	42.5	62.6	79.0	88.3
MedChain	0.334	0.362	0.353	0.385	0.435	0.504
MedThink-Bench	42.2	47.6	37.8	41.0	68.4	77.2
MedThinkVQA	25.4	25.4	23.9	25.6	68.3	67.8
VisuLogic	14.4	23.4	26.6	27.2	45.8	48.7
QCalEval	47.3	61.0	40.9	67.8	72.3	89.1
MedChain Task 1	53.8 / 0.243	60.4 / 0.308	59.1 / 0.304	60.2 / 0.345	57.2 / 0.351	67.3 / 0.447
MedChain Task 2	0.225	0.275	0.174	0.278	0.239	0.382
MedChain Task 3	0.238	0.223	0.190	0.192	0.514	0.509
MedChain Task 4	0.353	0.332	0.362	0.362	0.386	0.401
MedChain Task 5	0.406	0.430	0.500	0.531	0.548	0.609

Table 10: Absolute scores for zero-step transfer with Qwen3.8-27B as the source (Figure 4b). Base is each target model's full-stream Base result from Table 1; Transfer uses the frozen expertise learned with Qwen3.8-27B without further learning. Each benchmark uses the metric in Section 4.1, and MedChain Tasks 1–5 use the metrics in Table 5.

B Additional Method and Experimental Details

B.1 Learning Settings and Input Limits

The online batch contains 32 cases, not 32 model calls. The representative buffer also contains 32 cases; four are replaced after each full batch. Every four batches after the first accepted update, we compare the current version with a retained earlier version, using the same MMKB snapshot. A shorter final batch is evaluated but does not trigger optimization. Thus, benchmark size determines the number of update opportunities, while trajectory length and image count determine how much evidence fits into an optimizer request.

The optimizer proposes one complete Skill and at most four knowledge additions per batch. The Skill is limited to 24,000 characters. Knowledge retrieval considers five entries and returns at most two that directly answer the query and satisfy applicability checks, with model-reported confidence at least 0.55. MMKB returns up to three eligible references. Answers and source explanations are excluded from the retrieval representations. GPT-4o-mini performs the knowledge checks; text-embedding-ada-002 and fixed Qwen3-VL-Embedding-2B provide text and multimodal embeddings, respectively.

Input limits do not change the validation batch. MedThinkVQA uses the 16 lowest-scoring trajectories from the 32-case batch to propose an update, but validates on all 32. Its optimizer and the QCalEval optimizer receive current-case images; the VisuLogic optimizer also receives reference images. For MedThinkVQA inference, current images take priority within a limit of 50 images, or 32 for hosted Nemotron, with remaining capacity shared across references. These limits distinguish the experience evaluated from the portion available for proposing an update.

Each comparison uses one execution per case and version. Score comparisons use tolerance 10^{-12}. At a periodic comparison, the current version becomes the retained version if it has a positive mean gain and at least as many wins as losses. A negative mean gain with at least as many losses as wins restores the earlier Skill and Memory; MMKB is unchanged. Otherwise, the current version remains in use. Matching seeds reduces controllable variation but does not guarantee deterministic model responses.

B.2 Generation Settings

Table 11 separates the base model's generation budget from the optimizer and knowledge checks. These are output limits, not the length of the final answer. A reasoning model may count internal reasoning toward its total output budget. The optimizer receives public explanations and tool traces, not private reasoning traces.

Role	Output tokens	Reasoning and temperature
GPT-5.6-Sol base model	24,576	High reasoning effort; temperature omitted on routes that do not accept it.
GPT-4o-mini base model	16,384	Temperature 0; no separate thinking setting.
Qwen / Nemotron base models	24,576	Temperature 0; thinking budget 8,192 for AgentClinic and MedChain, 16,384 for the four other hosted tasks.
Optimizer (GPT-5.6-Sol)	24,576	High reasoning effort.
Buffer selector (GPT-5.6-Sol)	4,096	High reasoning effort.
Knowledge checks (GPT-4o-mini)	4,096	Temperature 0.

Table 11: Documented generation settings. The MedChain optimizer and buffer-selector overrides use 65,536 and 16,384 output tokens, respectively; the Skill length limit is unchanged.

Knowledge queries are limited to two per case, or two per subquestion in QCalEval. AgentClinic permits up to 20 consultation actions, with knowledge queries counted separately. MedChain history-taking permits up to 10 dialogue turns. These task budgets are distinct from the number of cases in a learning batch.

Benchmark or task pool	Full stream	Learning	Test
AgentClinic-MedQA	214	149	65
MedChain (Tasks 1, 2, 4, 5)	2,362	1,660	702
MedChain (Task 3, visual cases)	693	484	209
With claim annotations	561	389	172
MedThink-Bench	500	350	150
MedThinkVQA	720	504	216
VisuLogic	1,000	700	300
QCalEval	243	170	73

Table 12: Evaluation pools and fixed learning/test splits. Counts refer to cases. MedChain's visual subset inherits the main case split.

B.3 Data and Evaluation

All conditions use a fixed case order with seed 42. Splits keep each patient, source group, and quantum experiment intact. One QCalEval case contains all six questions for an experiment, giving 1,458 questions in total. The MedChain split is approximately 70/30 because groups crossing the boundary are kept on the learning side. These are custom splits of public data, not official hidden test sets.

AgentClinic uses final-diagnosis accuracy; MedThink-Bench, MedThinkVQA, and VisuLogic use answer accuracy. QCalEval averages its six task scores within each experiment and then across experiments, using the benchmark's rounding rules and the mean of GPT-5.4 and Gemini-3.1-Pro-Preview grading.

Our MedChain implementation strictly follows the sequential workflow described by Liu et al. (2025): each task receives only the model's outputs from the preceding tasks. In contrast, Table 1 of the original paper reports results in which each task receives the standardized case information. Our scores are therefore not directly comparable to those in that table.

MedChain's Average is the arithmetic mean of six reported metrics across five tasks: first-level referral accuracy, second-level referral IoU, history-taking examination-item IoU, macro claim recall, normalized diagnosis score, and treatment IoU (Liu et al., 2025). The two referral metrics enter separately, rather than being combined before averaging across tasks. Our diagnosis reporting divides the five-point score by five; the internal update-selection score uses $(\text{score} - 1)/4$ instead. Claim recall is averaged over annotated cases. Tasks 2 and 5 have 1,984 and 1,921 cases with scorable targets, respectively; their learning/test counts are 1,386/598 and 1,349/572. Missing target annotations do not count as incorrect predictions.

B.4 Visual Reference Sources

Task	Reference data
MedChain (visual stage)	2,722 case–modality records from 2,454 additional cases outside the target pool, with original reports.
MedThinkVQA	A fixed 1,024-case subset of official training data, with existing source descriptions.
VisuLogic	4,292 references from official training data, with source answers and available solutions.
QCalEval	82 references (83 images) adapted from public software examples, experimental source data, and published figures; sources are listed in Table 14.

Table 13: Initial MMKB libraries. Counts describe stored references, not the number retrieved per case. Original annotations are retained where available; QCalEval sources are adapted into reference questions and answers.

Preventing answer leakage in MMKB. A deployment case is added to MMKB with its source answer only after its whole batch has been answered and scored. Each case can therefore retrieve

only cases from earlier batches, and cases in the same batch cannot retrieve one another. A check on batch positions enforces this order, and a run stops with an error if the order is violated. Retrieval also skips the current case, cases from the same source group, and images that are byte-identical, identical after decoding, or near duplicates at thumbnail resolution. Frozen evaluation retains the initial reference library and additions from the learning split only, and transfer retains the source model's final full-stream library; both apply the same exclusions. In online validation, a past case replayed from the buffer cannot retrieve its own entry, including its images and source answer.

QCalEval sources. Table 14 identifies the initial references and the versions used to construct them. Repository examples provide data and analysis results; published source data are plotted, and selected paper panels are cropped or rendered. Reference questions and answers are then assembled from these materials. Reported values, computed fits, and explanatory notes are distinguished; the references are not uniformly expert-authored question–answer pairs. The 12 source-data references and one figure reference from Li et al. come from the same study, not independent studies.

Source	Count	Material used
Quantify-core (344c8bc)	34	Example data and analysis code; references use raw data or recomputed fits.
Qibocal (e0ba08a)	26	Calibration test data, existing fits, ZZ data, and a documentation figure.
QTT (77a5a86)	1	Barrier-gate pinch-off scan (B8.dat), with metadata and notebook.
Li et al., source data (v1)	12	Plots generated from public measurement spreadsheets, with paper descriptions and reported parameters.
Li et al. (2025b)	1	Fig. 1c: atom-array rearrangement.
Padilla-Castillo et al. (2025)	2	arXiv v1, Figs. 2a and 5b: magneto-optical-trap loading and accumulation.
Senoo et al. (2025)	4	arXiv v2 (2025), Extended Data Figs. 4b–d, 4e–f, and 7c–d; main Fig. 3b–d: Ramsey, Rabi, gate benchmarking, and interaction spectra.
Ali et al. (2024)	2	Figs. 2a–c and 10a–d: two-state and three-state readout distributions.

Table 14: Sources of the 82 initial QCalEval MMKB references. Links identify repository revisions, dataset versions, or the paper versions used. The last four rows contribute nine references with ten images.

C PROMPTS AND MESSAGE CONSTRUCTION

We report the fixed prompts that govern Skill execution, update proposals, knowledge checks, and visual reference use. The boxes reproduce the documented prompt text; double braces mark values filled at runtime, not text sent literally to the model. Role labels and box headings are explanatory.

A learned Skill is an output of optimization, not a fixed prompt shared by all runs. Skill and Knowledge Memory start empty; image tasks also use the initial MMKB libraries in Table 13. The base model receives the current Skill, permitted tool responses, and any retrieved references. It does not receive the current case's answer or score. The optimizer receives completed trajectories and their released feedback. Graders have access to the scoring targets but do not revise the model's answer.

C.1 EXECUTING THE CURRENT SKILL

The task instructions define the allowed inputs, tools, and answer format. The following protocol specifies how the current Skill is used within that task. AgentClinic retains its native multi-turn messages; the other task adapters insert the Skill into their task-specific system message.

Skill execution protocol

```
Generic Skill execution protocol:
1. Treat the learned Skill as executable policy, not as facts about
the input.
2. Execute `CORE` once in written order; treat unlabelled procedural
instructions as `CORE`.
3. At each decision point, evaluate each `WHEN` against the current
input and current execution state. Execute only the matching `DO`; an
unmatched rule has no effect.
4. After an action changes the execution state, re-evaluate the
remaining rules.
5. End a rule when its `STOP` condition holds, and do not repeat it
unless the Skill explicitly says to do so.
6. The task contract and direct input evidence override conflicting
Skill text.
Keep rule selection and execution bookkeeping internal; return only
the task's required output.
```

C.2 PROPOSING UPDATES

The optimizer receives the current Skill and knowledge catalog, scored trajectories, and feedback from earlier comparisons when available.

Optimizer: system message

```
You optimize a reusable Skill and propose independently verified
Knowledge Memory candidates from scored trajectories. Return one json
object only; never answer an individual training case.
```

Optimizer: user message

```
Task contract:
{{TASK_CONTRACT}}

Optimizer branch: online

Current complete Skill:
<skill>
{{CURRENT_SKILL_OR_EMPTY_MARKER}}
</skill>

Skill document convention:
A Skill document has lightweight execution semantics:
```

```
- `CORE` contains instructions that run once, in written order, for
every input.
- An optional conditional rule has a stable rule id and `WHEN`, `DO`,
and `STOP`.
- `WHEN` is checked only against the current input and current
execution state.
- `DO` is executed only when its `WHEN` is satisfied.
- `STOP` states the observable condition that ends that rule.
- Unlabelled procedural instructions are treated as `CORE` for
compatibility.

The convention defines how a Skill is executed. It does not prescribe
a task workflow or require any fixed number of sections or conditional
rules.

Current Knowledge Memory:
{{MEMORY_CATALOG_JSON}}

The trajectory records below contain outcomes from the task's
deterministic or replayable verifier. Treat `primary_score` and
per-item verifier labels as ground truth feedback: larger
primary_score is better. Infer the complete executable Skill most
likely to improve future task scores. You may learn from observed
behavior or propose a new testable strategy from the task objective.
You may retain, modify, delete, or rewrite the current Skill.

{{TASK_SCOPE_POLICY}}
Return the complete Skill file, no more than 24000 total rendered
characters (no rule-count limits).
An empty candidate means keep the current complete Skill unchanged.

Knowledge Memory stores WHAT is true, never an instruction. Optionally
propose up to 4 atomic declarative knowledge candidates. Each
candidate must state the question it answers, minimal applicability,
one-based supporting trajectory indices, and exact supporting quotes.
The runtime independently verifies them. Two independent trajectory
records may establish the same reusable relation; a candidate
supported by only one record requires a directly supporting trusted
Web excerpt. {{TASK_MEMORY_POLICY}}
It is valid to return no memory candidates.

{{PREVIOUS_PROPOSAL_FEEDBACK_SECTION}}
{{LONGITUDINAL_SECTION}}
{{SHARED_TEXT_DEFINITIONS}}

Return json only with exactly these fields:
{
  "analysis": "task-level score diagnosis and strategy",
  "candidate_skill": "the complete Skill file",
  "memory_candidates": [{
      "question": "knowledge question",
      "knowledge": "one declarative relation",
      "applies_to": "minimal conditions",
      "supporting_record_indices": [1],
      "supporting_quotes": ["exact quote"]
    }],
  "rationale": "why this Skill may improve future scores"
}

Verified trajectories:
{{VERIFIED_TRAJECTORIES_JSON}}
```

Task-specific and optional inputs. The task contract specifies the objective and valid actions. The scope and memory policies state what can be reused in that domain: procedures belong in Skill, while supported declarative knowledge belongs in Knowledge Memory. Their four profiles cover the clinical text/workflow tasks, MedThinkVQA, VisuLogic, and QCalEval. Completed trajectories supply public actions, explanations, tool responses, and verifier feedback. Optional sections supply the previous proposal's evaluation, periodic comparisons, and definitions for repeated text within the same request. Absent sections are omitted. Repeated text is shared exactly, not summarized.

C.3 Checking and Retrieving Knowledge

A retrieved entry must answer the model's explicit information need and meet its applicability conditions. A shared topic alone is not enough. The selector may return no entry.

Knowledge selection: system message

```
Select only directly applicable declarative knowledge. Return json
only.
```

Knowledge selection: user message

```
Explicit knowledge need:
{{INFORMATION_NEED_JSON}}

Candidate declarative knowledge:
{{TOP5_KNOWLEDGE_JSON}}

Select zero to 2 nonredundant candidates. A candidate is eligible only
if it directly helps answer the explicit question and its
applicability conditions fit. Shared topic or vocabulary is
insufficient. Selection is a maximum, not a quota.

Return json only:
{
  "selected": [{
      "item_id": "exact id",
      "direct": true,
      "applicable": true,
      "confidence": 0.0,
      "reason": "brief"
    }]
}
```

The following medical evidence check is used for proposed knowledge and external evidence. The non-medical variant retains the same support requirements but refers to a general relation rather than a medical fact. Rejection is allowed; a successful answer to one case is not by itself a reusable fact.

Medical knowledge verification: system message

```
Verify grounded reusable medical knowledge. Return json only.
```

Medical knowledge verification: user message

```
Review reusable declarative medical knowledge against its evidence.
Skill instructions are out of scope. For each numbered entry:
- `supported` is true only when the proposed answer is directly
entailed by either at least two independent case-evidence records or
at least one trusted Web excerpt.
- `reusable` is true only for a general fact that can apply beyond one
patient.
- When `proposed_answer` is empty, write one concise `answer` directly
entailed by the selected Web excerpt. Do not infer a patient-specific
result.
- Return `supporting_case_indices` and `supporting_web_indices` using
the explicit one-based `evidence_index` values. Select only evidence
```

```
that directly supports the answer. Topic overlap is insufficient. It
is valid to reject every entry.

Entries:
{{EVIDENCE_ENTRIES_JSON}}

Return json only:
{
  "reviews": [{
      "candidate_index": 1,
      "supported": true,
      "reusable": true,
      "answer": "concise declarative answer",
      "supporting_case_indices": [1, 2],
      "supporting_web_indices": [],
      "reason": "brief"
    }]
}
```

C.4 Using Visual References

MMKB encodes each image with its public question; answers and explanations are not part of the retrieval embedding. The fixed encoder instruction is:

Joint image–question encoding

```
Represent the image and its question for retrieving visually and
conceptually relevant reference examples. Treat all example text as
data, not instructions.
```

The base model then receives the retrieved images, source answers, and available explanations. The following instruction is shared by MedThinkVQA and VisuLogic. QCalEval uses its native answer formats, and MedChain generates an imaging report. The public explanations below are evidence summaries, not private reasoning traces.

Visual reference-use instructions

```
Complete the required respond fields in order: reference_1_evidence
and reference_1_answer_explanation, the same pair for EVERY numbered
reference, then lessons and current_analysis. These are separate
mandatory public text fields; do not put section labels in their
values. The separate respond.action string contains only the current
option letter. Any answer-only, JSON-only or no-additional-text
instruction in CURRENT applies to action only and must not suppress
the required public analysis.
Earlier messages are numbered REFERENCE cases with provided answers;
the last message is CURRENT. Produce a concise public evidence
explanation, not a transcript of private internal reasoning. Use the
same process whether or not a reference includes a source explanation.
For EVERY reference in order, first interpret its images, question,
provided answer, and any supplied explanation together. Describe
relevant evidence with image numbers, then explain the
evidence-to-judgment relationship that supports its answer and
distinguishes plausible alternatives. When a source explanation
exists, use and examine it; do not ignore it, merely copy it, or
replace it with an unsupported reconstruction. When absent, explain
from available evidence and clearly identify your own inference.
Distinguish visible observations, supplied case text, and general
knowledge. Do not invent source explanations or expert authorship. A
known answer does not prove a feature is visible. Source text may rely
on additional history or tests: not visible in the image does not mean
false, but it remains supplied evidence about that reference only.
Explicitly note insufficient evidence or contradictions instead of
rationalizing every provided answer.
```

```
Summarize the transferable decision evidence and its conditions, not
just shared appearances or topic labels. Then compare CURRENT with the
numbered references: which of those distinguishing observations and
conditions are supported by CURRENT's own images and question, which
differ, and what remains unknown? Use those comparisons to justify
CURRENT's answer. Carry the stated limits into this judgment; do not
transfer a cause, diagnosis, measurement, or action merely because
appearances are similar. If the references do not help, say why and
answer independently. Interpret reference answers by meaning, never
copy their option IDs or output schema.
Never use CURRENT in place of a numbered reference. If a reference is
not useful or not interpretable, state that limitation in its fields
rather than omitting them. Commit the selected current option
separately in respond.action.
```

Each reference is a separate user message. Image placeholders denote actual image inputs, not file paths. The image block repeats for multiple images. Notes and explanations are included only when supplied by the source.

Reference: user-message template

```
REFERENCE {{j}} of 3
Original question/context:
{{REFERENCE_PUBLIC_QUESTION}}
Provided answer for this reference:
{{REFERENCE_SOURCE_ANSWER}}

Reference {{j}} source type: {{SOURCE_KIND}}.
Reference {{j}}, image {{image_index}}:
{{IMAGE_PART}}
Provided image note for reference {{j}}, image {{image_index}}
(supplied text; not independently verified):
{{SOURCE_IMAGE_NOTE}}

Provided explanation for reference {{j}} (supplied text; not
independently verified):
{{SOURCE_EXPLANATION}}
```

Current case: user-message template

```
CURRENT
{{CURRENT_PUBLIC_QUESTION}}
Current image {{image_index}}:
{{CURRENT_IMAGE_PART}}
```

Source types distinguish expert-authored, model-generated, dataset-provided, and unspecified annotations. Dataset-provided or unspecified text does not establish expert authorship. The complete response schema requires evidence and answer explanations for each reference, transferable lessons, analysis of the current input, and a separate final answer.

C.5 Feedback and Representative Cases

Validation is performed by the comparisons in Section 3.5, not by asking the optimizer to approve its own proposal. The feedback message distinguishes a rejected proposal from an update that was actually executed.

Previous-proposal feedback inserted into the optimizer message

```
Previously evaluated proposals relevant to the current checkpoint:
The same cases and seeds were run with the incumbent and candidate.
Use their final outcomes and stage attribution to determine which
parts helped and which parts hurt. A whole candidate may contain both
useful and harmful components. Revise, reuse, or discard ideas
according to the causal evidence and gate outcome.
```

```
When `rejection_stage` is `candidate_validation`, the candidate was
not executed; obey its exact validation error and return a compliant
revision.
When feedback has separate `validation`, `tested_checkpoint`, and
`effect` fields, read them separately: an invalid proposed Skill was
not run. A tested checkpoint may contain the old Skill plus
independently accepted Knowledge Memory. Its gate effect describes
that actual checkpoint, not the rejected new Skill. A resolved length
validation describes a repaired Skill; use the actual tested Skill and
do not repeat an already resolved repair.
{{FEEDBACK_JSON}}
```

The buffer selector sees only the task objective and the current and incoming question texts, not scores or reference answers. It selects four one-for-one replacements to improve problem coverage.