FROM SELF-CORRECTION TO SELF-EVOLUTION: AGENTIC TEST-TIME LEARNING FOR MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in self-correction (SC) and recursive self-improvement (RSI) offer new ways to improve large language models' reasoning. A natural approach is to accumulate correction experience as reusable skills to guide subsequent problem solving. To this end, we propose SC2SE (Self-Correction to Self-Evolution), an agentic test-time learning framework for mathematical reasoning. With model parameters frozen and without requiring ground-truth answers as learning feedback, SC2SE learns from its own verification and revision experience through four collaborating agents for problem solving, stepwise correction, skill evolution, and role calibration. These agents distill correction experience into skills; role calibration checks each skill's suitability for the solver role before and after skill aggregation, turning diagnoses of existing solutions into actionable guidance for subsequent problem solving. Updated skills guide the next round of problem solving, and new answers undergo further verification and correction, forming a test-time self-evolution loop: correction $\rightarrow$ skill accumulation $\rightarrow$ skill-assisted answering $\rightarrow$ further correction. We conduct same-pool evaluation on four mainstream large language models and five public mathematics datasets. The results show that skill-assisted initial answers achieve higher mean accuracy (ACC) than the compared RSI baselines; with further correction, SC2SE improves mean ACC by 14.3 percentage points over the highest mean ACC among the compared baselines. Together with component comparisons and process case studies, these results support the complementary roles of self-correction and skill accumulation: correction provides experience for skill learning, while accumulated skills support subsequent problem solving and revision.

1 INTRODUCTION

Recent advances in self-correction (SC) and recursive self-improvement (RSI) offer new ways to improve large language models' reasoning (Song et al., 2025; Zelikman et al., 2024). Self-correction improves the current answer through verification and feedback-guided revision, while self-improvement research emphasizes accumulating and reusing strategies developed during problem solving to support subsequent answers (Suzgun et al., 2026; Zhang et al., 2026). For mathematical tasks involving multi-step derivations, condition handling, and precise computation, the former may improve the current solution's quality, while the latter may yield transferable problem-solving strategies.

Existing research has explored both self-correction and experience accumulation. For self-correction, generation-time interventions, self-correction training, and tool-based or programmatic feedback provide different routes to improved reasoning (Gao et al., 2026b; Kumar et al., 2025; Gou et al., 2024a; Song et al., 2025), with effectiveness depending on the feedback source and evaluation conditions (Huang et al., 2024; Kamoi et al., 2024). Related studies also examine error-step identification and revision given error information as separate capabilities (Tyen et al., 2024; Zheng et al., 2025). For experience accumulation, Reflexion and ExpeL support subsequent attempts through reflective memory and natural-language experience, respectively (Shinn et al., 2023; Zhao et al., 2024), while LEAP identifies errors in few-shot examples with reference answers and distills principles (Zhang et al., 2024). Dynamic Cheatsheet and EvoLib explore the continual extraction and

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.

maintenance of memories or skills (Suzgun et al., 2026; Xu et al., 2026). ReasoningBank and Agentic Context Engineering (ACE) accumulate and reuse problem-solving experience through reasoning memories and playbooks, respectively (Ouyang et al., 2026; Zhang et al., 2026).

These studies show that verification, feedback, and experience reuse can all improve reasoning. Building on these findings, we explore integrating stepwise mathematical correction with skill accumulation, transforming diagnostic and revision experience into reusable skills for subsequent problem solving. The rationale is that stepwise verification provides fine-grained evaluation signals for intermediate reasoning steps (Lightman et al., 2024), and accurate error-location information helps models correct reasoning errors (Tyen et al., 2024). Stepwise correction also provides information about error causes and revisions that can be distilled into actionable rules for current and subsequent problems. In Figure 1, the process on the left verifies and revises the current answer. The process on the right additionally accumulates correction experience as reusable skills for subsequent answers and continually updates them with new verification and correction experience.

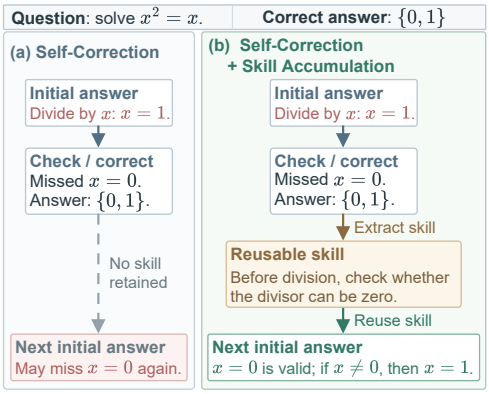

Figure 1: Self-correction with and without skill accumulation.

Turning this experience into reusable skills further requires a shift from the corrector's perspective to the solver's. Correction feedback explains what went wrong in an existing solution, whereas the solver must determine what action to take from the problem conditions. For example, the diagnosis in Figure 1, "this division omitted the zero case," must become "before dividing by an expression containing an unknown, check the zero case" to provide actionable guidance for subsequent problem solving. Newly extracted skills must also remain suitable for the solver role after aggregation and rewriting with the existing library.

We therefore propose SC2SE (Self-Correction to Self-Evolution), a test-time learning framework in which four agents collaborate to connect stepwise correction, experience transformation, and skill-assisted answering. First, the **solver agent** generates a solution with the current skill library. The **stepwise correction agent** then performs stepwise programmatic checks, invokes mathematical tools when needed, generates feedback, and revises the solution. Using this evidence for skill learning, the **skill evolution agent** combines the current round's correction experience with existing skills to distill problem-solving guidance and propose library updates. Aggregation and maintenance reconcile duplicate or conflicting guidance while retaining relevant applicability conditions. The **role calibration agent** then checks each new proposal before aggregation for direct usability by the solver, revises statements that depend on current verification findings or the correction process, and discards entries that remain unsuitable. Because aggregation merges and rewrites guidance, the agent rechecks its suitability for the solving stage afterward. The updated skills support the solver's next round of answering. New answers undergo verification and correction as needed, providing further learning experience and forming a test-time self-evolution loop: correction $\rightarrow$ skill accumulation $\rightarrow$ skill-assisted answering $\rightarrow$ further correction. With model parameters frozen, the framework updates an external skill library using verification and revision experience; ground-truth answers are used only for independent scoring of experimental results.

We conduct same-pool test-time learning experiments on four mainstream large language models and five public mathematics datasets, evaluating both skill-assisted initial answers and the same answers after further correction. The results show that skill-assisted initial answers achieve higher mean accuracy than the compared RSI baselines. With further correction, SC2SE improves mean accuracy by 14.3 percentage points over the highest mean ACC among the compared baselines. With DeepSeek Flash, leave-one-dataset-out skill-assisted initial answers achieve 60.0% ACC, versus 52.9% for Base (Appendix B.2).

Our main contributions are as follows: (1) We propose SC2SE, which combines stepwise mathematical correction with skill accumulation and reuse. With model parameters frozen and without ground-truth answers as learning feedback, it forms a test-time self-evolution loop: acquiring expe-

rience through correction → accumulating skills → solving with skills → correcting again. (2) We combine experience distillation with role calibration before and after skill aggregation to transform diagnoses of existing solutions into guidance the solver can apply based on problem conditions, while preserving the skills' suitability for the solver role after aggregation and rewriting. (3) Experiments show that skill-assisted initial answers achieve higher mean accuracy than the compared RSI baselines, and that mean accuracy after further correction exceeds that of all compared baselines. Component comparisons and process case studies support the complementary roles of self-correction and skill accumulation; leave-one-dataset-out evaluation further supports skill transfer across mathematics datasets.

2 Related Work

Our work relates to self-correction, test-time learning, and collaborative agent reasoning.

Self-correction. Research primarily considers feedback sources and verification granularity. Self-generated feedback, tool feedback, and programmatic verification support different revision procedures: Self-Refine iteratively revises answers using self-generated feedback, CRITIC introduces external tool feedback, and ProgCo organizes checks through verification pseudoprograms while jointly revising answers and verification programs (Madaan et al., 2023; Gou et al., 2024a; Song et al., 2025). Program execution and tool use also support problem solving: PAL passes generated programs to an interpreter, while ToRA combines natural-language reasoning with tool calls (Gao et al., 2023; Gou et al., 2024b). Verification has expanded from candidate-solution ranking and outcome feedback to step-level process supervision (Cobbe et al., 2021; Uesato et al., 2022), including process reward models trained on human-annotated or automatically constructed data (Lightman et al., 2024; Wang et al., 2024b). Error identification and answer revision are connected but distinct stages. Tyen et al. evaluate them separately and show that accurate error locations facilitate subsequent revision (Tyen et al., 2024), while ProcessBench evaluates identification of the first erroneous step (Zheng et al., 2025).

Test-time learning. Test-time learning studies how models use test-time feedback and experience to improve subsequent answers. Related work addresses experience representation and reuse, continual maintenance, and update optimization. For representation and reuse, trajectories are organized into reflective memories or natural-language experience (Shinn et al., 2023; Zhao et al., 2024). Other representations include reasoning modules and thought templates (Zhou et al., 2024; Yang et al., 2024b), code skills and workflows (Wang et al., 2024a; 2025), and retrievable subproblem–subprocedure knowledge (Wu et al., 2026). For continual maintenance, Dynamic Cheatsheet stores and reuses strategies, code snippets, and problem-solving experience, while ACE maintains playbooks through generation, reflection, and curation. ReasoningBank and EvoLib extract reasoning strategies or skills from problem-solving experience and continually update their memories or knowledge libraries (Suzgun et al., 2026; Zhang et al., 2026; Ouyang et al., 2026; Xu et al., 2026). For update optimization, language critiques, candidate scoring, and reflection on execution trajectories are used to improve prompts (Pryzant et al., 2023; Yang et al., 2024a; Agrawal et al., 2026). TextGrad propagates language feedback through a computation graph (Yuksekgonul et al., 2025), while SkillOpt and SkillAudit guide updates through budget-constrained structured edits and audits of paired trajectories with and without skills, respectively (Yang et al., 2026; Gao et al., 2026a). These methods operate under their respective feedback conditions: for example, LEAP derives principles from errors on examples with reference answers, while Training-Free GRPO compares groups of trajectories and accumulates experience using a small dataset with ground-truth answers (Zhang et al., 2024; Cai et al., 2025).

Collaborative agent reasoning. Related work focuses on interaction organization and reasoning improvement. For interaction organization, ReAct connects reasoning, actions, and external observations (Yao et al., 2023). Multi-agent frameworks further coordinate roles through conversational mechanisms or standardized workflows, such as AutoGen's customizable conversations and MetaGPT's workflows and intermediate-result checks (Wu et al., 2024; Hong et al., 2024). To improve reasoning, multi-round debate combines judgments through interactions among model instances (Du et al., 2024), while MAgICoRe uses step-level reward feedback to organize coarse-grained answer aggregation and fine-grained multi-agent revision (Chen et al., 2025). We further

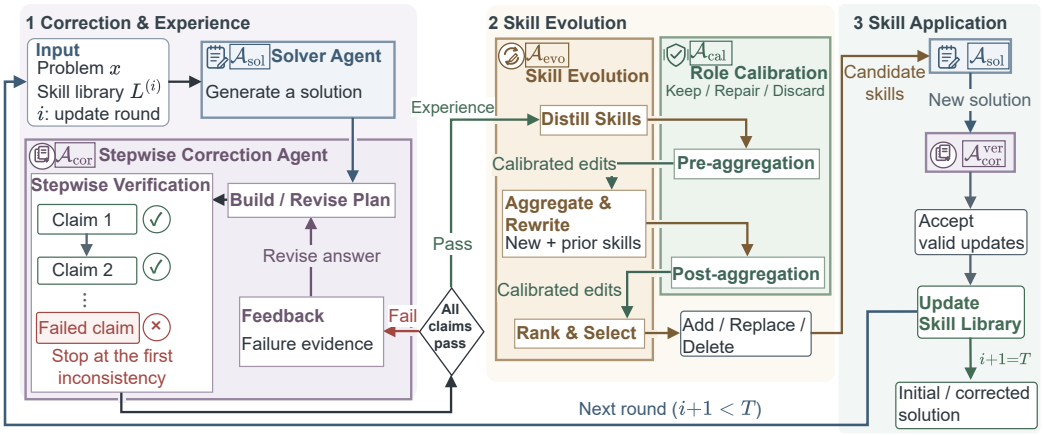

Figure 2: Overview of the SC2SE framework.

investigate how experience from stepwise mathematical correction can be transformed into reusable, continually updatable skills for the solving stage.

3 METHOD

As shown in Figure 2, SC2SE uses three stages to transform current-answer correction experience into reusable skills for subsequent answers. Pseudocode is provided in Appendix A.

Stage 1: correction and experience acquisition. The solver agent $\mathcal{A}_{\text{sol}}$ generates a solution with the current skill library. The stepwise correction agent $\mathcal{A}_{\text{cor}}$ follows a verification plan, gathers evidence with mathematical tools as needed, and revises the solution using feedback until verification passes or the correction budget is exhausted. The resulting experience feeds skill evolution. **Stage 2**: skill evolution. The skill evolution agent $\mathcal{A}_{\text{evo}}$ distills problem-solving guidance from the current round's experience and aggregates and rewrites it with existing skills. The role calibration agent $\mathcal{A}_{\text{cal}}$ checks each skill's suitability for solving before and after aggregation, rewriting statements tied to current verification findings as actionable guidance for the solver. Edits selected within the per-round proposal limit add, replace, or delete skills to form a candidate library. **Stage 3**: skill application. The same solver generates new initial answers with the candidate library; the stepwise correction agent checks them in verification-only mode and records the results. Valid updates passing role calibration, proposal selection, and structural checks are accepted. The figure's bottom loop uses the updated library for the next answering round and new verification and correction experience for further skill updates. After all T skill-update rounds, the final library $L^{(T)}$ is fixed, and skill-assisted initial answers and their further-corrected counterparts are evaluated separately. Additionally, model parameters remain frozen at test time, and skill learning uses no ground-truth answers.

3.1 CORRECTION AND EXPERIENCE ACQUISITION

Let x denote a problem, y a generated solution, and a the answer extracted from that solution. Let $L^{(i)}$ denote the skill library after i updates, with $L^{(0)} = \varnothing$. For $i = 0, \ldots, T-1$, round $i+1$ collects solving and correction trajectories with $L^{(i)}$ fixed, then updates the library once trajectory collection is complete.

During problem solving, the current skill library $L^{(i)}$ is included as guidance in the system prompt, while the problem is provided separately. The model generates a complete solution and extracts an answer:

$$(y_0, a_0) = \mathcal{A}_{\text{sol}}(x; L^{(i)}), \qquad \tau_x^{(i)} = \mathcal{A}_{\text{cor}}^{\text{corr}}(x, y_0, a_0). \tag{1}$$

Here, (y_0, a_0) is the initial answer, and (y_k, a_k) is the solution and extracted answer after the kth revision. In the first round, a matching shared initial answer (y_0, a_0) is reused when available. The correction agent then receives the problem, initial solution, and extracted answer, and alternates between verification and revision within the allotted correction rounds to produce a trajectory. The

superscript corr denotes verification–revision mode. The trajectory $\tau_x^{(i)}$ records verification, feedback, and revision for this problem in the current round, terminating when verification passes or the round budget is exhausted. If verification has not passed when the correction budget is exhausted, the latest solution is returned as the final output. The correction agent first constructs a verification plan, then checks the solution's key mathematical claims in the specified order. It invokes mathematical tools when precise computation, symbolic derivation, or enumeration is needed. At the first inconsistency, it stops subsequent checks and generates feedback and a complete revised solution. If feedback arises from an inadequate verification plan, revision may retain the original solution and explain why. During re-verification, the plan is rebuilt if the answer changes and revised if it remains unchanged. If an execution error prevents verification, the current solution is retained and the trajectory is excluded from experience distillation.

Each trajectory's success/failure label is determined by the complete self-verification result for its initial answer and remains fixed. Trajectories that pass verification are used to extract candidate strategies; those that fail, together with their revisions, are used to identify error patterns and responses. The complete record contains the problem, skill library used, initial answer, verification plan, execution evidence, judgment, feedback, and revision results.

3.2 Experience Transformation and Skill Optimization

Correction feedback explains what went wrong on the current problem, whereas skills must guide the solver's actions in similar situations. Transforming experience therefore requires a shift from the corrector's checking perspective to the solver's action perspective: retaining reusable mathematical content and applicability conditions while rewriting statements that depend on findings from the current verification or the internal correction process.

Skill-library maintenance builds on budget-constrained structured editing (Yang et al., 2026). We further use stepwise verification and revision evidence to drive proposals and introduce role calibration before and after aggregation. Pre-aggregation calibration aligns new proposals with the solving stage. Since aggregation with the existing library may merge or rewrite skills, post-aggregation calibration checks their intended role again to prevent the integrated guidance from depending on the correction process.

We first extract an evidence set $D^{(i)}$ from the current round's trajectories, including problems, initial answers, verification signals, feedback, and revisions, while retaining trajectory provenance and success/failure labels. Internal verification programs and tool logs are stored separately in the raw records. Selective compression is triggered only when the full request reaches the context-length threshold: key evidence, including the problem, extracted answer, verification signal, and first feedback item, is preserved verbatim, while longer initial, revised, and final solutions serve as compressible supporting material. The original trajectories are retained separately.

After reflection conditioned on the current library, proposals are calibrated before and after aggregation, then selected for editing:

$$\begin{aligned} E_{\mathrm{ref}}^{(i)} &= \mathcal{A}_{\mathrm{cal}}\Big(\mathcal{A}_{\mathrm{evo}}^{\mathrm{reflect}}(D^{(i)}; L^{(i)}); L^{(i)}\Big), \\ E_{\mathrm{merge}}^{(i)} &= \mathcal{A}_{\mathrm{cal}}\Big(\mathcal{A}_{\mathrm{evo}}^{\mathrm{merge}}(E_{\mathrm{ref}}^{(i)}; L^{(i)}); L^{(i)}\Big), \\ E_{\mathrm{sel}}^{(i)} &= \mathcal{A}_{\mathrm{evo}}^{\mathrm{rank}}(E_{\mathrm{merge}}^{(i)}; L^{(i)}, b_i). \end{aligned} \tag{2}$$

All three E terms denote sequences of structured edit proposals associated with library $L^{(i)}$. Here, $E_{\mathrm{ref}}^{(i)}$ contains the preliminary proposals after reflection and calibration, $E_{\mathrm{merge}}^{(i)}$ contains proposals aggregated with the existing library and calibrated again, and $E_{\mathrm{sel}}^{(i)}$ contains the selected proposals. Both calls to $\mathcal{A}_{\mathrm{cal}}$ take an edit-proposal sequence and the current library context as input and return a calibrated edit-proposal sequence.

The parameter b_i is the maximum number of edit proposals in round $i+1$, so $|E_{\mathrm{sel}}^{(i)}| \le b_i$. All proposals are retained when their number is within this limit; otherwise, the skill evolution model prioritizes them based on the current library and proposals. Each proposal may contain multiple skills and retains its operation type, target, and provenance information. The selected addition,

replacement, or deletion operations are executed programmatically after structural validity checks, yielding a candidate library $L^{(i)\prime}$.

Reflection distills feedback on individual problems into reusable mathematical knowledge, problem-solving strategies, or self-checking procedures. Each skill is recorded independently with a title and complete guidance, forming the smallest unit of content checking. $\mathcal{A}_{\text{cal}}$ checks whether each item of guidance can be used during problem solving based on the problem conditions: suitable skills are retained, repairable skills are revised and rechecked, and those that remain unsuitable are discarded.

Illustration: Correction → reusable skill.
Problem: $z^2 = z, z \in \mathbb{R}$.
Initial: divide by $z \Rightarrow \{1\}$ (misses $z = 0$).
Correction: $z(z-1) = 0 \Rightarrow \{0, 1\}$.
Skill: "Before division, check the zero case."
Next problem: $w^2 = 2w$.
With skill: $w(w-2) = 0 \Rightarrow \{0, 2\}$ ✓.

Let $c_{\text{cal}}(s)$ denote the calibration agent's internal processing of a single skill s:

$$c_{\text{cal}}(s) = \begin{cases} s, & \text{the original skill passes the role check,} \\ \widetilde{s}, & \text{the original fails, but the revision passes rechecking,} \\ \perp, & \text{repair is infeasible or the revision still fails.} \end{cases} \tag{3}$$

Here, $\widetilde{s}$ is the revised skill and $\perp$ denotes rejection. The function c_{cal} specifies the rule for a single skill, while $\mathcal{A}_{\text{cal}}$ organizes these operations at the edit-proposal level. Calibrated skills are inserted back into their original proposals, preserving the corresponding operation types and targets. An addition or replacement proposal is removed if all its skills are rejected. Pure deletion operations are handled through structural checks.

Pre-aggregation calibration checks whether new proposals are solver-oriented; post-aggregation calibration checks for role drift introduced by merging or rewriting. A previous passing result can be reused for skills with identical content. Aggregation reconciles redundancy and conflicts with the existing library, while role calibration keeps skills suitable for the solving stage.

3.3 Skill Application and Library Updates

For the candidate library $L^{(i)\prime}$, we use the same prompting procedure and call $\mathcal{A}_{\text{sol}}(x; L^{(i)\prime})$ on the same evaluation pool to generate initial answers (y_0, a_0). We then call $\mathcal{A}^{\text{ver}}_{\text{cor}}(x, y_0, a_0)$ to fully verify each initial answer once and record the result. The superscript ver denotes verification-only mode, which returns the same type of trajectory record as corr mode, but with an empty revision record.

After verification, valid updates are accepted, and the candidate becomes the library used in the next round. Edit execution and library updates are expressed as:

$$L^{(i)\prime} = \text{Apply}\left(L^{(i)}, E^{(i)}_{\text{sel}}\right), \qquad L^{(i+1)} = L^{(i)\prime}. \tag{4}$$

Here, Apply programmatically applies the selected structured edits to the existing library. If the valid edit set is empty, $L^{(i+1)} = L^{(i)}$.

After all updates, the final library $L^{(T)}$ is fixed, and skill-assisted initial answers and their further-corrected counterparts are evaluated separately. For each evaluation problem, the outputs of the two stages are:

$$(y_{\text{init}}, a_{\text{init}}) = \mathcal{A}_{\text{sol}}(x; L^{(T)}), \qquad (y_{\text{corr}}, a_{\text{corr}}) = \text{Final}(\mathcal{A}^{\text{corr}}_{\text{cor}}(x, y_{\text{init}}, a_{\text{init}})). \tag{5}$$

Final extracts the complete solution and its extracted answer at trajectory termination; if no revision occurred, it returns the initial answer. These two outputs correspond to the skill-assisted initial and further-corrected results reported in the experiments.

4 Experiments

We evaluate SC2SE on four base models and five mathematics datasets, examining overall performance, the complementary roles of skill accumulation and correction, and the process of transforming and reusing experience.

4.1 EXPERIMENTAL SETUP

Datasets, models, and baselines. We use five public mathematics datasets. AIME 2024 (A24) (Hugging Face H4, 2025) and AIME 2025 (A25) (Balunovic et al., 2025) emphasize multi-step reasoning and precise computation. BRUMO 2025 (B25) and HMMT February 2025 (HM) (Balunovic et al., 2025) cover competition topics in algebra, geometry, combinatorics, and number theory. AMO-Bench (AM) (Liu et al., 2026) provides more challenging Olympiad-level mathematical reasoning problems. For A24, we use Hugging Face H4's dataset; Hochlehnert et al. discuss evaluation reproducibility (Hochlehnert et al., 2025). We use 30 problems from each of the first four datasets and all 50 problems from AM. The base models are DeepSeek-V4.1-Flash (DeepSeek Flash) (DeepSeek, 2026), DeepSeek-V4-Pro (DeepSeek Pro), Qwen3.6-35B-A3B (Qwen 35B) (Qwen, 2026), and GPT-5.6-Luna (GPT Luna). Each model maintains a separate skill library for each dataset.

For natural-language correction, Reflex uses ProgCo's basic reflection configuration (Song et al., 2025); Self-Refine iteratively revises answers using self-generated feedback (Madaan et al., 2023); Self-Reflection follows Reflexion in using self-reflective feedback to guide revision (Shinn et al., 2023); and CheckList guides revision through checklists (Cook et al., 2024). CoT-Tool independently generates multiple chain-of-thought paths and selects an answer by self-consistency voting (Wei et al., 2022; Wang et al., 2023). For programmatic verification, ProgCo checks answers using verification pseudoprograms and jointly revises answers and verification programs, while ProgCo-Py enables Python execution feedback within this procedure (Song et al., 2025).

We additionally include Dynamic Cheatsheet (Suzgun et al., 2026) and ACE (Zhang et al., 2026) as experience-accumulation methods, which maintain a continually updated memory and a playbook, respectively. In our experiments, Dynamic Cheatsheet uses a cumulative configuration without ground-truth feedback, whereas ACE updates its experience using ground-truth feedback. SC2SE learns skills from its own verification and correction, using ground-truth answers only for evaluation.

Test-time adaptation and evaluation. In the main experiments, SC2SE starts from an empty library $L^{(0)}$ and performs four skill-update rounds on each dataset. Skill learning and evaluation use the same fixed problem pool. After four rounds, the final library $L^{(4)}$ is fixed, and skill-assisted initial answers and their further-corrected counterparts are evaluated separately.

Evaluation procedure and metrics. Answer quality is measured consistently by accuracy (ACC) using a three-step scoring procedure. Each model's results are aggregated across the five datasets with weights proportional to problem counts. Mean in the main table is the equally weighted average of the four model ACCs; gains are reported in percentage points. $\mathrm{SC2SE}_{\mathrm{init}}$ denotes initial answers generated with the final skill library, and $\mathrm{SC2SE}_{\mathrm{corr}}$ denotes the same answers after further verification and any necessary correction. The main experiment compares final ACC under each method's prescribed procedure. Scoring, shared-initial-answer reuse, and runtime configurations are detailed in Appendix C. Each method uses the computational budget configured for its own procedure; SC2SE's skill-learning and correction costs are reported in Appendix E.

4.2 MAIN RESULTS

$\mathrm{SC2SE}_{\mathrm{corr}}$ outperforms all listed baselines on all four models (Table 1). Mean ACC reaches 64.6%, 14.3 percentage points above Self-Refine (50.3%), the highest-mean-ACC baseline. Gains span models with different base capabilities: DeepSeek Pro's 18.8-percentage-point gain over its best baseline indicates benefits even with weaker shared initial answers.

At the dataset level, Figure 3 shows that SC2SE exceeds the corresponding best baseline in 19 of the 20 model–dataset combinations, except Qwen 35B on A25. Gains span different competition datasets and extend to AM, where shared initial-answer accuracy is low, indicating broad dataset coverage of the overall improvement.

Without ground-truth answers as learning feedback, skill-assisted initial answers ($\mathrm{SC2SE}_{\mathrm{init}}$) reach 47.1% mean ACC, exceeding both RSI baselines, Dynamic Cheatsheet and ACE. Further correction adds 17.5 percentage points. This indicates complementary roles for experience reuse and current-

Table 1: ACC (%) across five mathematics datasets and four models. Subscripts show percentage-point changes from shared initial answers (Base): green for increases, red for decreases. Boldface marks the best final result. Complete results: Appendix B, Tables 3–6.

Method	DeepSeek Flash	DeepSeek Pro	Qwen 35B	GPT Luna	Mean
Base	52.9	38.8	53.5	23.5	42.2
Reflex	60.0 $_{+7.1}$	37.6 $_{-1.2}$	54.7 $_{+1.2}$	31.2 $_{+7.6}$	45.9 $_{+3.7}$
Self-Refine	61.8 $_{+8.8}$	51.2 $_{+12.4}$	57.6 $_{+4.1}$	30.6 $_{+7.1}$	50.3 $_{+8.1}$
Self-Reflection	55.3 $_{+2.4}$	40.0 $_{+1.2}$	57.1 $_{+3.5}$	32.9 $_{+9.4}$	46.3 $_{+4.1}$
CheckList	55.3 $_{+2.4}$	41.2 $_{+2.4}$	57.1 $_{+3.5}$	30.0 $_{+6.5}$	45.9 $_{+3.7}$
CoT-Tool	54.1 $_{+1.2}$	40.0 $_{+1.2}$	41.8 $_{-11.8}$	22.4 $_{-1.2}$	39.6 $_{-2.6}$
ProgCo	58.2 $_{+5.3}$	41.2 $_{+2.4}$	57.1 $_{+3.5}$	25.9 $_{+2.4}$	45.6 $_{+3.4}$
ProgCo-Py	62.4 $_{+9.4}$	48.2 $_{+9.4}$	58.8 $_{+5.3}$	28.2 $_{+4.7}$	49.4 $_{+7.2}$
Dynamic Cheatsheet	58.2 $_{+5.3}$	44.7 $_{+5.9}$	52.4 $_{-1.2}$	15.9 $_{-7.6}$	42.8 $_{+0.6}$
ACE	55.3 $_{+2.4}$	45.9 $_{+7.1}$	53.5 $_{0.0}$	21.8 $_{-1.8}$	44.1 $_{+1.9}$
$\text{SC2SE}_{\text{init}}$	60.0 $_{+7.1}$	44.1 $_{+5.3}$	57.6 $_{+4.1}$	26.5 $_{+2.9}$	47.1 $_{+4.9}$
$\text{SC2SE}_{\text{corr}}$	**74.7** $_{+21.8}$	**70.0** $_{+31.2}$	**65.3** $_{+11.8}$	**48.2** $_{+24.7}$	**64.6** $_{+22.4}$

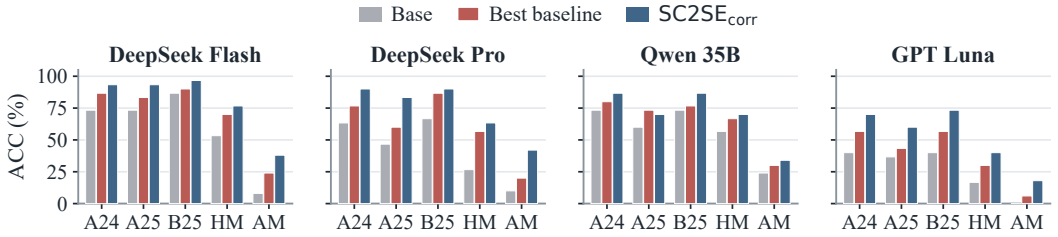

Figure 3: Per-dataset ACC for shared initial answers (Base), the best baseline, and $\text{SC2SE}_{\text{corr}}$. The best baseline has the highest ACC among compared baselines for the same model and dataset.

answer revision: accumulated skills improve initial answers, while verification and revision address their remaining problems.

SC2SE converts stepwise verification and revision evidence into problem-solving skills. Its initial-answer advantage may relate to this organization: after distillation and role calibration, diagnostic information may be easier for the model to use in subsequent solving. Dynamic Cheatsheet and ACE likewise note that experience reliability and models' ability to use it affect gains, while incorrect memories or low-quality reflections can introduce noise (Suzgun et al., 2026; Zhang et al., 2026).

4.3 Ablation Study

Table 2 uses an independently implemented SC2SE (w/o Skills) as a no-skill reference to compare the effects of skills on initial answers and corrected results separately. On Flash, Pro, and Qwen 35B, skill-assisted initial answers ($\text{SC2SE}_{\text{init}}$) outperform the shared initial answers (Base). After further correction, the full method ($\text{SC2SE}_{\text{corr}}$) still outperforms the no-skill reference. These two comparisons indicate that gains from skill accumulation extend from initial answering through subsequent correction, supporting the complementary roles of experience reuse and answer revision.

Table 2: Skill-component comparison. ACC (%).

Method	Flash	Pro	Qwen 35B
Base	52.9	38.8	53.5
$\text{SC2SE}_{\text{init}}$	**60.0**	**44.1**	**57.6**
SC2SE (w/o Skills)	72.4	66.5	64.1
$\text{SC2SE}_{\text{corr}}$	**74.7**	**70.0**	**65.3**

4.4 Skill Evolution and Experience Reuse

Further correction improves skill-assisted initial-answer ACC at every library version for all three models (Figure 4). For Flash, rapid early library construction gives way to fewer additions, while

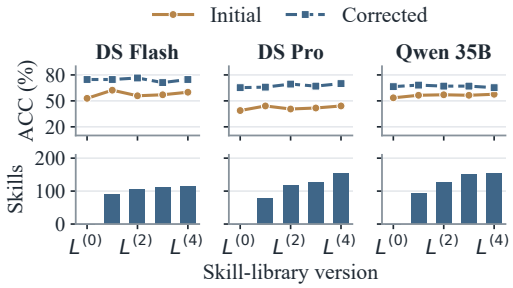

Figure 4: Skill-library evolution across three models: skill-assisted initial and corrected ACC (top), and skill counts (bottom). Per-dataset growth and update counts: Appendix D.

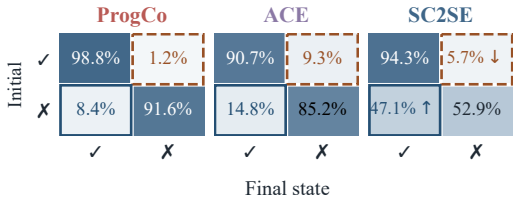

Figure 5: Transitions from shared initial answers (rows) to method outputs (columns); ✓/× denotes correct/incorrect. Each row is normalized to 100%. Blue/dashed orange outlines mark repair/regression rates; arrows indicate desired directions. All four models: Appendix Table 7.

skills continue supporting answering. Library size and ACC change at different rates, suggesting that skill-accumulation evaluation also requires attention to guidance content and actual use.

The following case illustrates correction experience becoming skills for subsequent initial answers.

Example: From Correction to Skill Reuse

Problem. After establishing $p = 17$, find the smallest positive integer m such that $m^4 + 1$ is divisible by 289. The initial answer, 134, is feasible but misses a smaller solution. Correction feedback identifies incomplete candidate-root enumeration, and the revised answer is 110.

From error to skill. This problem contributes to batch-level reflection, producing guidance to enumerate all roots modulo a prime before lifting them to higher prime-power moduli. The resulting skill passes the role check and is retained in $L^{(1)}$.

Skill reuse. With $L^{(1)}$, the model answers the problem again, completes the candidate enumeration and checks, and produces the correct initial answer, 110, before subsequent correction.

This case reuses the full library on the same problem, showing how error diagnoses become actionable checks in a subsequent initial answer. The complete case and role-calibration records appear in Appendix F.

4.5 Error Repair and Correct-Answer Preservation

To examine answer changes underlying ACC gains, we compare ProgCo as a representative programmatic correction method and ACE as an experience-accumulation representative selected by its mean ACC ranking within the RSI group. Pairing each method's final outputs with the same shared initial answers (Base) separates error repair from correct-answer preservation. Figure 5 pools per-problem paired results for Flash, Pro, and Qwen 35B.

SC2SE repairs 47.1% of shared-initial-answer errors while limiting regressions on initially correct answers to 5.7% (Figure 5). Its repair rate exceeds both comparators', while its regression rate is below ACE's. Accounting for initial correct/incorrect group sizes, repair gains substantially outweigh regression losses, indicating that overall improvement primarily comes from recovering previously failed problems. This links overall ACC gains to per-problem repairs and regressions; Appendix Table 10 reports within-method changes across the two stages.

5 Conclusion

SC2SE transforms stepwise mathematical correction experience into reusable skills through agent collaboration, experience distillation, and role calibration. With frozen model parameters, it forms a test-time self-evolution loop: correction → skill accumulation → skill-assisted answering → further correction. Experiments show higher mean accuracy for skill-assisted initial answers than the compared RSI baselines, and after further correction than all compared baselines. Further analysis supports the complementary roles of self-correction and skill accumulation. Future work will explore SC2SE in domains such as program reasoning and scientific problem solving, and investigate skill transfer and continual adaptation across tasks and environments.

Statement on AI Use

This research was led by the authors. Generative AI was used to support discussions of research design, implementation and analysis, and manuscript translation and editing. The authors are responsible for the research content and conclusions.

Reproducibility Statement

We provide code, recorded configurations, and question-level records for offline verification of the reported results in the supplementary materials. Experimental settings and supporting analyses are detailed in Appendices B–F.

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, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0679. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1d27c01ebd3e3aebe226b44fc970d803-Abstract-Datasets_and_Benchmarks_Track.html.

Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, Yong Mao, Ke Li, and Xing Sun. Training-free group relative policy optimization, 2025. URL https://arxiv.org/abs/2510.08191. arXiv preprint arXiv:2510.08191.

Justin Chen, Archiki Prasad, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. MAgICoRe: Multi-agent, iterative, coarse-to-fine refinement for reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32663–32686, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1660. URL https://aclanthology.org/2025.emnlp-main.1660/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168. arXiv preprint arXiv:2110.14168.

Jonathan Cook, Tim Rocktäschel, Jakob Foerster, Dennis Aumiller, and Alex Wang. TICKing all the boxes: Generated checklists improve LLM evaluation and generation. In *Language Gamification Workshop*, 2024. URL https://neurips.cc/virtual/2024/106582. NeurIPS 2024 workshop paper.

DeepSeek. Introducing DeepSeek-V4.1-Flash: Smarter, faster, more efficient. Official release announcement, 2026. URL https://www.deepseek.com/en/news/deepseek-v4-1-flash/. Accessed: 2026-09-17.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Haowen Gao, Haoran Chen, Can Wang, Shasha Guo, Liang Pang, Zhaoyang Liu, Huawei Shen, and Xueqi Cheng. SkillAudit: Ground-truth-free skill evolution via paired trajectory auditing, 2026a. URL https://arxiv.org/abs/2606.14239. arXiv preprint arXiv:2606.14239.

Jiaxun Gao, Him Wai (Michael) Ng, and Z. Jane Wang. Once-More: Continuous self-correction for large language models via perplexity-guided intervention. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0f402cddc6ab9e74c66e684d63651cf0-Abstract-Conference.html.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/d3cf1559a8795eb1ed2b3ad52409ac7d-Abstract-Conference.html.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=90UrTTxp5O.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Hugging Face H4. AIME 2024. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713. URL https://aclanthology.org/2024.tacl-1.78/.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/871ac99fdc5282d0301934d23945ebaa-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Junlin Liu, Shengnan An, Shuang Zhou, Dan Ma, Yehao Lin, Xinxuan Lv, Xuanlin Wang, Xiaoyu Li, Ziwen Wang, Xuezhi Cao, and Xunliang Cai. AMO-bench: Large language models still struggle in high school math competitions. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2120–2137, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.101. URL https://aclanthology.org/2026.findings-acl.101/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/980ea04d23d1f6908964eba2a74afe45-Abstract-Conference.html.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Qwen. Qwen3.6-35B-A3B-FP8. Official Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B-FP8. Accessed: 2026-09-17.

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*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Xiaoshuai Song, Yanan Wu, Weixun Wang, Jiaheng Liu, Wenbo Su, and Bo Zheng. ProgCo: Program helps self-correction of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 944–959, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-short.73. URL https://aclanthology.org/2025.acl-short.73/.

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, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.826. URL https://aclanthology.org/2024.findings-acl.826/.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275. arXiv preprint arXiv:2211.14275.

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*, 2024a. URL https://openreview.net/forum?id=ehfRiF0R3a.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Di Wu, Devendra Singh Sachan, Scott Yih, and Mingda Chen. Procedural knowledge at scale improves reasoning. In *Conference on Language Modeling*, 2026. URL https://colm.eventhosts.cc/Conferences/2026/AcceptedPapers.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Weijia Xu, Alessandro Sordoni, Chandan Singh, Zelalem Gero, Michel Galley, Xingdi Yuan, and Jianfeng Gao. Test-time learning with an evolving library, 2026. URL https://arxiv.org/abs/2605.14477. arXiv preprint arXiv:2605.14477.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3339f19c5fcee3ad74502947a32be9e6-Abstract-Conference.html.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 113519–113544, 2024b. doi: 10.52202/079017-3607. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/cde328b7bf6358f5ebb91fe9c539745e-Abstract-Conference.html.

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. arXiv preprint arXiv: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. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://doi.org/10.1038/s41586-025-08661-4.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=46Zgqo4QIU.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a94ff6f922d995d7d3f4ebf4143e442-Abstract-Conference.html.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59520–59558. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24at.html.

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. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. SELF-DISCOVER: Large language models self-compose reasoning structures. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126032–126058, 2024. doi: 10.52202/079017-4004. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e41efb03e20ca3c231940a3c6917ef6f-Abstract-Conference.html.

APPENDIX: SUPPORTING EVIDENCE AND REPRODUCIBILITY MATERIALS

Appendix A presents the pseudocode; B provides complete results, paired method comparisons, and cross-dataset transfer; C describes the data and configurations; D supplements the skill-evolution evidence; E reports costs; and F presents cases and prompts.

A SC2SE PSEUDOCODE

Algorithm 1 SC2SE: skill learning and final-library evaluation

Require: Problem pool $\mathcal{P}$; update rounds T; edit limits $\{b_i\}$; correction budget; optional shared initial answers
Ensure: Final library $L^{(T)}$ and paired initial and corrected outputs

Run per model–dataset pair with frozen model parameters.

$L^{(0)} \leftarrow \varnothing$; initialize role-approval and library-evaluation caches
Solve $\mathcal{P}$ with $L^{(0)}$, verify once, and cache the results
for $i = 0, \ldots, T-1$ **do**
Stage 1: correction and experience acquisition
  for each $x \in \mathcal{P}$ **do**
    Set (y_0, a_0) to the shared answer if $i = 0$ and available, or to $\mathcal{A}_{\mathrm{sol}}(x; L^{(i)})$
    $\tau_x^{(i)} \leftarrow \mathcal{A}_{\mathrm{cor}}^{\mathrm{corr}}(x, y_0, a_0)$ ▷ *stop: pass, exhausted budget, or execution error*
    Check claims in order with tools as needed; revise using feedback at the first inconsistency.
    Rebuild the plan if a changes; otherwise refine it. Retain the latest solution at termination.
  Extract $D^{(i)}$ from trajectories without verification execution errors
  Fix labels by each initial answer's first self-verification; raise an error if $D^{(i)} = \varnothing$
Stage 2: skill evolution
  $E \leftarrow \mathcal{A}_{\mathrm{evo}}^{\mathrm{reflect}}(D^{(i)}; L^{(i)})$ ▷ *separate success/failure minibatches*
  Compress supporting solutions at the context threshold; preserve key evidence verbatim.
  $E_{\mathrm{ref}}^{(i)} \leftarrow \mathcal{A}_{\mathrm{cal}}(E; L^{(i)})$ ▷ *itemwise solver-role checks*
  Reuse matching approvals; repair/recheck unsuitable skills, preserving mathematical content.
  Drop failed skills and empty add/replace proposals; structurally check deletions.
  if $E_{\mathrm{ref}}^{(i)} = \varnothing$ **then** save $L^{(i+1)} \leftarrow L^{(i)}$ and continue
  $E_{\mathrm{merge}}^{(i)} \leftarrow \mathcal{A}_{\mathrm{cal}}\left(\mathcal{A}_{\mathrm{evo}}^{\mathrm{merge}}(E_{\mathrm{ref}}^{(i)}; L^{(i)}); L^{(i)}\right)$ ▷ *same checks after aggregation*
  On technical errors in either calibration call, save $L^{(i+1)} \leftarrow L^{(i)}$ and continue.
  $E_{\mathrm{sel}}^{(i)} \leftarrow \mathcal{A}_{\mathrm{evo}}^{\mathrm{rank}}(E_{\mathrm{merge}}^{(i)}; L^{(i)}, b_i)$ ▷ *at most b_i proposals*
  Keep all if within the limit; use input order if ranking retries fail.
  $L^{(i)\prime} \leftarrow \mathrm{Apply}(L^{(i)}, E_{\mathrm{sel}}^{(i)})$ ▷ *validate edit types and targets*
  Keep the current library if no edit applies.
Stage 3: skill application
  if $L^{(i)\prime}$ has no cached evaluation **then**
    for each $x \in \mathcal{P}$ **do**
      $(y_0, a_0) \leftarrow \mathcal{A}_{\mathrm{sol}}(x; L^{(i)\prime})$; record $\mathcal{A}_{\mathrm{cor}}^{\mathrm{ver}}(x, y_0, a_0)$ ▷ *verify once; no revision*
    Cache the results
  $L^{(i+1)} \leftarrow L^{(i)\prime}$; save the library and round records ▷ *accept the valid candidate*
Final-library evaluation
Fix $L^{(T)}$
for each $x \in \mathcal{P}$ **do**
  $(y_{\mathrm{init}}, a_{\mathrm{init}}) \leftarrow \mathcal{A}_{\mathrm{sol}}(x; L^{(T)})$
  $(y_{\mathrm{corr}}, a_{\mathrm{corr}}) \leftarrow \mathrm{Final}(\mathcal{A}_{\mathrm{cor}}^{\mathrm{corr}}(x, y_{\mathrm{init}}, a_{\mathrm{init}}))$ ▷ *same correction procedure*
  Save both outputs ▷ *identical if unrevised*
return $L^{(T)}$ and all paired outputs

B Complete Main Results and Paired Method Comparisons

This section provides the complete results for four models and five datasets, corresponding to the main results table and gain comparison figure. Each model is evaluated on 170 problems. Avg is weighted by problem count, and its colored subscript indicates the ACC change relative to that model's Base, in percentage points. Boldface marks the best value among the baselines and full SC2SE results, including ties.

Table 3: DeepSeek-V4.1-Flash: per-dataset ACC (%).

Method	A24	A25	B25	HM	AM	Avg
Base	73.3	73.3	86.7	53.3	8.0	52.9
Reflex	83.3	80.0	90.0	60.0	16.0	60.0 $_{+7.1}$
Self-Refine	86.7	83.3	83.3	70.0	16.0	61.8 $_{+8.8}$
Self-Reflection	80.0	70.0	90.0	60.0	8.0	55.3 $_{+2.4}$
CheckList	76.7	76.7	86.7	53.3	12.0	55.3 $_{+2.4}$
CoT-Tool	83.3	73.3	76.7	56.7	10.0	54.1 $_{+1.2}$
ProgCo	86.7	73.3	86.7	60.0	14.0	58.2 $_{+5.3}$
ProgCo-Py	83.3	76.7	90.0	63.3	24.0	62.4 $_{+9.4}$
Dynamic Cheatsheet	83.3	83.3	80.0	63.3	12.0	58.2 $_{+5.3}$
ACE	80.0	76.7	86.7	53.3	10.0	55.3 $_{+2.4}$
$\text{SC2SE}_{\text{init}}$	86.7	76.7	90.0	56.7	18.0	60.0 $_{+7.1}$
$\text{SC2SE}_{\text{corr}}$	**93.3**	**93.3**	**96.7**	**76.7**	**38.0**	**74.7** $_{+21.8}$

Table 4: DeepSeek-V4-Pro: per-dataset ACC (%).

Method	A24	A25	B25	HM	AM	Avg
Base	63.3	46.7	66.7	26.7	10.0	38.8
Reflex	63.3	43.3	63.3	30.0	8.0	37.6 $_{-1.2}$
Self-Refine	70.0	60.0	86.7	56.7	10.0	51.2 $_{+12.4}$
Self-Reflection	76.7	46.7	66.7	26.7	6.0	40.0 $_{+1.2}$
CheckList	66.7	46.7	66.7	33.3	12.0	41.2 $_{+2.4}$
CoT-Tool	66.7	46.7	66.7	30.0	10.0	40.0 $_{+1.2}$
ProgCo	70.0	50.0	70.0	26.7	10.0	41.2 $_{+2.4}$
ProgCo-Py	70.0	56.7	83.3	43.3	12.0	48.2 $_{+9.4}$
Dynamic Cheatsheet	66.7	60.0	73.3	36.7	10.0	44.7 $_{+5.9}$
ACE	63.3	50.0	73.3	40.0	20.0	45.9 $_{+7.1}$
$\text{SC2SE}_{\text{init}}$	66.7	50.0	66.7	43.3	14.0	44.1 $_{+5.3}$
$\text{SC2SE}_{\text{corr}}$	**90.0**	**83.3**	**90.0**	**63.3**	**42.0**	**70.0** $_{+31.2}$

Table 5: Qwen3.6-35B-A3B: per-dataset ACC (%).

Method	A24	A25	B25	HM	AM	Avg
Base	73.3	60.0	73.3	56.7	24.0	53.5
Reflex	70.0	60.0	73.3	66.7	24.0	54.7 +1.2
Self-Refine	80.0	66.7	73.3	63.3	26.0	57.6 +4.1
Self-Reflection	73.3	60.0	76.7	63.3	30.0	57.1 +3.5
CheckList	80.0	63.3	73.3	63.3	26.0	57.1 +3.5
CoT-Tool	60.0	56.7	56.7	43.3	12.0	41.8 -11.8
ProgCo	80.0	70.0	73.3	60.0	24.0	57.1 +3.5
ProgCo-Py	80.0	**73.3**	76.7	60.0	26.0	58.8 +5.3
Dynamic Cheatsheet	73.3	70.0	76.7	43.3	20.0	52.4 -1.2
ACE	76.7	70.0	70.0	53.3	20.0	53.5 +0.0
$\text{SC2SE}_{\text{init}}$	86.7	53.3	86.7	60.0	24.0	57.6 +4.1
$\text{SC2SE}_{\text{corr}}$	**86.7**	70.0	**86.7**	**70.0**	**34.0**	**65.3** +11.8

Table 6: GPT-5.6-Luna: per-dataset ACC (%).

Method	A24	A25	B25	HM	AM	Avg
Base	40.0	36.7	40.0	16.7	0.0	23.5
Reflex	56.7	40.0	56.7	20.0	2.0	31.2 +7.6
Self-Refine	53.3	43.3	46.7	23.3	4.0	30.6 +7.1
Self-Reflection	56.7	40.0	53.3	30.0	4.0	32.9 +9.4
CheckList	53.3	40.0	43.3	23.3	6.0	30.0 +6.5
CoT-Tool	50.0	26.7	33.3	16.7	0.0	22.4 -1.2
ProgCo	53.3	36.7	43.3	13.3	0.0	25.9 +2.4
ProgCo-Py	46.7	40.0	46.7	23.3	2.0	28.2 +4.7
Dynamic Cheatsheet	33.3	13.3	33.3	10.0	0.0	15.9 -7.6
ACE	33.3	40.0	36.7	13.3	0.0	21.8 -1.8
$\text{SC2SE}_{\text{init}}$	36.7	50.0	40.0	16.7	4.0	26.5 +2.9
$\text{SC2SE}_{\text{corr}}$	**70.0**	**60.0**	**73.3**	**40.0**	**18.0**	**48.2** +24.7

B.1 Per-Model Transition Rates for Method Comparisons

Table 7 provides results for all four models corresponding to Figure 5; SC2SE uses the corrected outputs from the final skill library.

SC2SE achieves the highest error-repair rate on all four models, indicating that the repair gains observed in the main text extend across base models. Correct-answer preservation varies by model: SC2SE performs best on Pro, while ProgCo performs best on the other three. These metrics reveal different sources of benefit, with SC2SE's main advantage being the recovery of more previously incorrect answers.

B.2 Cross-Dataset Skill Transfer

We evaluate skill transfer with DeepSeek Flash in a leave-one-dataset-out setting. Each of the five mathematics datasets serves as the target in turn. For each target, we pool the independently learned final libraries $L^{(4)}$ from the other four datasets and use DeepSeek Flash to jointly select 30 skills, preserving their original wording. Selection uses only source-skill content and prioritizes actionability, coverage, and complementarity. The construction strategy was selected based on exploratory evaluation results.

All five target-specific libraries are fixed before target answering. Each library guides initial answers using SC2SE's solver prompt, with model parameters and the library held fixed throughout evaluation. We use the same three-step scoring procedure as in the main experiments; Base reuses the saved public initial answers. Generation uses non-thinking mode, temperature 0, and an 8,192-token

Table 7: Per-model conditional transition rates (%) from paired method comparisons. ✓/× denotes correct/incorrect. The first two columns use initially correct answers as the denominator, and the last two use initially incorrect answers; each pair sums to 100%. Boldface marks the best value in each column within a model, including ties. Higher correct-answer preservation and repair rates are better (↑), while lower regression and non-repair rates are better (↓).

Model	Method	✓→✓ ↑	✓→× ↓	×→✓ ↑	×→× ↓
DeepSeek Flash	ProgCo	**100.0%**	**0.0%**	11.2%	88.8%
	ACE	92.2%	7.8%	13.8%	86.2%
	SC2SE	95.6%	4.4%	**51.2%**	**48.8%**
DeepSeek Pro	ProgCo	95.5%	4.5%	6.7%	93.3%
	ACE	89.4%	10.6%	18.3%	81.7%
	SC2SE	**97.0%**	**3.0%**	**52.9%**	**47.1%**
Qwen 35B	ProgCo	**100.0%**	**0.0%**	7.6%	92.4%
	ACE	90.1%	9.9%	11.4%	88.6%
	SC2SE	91.2%	8.8%	**35.4%**	**64.6%**
GPT Luna	ProgCo	**97.5%**	**2.5%**	3.8%	96.2%
	ACE	80.0%	20.0%	3.8%	96.2%
	SC2SE	87.5%	12.5%	**36.2%**	**63.8%**

output limit. The rendered skill guidance is limited to 8,000 tokens. Source libraries, selected skills, prompts, and question-level outputs and scores are included in the supplementary materials.

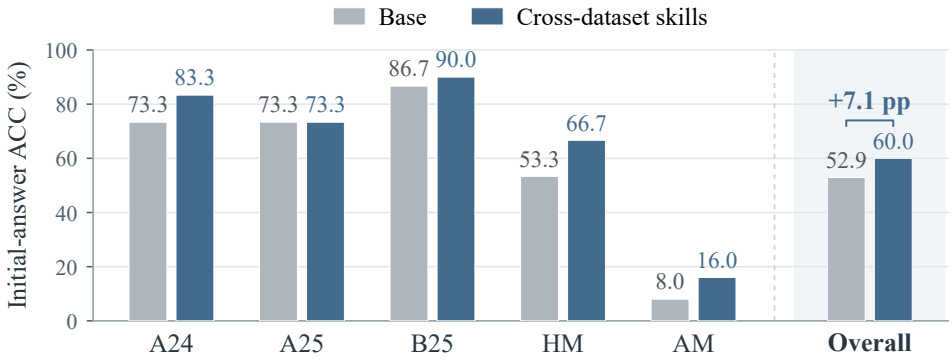

Figure 6: Leave-one-dataset-out initial-answer ACC (%) on DeepSeek Flash. Overall denotes pooled accuracy across the five target datasets; pp denotes percentage points.

Cross-dataset skill-assisted initial answers reach 60.0% ACC, compared with 52.9% for Base (Figure 6), a gain of 7.1 percentage points. Accuracy improves on four datasets and is unchanged on the fifth. These results support the reuse of learned skills across mathematics datasets.

C Data, Evaluation Protocol, and Experimental Configuration

C.1 Data, Models, and Shared Initial Answers

Problem sources are provided in the experimental setup in the main text and the corresponding dataset references. Each model maintains a separate skill library for each of the five datasets, with 170 problems in total; the problems and denominators are fixed across methods. Learning and evaluation in the main experiments use the same fixed problem pool, so those results reflect same-pool test-time adaptation.

The full names of the four base models are given in Section 4.1. Base denotes the shared initial answers for each model. The first round of SC2SE and the baselines that reuse shared initial answers

Table 8: Datasets and fixed evaluation sizes.

Short name	Dataset	Questions
A24	AIME 2024	30
A25	AIME 2025	30
B25	BRUMO 2025	30
HM	HMMT February 2025	30
AM	AMO-Bench	50

use the same complete answers and scores for that model. Skill-assisted initial answers are generated anew using the final skill library. The current implementation inserts the titles and solution guidance of all skills into the system prompt in library order, with the problem supplied separately.

ACC is computed using a three-step scoring procedure: exact matching, followed by numerical comparison, and, if the result remains unresolved, answer-equivalence judgment by a scoring model. For Qwen 35B, the shared initial answers and baselines retain their verified scores, whereas the second SC2SE batch uses the three-step scores from that batch; archived scores are used for the other models. A model's ACC is the total number of correct answers across the five datasets divided by 170, and Mean in the main table is the equally weighted average across the four models. Means and gains are computed from unrounded correct-answer counts and then reported to one decimal place.

C.2 Key Experimental Settings

Fair comparison primarily relies on consistent base models, problem pools, shared initial answers, and ACC evaluation criteria. The four agents share the current base model and differ in their prompts and responsibilities. The table below lists the key settings needed to interpret SC2SE's results; engineering details such as timeouts, retries, and concurrency are provided with the reproduction code and run records.

Table 9: Key experimental settings for SC2SE.

Configuration	Setting
Generation	Non-thinking mode; temperature 0 for solving and correction
Output budget	Up to 8,192 tokens for skill-related calls; shared initial answers, memory updates, and dynamic compression retain their respective budgets
Random seed	42 for skill updates and trajectory shuffling
Learning and evaluation	Four update rounds starting from an empty library; evaluation with the final library $L^{(4)}$ fixed
Correction rounds	The round parameter is set to 3 for both SC2SE and the correction baselines
Feedback	SC2SE learns from its own verification and correction; ground-truth answers are used only to report ACC

Each method retains its own procedure: CoT-Tool votes over three reasoning paths; ACE updates experience using ground-truth feedback; and Dynamic Cheatsheet's memory-update budget can extend to 16,384 tokens. Skill-assisted initial and corrected results are reported separately. Call and computation budgets follow each method's procedure, and SC2SE's additional learning costs are reported in Appendix E.

C.3 Runtime Environment and Reproducibility Materials

Flash, Pro, and Luna are accessed through their respective model APIs; SC2SE's Qwen 35B runs locally in FP8 on an NVIDIA A800. Both locally deployed and API-based main experiments are run from hosts with AMD EPYC 7413 CPUs, with an allocation of 32 vCPUs and a memory limit of 256 GiB.

All Qwen 35B initial-answer, corrected-answer, and process statistics use the second experiment conducted on 2026-09-22, with final results obtained from skill library $L^{(4)}$. Answers, skills, scores,

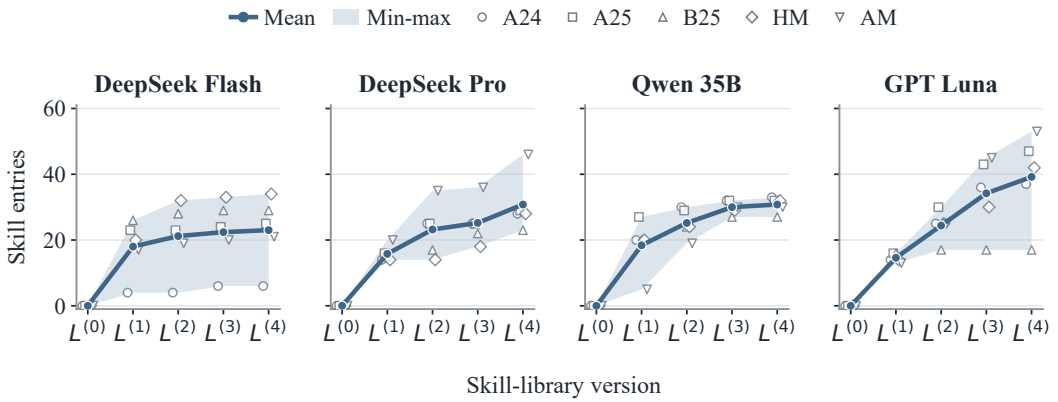

Figure 7: Skill growth for the four models. The blue line is the equally weighted mean number of skill entries across the five datasets, and the shaded area spans their minimum and maximum counts. Gray markers show the original values for each dataset, with small horizontal offsets to prevent overlap; all panels share the same vertical scale.

and configurations are saved by batch, and state transitions are computed by pairing per-problem records using problem IDs. Code and configurations are provided with the reproducibility materials.

D Skill Growth and Updates

This section provides further details on how skill libraries are constructed and maintained, and on performance changes across the two answering stages with the final library.

D.1 Skill Growth and Differences across Datasets

All four models continue accumulating experience after constructing their libraries in the first round, but library sizes gradually diverge across datasets within each model. This indicates that skill updates follow different accumulation paths depending on the task. Together with the answer-performance curves in the main text, these results allow further examination of how the accumulated experience is used in solving problems.

D.2 Content Maintenance and Changes in Answer Correctness

Table 10 shows that maintenance after initial library construction still consists mainly of adding entries, with few removals; the number of subsequent additions varies substantially across models. For answer correctness, skill-assisted initial answers yield net gains over shared initial answers for every model, and further correction yields additional gains, indicating that skill guidance and verification and revision of the current answer play complementary roles. Entry counts describe library maintenance, whereas per-problem changes describe answer outcomes; the two should be interpreted jointly rather than interchangeably.

E Computational Cost and Long-Trajectory Compression

Using DeepSeek Flash and DeepSeek Pro as examples, this section reports the time required for skill learning and the effects of context compression on long trajectories, showing the time breakdown by phase and changes in request length before and after compression.

Table 10: Skill updates and changes in answer correctness for the four models. (a) Initial construction and subsequent updates of skill entries; (b) changes across the two answering stages with the final skill library.

(a) Skill-entry updates (counts)

Model	Initial additions $L^{(0)} \to L^{(1)}$	Later additions $L^{(1)} \to L^{(4)}$	Later removals $L^{(1)} \to L^{(4)}$
DeepSeek Flash	90	29	4
DeepSeek Pro	79	80	5
Qwen 35B	92	66	4
GPT Luna	73	127	4

(b) Two-stage changes with the final skill library

	Base → Skill initial			Skill initial → Corrected		
Model	Repair (%)	Regression (%)	Net (pp)	Repair (%)	Regression (%)	Net (pp)
DeepSeek Flash	12.4	5.3	+7.1	14.7	0.0	+14.7
DeepSeek Pro	12.4	7.1	+5.3	28.2	2.4	+25.9
Qwen 35B	11.8	7.6	+4.1	7.6	0.0	+7.6
GPT Luna	10.6	7.6	+2.9	21.8	0.0	+21.8

Later additions and removals are summed over three consecutive updates; the same entry may be counted more than once, and rewriting may count as one addition and one removal. Repair/Regression denote incorrect-to-correct/correct-to-incorrect transitions as percentages of all 170 problems per model; Net is their difference in percentage points (pp), computed before rounding. Entry updates and changes in answer correctness do not imply a one-to-one causal relationship.

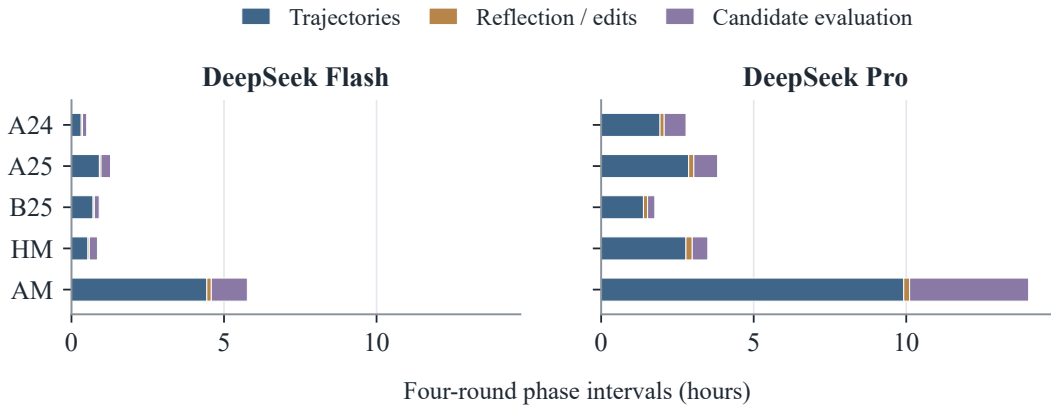

Figure 8: Time spent in each phase of four-round learning for DeepSeek Flash and Pro (hours). Times are recorded in their respective runtime environments, and a shared horizontal axis shows the phase breakdown. Datasets are run in parallel, and times are reported individually.

E.1 Learning Time and Phase Breakdown

Most time is spent generating learning trajectories and evaluating candidate libraries, with longer phase times on AM. This indicates that model calls for solving and verification are the main sources of learning cost. The phase plot covers four learning rounds, while Table 11 separately reports the combined time for skill learning and final-library evaluation.

Table 11: Recorded time for learning and final-library evaluation (minutes).

Dataset	DeepSeek Flash	DeepSeek Pro
A24	40.2	185.3
A25	85.1	329.9
B25	66.6	118.3
HM	61.6	266.4
AM	407.4	937.2

The table summarizes the time for skill learning and final-library evaluation from each dataset's run records, in minutes.

E.2 Effects of Request-Context Compression

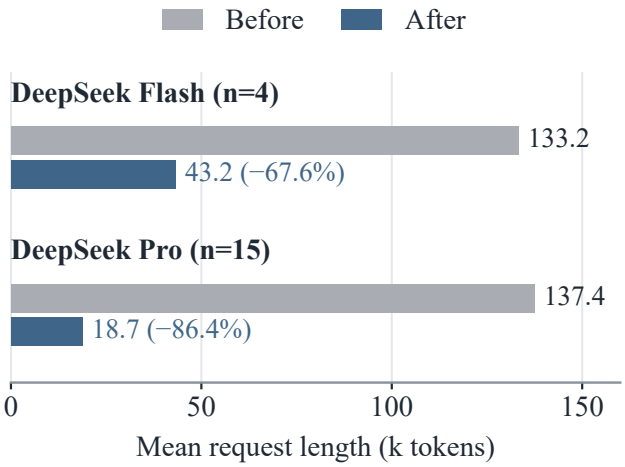

Figure 9: Context compression for the two DeepSeek models. Each group shows mean request lengths before and after compression over all recorded operations (thousands of tokens), with n denoting the number of operations. The overall context reduction ratio is shown on the right; the measured quantity is request length.

Figure 9 directly compares request lengths at the point when compression is triggered with their post-compression lengths. Each group aggregates all recorded operations for that model, using mean lengths to show changes in context load. The reduction ratio is calculated as the total reduction divided by the total pre-compression length.

All 4 operations for Flash and 15 for Pro are included, including one with a near-zero reduction. For both models, mean request length is substantially shorter after compression, indicating that selective compression can reduce the context occupied by long trajectories and provide more compact inputs for subsequent reflection on experience.

Records of individual compression operations are provided with the reproducibility materials.

F Case Evidence and Agent Prompts

This section supplements the case in the main text showing how correction experience becomes usable skills. We first present role-calibration execution records, then provide the evidence chain for correction, experience transformation, and skill reuse, and finally show the prompts used in the implementation to connect behavioral explanations to the actual interfaces.

F.1 Role-Calibration Execution Records

Table 12: Role-calibration call records (event counts).

Model	Fresh checks	Reused checks	Repair attempts
DeepSeek Flash	531	162	1
DeepSeek Pro	293	235	0
Qwen 35B	294	239	1
GPT Luna	512	58	0

Role calibration constrains the intended user of a skill: correction may refer to existing errors and feedback, whereas problem-solving skills must transform this experience into guidance that a model can use from the problem conditions alone. The table counts calibration events before and after aggregation; skills with identical content that have already passed the check can reuse the check result.

F.2 Evidence Chain from Correction to Reuse

Full Case: From Missing Modular Roots to Skill Reuse

This case uses the second Qwen 35B experiment on A24 (2026-09-22), with Problem 13 from the 2024 AIME I.

Original problem and initial answer. Let p be the smallest prime such that there exists a positive integer n for which $n^4 + 1$ is divisible by p^2. Find the smallest positive integer m such that $m^4 + 1$ is divisible by p^2.

With the empty skill library $L^{(0)}$, the model reuses the shared initial answer, which identifies $p = 17$ but does not enumerate all candidate roots and gives a final answer of 134. This value satisfies the divisibility condition modulo 289 but is not the required smallest positive integer, so the answer is scored as incorrect.

Verification and revision. Verification feedback identifies an incomplete candidate set and calls for re-enumerating the roots modulo 17 and lifting each root to find the global minimum. Following this feedback, the revised response changes the final answer to 110 and subsequently passes both re-verification and independent answer scoring.

Skill formation and addition to the library. This problem and other failed trajectories in the same batch are jointly used for reflection, producing the skill **Lifting All Roots in Hensel's Lemma**. Its core guidance is:

> When solving $f(x) \equiv 0 \pmod{p^k}$ by lifting solutions from $\pmod{p^{k-1}}$, you must first find ALL distinct roots of $f(x) \equiv 0 \pmod{p}$.

This skill distills the specific error into a general problem-solving procedure: enumerate all roots before lifting. It passes the role check, reuses the existing check result after aggregation, and is retained verbatim in $L^{(1)}$, which contains 20 skills.

Subsequent reuse. In the second answering round, this skill is included in the solver model's system prompt as part of $L^{(1)}$. The model re-enumerates and checks the candidate roots, obtaining 110, 134, 155, and 179, and answers with the minimum, 110, before further correction. The answer passes independent scoring.

This case illustrates experience reuse from detecting omitted candidates, to forming general enumeration guidance, to producing a correct initial answer in a later round.

This case presents key steps in reusing the full library on the same problem; the outcome is not attributed to any individual skill. The original trajectories still contain intermediate calculation errors; correctness here refers to the final answer. Complete outputs are provided in the reproducibility materials.

F.3 Key Prompts for the Four Agents

The following excerpts present key prompts from the archived implementation for the four roles: solving, stepwise correction, skill evolution, and role calibration. The input and output descriptions summarize each role's interface, and the English instructions are excerpted from the corresponding templates. Square brackets indicate skill content inserted at runtime. When a role involves multiple internal calls, the stage associated with each excerpt is identified. Complete templates are retained in the code.

Solver Agent

Input: The current problem and skill library. Skill content is inserted into the system prompt, and the problem is supplied separately.

You are a mathematical reasoning agent.

[Current skill library]

Solve the given mathematics problem carefully.

Use the skill document only as reusable problem-solving experience.

Do not treat the skill document as a source of the current problem's answer.

Do not copy problem-specific numbers, conclusions, or final answers from the skill unless you derive them again for the current problem.

Give a clear reasoning process and state the final answer clearly.

Output: A complete mathematical solution and an explicit final answer.

Stepwise Correction Agent

Input: The problem, current solution, and extracted answer; the revision stage additionally receives verification feedback.

Verification-stage excerpt:

Given a problem, a current solution response, and its extracted final answer, generate a forward mathematical verification flow for that response.

The flow should inspect the current solution response, identify the key mathematical claims, and verify those claims from the original problem conditions. It should also verify that the extracted final answer matches the complete target object required by the problem.

Revision-stage excerpt:

If the feedback is correct, revise the solution.

If the solution is correct and the feedback is caused by a weak verification flow, keep the solution and state why.

When the feedback identifies an incorrect intermediate claim, do not only patch that single line. Re-derive all downstream steps that depend on that claim and make sure the final answer follows from the corrected reasoning.

Output: A verification flow and feedback, together with a complete revised solution when needed.

Skill Evolution Agent

Input: A batch of solving and correction trajectories and the existing skill library. The excerpt below illustrates the reflection stage for failed trajectories.

For each trajectory, distinguish the Initial Response from all downstream Forward verification, feedback, tool execution, and revision records.

Use the problem, Initial Response, Forward evidence, and Revised Response, when available, to diagnose the first substantive mathematical failure in the Initial Response.

Convert each selected pattern into concise, actionable guidance that the initial mathematical solver can apply while solving a new problem.

Skills must generalize beyond the observed problems. Do not copy task-specific values, answers, names, or derivations.

Output: Structured skill-edit proposals, with a title and solution guidance for each skill. After aggregation, proposals are selected and executed within the allowed number of proposals per round.

Role Calibration Agent

Input: Candidate skills and target-role requirements.

Target-role excerpt:

The sole user of the candidate Skill is a mathematics problem solver.

The analysis used to generate a candidate Skill may refer to Forward verification, feedback, and revision records, but the final Skill must transform that information into guidance that a mathematics problem solver can use independently, without mentioning or depending on such subsequent information.

Repair-principle excerpt:

Modify only content necessary for role cleanup. Preserve already role-consistent text as literally as possible. Do not polish, summarize, expand, add examples, or standardize style.

Output: Role-check decisions; when repair is needed, either replacement skills or a decision not to submit that repair. Calibration focuses on roles and information dependencies while preserving the original mathematical content and applicability conditions.