GROUP-RELATIVE SKILL OPTIMIZATION FOR FROZEN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing external natural-language skills has emerged as an effective approach to adapting frozen large language models. Existing methods have improved task performance and stabilized skill updates through held-out validation and incremental editing. However, we identify two limitations of batch-level skill optimization: (1) common-denominator bias, where reflection over trajectories from different questions favors generic guidance over question-specific knowledge; and (2) all-or-nothing acceptance, where edits from different questions are accepted or rejected as a single update, making it difficult to retain useful edits while rejecting harmful ones. We introduce Group-Relative Skill Optimization (GRSO), which aligns reflection and verification at the question level. For each question, GRSO samples a group of trajectories and uses within-question reflection to propose itemized edits, contrasting successful and failed attempts. Each edit bundle is verified on its source question before accepted bundles are merged. Dynamic sampling and retry concentrate additional computation on difficult questions. This design preserves question-specific learning signals without held-out validation during optimization. Across SearchQA, DocVQA, and LiveMathBench, GRSO achieves the best or tied-best accuracy in all six target–benchmark settings. Compared with the unadapted models, it improves average accuracy across the three benchmarks by 15.2 and 22.0 percentage points for Qwen3.5-4B and Seed1.8, respectively.

1 INTRODUCTION

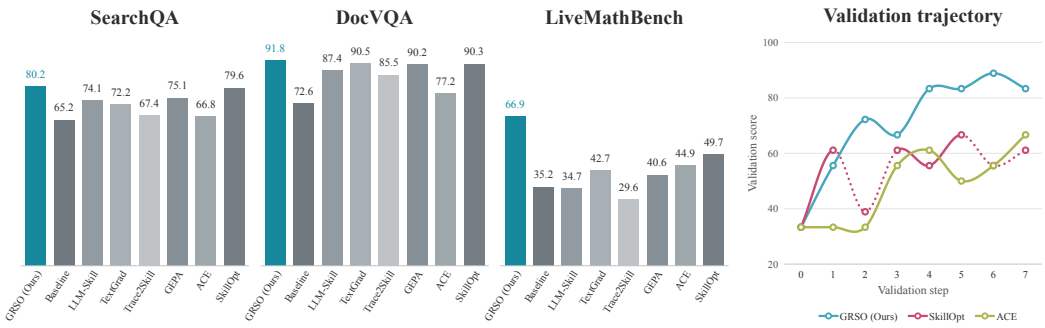

Figure 1: **Performance results.** Test performance across three benchmarks and validation trajectories on LiveMathBench. GRSO does not use validation data during optimization, and its validation scores are computed only for post hoc analysis. For validation-gated baselines, dashed segments denote candidate updates that are rejected and rolled back.

Adapting large language models to new tasks increasingly involves external *skills* (Zhou et al., 2026): natural-language instructions read at inference time that guide domain conventions, reasoning procedures, and answer formats (Li et al., 2026; Jiang et al., 2026b; Wang et al., 2025). Such skills also support tool use and interaction in LLM-based agents (Yao et al., 2023; Yang et al., 2024b; Wang et al., 2023a). Many capable models have weights that are inaccessible or impractical to finetune,

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.

so a growing family of methods instead treats this text as a *trainable* object, improved automatically from the model's own execution experience while its weights stay untouched (Zhang et al., 2024; Yang et al., 2026b; Ouyang et al., 2026; Jiang et al., 2026a).

Skill optimization retains advantages that weight updates cannot offer. The learning signal is legible: the optimizer LLM reflects on execution traces and writes a *textual gradient*, a natural-language account of what failed and how the skill should change (Pryzant et al., 2023; Mi et al., 2026). The trained skill is an inspectable, portable artifact (Chen et al., 2024; Fang et al., 2026; Ma et al., 2026), and adaptation completes in an order of magnitude fewer rollouts than reinforcement learning (Agrawal et al., 2025). Editing text has pitfalls of its own: a misguided update can send the skill off course, results may depend on the order in which lessons arrive (Ni et al., 2026), and repeated rewriting can erode accumulated detail (Zhang et al., 2025). Existing methods counter these pitfalls with dedicated machinery, including held-out validation gates (Alzubi et al., 2026; Yang et al., 2026a), itemized incremental updates (Zhang et al., 2025), and population-based selection (Guo et al., 2024; Fernando et al., 2024). With such safeguards in place, trained skills deliver consistent gains and can match or surpass weight-based adaptation at a fraction of its rollouts (Agrawal et al., 2025).

Yet beneath these safeguards, two limitations persist. First, *common-denominator bias*: handed trajectories from *different* questions (Yuksekgonul et al., 2024; Agrawal et al., 2025; Yang et al., 2026a), the optimizer distills lessons that skew toward common denominators such as output format, phrasing conventions, and generic caution. Such lessons repeat step after step: quick early gains from surface fixes, then a plateau short of what the model could reach, with question-specific reasoning failures rarely touched. Second, *all-or-nothing acceptance*: the whole update must improve the held-out score (Yang et al., 2026a). The check demands a large split yet is re-run at every step. Because the edits of an entire batch are scored as one, its single pass/fail cannot tell which question's edits did the work.

To address these limitations, we introduce GRSO (Group-Relative Skill Optimization), a skill optimization system that scopes every learning signal to a single question: every edit is distilled from a single question, and each question's edits are accepted or rejected on that question alone. GRSO realizes this fine-grained signal through two mechanisms. (1) *Group sampling with contrastive reflection*: following the spirit of group-relative policy optimization (Shao et al., 2024), each step samples a batch of training questions and, for every question, rolls out a group of attempts under the current skill. The optimizer then contrasts the group's successful and failed trajectories to propose skill edits. (2) *Per-question acceptance*: the edits distilled from a question are merged into the step-start skill to form a trial, which is verified on that question alone. Edits are kept when the trial meets the success criterion defined in Section 3.2. Failed trials may trigger revision using the recorded failure.

We evaluate GRSO on three benchmarks: SearchQA (Dunn et al., 2017) for retrieval QA, DocVQA (Mathew et al., 2021) for document understanding, and LiveMathBench (Liu et al., 2024) for mathematical reasoning. We use two target models spanning small open-weight and frontier-scale regimes: Qwen3.5-4B and Seed1.8. Our experiments yield two main observations. (1) GRSO consistently outperforms all baselines with Seed1.8 as the target. As shown by the bar charts in Figure 1, under a matched evaluation harness and identical data splits, GRSO achieves the best accuracy on all three benchmarks. Without modifying any target-model parameters, it improves average accuracy over the base model by 22.0 percentage points. (2) GRSO uses only task-native environment signals to verify updates and maintains stable skill optimization without a held-out validation gate. The right panel of Figure 1 shows that its optimization trajectory still trends upward overall even when the score on the full validation split does not improve at individual steps. This indicates that per-question acceptance prevents local regressions from accumulating and sending the skill off course.

2 Background and Motivation

2.1 Skill Optimization

Skill optimization improves a frozen language model by constructing or revising the natural language text it reads. The current state of the art builds on natural-language feedback (Cheng et al., 2024; Agarwal et al., 2024; Yan et al., 2025): an optimizer model inspects execution traces and writes feedback on how the skill should change (Liu et al., 2026). Reflexion (Shinn et al., 2023)

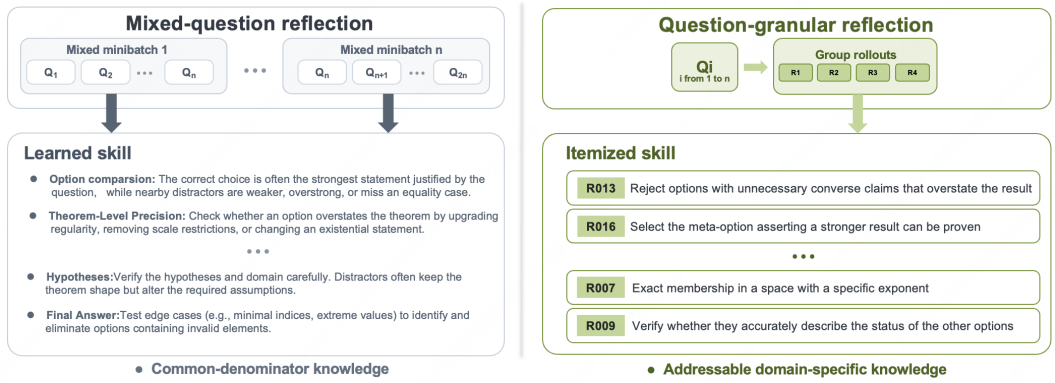

Figure 2: **How the reflection unit shapes learned skill content.** Mixed-pool reflection aggregates evidence across questions and favors common-denominator heuristics (left), whereas question-level reflection preserves the general layer while making domain-specific rules individually addressable (right).

reflects on failed attempts to revise agent behavior. Dynamic Cheatsheet (Suzgun et al., 2025) accumulates lessons in an external test-time memory. TextGrad (Yuksekgonul et al., 2024) treats feedback as gradients, aggregated over small batches and gated by a validation check. GEPA (Agrawal et al., 2025) evolves prompts by reflective mutation and selects candidates on a validation pool. ACE (Zhang et al., 2025) maintains an itemized playbook updated by incremental deltas. Trace2Skill (Ni et al., 2026) distills per-trajectory patches and consolidates them into one skill. SKILLOPT (Yang et al., 2026a) runs a full training loop with batched reflection and a strict per-step held-out gate, and is the strongest prior method in our evaluation. The optimized text is called a prompt, a playbook, or a skill in these works. We refer to it uniformly as the skill. A parallel line (Zhou et al., 2023; Yang et al., 2024a; Khattab et al., 2023; Opsahl-Ong et al., 2024; Wan et al., 2024) searches prompts under scalar scores rather than natural-language feedback.

2.2 COMMON-DENOMINATOR BIAS

TextGrad aggregates feedback over mixed batches, GEPA mutates prompts from pooled minibatch rollouts, and SkillOpt reflects over batches mixing failures and successes (Yuksekgonul et al., 2024; Agrawal et al., 2025; Yang et al., 2026a). In each case, the optimizer reflects over a pool of different questions. Asked for lessons that hold across such a pool, it is pushed toward what the questions have in common. This is rarely more than surface conventions of format and phrasing, and the tendency is the common-denominator bias. A similar convergence toward near-identical generic instructions has also been documented in prompt optimization (Gao et al., 2025). In our SKILLOPT reproduction, the skill trained under mixed-question reflection consists almost entirely of generic meta-strategies, with no rule tied to the domain knowledge that individual questions require (Figure 2). Figure 1 illustrates this pattern for SKILLOPT on LiveMathBench: the first step captures broadly shared corrections and yields a large validation gain, but subsequent mixed-question reflections struggle to improve beyond these common patterns, so the skill is updated only at steps 1 and 5.

Escaping the bias calls for a learning signal that reaches question-specific knowledge. However, extracting a reliable training signal from a single question is not trivial. Trace2Skill (Ni et al., 2026) and ACE (Zhang et al., 2025) both reflect on individual trajectories, but a lone trajectory provides no reference for telling a systematic error from an accident, and when the optimizer and the target are the same model, reflection on one's own failed trace has little beyond the attempt itself to draw on. To address this limitation, we construct a reference within each question through contrastive reflection over multiple trajectories.

2.3 THE VALIDATION DILEMMA

Held-out validation gates help screen out harmful updates before they enter the skill (Agrawal et al., 2025; Yang et al., 2026a). We observe that existing gates face two difficulties.

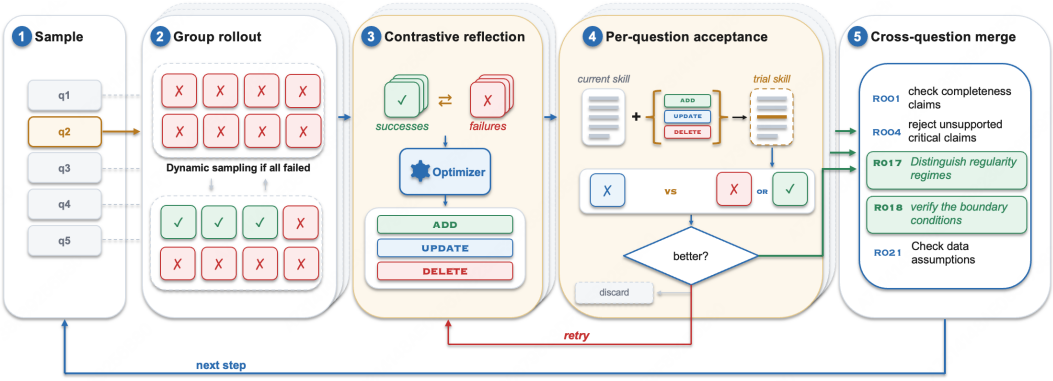

Figure 3: **The GRSO training pipeline.** Each question forms an independent rollout group. Its proposed edits are checked against the step-start skill before accepted edits are merged.

First, a gate is reliable only when the validation split is large enough for its pass/fail signal to rise above sampling noise. Consequently, at each training step, the rollouts required for validation can exceed those used to generate the training traces themselves.

Second, the edits distilled from an entire batch are merged and judged as one update, resulting in all-or-nothing acceptance. The edit bundles submitted to a held-out validation gate mix useful and harmful edits. As Section 4.3 shows, this mixture leads to low retained-update rates under coarse validation gates. Even when an entire bundle passes, harmful edits may be accepted alongside useful ones.

Both difficulties trace to the same mismatch: the unit of verification, a full split and a whole bundle, is far coarser than the unit of learning.

3 METHOD

We propose GRSO (Group-Relative Skill Optimization), a skill optimization framework that brings group-relative learning (Shao et al., 2024) into textual gradients. GRSO takes the individual question as its unit of learning and obtains question-specific update signals by contrasting multiple reasoning trajectories from the same question. The framework comprises five components. *Sample* selects the questions for the current update. *Group rollout* uses the frozen target model to generate diverse reasoning trajectories for each question (Wei et al., 2022; Wang et al., 2023b). During *contrastive reflection*, the optimizer model reflects on each rollout group to propose skill edits, contrasting successes and failures when both are available. *Per-question acceptance* verifies the proposed edits on their source question. *Cross-question merge* consolidates the accepted edits across questions in the same batch.

As illustrated in Figure 3, each training step samples a batch of questions from the training split. Each sampled question forms an independent minibatch, with group rollout, contrastive reflection, and per-question acceptance performed against the same step-start skill. For acceptance, each question's edit bundle is applied independently to the step-start skill to form a trial skill, which is evaluated on the source question. Accepted bundles are merged only after all question-level checks are complete, forming the skill for the next step. This shared starting point allows the question-level computations to run in parallel.

Training runs for a fixed update budget, and the delivered skill is the one standing after the final step.

3.1 GROUP SAMPLING WITH CONTRASTIVE REFLECTION

Each minibatch corresponds to one question. The target model samples a group of G trajectories under the current skill. A task-specific evaluator assigns each trajectory a strict score and, when supported by the task, an additional soft score. The strict score indicates exact correctness, whereas the soft score is an environment-specific measure of partial correctness, such as token-level F1 or ANLS.

When the group contains both successes and failures, the optimizer model contrasts them, asking what the successful attempts did that the failed ones lacked, and writes the resulting improvements as candidate edits for that question.

Reflection is not limited to mixed groups. When every trajectory succeeds, the group is still reflected on: although a scalar advantage vanishes under uniform outcomes, textual reflection can still extract the practices shared by the successful attempts and write them as candidate edits. All-failure groups receive additional treatment because reflection over failures alone tends to drift toward generic conclusions. GRSO therefore invokes *dynamic sampling*, drawing one additional group of G trajectories and increasing the available pool to $2G$ in search of a success. If all $2G$ attempts still fail and a soft score is available, GRSO forms a pseudo-contrast group of size G by selecting equal numbers of the highest- and lowest-scoring attempts, keeping reflection anchored in a better-versus-worse comparison. If the task does not provide a soft score, GRSO instead randomly selects G trajectories from the enlarged all-failure pool.

3.2 Per-Question Acceptance

GRSO replaces full-split validation with a lightweight check on the question that produced an edit. To make question-level updates independently addressable, the skill is represented as an itemized list following ACE (Zhang et al., 2025): each edit adds, updates, or deletes one item, and all edits proposed for a question form one atomic bundle. The bundle is applied to the step-start skill to form a trial skill. Both the step-start and trial skills are run once on the source question under greedy decoding. The acceptance gate admits the bundle only when the trial skill corrects a failure of the step-start skill or preserves an already successful outcome. If the trial turns a success into a failure, the edits from that question are discarded and do not enter the cross-question merge stage.

To concentrate additional optimization on questions that a single edit cannot correct, GRSO introduces a retry strategy. When both the step-start skill and the initial trial fail, GRSO returns the rejected bundle and its trial outcome to the optimizer. Using the same rollout group and the recorded failure, the optimizer revises the bundle and submits it for another check. GRSO allows at most two retries and discards the bundle if the revised trials still fail.

Because the step-start and trial skills differ only by one question's edit bundle and are evaluated on the same source question under greedy decoding, the verdict isolates whether that bundle improves its target, yielding a fine-grained, question-level improvement signal. To prevent locally accepted edits from accumulating as question-specific answers, the cross-question merge stage semantically consolidates edits from different questions, removes duplicates, and resolves conflicts. Entries useful only to a single source question are absorbed into broader rules or removed during consolidation.

4 Experiments

4.1 Setup

Benchmarks. We evaluate on three benchmarks that cover complementary question-answering tasks: retrieval-augmented QA (Lewis et al., 2020) on SearchQA (Dunn et al., 2017) (400/200/1,400 train/validation/test questions), document-image understanding on DocVQA (Mathew et al., 2021) (107/53/374), competition-level mathematical reasoning on LiveMathBench (Liu et al., 2024) (35/18/124). Every method is optimized on the training split and evaluated on the held-out test split with the same target model and task interface. We report strict accuracy. Appendix D defines the strict and soft scores for each task. ACE, GEPA, and SkillOpt use the validation splits for their method-specific validation or selection procedures, whereas GRSO does not use validation data. For the main results, we conduct three independent training runs per trainable method and evaluate each resulting skill once at temperature 0, reporting the mean of the three test scores.

Models. We use two target–optimizer pairs matching the two deployment regimes of Section 1: a small open-weight target, Qwen3.5-4B (Qwen Team, 2026), optimized by Seed1.8, and a frontier-scale pair with Seed1.8 serving as both target and optimizer.

Table 1: Main results with Qwen3.5-4B as the target and Seed1.8 as the optimizer. Scores average three test results, one per independent training run, and Aggregate averages the three benchmarks. Small green/red subscripts show the absolute change from No Skill, computed before rounding. Best results are bold and second-best results are underlined.

Method	SearchQA	DocVQA	LiveMathBench	Aggregate
No Skill	69.2	87.4	25.5	60.7
LLM-Skill	70.3 +1.1	88.8 +1.3	27.2 +1.6	62.1 +1.3
TextGrad	70.5 +1.3	90.4 +2.9	27.7 +2.2	62.9 +2.1
Trace2Skill	67.4 −1.8	<u>90.6</u> +3.1	30.6 +5.1	62.9 +2.1
GEPA	71.0 +1.8	89.2 +1.8	32.8 +7.3	64.3 +3.6
ACE	68.8 −0.4	87.5 +0.1	31.5 +5.9	62.6 +1.9
SKILLOPT	<u>73.9</u> +4.6	**91.4** +4.0	<u>38.2</u> +12.6	<u>67.8</u> +7.1
GRSO (ours)	**75.3** +6.1	**91.4** +4.0	**61.0** +35.5	**75.9** +15.2

Table 2: Main results with Seed1.8 as both target and optimizer. Scores average three test results, one per independent training run. Aggregate averages the three benchmarks. Colored subscripts follow the convention above. Best results are bold and second-best results are underlined.

Method	SearchQA	DocVQA	LiveMathBench	Aggregate
No Skill	65.2	72.6	35.2	57.7
LLM-Skill	74.1 +8.8	87.4 +14.8	34.7 −0.5	65.4 +7.7
TextGrad	72.2 +7.0	<u>90.5</u> +17.8	42.7 +7.5	68.5 +10.8
Trace2Skill	67.4 +2.1	85.5 +12.8	29.6 −5.6	60.8 +3.1
GEPA	75.1 +9.9	90.2 +17.6	40.6 +5.4	68.6 +10.9
ACE	66.8 +1.6	77.2 +4.5	44.9 +9.7	63.0 +5.3
SKILLOPT	<u>79.6</u> +14.4	90.3 +17.6	<u>49.7</u> +14.5	<u>73.2</u> +15.5
GRSO (ours)	**80.2** +15.0	**91.8** +19.2	**66.9** +31.7	**79.7** +22.0

Baselines. We compare against seven baselines, ranging from no adaptation and one-shot skill generation to iterative prompt and skill optimization: no-skill (the frozen target with the benchmark's default prompt; the Baseline row in tables), LLM-Skill (a skill written by the optimizer model in one shot from the task description and never updated), TextGrad (Yuksekgonul et al., 2024) (gradient-style natural-language prompt optimization), GEPA (Agrawal et al., 2025) (Pareto reflective prompt evolution), ACE (Zhang et al., 2025) (incremental playbook optimization), Trace2Skill (Ni et al., 2026) (trajectory-level skill distillation), and SKILLOPT (Yang et al., 2026a) (batched skill training with a per-step held-out gate), the strongest prior method. All baselines run from their released implementations under the shared target model and splits.

GRSO configuration. Each step samples five questions and draws a group of $G{=}8$ trajectories per question at temperature 0.8, yielding 40 base rollouts and five per-question reflections. When all eight attempts fail, dynamic sampling draws one additional group of eight. If all 16 attempts still fail, the method constructs a group of eight as described in Section 3.1. During acceptance, the step-start and trial skills are evaluated on the source question using greedy decoding. Rejected bundles may be revised and checked again, with at most two retries. Based on the size and difficulty of each dataset, we use 40, 12, and 7 optimization steps for SearchQA, DocVQA, and LiveMathBench, respectively. The skill obtained after the final step is used for test evaluation.

4.2 MAIN RESULTS

Tables 1 and 2 report the main comparison. GRSO achieves the best or tied-best accuracy in all six target–benchmark settings and the highest aggregate score with both target models. Appendix E shows the three individual results for each method and benchmark. Its largest gains occur on LiveMathBench: compared with No Skill, accuracy improves by 35.5 points with Qwen and 31.7 points with Seed1.8. Compared with SKILLOPT, the corresponding gains are 22.8 and 17.2 points. On SearchQA, GRSO improves over No Skill by 6.1 and 15.0 points with the Qwen and Seed1.8 targets, respectively. On DocVQA, the corresponding improvements are 4.0 and 19.2 points.

Table 3: End-to-end training cost with Seed1.8 as both target and optimizer. We report update steps, mean target-model rollouts per step, and total input and output tokens over the full training run.

Benchmark	Method	Steps	Rollouts/step	Input tok.	Output tok.
SearchQA	GRSO	40	52.7	11.80M	2.37M
SearchQA	SKILLOPT	40	240	31.26M	5.93M
DocVQA	GRSO	12	53.3	2.83M	0.38M
DocVQA	SKILLOPT	12	93	6.16M	0.53M
LiveMathBench	GRSO	7	60.9	3.53M	2.65M
LiveMathBench	SKILLOPT	7	53	3.27M	2.82M

Seed1.8's lower no-skill score on DocVQA largely reflects a mismatch with the strict answer format. Its responses frequently contain the correct answer while also including explanations, units, or additional context, causing them to fail exact matching. In a no-skill diagnostic, 94.9% of Seed1.8 predictions contain the gold answer, compared with 92.8% for Qwen (Appendix B). After optimization with GRSO, the strict-score gap between the two targets narrows to only 0.4 points (91.4 for Qwen vs. 91.8 for Seed1.8), consistent with the learned skill helping Seed1.8 follow the task-specific answer format.

GRSO's largest gains occur on LiveMathBench, whose competition-level mathematical reasoning requires detailed, problem-specific knowledge.

4.3 COST AND ACCEPTANCE EFFICIENCY

For a step-matched comparison, GRSO samples $G = 8$ trajectories for each of $B = 5$ questions, using $BG = 40$ base rollouts per step. SKILLOPT evaluates 40 distinct training questions per step on SearchQA and DocVQA, and all 35 training questions on LiveMathBench. Their rollout costs also differ in verification. GRSO verifies edits locally on their source questions, requiring one step-start evaluation, one initial trial, and at most two retry trials per question. This adds at most $4B$ verification rollouts per step. Dynamic sampling contributes an additional $G\bar{A}$ rollouts, where $\bar{A}$ is the observed mean number of groups per step that trigger additional sampling. In contrast, SKILLOPT evaluates each candidate update by sweeping the entire held-out validation split.

Accordingly, their per-step target-model rollout costs are

$$C_{\text{GRSO}} = BG + G\bar{A} + \bar{V}, \qquad C_{\text{SKILLOPT}} = \min(40, |\mathcal{D}_{\text{tr}}|) + |\mathcal{D}_{\text{val}}|,$$

where $\bar{V} \leq 4B$ is the observed mean number of local verification rollouts per step, including step-start evaluations and retry trials. GRSO does not use a validation split during optimization.

As shown in Table 3, the resulting rollout reduction depends on the size of the validation split. On SearchQA, SKILLOPT requires 240 rollouts per step, compared with 52.7 for GRSO, a $4.6\times$ reduction. On DocVQA, the corresponding costs are 93 and 53.3 rollouts per step, yielding a $1.7\times$ reduction. On LiveMathBench, however, SKILLOPT uses 35 training rollouts plus 18 validation rollouts, totaling 53 per step, compared with 60.9 for GRSO. The smaller validation split and training batch therefore remove GRSO's rollout-cost advantage on this benchmark. Beyond rollout cost, GRSO improves update efficiency through question-level verification. Edit bundles within the same batch are evaluated independently, and only accepted bundles enter the cross-question merge. Rejecting one bundle therefore does not block useful edits from other questions. A step leaves the skill unchanged only when none of its proposed bundles is accepted. We measure update efficiency by the fraction of training steps or iterations that produce a retained update. With Seed1.8 as both target and optimizer, GRSO updates its skill in 98.3% of training steps across the three benchmarks, whereas only 14.3% of SKILLOPT's step-level candidates and 20.8% of GEPA's iteration-level candidates pass their respective validation gates.

4.4 ABLATION

Table 4 reports ablation studies on LiveMathBench with both target–optimizer pairs, examining the role each independent component plays in GRSO's optimization. We examine three components.

Table 4: Component ablations on LiveMathBench. Acc is single-run test accuracy; Mixed and Fail are the fractions of mixed and all-failure rollout groups; Mixed succ. is the mean success fraction within mixed groups; Rollouts is the mean number of sampling rollouts per group; and Accept is the fraction of proposed edit bundles eventually accepted, including after retry. Learned-skill rows use the skill obtained after one epoch.

Configuration	Acc	Mixed	Fail	Mixed succ.	Rollouts	Accept
Target: Qwen3.5-4B, optimizer: Seed1.8						
no-skill	25.8	–	–	–	–	–
w/o per-question acceptance	31.5	77.1%	20.0%	0.308	10.74	–
w/o retry	41.1	62.9%	34.3%	0.412	11.89	26.5%
w/o dynamic sampling	56.5	71.4%	22.9%	0.490	8.00	64.5%
GRSO (full)	61.3	77.1%	17.1%	0.442	10.29	69.7%
Target and optimizer: Seed1.8						
no-skill	35.5	–	–	–	–	–
w/o per-question acceptance	46.8	60.0%	22.9%	0.348	10.51	–
w/o retry	51.6	51.4%	28.6%	0.382	10.97	44.4%
w/o dynamic sampling	59.7	37.1%	34.3%	0.500	8.00	60.0%
GRSO (full)	67.7	54.3%	17.1%	0.467	9.37	52.0%

Dynamic sampling targets all-failure groups by drawing one additional batch of eight trajectories. Without it, the all-failure share rises from 17.1% to 34.3% and the mixed share falls from 54.3% to 37.1% for the Seed1.8 target, reducing accuracy by 8.0 points. The Qwen target shows the same pattern: the all-failure share rises from 17.1% to 22.9%, the mixed share falls from 77.1% to 71.4%, and accuracy drops by 4.8 points. Notably, the ablated variant has a higher mixed-group success rate than the full method (0.500 vs. 0.467 on Seed1.8 and 0.490 vs. 0.442 on Qwen). This is expected because dynamic sampling recovers difficult, initially all-failure groups and moves them into the mixed category, often with only a few successful trajectories.

Retry targets questions that a single edit does not correct. Even with dynamic sampling retained, removing retry raises the all-failure share from 17.1% to 28.6% for Seed1.8 and from 17.1% to 34.3% for Qwen, while reducing accuracy from 67.7 to 51.6 and from 61.3 to 41.1, respectively. Additional sampling alone is therefore insufficient to resolve these difficult groups. In the full runs, 15.4% of accepted bundles for the Seed1.8 target and 39.1% for Qwen3.5-4B are admitted only after retry, showing that a single update attempt misses useful edits. By revising failed edits, retry reduces the share of all-failure groups and improves final accuracy.

Without *per-question acceptance*, every proposed edit is merged without verification, allowing useful edits to coexist with low-quality and sometimes conflicting ones. Consequently, the target model produces inconsistent outcomes across rollouts of the same question. In both settings, this instability manifests as the highest or tied-highest mixed-group share and the lowest mixed-group success rate, accompanied by substantial accuracy drops: from 67.7 to 46.8 for the Seed1.8 target and from 61.3 to 31.5 for Qwen.

4.5 Skill Analysis

Cross-model skill transfer. We transfer final skills between Qwen3.5-4B and Seed1.8 without further optimization. Both directions improve over No Skill on all three benchmarks, but underperform skills optimized for the recipient itself. This suggests that transferable skills still reflect the error patterns of the target used during optimization. Appendix A provides the results and analysis.

Update anatomy. Appendix C illustrates one question-level update on LiveMathBench and the subsequent merge of accepted bundles from the same step, showing how edit proposals are checked on their source questions and then merged into the stored skill.

5 CONCLUSION

We introduced GRSO, a question-granular framework for optimizing external skills while keeping the target model frozen. Within-question contrastive reflection preserves question-specific learning signals, while source-question verification checks each edit bundle independently before accepted updates are merged. Across three benchmarks and two target models, GRSO matches or exceeds the evaluated baselines without held-out validation during optimization. These results suggest that aligning reflection and verification at the question level is an effective way to turn execution experience into reusable skill knowledge.

6 LIMITATIONS

GRSO is most beneficial when individual training questions expose fine-grained, transferable domain knowledge that mixed-question reflection would dilute. Tasks mainly requiring high-level strategies, formatting conventions, or knowledge already present in the model may gain little from question-level contrast, limiting the value of additional group rollouts. The method also relies on a capable optimizer to extract meaningful differences between successful and failed trajectories, formulate edits, and revise rejected bundles from sparse feedback. Otherwise, learned skills may become superficial, redundant, or conflicting. Acceptance checks can reject harmful edits but cannot supply knowledge the optimizer fails to infer. Learned skills also reflect the error patterns of the target used during optimization. Although transferable, they may not fully address failures specific to another recipient, limiting their effectiveness relative to skills optimized for that recipient. Generalization to long-horizon interactive tasks remains untested.

AI USE STATEMENT

This manuscript was written entirely by the authors, with LLMs used only for grammar and wording refinement.

REFERENCES

Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. PromptWizard: Task-aware prompt optimization framework. *arXiv preprint arXiv:2405.18369*, 2024. URL https://arxiv.org/abs/2405.18369.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

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. URL https://arxiv.org/abs/2603.02766.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0019.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next AutoDiff: Generative optimization with rich feedback, execution traces, and LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/83ba7056bce2c3c3c27e17397cf3e1f0-Abstract-Conference.html.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. SearchQA: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544, 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Shuzheng Gao, Chaozheng Wang, Cuiyun Gao, Xiaoqian Jiao, Chun Yong Chong, Shan Gao, and Michael Lyu. The prompt alchemist: Automated LLM-tailored prompt optimization for test case generation. *arXiv preprint arXiv:2501.01329*, 2025.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZG3RaNIsO8.

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. URL https://arxiv.org/abs/2603.12056.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. SoK: Agentic skills: Beyond tool use in LLM agents. *arXiv preprint arXiv:2602.20867*, 2026b.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2005.11401.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. Are your LLMs capable of stable reasoning? *arXiv preprint arXiv:2412.13147*, 2024.

Xingyan Liu, Xiyue Luo, Linyu Li, Ganghong Huang, Jianfeng Liu, and Honglin Qiao. SkillForge: Forging domain-specific, self-evolving agent skills in cloud technical support. *arXiv preprint arXiv:2604.08618*, 2026. URL https://arxiv.org/abs/2604.08618.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. SkillGen: Verified inference-time agent skill synthesis. *arXiv preprint arXiv:2605.10999*, 2026. URL https://arxiv.org/abs/2605.10999.

Minesh Mathew, Dimosthenis Karatzas, and C V Jawahar. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-Pro: Learning reusable skills from experience via non-parametric PPO for LLM agents. *arXiv preprint arXiv:2602.01869*, 2026. URL https://arxiv.org/abs/2602.01869.

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.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

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

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, 2023. doi: 10.18653/v1/2023.emnlp-main.494.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y K Li, Y Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*, 2025.

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, and Sercan Ö. Arık. Teach better or show smarter? on instructions and exemplars in automatic prompt optimization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 58174–58244, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6b031defd145b02bed031093d8797bb3-Abstract-Conference.html.

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. *arXiv preprint arXiv:2305.16291*, 2023a. URL https://arxiv.org/abs/2305.16291.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023b. URL https://arxiv.org/abs/2203.11171.

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, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. 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. URL https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Cilin Yan, Jingyun Wang, Lin Zhang, Ruihui Zhao, Xiaopu Wu, Kai Xiong, Qingsong Liu, Guoliang Kang, and Yangyang Kang. Efficient and accurate prompt optimization: the benefit of memory in exemplar-guided reflection. In *Proceedings of the 63rd Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pp. 753–779, 2025. doi: 10.18653/v1/2025.acl-long.37.

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.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024b.

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*, 2026a.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. AutoSkill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026b. URL https://arxiv.org/abs/2603.01145.

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.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-Pro: Learning to evolve via policy-level reflection and optimization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5348–5375, 2024. doi: 10.18653/v1/2024.acl-long.292. URL https://aclanthology.org/2024.acl-long.292/.

Yingli Zhou, Wang Shu, Yaodong Su, Wenchuan Du, Yixiang Fang, and Xuemin Lin. A comprehensive survey on agent skills: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2605.07358*, 2026. URL https://arxiv.org/abs/2605.07358.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=92gvk82DE-.

A Cross-model Skill Transfer

We examine whether the learned skills remain useful when the target model changes. For each benchmark, we transfer the final skill optimized for Qwen3.5-4B to Seed1.8, and the final skill optimized for Seed1.8 to Qwen3.5-4B, without further skill optimization. Each recipient model is evaluated with and without the transferred skill under the same test protocol. Table 5 reports both transfer directions. The skill's training target identifies the model whose trajectories and feedback were used during optimization. Seed1.8 serves as the optimizer in both source settings.

Table 5: Cross-model transfer of benchmark-specific GRSO skills. Each skill is transferred unchanged to the other target model without further optimization. The skill column identifies the source training target. Seed1.8 is the optimizer in both source settings.

Recipient	Skill	SearchQA	DocVQA	LiveMathBench
Seed1.8	No Skill	65.2	72.6	35.2
Seed1.8	GRSO (trained for Qwen3.5-4B)	77.1	78.3	59.7
Qwen3.5-4B	No Skill	69.2	87.4	25.5
Qwen3.5-4B	GRSO (trained for Seed1.8)	71.6	90.6	49.2

The No Skill rows use the corresponding baselines from the main tables under matching evaluation configurations. The two directions test transfer between training targets with a shared optimizer. They do not establish independence from the optimizer.

Transferred skills improve performance in both directions, but remain below skills optimized for the recipient model itself. On DocVQA, for example, Seed1.8 reaches 78.3 with Qwen-trained skills, compared with 91.8 with its own skills. This gap is consistent with differences in the errors encountered by the two targets. Appendix B shows that Seed1.8 frequently includes additional content that causes exact-match failures. Such errors are less prominent in Qwen's responses, providing fewer opportunities to learn the corresponding corrections. Learned skills therefore convey useful experience across models, while their coverage remains shaped by the target used during optimization.

B DocVQA Answer Analysis

To examine the role of answer formulation in the DocVQA performance gap, we compare predictions from the two target models on the same 374 test questions. Table 6 reports exact match (EM) and a diagnostic indicating whether an answer contains a gold answer. Seed1.8 has lower EM than Qwen3.5-4B in this evaluation (72.2% vs. 89.8%), although its gold-containment rate is higher (94.9% vs. 92.8%). Among Seed1.8's 104 EM errors, 85 (81.7%) contain a gold answer, compared with 11 of 38 (28.9%) for Qwen3.5-4B.

Table 6: Single-run diagnostics on 374 DocVQA test questions. Gold containment is an auxiliary diagnostic, not a replacement for EM.

Target	EM (%)	Contains gold (%)	Contains gold among EM errors
Qwen3.5-4B	89.8	92.8	11/38 (28.9%)
Seed1.8	72.2	94.9	85/104 (81.7%)

Individual examples illustrate how additional context leads to EM failures. For a reference answer of "Washington university," Seed1.8 responds "Washington University (St. Louis, Missouri)," while Qwen3.5-4B responds "Washington University." For a reference answer of "5," Seed1.8 produces "5 (including cover sheet)," while Qwen3.5-4B returns "5." These examples and the containment statistics indicate that answer formulation contributes to Seed1.8's lower EM: many responses include the requested answer alongside additional explanation or context. Containment alone does not establish semantic correctness for every case.

C UPDATE ANATOMY

Figure 4 traces the second minibatch of step 2 on LiveMathBench. Its reflection group contains two correct and six incorrect trajectories. Contrastive reflection proposes edits concerning critical parameter boundaries and endpoint regularity. After applying the effective edits, the source-question outcome changes from incorrect to correct. The lower part shows the subsequent merge of four accepted bundles from the same step: three bundles contribute complementary updates to R001, while the retained additions are stored as separate rules. Rule IDs for additions are assigned when the merged edits are applied to the skill.

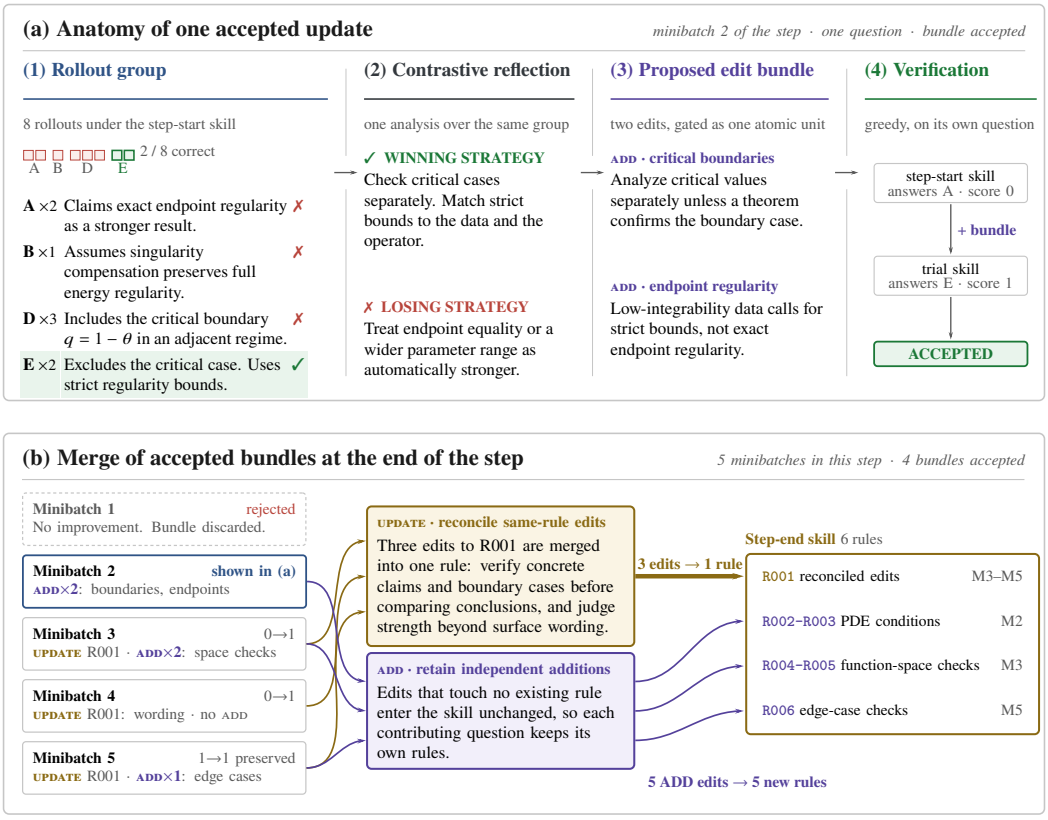

Figure 4: One minibatch update and the subsequent step-level merge. The upper row expands a single question's rollout evidence, reflection, proposed edits, and source-question verification. The lower row shows selected edits from four accepted bundles, their consolidation, and the stored skill. Text is condensed and paraphrased for readability. Only the two effective additions of the detailed minibatch are shown. A third proposed addition was filtered before the trial. Acceptance outcomes precede the cross-question merge.

D TASK-SPECIFIC SCORING

The main results use strict accuracy. The task-specific soft scores provide additional feedback during optimization and are distinct from the reported strict metric.

SearchQA. Answers are normalized by lowercasing, removing punctuation and the articles "a," "an," and "the," and collapsing whitespace. The strict score is exact match against any normalized gold answer. The soft score is token-level F1, taking the maximum over gold answers. The evaluator also records a bidirectional substring-match diagnostic (`sub_em`): either the normalized prediction contains a gold answer or vice versa. This diagnostic does not contribute to the strict or soft score.

DocVQA. Normalization lowercases the answer and collapses whitespace, preserving punctuation and articles. For a prediction p and a gold answer g, let $d(p, g) = \mathrm{ED}(p, g)/\max(|p|, |g|)$ denote normalized Levenshtein distance. Their similarity is $1 - d(p, g)$ when $d(p, g) < 0.5$, and zero otherwise. The soft score is ANLS, obtained by taking the maximum similarity over gold answers. The strict score is $\mathbf{1}[\mathrm{ANLS} \geq 0.999]$. Thus, the main tables report strict accuracy, rather than mean ANLS. This normalization is stricter than SearchQA's with respect to punctuation and articles.

LiveMathBench. The evaluated questions are multiple choice. The strict score is one if the predicted option equals the gold option, and zero otherwise. For interface compatibility, the `em`, `f1`, and `sub_em` fields all contain this same binary value. The `f1` field therefore provides no additional partial-correctness signal on this task.

E Results Across Independent Runs

Figure 5 shows the three test results underlying Tables 1 and 2. Each trained method is optimized independently three times, and each resulting skill is evaluated once. No Skill undergoes three test evaluations without training. Scores are computed from the correct-answer count and test-set size for each run, then averaged before rounding. The variation shown includes overlapping results on some tasks: with the Qwen target, GRSO and SkillOpt have identical mean DocVQA accuracy, whereas GRSO's LiveMathBench results exceed SkillOpt's in all three runs for both targets.

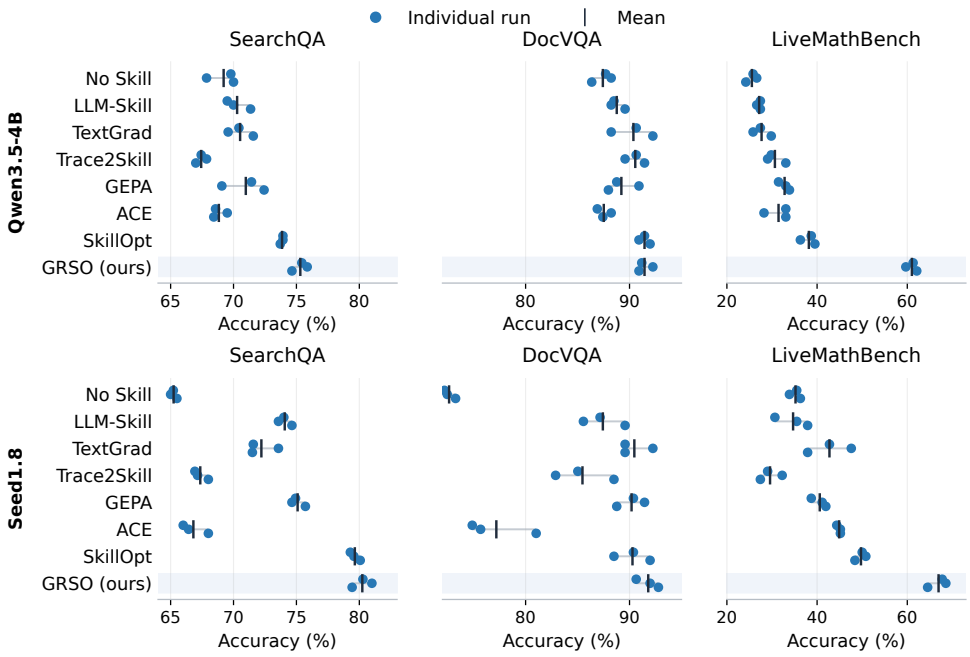

Figure 5: Individual-run strict accuracies and their means. Rows correspond to target models and columns to benchmarks. Each dot represents one run, black ticks mark their means, and gray segments span the minimum and maximum. Small vertical offsets separate overlapping markers. They do not encode an additional variable. All methods use the same test split within each benchmark.

F Post-merge Verification

Locally accepted edit bundles may interact or be rewritten during merging, so their individual acceptance does not guarantee that the merged skill preserves the same outcomes. We compare GRSO with an additional post-merge gate using Seed1.8 as both target and optimizer. The gate evaluates the merged skill on all five questions in the current step and retains it if mean strict accuracy does

not decrease from the step-start skill. Otherwise, it rolls back the update. All other components remain unchanged. The gate uses greedy decoding, reuses available step-start evaluations, and adds no further retry.

Table 7: Post-merge verification. Acc is test accuracy (%). Input and output tokens are full-training totals across both models. GRSO accuracies and costs are from Tables 2 and 3, respectively. LiveMathBench final-gate token totals are approximated from the reported relative changes.

Benchmark	Variant	Acc	Input (M)	Output (M)
SearchQA	GRSO	80.2	11.80	2.37
	+ final gate	80.0	9.71	2.95
DocVQA	GRSO	91.8	2.83	0.38
	+ final gate	92.1	2.41	0.45
LiveMathBench	GRSO	66.9	3.53	2.65
	+ final gate	52.7	3.04	2.93

The final gate changes accuracy by -0.2 points on SearchQA and $+0.3$ points on DocVQA, but reduces LiveMathBench accuracy from 66.9% to 52.7%. Across all three benchmarks, input tokens decrease and output tokens increase, with lower total token consumption. On LiveMathBench, input tokens decrease by 13.8% and output tokens increase by 10.4%: more frequent rollbacks leave shorter skills, reducing input consumption despite five additional verification rollouts per step. Thus, the additional gate reduces token use but does not consistently improve test accuracy. This comparison does not establish that merging always preserves locally successful outcomes.