METAEXP: LEARNING TO SELF-IMPROVE THROUGH EXPERIENCE VIA VERIFIABLE CROSS-TASK ENVIRONMENT SCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents can accumulate interaction feedback, but reliably turning that feedback into reusable experience remains challenging. Existing methods either optimize external experience scaffolds or train experience-related behavior on specific task distributions, leaving open whether experience learning itself can become a transferable model capability. Training such a capability calls for supervision beyond isolated tasks, which mainly support in-episode problem solving rather than revealing what knowledge remains useful across episodes. In this work, we introduce METAEXP, a scalable framework that trains this capability through verifiable task families with controlled cross-task structure. Each family preserves a reusable factor while varying task-specific realizations, enabling transfer without direct solution reuse. We propose two complementary designs: Shared-Tool families keep black-box tools fixed across tasks to elicit operational experience, while Shared-Workflow families preserve latent procedures across changing realizations to elicit procedural experience. We scale environments under both designs, collect and filter sequential interaction trajectories, and fine-tune Qwen3.5-122B-A10B on 4,956 trajectories from 1,000 task families. With parameters frozen during downstream evolution, METAEXP improves by $+8.5$ points on SpreadsheetBench, peaks at $+2.3$ on APEX-Agents, and improves by $+14.8$ on AutomationBench. The gains persist under domain-excluded training, supporting experience learning as a transferable capability for continued non-parametric improvement after deployment.

1 INTRODUCTION

Large language models (LLMs) increasingly act in interactive environments, using tools to complete long-horizon tasks and respond to execution feedback (Yao et al., 2023). Yet most agents remain largely episodic: solving one task does not by itself make the agent more effective on the next, and useful procedures discovered through interaction are often lost once the episode ends (Zhao et al., 2024; Wang et al., 2025b). Experience-driven self-evolution addresses this limitation by turning past trajectories and feedback into persistent experience that can guide future behavior (Shinn et al., 2023). However, retaining experience is not equivalent to learning from it. An effective agent must identify what caused an outcome, abstract what remains useful, update its experience accordingly, and reuse it in later tasks. This raises a fundamental question: can a model be trained once to acquire, update, and reuse experience more effectively on previously unseen task distributions?

Existing work approaches experience learning at either the system or model level. System-level methods keep the backbone fixed and improve how external experience is represented, extracted, revised, retrieved, or validated (Ni et al., 2026; Yu et al., 2026; Wang et al., 2026a; Yang et al., 2026b). They demonstrate that external experience can support parameter-free adaptation, but its quality still depends on the pretrained model's ability to diagnose trajectories, abstract reusable knowledge, and apply it appropriately. Model-level approaches instead optimize experience acquisition, maintenance, or skill-conditioned execution through parameter training (Wang et al., 2026b; Xia et al., 2026; Fan et al., 2026a). However, their optimization signals and downstream evaluation are typically tied to the same or closely related task distributions, making general experience-learning ability

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.

difficult to separate from task-specific adaptation. It therefore remains unclear whether experience learning itself can be trained as a transferable model capability.

This question is particularly relevant to realistic agent deployment. An agent may encounter a new task distribution without a curated skill library, large collections of verified target-domain trajectories, or the opportunity for benchmark-specific retraining. It must instead improve incrementally from the tasks and feedback it encounters, even when later task variants differ substantially from earlier ones. Motivated by this setting, we study *meta experience learning*: after a single general training stage, the model parameters remain fixed, and the agent starts each held-out task distribution with an empty experience store. It must construct, revise, and reuse experience solely through sequential interaction and feedback. In this work, we instantiate external experience as structured Agent Skills (Anthropic, 2025). This setting separates learning how to acquire and use experience from pre-acquiring or carrying target-domain experience itself.

Training this capability requires supervision beyond isolated task solving. Independent task trajectories primarily supervise in-episode problem solving, but provide limited evidence about which parts of an interaction will remain useful on later tasks. We therefore organize trajectory collection around structured task families that provide controlled cross-episode supervision. METAEXP operationalizes this idea by constructing verifiable families under the principle of *transfer without direct solution reuse*: each family preserves a reusable factor across tasks while varying task-specific realizations, so that experience acquired from earlier members can help with later ones without allowing previous solutions to be directly reused. We instantiate this principle through two complementary designs. *Shared-Tool* families keep black-box tools fixed across tasks while varying their concrete problems, encouraging operational experience about recurring interfaces and tool composition. *Shared-Workflow* families instead preserve a latent multi-step procedure across changing domains, tools, and data, encouraging higher-level procedural experience across different realizations. Sequential trajectories collected from environments scaled under both designs are then used to train the model to acquire, update, and reuse experience.

We fine-tune Qwen3.5-122B-A10B on 4,956 filtered trajectories collected from 1,000 synthetic task families and evaluate it under cold-start sequential evolution on three held-out agent benchmarks: SpreadsheetBench, APEX-Agents, and AutomationBench. As test-time experience accumulates, METAEXP improves by $+8.5$ points on SpreadsheetBench, peaks at $+2.3$ points on APEX-Agents, and improves by $+14.8$ points on AutomationBench. In contrast, the matched Qwen3.5 backbone degrades by 2.2 points on SpreadsheetBench under the same evolution protocol, showing that strong task-solving ability does not necessarily translate into effective experience learning. The gains remain substantial when semantically related training domains are excluded, and METAEXP also makes better use of externally provided skills on SkillsBench. Together, these results support experience learning as a transferable model capability that can drive continued improvement through non-parametric experience accumulation on unseen task distributions.

Our contributions are threefold: (i) We formulate *meta experience learning* as a transferable model capability: after a single general training stage, an agent should be able to build and reuse external experience from scratch on unseen task distributions without further parameter updates. (ii) We introduce METAEXP, a scalable framework that constructs verifiable task families with controlled cross-task structure to generate cross-episode supervision for experience learning, instantiated through complementary Shared-Tool and Shared-Workflow designs. (iii) We demonstrate transferable cold-start experience learning across heterogeneous held-out agent benchmarks, with substantial gains retained under domain-excluded training and improved utilization of curated skills.

2 RELATED WORK

Experience-driven agent self-evolution. Existing approaches improve agents through persistent experience at either the system or model level. System-level methods keep the backbone fixed and externalize experience as verbal memories, codes, or executable skills (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024; 2025b;a). Recent work further improves how such experience is extracted, revised, retrieved, and validated (Ni et al., 2026; Qiu et al., 2026; Jiang et al., 2026a; Lin et al., 2026; Yu et al., 2026; Li et al., 2026c; Wang et al., 2026a; Yang et al., 2026b). These approaches enable parameter-free adaptation, but the resulting experience remains dependent on the frozen model's ability to diagnose trajectories and abstract reusable knowledge, and some rely

on substantial target-distribution trials or in-distribution validation (Zhang et al., 2026a; Liu et al., 2026). Model-level methods instead update parameters to optimize experience acquisition, maintenance, or skill-conditioned execution (Lu et al., 2026; Zhu et al., 2026a; Wu et al., 2025; Wang et al., 2026b; Xia et al., 2026; Shi et al., 2026; Yang et al., 2026a; He et al., 2026; Zhang et al., 2026b; Yao et al., 2026; Li et al., 2026b; Vishe et al., 2026; Fan et al., 2026a). However, their optimization signals and downstream evaluation are typically tied to the same or closely related task distributions, making general experience-learning ability difficult to separate from task-specific adaptation. In contrast, METAEXP trains on diverse source task families and evaluates whether experience acquisition and reuse transfer to held-out distributions under frozen parameters and cold-start evolution.

Environment scaling and evaluation. Agent training data can be scaled through verifiable task and environment synthesis, including task proposal, capability composition, and automated construction of executable environments (Zhou et al., 2025; Xie et al., 2026; Zhu et al., 2026b; Hua et al., 2026; Wei et al., 2026). Some methods organize generation around predefined skills or skill structures (Zheng et al., 2025; Fan et al., 2026b; Cheng et al., 2026; Tan et al., 2026). These approaches primarily scale individual tasks or environments, whereas METAEXP scales *task families* with controlled cross-task structure so that experience acquired from earlier tasks can remain useful for later ones without enabling direct solution reuse. Complementary benchmarks evaluate either the utility of provided skills (Li et al., 2026a; Han et al., 2026) or continual skill generation and refinement over task streams (Zhong et al., 2026; Zhou et al., 2026b; Zhang et al., 2026c; Lei et al., 2026; Jiang et al., 2026b; Gao et al., 2026; Zhou et al., 2026a). Our evaluation instead focuses on cold-start sequential evolution across heterogeneous held-out task distributions.

3 PROBLEM SETUP

We study whether a language model can be trained once to better acquire, update, and reuse external experience on previously unseen task distributions. During test-time evolution, the model parameters remain fixed and only the accumulated experience changes. We first define this sequential experience-learning setting and then introduce the cold-start protocol used for evaluation.

3.1 EXPERIENCE LEARNING SETTING

We consider a sequence of tasks, where each task $\tau_i = (e_i, q_i, v_i)$ consists of an interactive environment e_i, a textual task request q_i, and an evaluator v_i. Given a model with parameters θ and an external experience store B_{i-1}, the agent interacts with the environment and produces a trajectory

$$h_i \sim \pi_\theta(\cdot \mid e_i, q_i, B_{i-1}). \tag{1}$$

The evaluator then returns an outcome score r_i together with task-level feedback f_i,

$$(r_i, f_i) = v_i(h_i). \tag{2}$$

The score measures task completion, while the feedback contains richer information about successes and failures. Based on the trajectory and feedback, the agent may update its experience:

$$B_i = U_\theta(B_{i-1}, \tau_i, h_i, f_i), \tag{3}$$

where U_θ denotes the model's experience-update behavior. The resulting B_i persists across tasks and can influence subsequent execution. Experience may take different forms; in this work, we instantiate B_i as a file-based bank of structured agent skills. Experience learning therefore involves both deciding what skills should be updated according to previous interactions and using the resulting skills appropriately on future tasks.

Our goal is not to learn target-domain experience during parameter training. Instead, we ask whether a single *general training stage* can improve the model's ability to acquire and reuse experience under this sequential process on new task distributions. Training produces parameters θ_{ME} from a base model θ_0 using only source task families. At evaluation time, neither model receives target-benchmark parameter updates or pre-accumulated target-domain experience. Their parameters remain fixed, and adaptation occurs only through the evolving experience store B. We refer to this property as *meta experience learning*: the ability to acquire useful experience from interaction feedback and reuse it on subsequent tasks, such that this behavior transfers beyond the task distributions used for parameter training. Operationally, it requires the model to construct experience from scratch, revise it as new feedback arrives, and selectively apply it when solving later tasks.

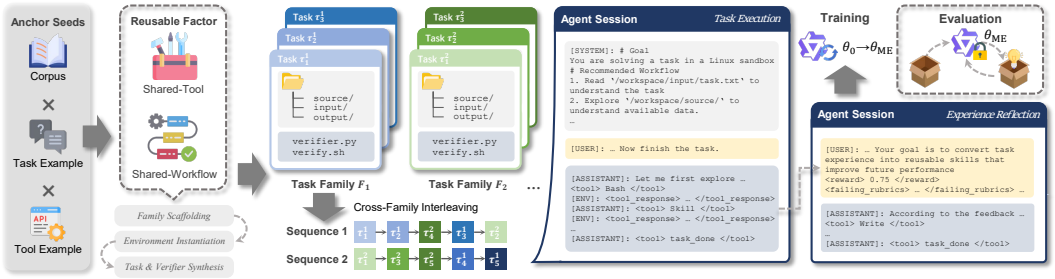


Figure 1: Overview of METAEXP. Heterogeneous seeds are expanded into verifiable task families with complementary Shared-Tool and Shared-Workflow structure. Family-centered task sequences are used to collect execution and feedback-driven reflection trajectories, which train the model to acquire, update, and reuse experience. At evaluation, model parameters are frozen and experience is accumulated from an empty bank on unseen task distributions.

3.2 COLD-START SEQUENTIAL EVOLUTION

We introduce a cold-start sequential evolution protocol (SeqEvo) to evaluate experience learning on held-out task distributions. For a fixed evaluation domain, we omit the domain index for clarity. Both the adaptation and target tasks come from a task distribution that is not used for parameter optimization. The agent starts with an empty experience store,

$$B_0 = \varnothing, \tag{4}$$

and processes K adaptation tasks $(\tau_1, \ldots, \tau_K)$ one at a time. Following the update process above, the experience bank obtained after task τ_i is carried forward to τ_{i+1}, yielding B_k after the first k tasks. No future adaptation task or target-set feedback is available when constructing B_k. Here, k denotes the number of test-time adaptation tasks rather than gradient updates.

We evaluate each B_k on a fixed target set E that is disjoint from the adaptation tasks, while keeping the experience bank unchanged during evaluation. We define

$$S(k;\theta) = \frac{1}{|E|} \sum_{\tau \in E} r_\tau^{(k,\theta)}, \tag{5}$$

where $r_\tau^{(k,\theta)}$ denotes the outcome score obtained by model θ on task τ using the accumulated experience bank B_k, and $S(0;\theta)$ is the performance without accumulated experience. The effect of test-time experience accumulation after k adaptation tasks is

$$\Delta(k;\theta) = S(k;\theta) - S(0;\theta). \tag{6}$$

We report both $S(0;\theta)$ and $\Delta(k;\theta)$. The former reflects zero-experience task-solving ability, while the latter measures how performance changes as test-time experience accumulates. Together, they separate zero-experience task performance from the change induced by accumulated experience.

4 METAEXP: VERIFIABLE CROSS-TASK ENVIRONMENT SCALING

METAEXP trains transferable experience-learning behavior by constructing controlled cross-task supervision. As shown in Figure 1, it builds two complementary types of task families with reusable cross-task structure, realizes them as verifiable executable environments, collects sequential experience-learning trajectories, and uses the resulting data for model training.

4.1 CROSS-TASK FAMILY DESIGN

The key design principle is to preserve a reusable source of commonality across tasks while varying their concrete realizations. We represent a task family as

$$\mathcal{F}(z) = \{\tau_i = G(z, \xi_i)\}_{i=1}^{m}, \tag{7}$$

where z denotes the family-level reusable factor and ξ_i captures task-specific variation. We follow the principle of *transfer without direct solution reuse*: z should make experience acquired from earlier members potentially useful for later ones, while variation in ξ_i prevents previous solutions from being directly reused.

We propose two complementary designs. (i) **Shared-Tool families** keep a black-box tool library and its documentation fixed across members while varying task data and requests. Here, z corresponds to stable tool semantics and interfaces, encouraging relatively concrete operational experience about tool behavior, argument conventions, and composition. (ii) **Shared-Workflow families** instead preserve a latent multi-step procedure while varying its concrete realization, including domain, APIs and documentation, data, and request. The latent workflow guides generation but is never exposed to the solver, encouraging more abstract procedural experience that transfers across changing tools and domains. Together, the two designs expose reusable structure at complementary levels, from concrete operational regularities to abstract procedural patterns.

4.2 Verifiable Family Scaling

We scale these families from heterogeneous seeds drawn from pretraining-style corpora, task examples, and tool schemas, which provide diverse semantic contexts, procedural patterns, and executable affordances. Each family is then realized through three stages. **Family scaffolding** constructs its shared factor z once per family. For Shared-Tool families, we generate and validate an executable tool library together with its documentation, which are then fixed across members. For Shared-Workflow families, we abstract a domain-independent latent workflow from a seed task example.

Environment instantiation produces multiple family members by varying task-specific factors ξ_i. Shared-Tool members reuse the same tools with independently populated data, whereas Shared-Workflow members instantiate the same latent procedure with new domains, APIs and documentation, and task-specific data. We additionally vary data scale, output structure, and difficulty to increase task diversity while preserving the intended commonality.

Finally, **task-and-verifier synthesis** uses a propose–solve–confirm loop to generate each task request together with a reference solution and executable checks. The generator explores the instantiated tools and data before proposing and solving the task. The complete package is rerun in a clean sandbox and retained only if the reference solution executes successfully and passes verification. Validation is asymmetric: reference solutions, verifiers, and latent workflows may be used during synthesis but are never exposed to the solver.

4.3 Sequential Experience Trajectory Collection

For each validated family $\mathcal{F} = \{\tau_1, \ldots, \tau_m\}$, we order its members by increasing difficulty and assign them sequentially to a teacher model starting from an empty skill bank. Each collection sequence is centered on one family but interleaves a small fraction of tasks sampled from other families. These cross-family tasks introduce potentially irrelevant experience into the bank, encouraging the model to learn selective reuse instead of indiscriminately applying accumulated skills. The teacher follows the sequential experience-learning process defined in Section 3: it solves each task conditioned on the current bank, receives executable feedback, and updates the bank before proceeding to the next task.

Each task runs in a fresh sandbox with the current skill bank mounted. The teacher interacts through a general agent harness and invokes a `task_done` tool upon completion, after which the task-specific verifier is introduced and executed. An LLM judge reviews only failed rubric items to correct potential false negatives and produces the final score and feedback. The feedback is then injected through a fixed reflection prompt, allowing the teacher to create, revise, or delete skills before the updated bank is carried forward. We retain only trajectories with coherent execution, evaluation, and reflection, verified task completion, and valid skill artifacts.

4.4 Meta-Experience Training

The retained trajectories supervise two complementary behaviors: execution-stage actions train experience-conditioned task solving, while reflection-stage actions train feedback-driven experience

Table 1: Performance and efficiency on SpreadsheetBench. Values are mean ± population standard deviation, with $\Delta_k = S(k) - S(0)$ shown in parentheses. Green/red denotes improvement/degradation relative to $k = 0$. EvoSkill and SkillGrad results are taken from Wang et al. (2026a) and follow their originally reported evolution protocols; all other results use our cold-start SeqEvo protocol. "–" denotes an unreported checkpoint.

Model	Protocol	$k = 0$	$k = 5$	$k = 10$	$k = 20$	$k = 40$
			Task success rate (%, ↑)			
GPT-4.1	EvoSkill	44.2	–	–	–	37.2 ± 2.6 (−7.0)
GPT-5.4	EvoSkill	62.5	–	–	–	68.1 ± 0.5 (+5.6)
GPT-4.1	SkillGrad	44.2	–	–	–	54.2 ± 3.5 (+10.0)
GPT-5.4	SkillGrad	62.5	–	–	–	71.1 ± 1.7 (+8.6)
GLM-5.2	SeqEvo	88.3 ± 1.2	88.9 ± 0.5 (+0.6)	89.5 ± 0.7 (+1.2)	89.6 ± 0.4 (+1.3)	89.4 ± 0.4 (+1.1)
Qwen3.6-Plus	SeqEvo	76.0 ± 0.9	71.8 ± 2.5 (−4.2)	72.9 ± 3.1 (−3.1)	73.1 ± 3.1 (−2.9)	74.3 ± 3.1 (−1.7)
Qwen3.5-122B-A10B	SeqEvo	68.9 ± 1.8	67.3 ± 1.8 (−1.6)	67.7 ± 0.5 (−1.2)	67.0 ± 0.9 (−1.9)	66.7 ± 1.3 (−2.2)
METAEXP-Qwen3.5-122B-A10B	SeqEvo	65.0 ± 2.2	69.5 ± 3.3 (+4.5)	72.9 ± 3.7 (+7.9)	73.1 ± 2.8 (+8.1)	73.5 ± 3.4 (+8.5)
			Execution turns (↓)			
GLM-5.2	SeqEvo	10.4 ± 0.4	12.3 ± 0.2 (+1.9)	12.1 ± 0.1 (+1.7)	12.4 ± 0.5 (+2.0)	13.1 ± 0.6 (+2.7)
Qwen3.6-Plus	SeqEvo	7.8 ± 0.1	8.0 ± 0.3 (+0.2)	8.0 ± 0.2 (+0.2)	8.1 ± 0.2 (+0.3)	7.7 ± 0.2 (−0.1)
Qwen3.5-122B-A10B	SeqEvo	7.6 ± 0.5	7.2 ± 0.1 (−0.4)	7.3 ± 0.0 (−0.3)	7.4 ± 0.1 (−0.2)	7.2 ± 0.0 (−0.4)
METAEXP-Qwen3.5-122B-A10B	SeqEvo	8.1 ± 0.1	8.1 ± 0.2 (0.0)	8.2 ± 0.3 (+0.1)	8.5 ± 0.9 (+0.4)	8.6 ± 0.7 (+0.5)

updating. Let $\mathcal{D}_{\mathrm{ME}}$ denote records from both stages. We fine-tune the base model with the standard assistant-token objective

$$\mathcal{L}_{\mathrm{ME}}(\theta) = -\sum_{x \in \mathcal{D}_{\mathrm{ME}}} \sum_{t \in \mathcal{A}(x)} \log p_\theta(x_t \mid x_{<t}), \tag{8}$$

where $\mathcal{A}(x)$ denotes assistant-generated token positions; system messages, user messages, and environment observations are masked from the loss.

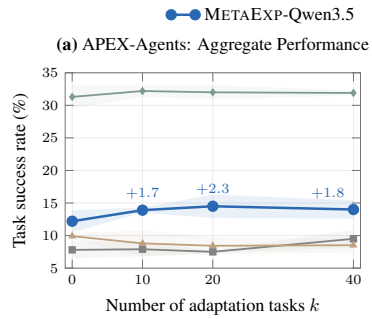
Optimizing this objective from θ_0 yields θ_{ME}. The skill banks accumulated during trajectory collection are discarded after model training; what is retained is the model's learned behavior for acquiring, updating, and reusing experience. At evaluation time, θ_{ME} remains frozen and each held-out task distribution starts from an empty bank under SeqEvo, where adaptation occurs solely through newly accumulated experience.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

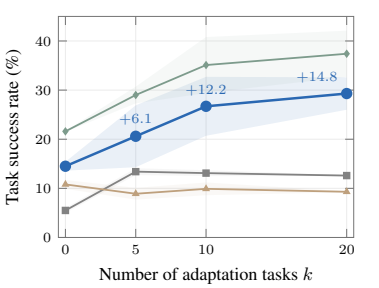
Training configuration. Following Section 4, we construct 500 Shared-Tool and 500 Shared-Workflow families, each containing five task instances. We use GLM-5.2 (Zeng et al., 2026) as the teacher model and Claude Code as the agent harness for trajectory collection, yielding 4,956 trajectories after filtering. We fine-tune Qwen3.5-122B-A10B (Qwen Team, 2026) for two epochs with a 256K context window. The loss is computed only over assistant-generated tokens, while system messages, user messages, and environment observations are masked out.

Datasets and evaluation. We evaluate cold-start experience evolution on three challenging benchmarks spanning distinct task distributions. We adapt their original environments and evaluators into executable sandboxes compatible with our agent harness. All models evaluated by us follow the SeqEvo protocol defined in Section 3.2, using Claude Code as the same harness employed during trajectory collection. For each benchmark, adaptation and target sets are disjoint, and we repeat each experiment three times with independently randomized adaptation orders. Each adaptation task allows up to three reflection retries; if all retries fail, the latest verified skill bank is carried forward. **SpreadsheetBench-v1-verified** contains 400 real-world spreadsheet manipulation tasks at the cell and sheet levels. We use 40 tasks for adaptation and 350 disjoint tasks for evaluation, with a shared skill bank capped at 10 skills. Verification executes the generated solution on the input workbook, recalculates formulas, and compares annotated output ranges with the golden workbook; a task succeeds only if all required cells match. **APEX-Agents** contains 480 long-horizon professional tasks across investment banking, law, and management consulting. For each domain, we use 40 tasks



(b) APEX-Agents: Domain-wise Performance

	METAEXP-Qwen3.5				Baseline-Qwen3.5			
Domain	$k=0$	Δ_{10}	Δ_{20}	Δ_{40}	$k=0$	Δ_{10}	Δ_{20}	Δ_{40}
Investment banking	16.1	−1.9	−0.5	−2.5	4.7	+3.4	+2.2	+3.7
Law	11.9	+3.9	+3.9	+2.8	12.8	−2.8	−2.0	+0.8
Mgmt. consulting	8.6	+3.1	+3.6	+5.0	5.8	−0.2	−1.1	+0.9



(d) AutomationBench: Domain-wise Performance

	METAEXP-Qwen3.5				Baseline-Qwen3.5			
Domain	$k=0$	Δ_5	Δ_{10}	Δ_{20}	$k=0$	Δ_5	Δ_{10}	Δ_{20}
Finance	17.9	+1.3	+25.4	+32.9	10.4	+10.4	+7.9	+9.2
HR	12.1	+15.8	+17.1	+14.6	3.8	+11.2	+7.4	+7.4
Marketing	14.2	+15.8	+18.7	+16.6	4.2	+5.0	+5.8	+3.7
Operations	17.9	+3.8	+6.7	+10.9	8.3	+5.5	+7.1	+5.5
Sales	13.8	+0.8	+3.7	+0.8	3.8	+11.2	+12.9	+11.2
Support	11.2	−1.2	+1.7	+13.0	2.5	+4.2	+4.6	+5.8

$\Delta_k = S_k - S_0$: $\Delta < 0$ | $0 \le \Delta < 5$ | $5 \le \Delta < 15$ | $\Delta \ge 15$

Figure 2: Performance on APEX-Agents and AutomationBench. Left: aggregate success rate over adaptation steps, with shaded population standard deviations over three runs. Right: domain-wise success rate and changes Δ_k on each adaptation step for METAEXP-Qwen3.5-122B-A10B and Qwen3.5-122B-A10B; green/red denotes improvement/degradation.

for adaptation and 120 disjoint tasks for evaluation, maintaining a separate skill bank capped at 5 skills. An LLM judge evaluates the agent response, generated artifacts, and environment changes against expert-authored binary rubrics; a task succeeds only if all required criteria are satisfied. **AutomationBench** contains 600 cross-application API workflow tasks across finance, HR, marketing, operations, sales, and support. For each domain, we use 20 tasks for adaptation and 80 disjoint tasks for evaluation, with an independent skill bank capped at 10 skills. Verification compares the final environment state against deterministic assertions. We report average task success rate on all three benchmarks. Each adaptation task is encountered only once per run, preventing repeated optimization against the same verifier feedback. We additionally evaluate the ability to utilize externally provided experience on **SkillsBench-v1.1**, which contains 87 expert-level tasks across 11 domains, each paired with curated agent skills. All models receive the same skills for each task, and we report average score and pass rate over three runs.

Models and baselines. We denote our trained checkpoint as METAEXP-Qwen3.5-122B-A10B and use the original Qwen3.5-122B-A10B as the matched backbone baseline. We additionally evaluate stronger models, including Qwen3.6-Plus and GLM-5.2, with reasoning enabled at maximum effort. These models are evaluated using the same harness and SeqEvo setup. On SpreadsheetBench, we also report published EvoSkill (Alzubi et al., 2026) and SkillGrad results from Wang et al. (2026a) as representative system-level experience-learning approaches; these follow their originally reported evolution protocols rather than our SeqEvo setup.

5.2 MAIN RESULTS

SpreadsheetBench. Table 1 reports performance and efficiency on SpreadsheetBench across different backbones and evolution protocols. Under SeqEvo, METAEXP improves from 65.0 at $k = 0$ to 73.5 at $k = 40$ (+8.5), with most gains emerging within the first 10 adaptation tasks. In contrast, Qwen3.5-122B-A10B and Qwen3.6-Plus finish below their zero-experience baselines, while GLM-5.2 remains the strongest absolute solver but improves only marginally, peaking at +1.3 points. The gain is accompanied by only a modest increase in execution turns, from 8.1 at $k = 0$ to 8.6 at $k = 40$,

Table 3: SpreadsheetBench task success under domain-excluded training. Values are mean ± population standard deviation over three runs, with $\Delta_k = S(k) - S(0)$ shown in parentheses.

Model	$k = 0$	$k = 5$	$k = 10$	$k = 20$	$k = 40$
			Task success rate (%, ↑)		
Qwen3.5-122B-A10B	68.9 ± 1.8	67.3 ± 1.8 (−1.6)	67.7 ± 0.5 (−1.2)	67.0 ± 0.9 (−1.9)	66.7 ± 1.3 (−2.2)
- METAEXP	65.0 ± 2.2	69.5 ± 3.3 (+4.5)	72.9 ± 3.7 (+7.9)	73.1 ± 2.8 (+8.1)	73.5 ± 3.4 (+8.5)
- **Strict-OOD METAEXP**	62.7 ± 1.4	66.8 ± 5.0 (+4.1)	66.2 ± 4.0 (+3.5)	72.5 ± 3.0 (+9.8)	72.0 ± 0.8 (+9.3)

indicating that the improvement is not simply associated with substantially more interaction. Published EvoSkill and SkillGrad results further show that system-level skill evolution can also yield substantial gains, although they follow different evolution protocols and typically use more verified trajectories during adaptation.

APEX-Agents. Figure 2(a,b) shows a more heterogeneous pattern of experience transfer on APEX-Agents. At the aggregate level, METAEXP improves from 12.2 at $k = 0$ to a peak of 14.5 at $k = 20$, retaining a +1.8-point gain at $k = 40$. This modest average masks clear domain differences: at $k = 40$, METAEXP gains +2.8 points in law and +5.0 in management consulting, compared with only +0.8 and +0.9 for the matched Qwen3.5 baseline.

Investment banking is the main exception, where METAEXP decreases by 2.5 points while Qwen3.5 improves by 3.7. This reversal is partly amplified by the near-binary exact-value evaluation and different cold-start headroom, although some case-specific skills also overgeneralize and interfere with otherwise correct solutions. GLM-5.2 remains the strongest absolute solver but shows little further improvement from accumulated experience. Overall, METAEXP yields positive aggregate evolution, with particularly clear gains in law and management consulting.

AutomationBench. Figure 2(c,d) shows a broader and more sustained evolution trend on AutomationBench. METAEXP improves monotonically from 14.5 to 29.3, yielding a +14.8-point gain, roughly twice that of its matched Qwen3.5 baseline. By $k = 10$, METAEXP already improves over its zero-experience performance in all six domains, and by $k = 20$ its gain exceeds that of Qwen3.5 in five of them. The strongest improvement appears in finance (+32.9), with substantial gains also observed across HR, marketing, operations, and support. GLM-5.2 remains the strongest absolute solver and also benefits from accumulated experience. Overall, AutomationBench shows that METAEXP substantially increases experience-driven improvement.

SkillsBench. Unlike the preceding benchmarks, SkillsBench holds the experience itself fixed: every model receives the same expert-curated skill, allowing us to isolate how effectively provided experience is used during execution. As shown in Table 2, METAEXP improves over its matched backbone from 33.3 to 41.4 in average score and from 47.1% to 62.1% in average pass rate. These results provide complementary evidence that METAEXP improves not only experience acquisition but also the ability to translate available experience into successful task execution.

Table 2: Model performance on SkillsBench. Values are averaged across three runs.

Model	Score (%)	Pass (%)
GLM-5.2	65.1	74.7
Baseline	33.3	47.1
METAEXP	**41.4** (+8.1)	**62.1** (+15.0)

5.3 DOMAIN-EXCLUDED OOD EVALUATION

Although the main evaluation benchmarks are excluded from training, the scaled training corpus may still contain semantically related domains. To impose a stricter distribution shift, we remove Software Engineering & Programming, Finance & Business, and IT Software & Web Services from the training data. We apply a strict-prefix filter: a trajectory is excluded if either its current task or any preceding task in the same family belongs to one of these domains, preventing retained trajectories from depending on earlier in-domain experience. This leaves 3,634 trajectories, including 1,939 Shared-Tool and 1,695 Shared-Workflow trajectories. Starting from the same Qwen3.5-122B-A10B checkpoint, we train a Strict-OOD variant with the same configuration and evaluate its epoch-2 checkpoint on the same SpreadsheetBench splits under SeqEvo.

Table 4: Ablation of cross-family task designs on models. Values are mean ± population standard deviation over three runs, with $\Delta = S_{\text{final}} - S_{\text{initial}}$ shown in parentheses.

Benchmark	Shared-Tool Only		Shared-Workflow Only		Full METAEXP	
	Initial	Final	Initial	Final	Initial	Final
Task success rate (%, ↑)						
SpreadsheetBench	62.7 ± 1.0	72.4 ± 7.3 (+9.7)	63.6 ± 1.5	69.7 ± 2.7 (+6.1)	65.0 ± 2.2	73.5 ± 3.4 (+8.5)
APEX-Agents	13.2 ± 1.3	12.7 ± 1.3 (−0.5)	12.2 ± 0.6	12.6 ± 0.7 (+0.4)	12.2 ± 1.7	14.0 ± 1.4 (+1.8)
AutomationBench	7.2 ± 0.5	14.8 ± 3.8 (+7.6)	8.5 ± 0.5	15.6 ± 3.6 (+7.1)	14.5 ± 0.9	29.3 ± 3.3 (+14.8)
Execution turns (↓)						
SpreadsheetBench	7.2 ± 0.3	10.5 ± 1.7 (+3.3)	7.8 ± 0.2	9.0 ± 0.4 (+1.2)	8.1 ± 0.1	8.6 ± 0.7 (+0.5)
APEX-Agents	29.0 ± 0.7	26.2 ± 0.4 (−2.8)	31.2 ± 0.5	28.6 ± 0.2 (−2.6)	33.3 ± 0.6	33.1 ± 1.6 (−0.2)
AutomationBench	43.0 ± 0.7	44.5 ± 6.4 (+1.5)	45.7 ± 1.2	43.3 ± 5.2 (−2.4)	49.4 ± 1.1	42.4 ± 3.1 (−7.0)

Table 3 shows that the experience-learning gains persist even after removing training domains most closely related to SpreadsheetBench. Strict-OOD METAEXP reaches $\Delta_{20} = +9.8$ and $\Delta_{40} = +9.3$, comparable to the gains of the full METAEXP model. Despite its lower zero-experience performance, it exhibits substantial positive evolution as experience accumulates, whereas the matched untrained backbone consistently degrades under the same SeqEvo setting. This contrast provides stronger evidence that the learned experience-learning behavior transfers across domains and does not depend on training examples from related task domains.

5.4 ABLATION STUDY

We ablate the two cross-task family designs by training separate models on Shared-Tool or Shared-Workflow trajectories, starting from the same base checkpoint. Table 4 shows that both designs can induce experience-driven improvement, but with different behavioral patterns. Shared-Tool yields the larger gain on SpreadsheetBench (+9.7 vs. +6.1 points), consistent with learning reusable operational experience over stable black-box tools. Shared-Workflow, in contrast, more consistently improves execution efficiency on long-horizon tasks, reducing execution by 2.6 turns on APEX-Agents and 2.4 turns on AutomationBench, consistent with learning higher-level procedural experience across changing tool realizations. Combining both designs yields the broadest improvement across benchmarks. On AutomationBench in particular, full METAEXP reaches a +14.8-point gain, substantially exceeding either Shared-Tool (+7.6) or Shared-Workflow (+7.1) alone, while also reducing execution by 7.0 turns. These results suggest that the two family designs provide complementary supervision for transferable experience learning.

5.5 BEHAVIORAL ANALYSIS

To better understand the source of the experience-learning gains, we analyze how accumulated skills are used and updated on SpreadsheetBench. METAEXP-Qwen3.5 invokes existing skills far more frequently than its matched backbone despite maintaining a smaller bank, and its positive performance changes are concentrated on trajectories where skills are actually invoked. We further find that METAEXP-Qwen3.5 consolidates feedback into fewer, longer skills with more failure- and verification-oriented content and less instance-specific structure. These patterns are consistent with the two behaviors targeted during training. See full analysis in Appendix A.

6 CONCLUSION

We study whether experience learning itself can be trained as a transferable model capability. METAEXP uses verifiable cross-task families to train models to acquire, update, and reuse external experience, while leaving model parameters fixed during downstream evolution. Across heterogeneous held-out benchmarks, the trained model consistently benefits more from accumulated experience than its matched backbone, with the effect persisting under domain-excluded training and extending to the use of externally provided skills. These results suggest that experience learning can be learned and transferred beyond the task distributions used for training.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, literature organization, and code used for experiment analysis. All scientific claims, citations, reported measurements, and AI-assisted text were reviewed by the authors, who take responsibility for the final content.

ETHICS STATEMENT

No human subjects took part, and the study collected no new human-subject data. All experiments were conducted in generated or benchmark environments rather than live operations.

REPRODUCIBILITY STATEMENT

We provide the task-family formulation, environment-scaling pipeline, sequential trajectory collection, filtering procedure, and training objective in Sections 3–4. Section 5 specifies the training configuration, benchmark splits, cold-start evaluation protocol, metrics, and repeated-run setup. Appendix B reports the exact harness-level execution and reflection templates, while Appendix C characterizes the seed sources used for environment scaling. Aggregate results are reported over three independent runs unless otherwise noted.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Anthropic. Equipping agents for the real world with agent skills. https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills, 2025.

Zihao Cheng, Hongru Wang, Zeming Liu, Xinyi Wang, Xiangrong Zhu, Yuhang Guo, Wei Lin, Jeff Z. Pan, and Yunhong Wang. Terminal-World: Scaling terminal-agent environments via agent skills. *arXiv preprint arXiv:2605.20876*, 2026.

Zhiyuan Fan, Wenwei Jin, Feng Zhang, Bin Li, Yihong Dong, Yao Hu, and Jiawei Li. Evolving-rl: End-to-end optimization of experience-driven self-evolving capability within agents. *arXiv preprint arXiv:2605.10663*, 2026a.

Zhiyuan Fan, Tinghao Yu, Yuanjun Cai, Jiangtao Guan, Yun Yang, Dingxin Hu, Jiang Zhou, Xing Wu, Zhuo Han, Feng Zhang, and Lilin Wang. Toward scalable terminal task synthesis via skill graphs. *arXiv preprint arXiv:2604.25727*, 2026b.

Xingze Gao, Chuanrui Hu, Hongda Chen, Pengfei Yao, Zhao Wang, Yi Bai, Zhengwei Wu, Yunyun Han, Xiaofeng Cong, Jie Gui, et al. EvoAgentBench: Benchmarking agent self-evolution via ability transfer. *arXiv preprint arXiv:2607.05202*, 2026.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. SWE-Skills-Bench: Do agent skills actually help in real-world software engineering? *arXiv preprint arXiv:2603.15401*, 2026.

Zelin He, Haotian Lin, Boran Han, Wei Zhu, Haoyang Fang, Bernie Wang, Xuan Zhu, Runze Li, and Matthew Reimherr. Reskill: Reconciling skill creation with policy optimization in agentic rl. *arXiv preprint arXiv:2606.01619*, 2026.

Zhanbo Hua, Yifan Yao, Weihao Xie, Yongchi Zhao, Minghao Liu, Ruizhi Qiu, Zhewei Huang, Zun Wang, Yiyan Ji, Yunhai Ye, Letian Zhu, Xinping Lei, Han Li, Zhiyuan Ma, Zili Wang, Zhaoxiang Zhang, and Jiaheng Liu. CLI-Universe: Towards verifiable task synthesis engine for terminal agents. *arXiv preprint arXiv:2606.22883*, 2026.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R Fung. XSkill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026a.

Sihang Jiang, Lipeng Ma, Zhonghua Hong, Keyi Wang, Zhiyu Lu, Tengfei Wang, Shisong Chen, Jinghao Zhang, Tianjun Pan, Weijia Li, et al. SEA-Eval: A benchmark for evaluating self-evolving agents beyond episodic assessment. *arXiv preprint arXiv:2604.08988*, 2026b.

Yingtie Lei, Zhongwei Wan, Jiankun Zhang, Samiul Alam, Zixuan Zhong, Peizhou Huang, Xin Wang, Jingxuan Zhang, Donghao Zhou, Yunta Hsieh, et al. SkillEvolBench: Benchmarking the evolution from episodic experience to procedural skills. *arXiv preprint arXiv:2605.24117*, 2026.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han-chung Lee, and Dawn Song. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026a.

Yanzhou Li, Yiran Zhang, Xiaoyu Zhang, Xiaoxia Liu, and Yang Liu. CodeSkill: Learning self-evolving skills for coding agents. *arXiv preprint arXiv:2605.25430*, 2026b.

Yixuan Li, Mingshu Cai, Ziyang Xiao, Wanyuan Wang, Yanchen Deng, and Bo An. Mind-skill: Quality-guaranteed skill generation via multi-agent induction and deduction. *arXiv preprint arXiv:2605.08670*, 2026c.

Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. Muse-AutoSkill: Self-evolving agents via skill creation, memory, management, and evaluation. *arXiv preprint arXiv:2605.27366*, 2026.

Yuxuan Liu, Zhaochen Su, Lingyun Xie, Yuhao Zhang, Qing Zong, Jiahe Guo, Zhongwei Xie, Yiyan Ji, Yauwai Yim, Hongyu Luo, et al. SkillRevise: Improving llm-authored agent skills via trace-conditioned skill revision. *arXiv preprint arXiv:2606.01139*, 2026.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026.

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. *arXiv preprint arXiv:2603.25158*, 2026.

Libin Qiu, Zhirong Gao, Junfu Chen, Yuhang Ye, Weizhi Huang, Xiaobo Xue, Wenkai Qiu, and Shuo Tang. AutoRefine: From trajectories to reusable expertise for continual llm agent refinement. *arXiv preprint arXiv:2601.22758*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Zelin Tan, Yiqun Zhang, Hao Li, Zhiyao Cui, Hejia Geng, Shao Zhang, Hangfan Zhang, Yang Chen, Xiaosong Wang, Lilong Wang, Zhenfei Yin, Shuyue Hu, Chen Zhang, and Lei Bai. SKT: Skill-use training at scale via verified synthetic data generation. *arXiv preprint arXiv:2608.02287*, 2026.

Yash Vishe, Rohan Surana, Xunyi Jiang, Zihan Huang, Xintong Li, Nikki Lijing Kuang, Tong Yu, Ryan A Rossi, Jingbo Shang, Julian McAuley, et al. Skill-R1: Agent skill evolution via reinforcement learning. *arXiv preprint arXiv:2605.09359*, 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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Hanyu Wang, Yifan Lan, Bochuan Cao, Lu Lin, and Jinghui Chen. SkillGrad: Optimizing agent skills like gradient descent. *arXiv preprint arXiv:2605.27760*, 2026a.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1529–1550, 2026b.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025b.

Yifan Wei, Li Du, Xiaoyan Yu, Yang Feng, and Angsheng Li. Towards compositional generalization of LLMs via skill taxonomy guided data synthesis. *arXiv preprint arXiv:2601.03676*, 2026.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. EvolveR: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Jingxu Xie, Dylan Xu, Xuandong Zhao, and Dawn Song. AgentSynth: Scalable task generation for generalist computer-use agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

Min Yang, Jinghua Piao, Xu Xia, Xiaochong Lan, Jiaju Chen, Yongshun Gong, and Yong Li. SkillMaster: Toward autonomous skill mastery in llm agents. *arXiv preprint arXiv:2605.08693*, 2026a.

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. *arXiv preprint arXiv:2605.23904*, 2026b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Zhiyuan Yao, Yuxin Chen, Zhengxi Lu, Zishan Xu, Yueqing Sun, Yifu Guo, Yuquan Lu, Zhengzhou Cai, Kangning Zhang, Zhuowen Han, Zi-Han Wang, Ziang Ye, Qi Gu, Xunliang Cai, Weiwen Liu, and Yongliang Shen. SkillRise: Agentic reinforcement learning for cross-task skill evolution. *arXiv preprint arXiv:2607.26784*, 2026.

Zhuoyun Yu, Xin Xie, Wuguannan Yao, Chenxi Wang, Lei Liang, Xiang Qi, and Shumin Deng. SkillAdaptor: Self-adapting skills for llm agents from trajectories. *arXiv preprint arXiv:2606.01311*, 2026.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification. In *Third Conference on Language Modeling*, 2026a.

Qi Zhang, Zhaopeng Feng, Xiaonan Shi, Xiaomeng Hu, Chu Liu, Pengjun Xie, Xiaobin Wang, Jieping Ye, Bryan Hooi, Haobo Wang, et al. Skillcomposer: Learning to evolve agent skills for specification and generalization. *arXiv preprint arXiv:2606.06079*, 2026b.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, et al. SkillFlow: Benchmarking lifelong skill discovery and evolution for autonomous agents. *arXiv preprint arXiv:2604.17308*, 2026c.

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, et al. SkillWeaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo FR Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. SkillLearnBench: Benchmarking continual learning methods for agent skill generation on real-world tasks. *arXiv preprint arXiv:2604.20087*, 2026.

Leijun Zhou, Zhihao Liu, Xiang Qu, Chenxu Liu, Yifei Liu, Yanke Yu, Jingzhe Xu, Xuejun Wu, Buyue Qian, Xi Chen, et al. GDPevo: Evaluating agent self-evolution on real business tasks. *arXiv preprint arXiv:2608.03764*, 2026a.

Yifan Zhou, Zhentao Zhang, Ziming Cheng, Shuo Zhang, Qizhen Lan, Zhangquan Chen, Zhi Yang, Ronghao Chen, Huacan Wang, Sen Hu, et al. SkillGenBench: Benchmarking skill generation pipelines for llm agents. *arXiv preprint arXiv:2605.18693*, 2026b.

Yifei Zhou, Qianlan Yang, Kaixiang Lin, Min Bai, Xiong Zhou, Yu-Xiong Wang, Sergey Levine, and Li Erran Li. Proposer-agent-evaluator (PAE): Autonomous skill discovery for foundation model internet agents. In *Forty-second International Conference on Machine Learning*, 2025.

Jiapeng Zhu, Jianxiang Yu, Yibo Zhao, Chengcheng Han, Qi Gu, Xunliang Cai, Xiang Li, and Weining Qian. Skill0.5: Joint skill internalization and utilization for out-of-distribution generalization in agentic reinforcement learning. *arXiv preprint arXiv:2605.28424*, 2026a.

Kaijie Zhu, Yuzhou Nie, Yijiang Li, Yiming Huang, Jialian Wu, Jiang Liu, Ximeng Sun, Zhenfei Yin, Lun Wang, Zicheng Liu, Emad Barsoum, William Yang Wang, and Wenbo Guo. TermiGen: High-fidelity environment and robust trajectory synthesis for terminal agents. *arXiv preprint arXiv:2602.07274*, 2026b.

Table 5: SpreadsheetBench experience-access behavior. Values are means ± population standard deviations over three seeds.

Model	Condition	Skills in bank	Any invocation (%)	Skill as first tool (%)	Calls/task
Baseline-Qwen3.5	$k = 5$	3.0 ± 0.0	9.3 ± 2.0	2.4 ± 1.3	0.094 ± 0.021
	$k = 10$	5.3 ± 0.5	11.3 ± 2.5	4.2 ± 1.5	0.113 ± 0.025
	$k = 20$	10.0 ± 0.0	18.6 ± 4.7	7.8 ± 0.1	0.188 ± 0.049
	$k = 40$	10.0 ± 0.0	17.0 ± 8.2	5.0 ± 2.6	0.172 ± 0.084
METAEXP-Qwen3.5	$k = 5$	2.0 ± 0.8	76.4 ± 16.4	63.1 ± 23.2	0.765 ± 0.162
	$k = 10$	3.3 ± 2.6	79.5 ± 16.9	67.0 ± 23.0	0.800 ± 0.164
	$k = 20$	4.3 ± 4.0	81.0 ± 12.3	67.8 ± 15.4	0.811 ± 0.121
	$k = 40$	4.7 ± 3.9	81.5 ± 10.0	70.6 ± 15.0	0.819 ± 0.094

Table 6: Paired success-rate change relative to the same task under an empty-bank condition (%).

Model	Conditions	Invoked skill	Did not invoke
Baseline-Qwen3.5	All k pooled	-4.2	-1.2
METAEXP-Qwen3.5	All k pooled	$+9.5$	-1.9
METAEXP-Qwen3.5	$k = 20$	$+10.8 \pm 4.7$	-0.3 ± 4.8
METAEXP-Qwen3.5	$k = 40$	$+10.2 \pm 6.4$	-4.7 ± 6.7

A FURTHER ANALYSIS

The main results show that METAEXP-Qwen3.5 benefits substantially more from accumulated experience than its matched backbone. We further examine how this difference manifests in the two behaviors targeted during training: using accumulated experience during task execution and updating that experience from feedback. Our analysis focuses on SpreadsheetBench and compares METAEXP-Qwen3.5-122B-A10B with its matched Qwen3.5-122B-A10B backbone, referred to below as METAEXP-Qwen3.5 and Baseline-Qwen3.5.

A.1 EXPERIENCE-CONDITIONED EXECUTION

We analyze 8,399 valid SpreadsheetBench evaluation trajectories across both models, three seeds, and $k \in \{5, 10, 20, 40\}$. A skill invocation is counted only when the model calls a generated skill available in the current bank; built-in harness skills and invalid skill names are excluded. Table 5 summarizes how often and how early each model accesses its accumulated skills during execution. Across the four conditions, Baseline-Qwen3.5 invokes a skill on 14.1% of tasks, compared with 79.6% for METAEXP-Qwen3.5. The difference is even larger for early use: a skill is the first tool invoked on 4.8% of tasks for the baseline and 67.1% for METAEXP-Qwen3.5. This behavior cannot be attributed to a larger skill bank, since METAEXP-Qwen3.5 maintains fewer skills throughout adaptation. Instead, the trained model consults its accumulated experience much more consistently and earlier in the execution process.

More frequent invocation does not by itself indicate that the retrieved experience is beneficial. We therefore pair each task with the corresponding run from the same model and seed under an empty-bank condition. Table 6 reports the change in success rate relative to this matched run. For METAEXP-Qwen3.5, tasks on which a skill is invoked improve by $+9.5$ points on average relative to their matched empty-bank runs. By contrast, the invoked subset of Baseline-Qwen3.5 performs 4.2 points below its empty-bank counterpart. The gains of METAEXP-Qwen3.5 are therefore closely associated with trajectories in which accumulated experience is actually used during execution.

A.2 FEEDBACK-DRIVEN EXPERIENCE UPDATING

We next examine the skill banks produced through repeated feedback-driven updates. Because these banks reflect the model's decisions about what to retain, revise, or discard over adaptation, their final structure offers a complementary view of the learned experience-update behavior. Table 7 compares the banks after 40 adaptation tasks. Although the two models produce similar total text volume, they organize that content quite differently. METAEXP-Qwen3.5 stores substantially fewer skills, resulting in considerably longer skills on average. Its banks also contain more language related to failures

Table 7: Structure of the final SpreadsheetBench skill banks at $k = 40$, pooled over three seeds.

Metric	Baseline-Qwen3.5	METAEXP-Qwen3.5
Skills	30	14
Lexical tokens	29,679	31,155
Tokens per skill	989.3	2,225.4
Failure terms / 1K tokens	1.42	5.42
Verification terms / 1K tokens	3.13	4.56
Opaque IDs / 1K tokens	2.26	0.00
Top-2 invocation share	32.2%	74.5%

Table 8: Case study on how different models encode experiences on SpreadsheetBench.

Case	Baseline-Qwen3.5	METAEXP-Qwen3.5
Cross-sheet lookup	A local VLOOKUP/INDEX–MATCH recipe with several formula variants and a recommendation to write static values.	A symptom–cause–repair–verification taxonomy covering truncated ranges, wrong-sheet references, hidden whitespace, cached formula values, and cell-level validation.
Boundary-sensitive editing	A generic checklist: preserve values, use absolute references, handle missing keys, and verify the output.	Conditional diagnostic rules for inflated worksheet bounds and locale-dependent formula separators, tied to observable failure signals and corrective checks.

and verification, no detected opaque identifiers, and a larger share of total invocations concentrated on a small number of skills. This pattern suggests stronger consolidation around reusable diagnostic and validation-oriented knowledge, with less fragmentation into narrowly scoped or instance-specific records.

Because greater length or concentration can also introduce risks such as overgeneralization, we emphasize that these statistics characterize bank structure rather than skill quality directly. Table 8 provides qualitative examples of how these structural differences appear in the skill contents. Baseline-Qwen3.5 tends to record direct procedures or local recipes for completing specific operations. By contrast, METAEXP-Qwen3.5 more often organizes experience around applicability conditions, possible causes, corrective actions, and subsequent verification. These examples align with the quantitative bank-level patterns above and suggest that meta-experience training encourages feedback to be consolidated into more structured and broadly reusable guidance, while reducing reliance on task-local procedures.

Taken together, the analyses indicate that meta-experience training changes both sides of the experience-learning process. METAEXP-Qwen3.5 makes accumulated skills a more active part of subsequent task execution and organizes interaction feedback into a more consolidated form for future reuse.

B PROMPT TEMPLATES FOR EXPERIENCE LEARNING

As described in Section 4, an agent trajectory consists of task execution followed by feedback-driven reflection. We use fixed harness-level templates for these two stages during both training-trajectory collection and experience-learning evaluation. The templates shown below are shared across models; runtime-provided context, such as the underlying agent system message, available-skill inventory, and execution reminders, are supplied separately by the specific agent harness.

Task execution. The execution template initializes each task, specifies the sandbox layout, and instructs the agent to consult relevant accumulated skills when available.

```
# Goal
You are solving a task in a linux sandbox.
# Environment
- `/workspace/source/`: Data, tool packages, and README.md file
```

```
- `/workspace/input/`: Read-only task description and sub-task files
- `/workspace/output/`: Write ALL output files here (do not write
    elsewhere)
# Recommended Workflow
1. Read `/workspace/input/task.txt` to understand the task.
2. Explore `/workspace/source/` to understand available data.
3. If any available skill covers the task or a sub-step, invoke it first.
4. Write and execute bash/Python scripts to produce the required output.
5. Verify that all output files are in `/workspace/output/`.
6. Call `task_done` when finished.
```

Experience reflection. After task evaluation, feedback is injected into the same interaction session using the following template. The agent can therefore inspect its preceding execution trajectory together with the evaluation outcome before deciding whether to create, revise, or retain skills in the current bank.

```
You are a reflective learning agent. Your goal is to convert task
    experience into reusable skills that improve future performance.
### Evaluation Result
Your performance on the previously assigned task has been evaluated. The
    reward score is:
<reward>
{reward}
</reward>
The score ranges from 0 (complete failure) to 1 (complete success).
<failing_rubrics>
{failing_rubrics}
</failing_rubrics>
### Objective
Based on this outcome, distill your experience into a reusable skill that
     will help you or another AI assistant perform better on similar
    tasks in the future. You may access and save skills to the following
    directory:
<skills_dir>
{skills_dir}
</skills_dir>
Each skill is a directory containing a SKILL.md file with YAML
    frontmatter (name, description) and markdown instructions. Use the
    skill-creator skill for guidance on creating or updating skills.
### Workflow
1. Analyze your task trajectory in light of the reward signal.
2. Check existing skills and decide whether to create a new one, update
    an existing one, or leave them unchanged.
3. Call `task_done` when complete.
### Rules
- Always use the skill-creator skill to create or update skills.
- DO NOT modify the `skill-creator` skill itself.
- The total number of available skills must not exceed **{max_skill_num
    }**. Only create a new skill when no existing skill adequately covers
     the scenario.
```

At runtime, `reward` contains the final task score, while `failing_rubrics` contains either unsatisfied verifier criteria or error assertions, depending on the verifier type. `skills_dir` specifies the mounted skill bank, and `max_skill_num` controls its capacity.

C SEED DIVERSITY FOR ENVIRONMENT SCALING

The family-scaling pipeline draws on heterogeneous seeds from pretraining-style corpora, task examples, and tool schemas. These sources provide complementary starting points for construction: corpus passages supply semantic contexts, task examples provide procedural patterns for workflow abstraction, and tool schemas provide executable affordances for tool-based interaction. We charac-

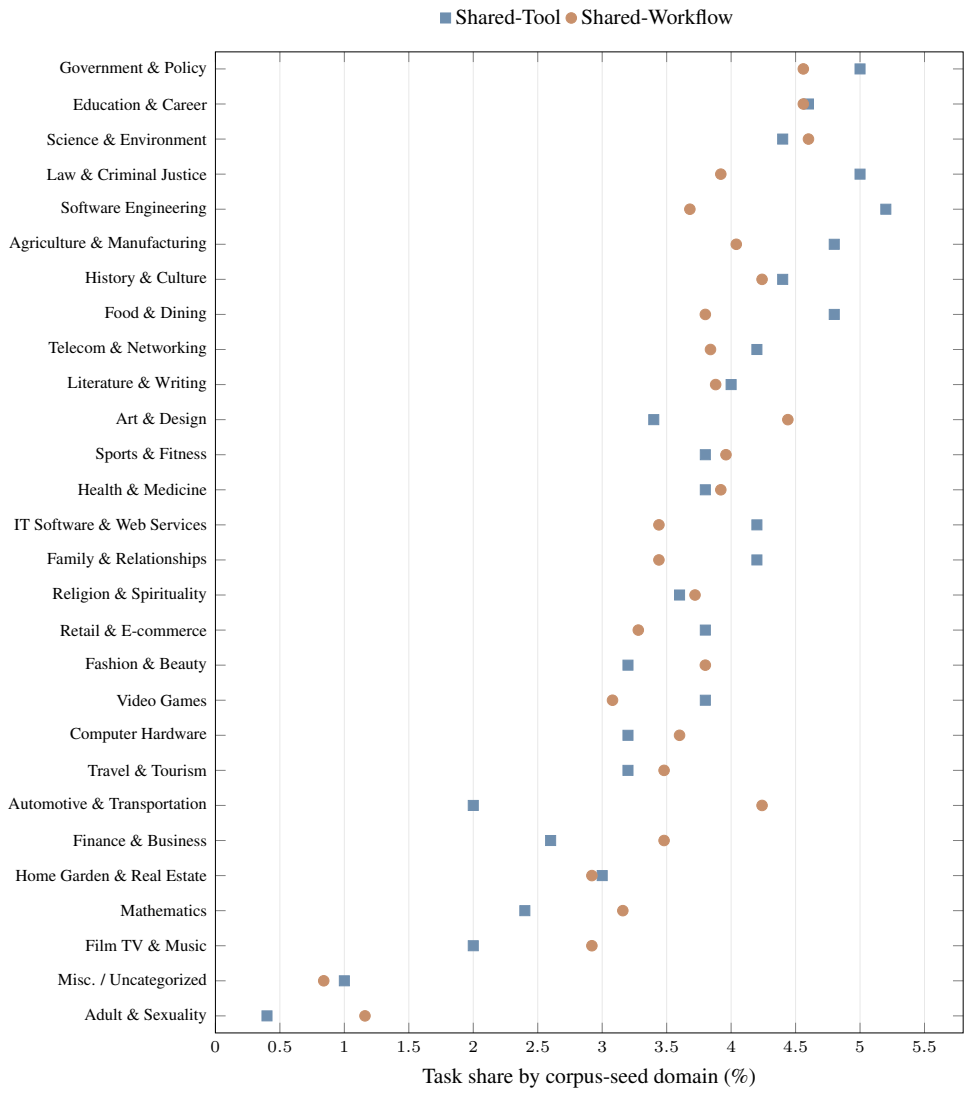


Figure 3: Domain distribution of corpus seeds under Shared-Tool and Shared-Workflow construction.

terize their diversity below to illustrate the breadth of the substrate available for scaling controlled task families.

Corpus seeds. Corpus passages provide the semantic context used during environment construction. Shared-Tool uses 500 independently sampled passages, each reused across the five members of one family, whereas Shared-Workflow independently samples passages for all 2,500 task instances. As shown in Figure 3, both construction modes cover all 28 domains, while independent sampling yields a slightly more even task-level distribution for Shared-Workflow.

Task-example seeds. The 500 task examples provide candidate procedural patterns from which Shared-Workflow families can be abstracted. For analysis, each example is assigned one primary intent using a reproducible researcher annotation rather than a source-provided label. Figure 4 shows that the seed set emphasizes agent configuration, software and system operations, automation, and finance, while retaining a long tail of other intents.

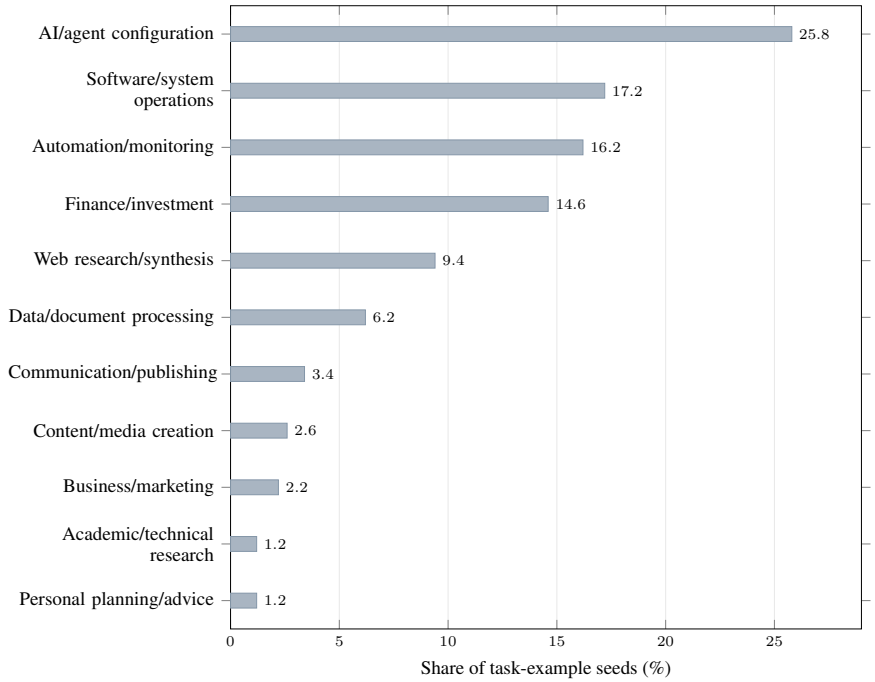


Figure 4: Primary-intent distribution of the task-example seeds used in METAEXP.

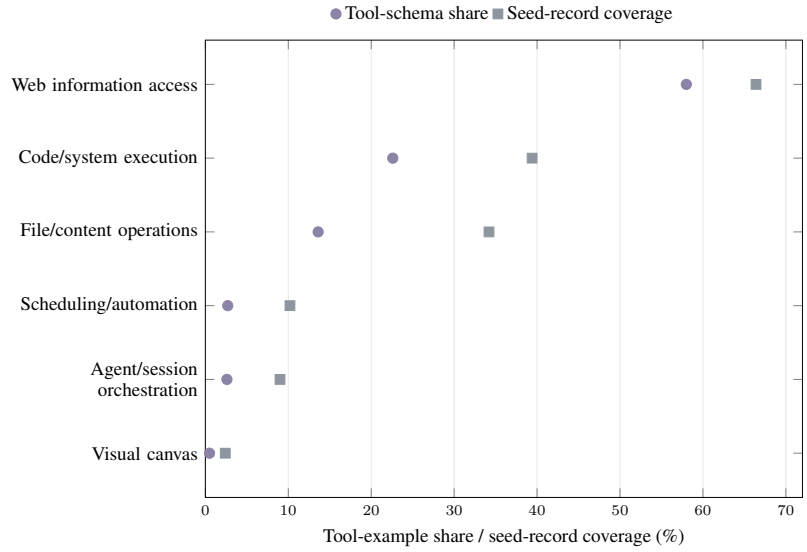


Figure 5: Capability distribution of tool examples and their coverage across seed records.

Tool-example seeds. The 500 seed records contain 2,338 tool-use examples, each pairing an interface schema with a corresponding usage example. These examples span 16 top-level tool names, which we deterministically group into six capability categories for analysis. Each seed record contains between 1 and 10 tool examples, with 4.676 examples on average. Figure 5 reports two complementary statistics. *Tool-example share* measures the fraction of the 2,338 examples assigned to each capability category, while *seed-record coverage* measures the fraction of the 500 seed records containing at least one example from that category.

Together, the three sources provide heterogeneous semantic contexts, procedural patterns, and executable affordances, giving the family-scaling pipeline diverse starting points without tying construc-

tion to a narrow set of manually specified environments. These statistics characterize the breadth of the scaling seeds rather than the final training distribution.

D DISCUSSION

Our current training strategy uses verified teacher trajectories to supervise experience-conditioned execution and feedback-driven experience updating. While this provides an efficient way to bootstrap transferable experience-learning behavior, it does not explicitly optimize the downstream utility of each experience update. A natural next question is therefore how future task performance can be incorporated more directly into training.

Why supervised fine-tuning? The environments, tasks, and executable verifiers constructed by METAEXP readily support reinforcement learning. However, experience learning introduces a delayed cross-episode objective that is more difficult to optimize than task-level reward. An update to the skill bank may appear reasonable on the task that produced it, yet its true utility is only revealed when that experience is reused on subsequent tasks. Reinforcement learning would therefore require attributing future performance changes back to earlier execution and reflection decisions, creating a nontrivial credit-assignment problem. It would also require rolling out multiple downstream tasks to evaluate earlier updates, substantially increasing the cost of optimization over long sequential task streams.

For this reason, we use rejection sampling followed by supervised fine-tuning as a practical first step. Executable verification allows us to filter reliable teacher trajectories at substantially lower cost, providing the base model with an initial capability for acquiring and reusing experience before introducing more expensive cross-episode optimization.

Filtering by downstream experience utility. The task-family structure of METAEXP also supports an intermediate approach between local trajectory filtering and full reinforcement learning. Because a family can contain many related but non-identical tasks, some members can be reserved as held-out validation tasks while the remaining members are used for sequential experience collection. After an update from B_{i-1} to B_i, the two banks can be evaluated on the same held-out family members, yielding

$$\Delta_i^{\mathrm{exp}} = S_{\mathrm{val}}(B_i) - S_{\mathrm{val}}(B_{i-1}). \tag{9}$$

Unlike the task verifier, which measures whether the current task was solved correctly, Δ_i^{exp} measures whether the resulting experience improves performance on related tasks that were not used to construct the bank. It can therefore provide a stronger rejection criterion for selecting teacher trajectories whose experience updates exhibit positive downstream transfer.

This strategy aligns trajectory selection more directly with the objective of experience learning, but requires additional evaluation rollouts for each candidate bank update and is consequently more expensive than the filtering used in this work. We leave downstream-utility-based filtering and reinforcement learning with cross-episode reward signals as promising directions for further improving meta-experience learning.