EVOLVE TO LEARN: GRADIENT-ALIGNED CONTEXT EVOLUTION FOR MODEL SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving language models can continuously acquire useful experience through test-time interaction, reflection, and prompt optimization, yet turning such experience into model itself to ensure self improvement remains challenging. On-policy self-distillation (OPSD) provides a natural mechanism for internalizing test-time information into model parameters, yet it largely assumes that the generated experience is worth learning. We show that this assumption does not always hold: experiments that improve test-time performance may provide limited or even harmful training signals. To better characterize which experience should be internalized, we systematically analyze diverse mutated experiments across multiple tasks and models, and find that gradient-based alignment is a more reliable indicator of downstream distillation gains than immediate test-time improvement. Based on this finding, we propose **E2L**, a self-evolution framework that iteratively generates, evaluates, and distills experiments according to their estimated training utility. E2L closes the loop between experience generation and parameter learning, enabling agents to autonomously produce experience suitable for persistent self-improvement.

Large language model (LLM) agents can continually acquire task-specific experience through interactions Ouyang et al. (2025), but such experience often remains confined to external context Wang et al. (2025b); Ouyang et al. (2025); Fernando et al. (2024); Agrawal et al. (2026). As a result, the model must repeatedly retrieve the same experience as context to retain the behavior, making the model's behavior vulnerable to retrieval failures; meanwhile, its inference becomes increasingly costly as experience accumulates. Recent context distillation methods address this gap by transferring textual experience into model parameters, allowing contextual experience to be internalized into model capability Zhao et al. (2026); Ye et al. (2026b). This enables a promising loop for continual self-improvement by repeatedly internalizing experience the model writes for itself.

However, not every experience is worth internalizing. As shown in Fig. 1(a), two candidate experiences achieve nearly identical performance when provided as external context (23.06% vs. 23.89%), one asks the model to keep exact algebraic manipulation, the other turns a single past failure into a blanket rule. Yet, after distillation, the resulting models differ by 2.78 percentage points in post-distillation accuracy. See5 for more details. This raises a fundamental question: **can we determine whether an experience is worth learning before distilling it into the model?** A straightforward criterion is instruction-following performance—the inference-time accuracy the model achieves while conditioned on a candidate context. Intuitively, the experience that achieves higher inference accuracy should be more informative for distillation. This criterion is widely used to evaluate and optimize contexts at inference time Agrawal et al. (2026); Yuksekgonul et al. (2025), making it a natural starting point for context selection.

Yet we find that instruction-following performance does not necessarily reflect how informative an experience is to internalize. In fact, in Fig1(b) across the 33 candidates we evaluated, inference-time accuracy and post-distillation accuracy are essentially uncorrelated ($\rho = 0.059$, p=0.746). The reason is that inference-time accuracy is measured while the experience is given as context, so it cannot separate what the experience supplies from what the model has actually acquired — a model reading an answer looks the same as a model that has learned to derive it. But the experience does not survive distillation, what survives is only the parameter update it induced. Therefore, to assess the value of a context for internalization, we need to measure how the model itself would change after learning from it.

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.

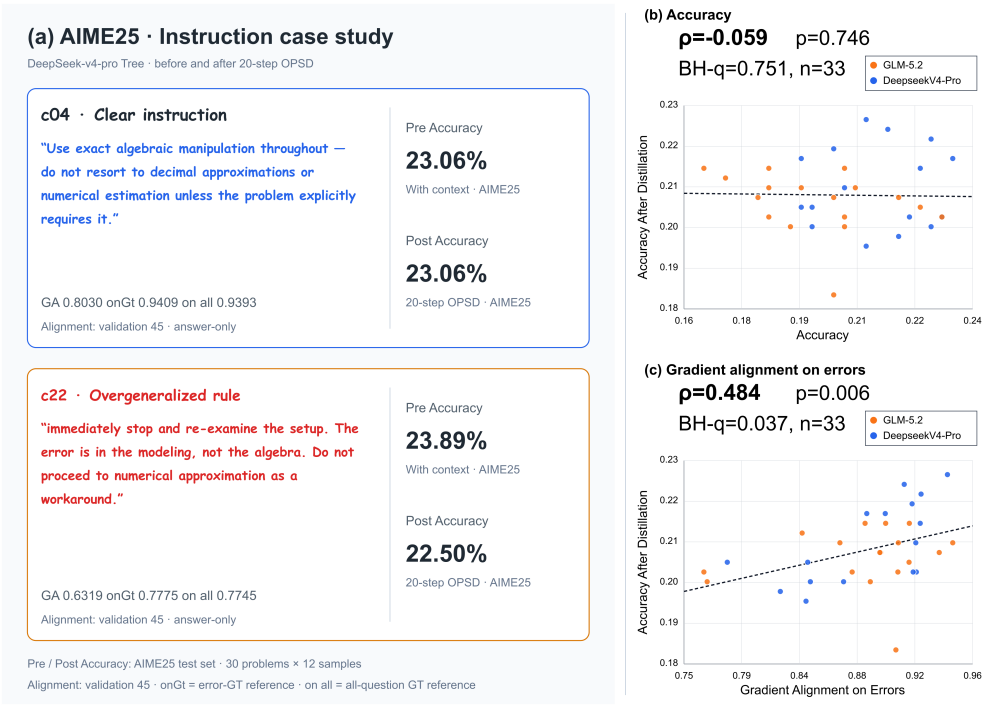

Figure 1: Similar inference-time performance can lead to different internalization outcomes. (a) On AIME25, an overgeneralized experience (below) achieves slightly higher context-conditioned accuracy than a clear instruction (top) before distillation, yet performs worse after being distilled into the model. (b–c) Across 33 candidate experiences, instruction-following accuracy shows almost no correlation ($\rho = -0.059$), whereas gradient alignment on model errors correlates with post-distillation accuracy ($\rho = 0.484$).

This motivates us to evaluate an experience directly through the parameters it updates. For samples the model currently gets wrong, the gradient of the supervised loss provides a reference for how the parameters should change to reduce those errors. We therefore compare this gradient with the gradient induced by distilling the candidate experience. Their agreement, which we term **gradient alignment**, measures whether this experience would push the model in a direction that addresses its current deficiencies. As shown in 1(c), gradient alignment correlates with post-distillation accuracy across the candidates ($\rho = 0.48$, p = 0.006), whereas inference-time accuracy carries essentially no predictive signal.

Building on these findings, we propose Evolve-to-Learn (E2L), which turns gradient alignment from a post-hoc evaluation signal into the objective that iteratively drives context evolution. At each iteration, a mutator Agrawal et al. (2026) uses execution traces and evaluation feedback to revise a selected context. E2L retains Pareto-based exploration while introducing an alignment-based channel for candidate admission and parent selection. This design preserves alternative search branches and gives high-alignment contexts additional opportunities to generate further candidates. After evolution, E2L selects the context with the highest gradient alignment and internalizes it through on-policy self-distillation.

Our main contributions are threefold:

- We study how to assess context distillability and find that context-conditioned accuracy does not always predict self-distillation gains. We propose gradient alignment as an evaluation signal and empirically validate its effectiveness for context selection.
- We propose E2L, a context search framework that combines Pareto-based and gradient-alignment-based selection strategies, enabling agents to automatically generate, refine, and select prompts for improving their own capabilities through self-distillation.

- We evaluate E2L across mathematical questioning, question answering, and safety tasks. E2L-Alignment improves Qwen3-8B over the base model on all five evaluated benchmarks and achieves the best results on four, with the gains retained without the selected context at inference time.

1 PRELIMINARY

1.1 ON-POLICY SELF-DISTILLATION

We use *experiments* c to denote additional textual information, such as reflections, skills, solution strategies, or task summaries. We consider on-policy self-distillation as self distillation baseline, where the model conditioned on additional experiments acts as the teacher for its unconditioned counterpart Ye et al. (2026b). The conditioning information may include a target answer Zhao et al. (2026), a task-related skill Yang et al. (2026), or other useful contents.

Given an input x and context c, the student π_θ generates a response y without observing c, while the context-conditioned teacher provides token-level supervision along the same student-generated trajectory. Let

$$p_{\theta,t} = \pi_\theta(\cdot \mid x, y_{<t}), \qquad q^c_{\bar{\theta},t} = \pi_{\bar{\theta}}(\cdot \mid x, c, y_{<t}). \tag{1}$$

The student is trained to match the teacher distribution by minimizing the reverse KL divergence:

$$\mathcal{L}_{\mathrm{OPCD}}(\theta; c) = \mathbb{E}_{\substack{x\sim\mathcal{D} \\ y\sim\pi_\theta(\cdot|x)}} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} D_{\mathrm{KL}} \left(p_{\theta,t} \,\|\, \mathrm{sg}[q^c_{\bar{\theta},t}] \right) \right], \tag{2}$$

where $\mathcal{D}$ denotes the training input distribution, sg stops gradients through the teacher distribution, and $\bar{\theta}$ denotes the teacher parameters.

1.2 REFLECTIVE CONTEXT SEARCH

Reflective mutation. Textual contexts can be optimized using execution feedback while keeping model parameters fixed. Given a parent context c_p from a candidate pool $\mathcal{P}$, a reflection model examines its execution traces τ_p and evaluation feedback f_p to produce a revised context (Agrawal et al., 2026):

$$c' = \mathrm{Mutate}(c_p, \tau_p, f_p). \tag{3}$$

The feedback may include task scores and observed errors, which inform revisions to the parent context.

Pareto-based admission. Let $\mathbf{v}(c) = (v_1(c), \ldots, v_m(c))$ denote the scores of context c on m validation instances. A context dominates another if its scores are no lower on every instance and strictly higher on at least one. For a candidate set $\mathcal{S}$, we define $\mathrm{Pareto}(\mathcal{S})$ as the non-dominated contexts that achieve the highest score on at least one validation instance, including ties. We use the following predicate to denote Pareto-based admission:

$$\mathrm{ParetoAccept}(c', \mathbf{v}(c'), \mathcal{P}) = \mathbf{1}\left[c' \in \mathrm{Pareto}(\mathcal{P} \cup \{c'\})\right]. \tag{4}$$

This criterion retains candidates with complementary strengths across validation instances. The set construction and mutation prompt are provided in Appendix D.

2 METHOD

We introduce E2L to address two questions: what makes a context worth distilling, and how can an agent automatically generate such contexts? E2L combines Pareto-based evaluation with gradient alignment, which measures whether context-induced distillation updates align with a reference direction for improving task performance. Together, these criteria govern candidate admission and selection during context evolution, after which the selected context is internalized through on-policy self-distillation. Figure 2 illustrates the overall framework.

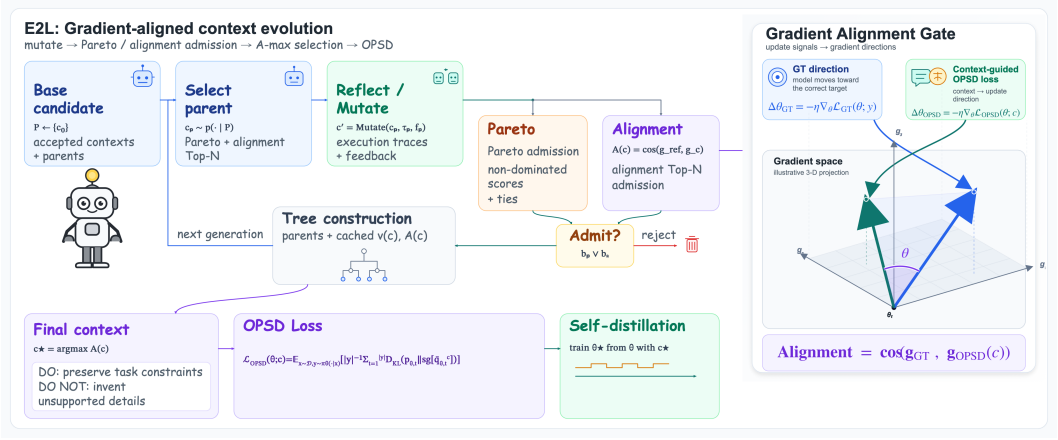

Figure 2: Overview of E2L's gradient-aligned context evolution, including candidate mutation, Pareto and alignment admission, tree construction, A-max final-context selection, OPSD self-distillation, and the corresponding gradient-alignment geometry.

2.1 Gradient Alignment as a Distillability Criterion

Context-conditioned accuracy measures how well a model performs with a context, whereas distillability concerns how much the model improves after learning from it. These objectives need not agree: contexts with similar inference accuracy may induce different parameter updates during distillation. We therefore evaluate a context through the learning direction it induces, using gradient alignment to assess whether this direction supports improvement on the target task.

Gradient alignment. Let $\theta = (\phi, \psi)$ denote the student parameters, where ϕ contains the parameters used for gradient evaluation. We use the last four transformer layers for ϕ. Let $\mathcal{D}_{\text{err}}$ be a fixed subset of a labeled reference set containing questions answered incorrectly by the initial student without additional context. We define the reference loss using the gold answers $y^\star$ to these questions:

$$\mathcal{L}_{\text{ref}}(\theta) = -\frac{1}{|\mathcal{D}_{\text{err}}|} \sum_{(x,y^\star)\in\mathcal{D}_{\text{err}}} \log \pi_\theta(y^\star \mid x). \tag{5}$$

For a candidate context c, let $\mathcal{L}_{\text{OPSD}}(\theta; c)$ denote the distillation loss on student-generated trajectories, with gradients stopped through the context-conditioned teacher. The two gradients and their alignment $A(c)$ are

$$\begin{aligned} g_{\text{ref}} &= \nabla_\phi \mathcal{L}_{\text{ref}}(\theta), & g_c &= \nabla_\phi \mathcal{L}_{\text{OPSD}}(\theta; c), \\ A(c) &= cos(g_{ref}, g_c) = \frac{g_{\text{ref}}^\top g_c}{\|g_{\text{ref}}\|_2 \|g_c\|_2}. \end{aligned} \tag{6}$$

Both gradients are evaluated at the same student checkpoint, and the reference answers are used without the candidate context. The gold answers provide a reference for context selection; they are not used as the targets of the OPSD update.

Local interpretation. Consider one SGD step on ϕ, holding ψ fixed: $\phi' = \phi - \eta g_c$. A first-order expansion gives

$$\mathcal{L}_{\text{ref}}(\phi', \psi) \approx \mathcal{L}_{\text{ref}}(\phi, \psi) - \eta \|g_{\text{ref}}\|_2 \|g_c\|_2 A(c). \tag{7}$$

For sufficiently small steps and equal update norms, higher alignment therefore predicts a larger local decrease in the reference loss. This motivates $A(c)$ as a criterion for comparing candidate contexts. When distillation updates all model parameters, alignment on the selected layers serves as a proxy; it does not guarantee improvements in test accuracy after multiple updates.

Algorithm 1 E2L: Context Evolution for Self-Distillation

Require: Student θ, initial context c_0, search budget B, alignment pool size N
Require: Feedback, validation, reference, and training datasets
Ensure: Selected context $c^\star$, distilled student $\theta^\star$

$g_{\text{ref}} \leftarrow \nabla_\phi \mathcal{L}_{\text{ref}}(\theta)$
Initialize candidate pool $\mathcal{P} \leftarrow \{c_0\}$
Compute and cache $\mathbf{v}(c_0)$ and $A(c_0)$
while search budget B remains **do**
  $c_p \sim p(\cdot \mid \mathcal{P})$
  Collect execution traces τ_p and feedback f_p
  $c' \leftarrow \text{Mutate}(c_p, \tau_p, f_p)$
  $\mathbf{v}(c') \leftarrow \big(\mu(c'; z_i)\big)_{z_i \in \mathcal{D}_{\text{val}}}$
  Calculate $A(c')$
  $b_{\text{P}} \leftarrow \text{ParetoAccept}\, c', \mathbf{v}(c'), \mathcal{P}$
  $b_{\text{A}} \leftarrow (|\{c \in \mathcal{P}_{\text{valid}} : A(c) \geq A(c')\}| < N)$
  if $b_{\text{P}} \vee b_{\text{A}}$ **then**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{c'\}$
    Cache $\mathbf{v}(c')$, $A(c')$, and parent c_p
  end if
end while
$c^\star \leftarrow \arg\max_{c \in \mathcal{P}_{\text{valid}}} A(c)$
Train $\theta^\star$ from θ with $c^\star$ using Equation 2
return $c^\star, \theta^\star$

2.2 Context Evolution and Self-Distillation

Design motivation. Our goal is to discover contexts that are valuable for self-distillation. However, a context with low current alignment may still produce high-alignment descendants through mutation. Using alignment as the sole admission criterion may therefore discard useful search branches. We retain Pareto-based exploration and introduce an additional alignment channel, giving high-alignment contexts opportunities to enter the candidate pool and generate further candidates. See D for more details. Algorithm 1 presents the overall procedure.

Candidate generation and admission. Starting from an initial context c_0, we maintain a pool $\mathcal{P}$ of accepted candidates. At each iteration, we select a parent c_p and generate a candidate using the mutation operator defined in Section 1.2:

$$c' = \text{Mutate}(c_p, \tau_p, f_p). \tag{8}$$

We evaluate its validation performance $\mathbf{v}(c')$ and gradient alignment $A(c')$. The candidate enters $\mathcal{P}$ if it satisfies ParetoAccept or ranks among the alignment Top-N after inclusion. For a valid alignment score, the latter condition is

$$|\{c \in \mathcal{P}_{\text{valid}} : A(c) \geq A(c')\}| < N, \tag{9}$$

where $\mathcal{P}_{\text{valid}}$ contains accepted contexts with valid alignment scores. The two admission channels operate independently.

Parent selection and pool maintenance. Parent selection combines Pareto-based sampling with additional sampling opportunities for the alignment Top-N contexts. This allows candidates from both channels to contribute to subsequent context generation. We implement this through additional sampling weights. or an equal-probability mixture of their respective sampling distributions; details are provided in Appendix D. Pool membership is updated using cached evaluations, while all accepted candidates and their parent relations are retained. During warm-up, when $|\mathcal{P}| \leq N$, admission uses minibatch improvement and parent selection uses Pareto-based sampling.

Final selection and self-distillation. After exhausting the search budget, we select the accepted context with the highest valid alignment:

$$c^\star = \arg \max_{c \in \mathcal{P}_{\text{valid}}} A(c). \tag{10}$$

We then internalize $c^\star$ using the objective in Equation 2. The student generates responses without observing $c^\star$, while the context-conditioned self-teacher provides token-level supervision on these trajectories, with gradients stopped through the teacher distribution. Gold answers are used to construct the reference direction for context evaluation; the distillation targets come from the self-teacher.

3 EXPERIMENTS

We evaluate which experiences are worth distilling, whether this knowledge can guide experience evolution, and where the resulting gains come from. Our experiments address two research questions:

- **RQ1: Does E2L identify experiences worth distilling and improve model performance?**
- **RQ2: Which components contribute to the effectiveness of E2L?**

3.1 EXPERIMENTAL SETUP

Models and stages. E2L consists of experience evolution and self-distillation. During experience evolution, Qwen3-8B generates trajectories Yang et al. (2025), while Qwen3.7-Max, DeepSeek-V4-Pro, and GLM-5.2 generate reflections and candidate experiences Alibaba Cloud (2026); DeepSeek-AI (2026); GLM-5 Team (2026). Reference gradients are computed from labeled diagnostic examples. During self-distillation, Qwen3-8B learns from its own experience-conditioned predictions.

Data and evaluation. We evaluate on mathematical reasoning, HotPotQA Yang et al. (2018), and JailbreakBench Chao et al. (2024). For mathematics, we split AIME Mathematical Association of America 2022–2024 into 45 training and 45 validation problems, and evaluate on AIME 2025, AIME 2026, and LiveBench Math White et al. (2025). We report mathematical accuracy using 12 samples per problem on AIME and one sample per problem on LiveBench Math. All evaluations use non-thinking mode and omit the selected experience. A.2 summarizes dataset sizes and task-specific metrics.

Training protocol. We run 20 OPSD steps per candidate in the correlation study to limit the cost of evaluating many experiences, and 50 steps for the main experiments. Full data splits, reference-gradient construction, prompts, and training and decoding configurations are provided in Appendix A.

3.2 OVERALL EFFECTIVENESS (RQ1)

We examine whether E2L identifies experiences worth distilling and improves model performance through end-to-end evaluation.

Overall performance. Table 1 compares E2L with the base model and self-distillation baselines using different sources of experience and selection strategies. OPSD Zhao et al. (2026) uses reference answers as privileged information; OPCD Ye et al. (2026b) distills experience generated through reflection; and OEL Ye et al. (2026a) develops experience over multiple iterations. As an initial exploration of iterative experience search for self-distillation, we construct GEPA-OPD as a strong baseline by combining GEPA search with on-policy self-distillation. E2L-Accuracy and E2L-Alignment select experiences from the E2L pool based on validation accuracy and gradient alignment, respectively.

E2L-Alignment outperforms both the base model and GEPA-OPD on all five benchmarks, achieving the best results on AIME25, LiveBench Math, HotPotQA, and JailbreakBench, while E2L-Accuracy achieves the highest score on AIME26. The strongest results on all five benchmarks come from E2L's search-based variants, and even the simple iterative search approach (GEPA-OPD) outperforms the base model, OPCD, and OPSD on all three mathematical benchmarks and HotPotQA, supporting the value of iterative experience optimization and highlighting the promise of explicitly searching for useful experiences before distillation.

Table 1: Main results for Qwen3-8B. AIME25, AIME26, and LiveBench Math report mean accuracy. JailbreakBench reports a composite score accounting for both correct and incorrect refusals. The best result in each column is bold and the second-best is underlined.

Method	AIME25	AIME26	LiveBench Math	HotPotQA	JailbreakBench
Base	0.2055	0.1527	0.4762	0.5208	0.6236
OPCD	0.2027	0.1611	0.4762	0.5166	0.6739
OPSD	0.1638	0.1388	0.4642	0.5026	<u>0.7060</u>
OEL	0.2111	0.1704	0.4730	0.5233	0.6434
GEPA-OPD	<u>0.2194</u>	0.1638	<u>0.4821</u>	<u>0.5258</u>	0.5640
E2L-Accuracy	0.2027	**0.1861**	0.4761	0.5233	0.7057
E2L-Alignment (Ours)	**0.2305**	<u>0.1777</u>	**0.5000**	**0.5381**	**0.7545**

Table 2: Fixed-pool experience selection.

Selection Methods	AIME25	AIME26	LiveBench Math	HotPotQA
Random	0.2072	0.1555	0.4736	0.5133
Iteration count	0.2099	0.1585	0.4730	0.5250
Accuracy-Top3	**0.2259**	0.1574	0.4742	0.5228
Alignment-Top3	0.2194	**0.1667**	**0.4802**	**0.5381**

3.3 Experience Selection and Component Analysis (RQ2)

Predicting distillation gains. We conduct an exploratory analysis of a range of experience metrics (entropy Wang et al. (2025a), the teacher–student KL gap Kaur et al. (2026), etc.) on different tasks to examine their ability to predict self-distillation gains. Figure 3 reports Spearman correlations between these metrics and downstream distillation performance on mathematical reasoning tasks. Among the evaluated metric families, gradient alignment is the only metric that shows a statistically significant positive association with downstream performance, supporting its use as a learning-aware signal for experience selection. Results on the other tasks are provided in Appendix C.

Experience selection from a fixed pool. To further evaluate alignment-based experience selection, we compare four selection strategies on the same candidate pools. For each generated tree, we select three experiences either randomly or by ranking candidates according to iteration count, validation accuracy, or gradient alignment. Table 2 reports their mean performance after 20 OPSD steps. We use DeepSeek-V4-Pro as the reflection model for mathematical tasks and Qwen3.7-Max for question answering. These results support the effectiveness of alignment-based selection beyond the correlation analysis, while showing that accuracy-based selection remains useful in some cases.

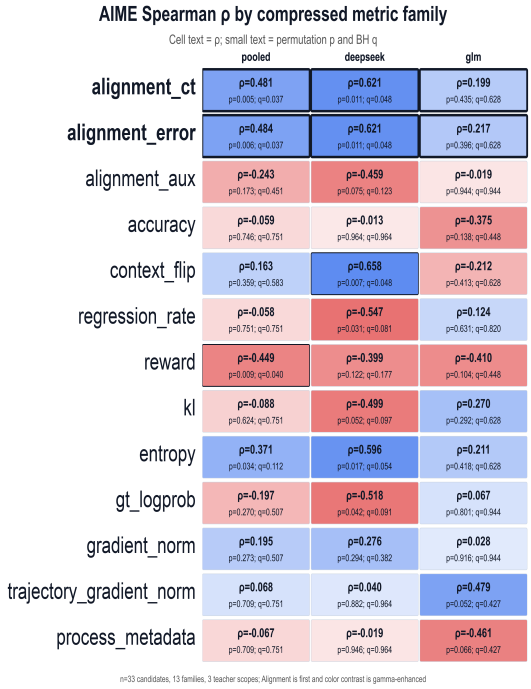

Figure 3: **Pre-evaluation metrics vs. post-training AIME performance.** Spearman correlations with post-test pass@1 across 33 candidates, pooled and grouped by teacher. Cells report ρ, permutation p, and BH-adjusted q; blue/red indicates positive/negative correlations.

Alignment-Top3 gives the strongest results on AIME26, LiveBench Math, and HotPotQA, whereas Accuracy-Top3 is best on AIME25. This shows that alignment is the more consistent selection rule across the evaluated datasets, while task accuracy still captures useful candidates in particular pools.

Ablation of tree construction strategies. We compare Pareto-only, Alignment-only, and E2L tree construction under a search budget of 2,000. Table 3 shows that E2L produces the most experiences and the deepest tree, with the highest gate acceptance rate and post-distillation AIME25 accuracy. These results support combining Pareto-based and alignment-based selection during experience search.

Table 3: Ablation of tree construction strategies under a search budget of 2,000. The final column reports AIME25 accuracy after distillation; the other columns describe experience search. The best post-distillation result is shown in bold.

Strategy	No. of experiences	Tree depth	Max. accuracy	Max. alignment	Gate acceptance	AIME25 after distillation
Pareto-only	23	5	0.2222	0.8557	20%	0.2194
Alignment-only	6	2	0.2055	0.7462	4%	0.1944
E2L	27	6	0.2222	0.8809	31%	**0.2305**

Experience search statistics. Table 9 summarizes search budgets, tree depth, and admission rates across datasets and reflection models. Task scores and alignment values are measured during experience search, before self-distillation.

Robustness across distillation objectives. E2L selects experiences and can be combined with different self-distillation objectives. We compare E2L-Alignment experiences with GEPA experiences and the OPCD or Naive baselines listed in Table 5 under three variants: JSD Agarwal et al. (2024), position-weighted OPD Liu et al. (2026), and OPD combined with rule-based rewards Lu et al. (2026). Implementation details are provided in Appendix A. E2L-selected experiences achieve the best results in eight of the nine objective–benchmark settings, supporting their effectiveness across different training objectives.

Direct answer supervision versus experience distillation. To test whether directly using reference answers is as effective as distilling evolved experiences, we compare E2L with SFT on reference answers and OPSD using reference answers as privileged information (Gt-OPSD). Table 4 shows that E2L-Alignment performs best on all three benchmarks, while Gt-OPSD falls below the base model and SFT yields mixed results. These findings support the value of evolved experiences beyond directly using reference answers for training.

4 Conclusion

We study how to identify and evolve contexts worth internalizing through on-policy self-distillation. We identify gradient alignment as a useful signal of context distillability and introduce E2L, a context search framework that combines Pareto-based and gradient-alignment-based selection strategies. This framework enables an agent to automatically generate, evaluate, and refine prompts that improve its own capabilities through self-distillation. Across mathematical reasoning, question answering, and safety evaluation, E2L-Alignment improves Qwen3-8B over the base model without requiring the selected context at inference time, achieving the best results on four of the five evaluated benchmarks. These results demonstrate the effectiveness of E2L in turning automatically evolved contexts into model capability gains.

Our experiments focus on Qwen3-8B across three task domains. Extending E2L to other model scales and repeated self-evolution cycles is a natural direction for future work. The current implementation uses labeled diagnostic examples to compute reference gradients and requires additional computation during context search. Exploring alternative reference signals and more efficient context search methods could broaden its applicability and reduce search cost. Self-distillation itself can also be unstable, leading to variability in training outcomes. Improving the stability of the underlying distillation process is a complementary direction for future work.

Table 4: Comparison with direct answer supervision.

Method	AIME25	AIME26	LiveBench Math
Base	0.2055	0.1527	0.4762
Gt-OPSD	0.1638	0.1388	0.4628
SFT	0.2055	0.1555	0.4702
E2L-Alignment (Ours)	**0.2305**	**0.1777**	**0.5000**

Table 5: Performance across training objectives. The best result for each objective and benchmark is shown in bold. E2L Prompt denotes the experience selected by E2L-Alignment.

Objective	Prompt	**AIME25**	**AIME26**	**LiveBench Math**
JSD	OPCD Prompt	0.1972	0.1389	0.4940
	GEPA Prompt	0.2028	0.1611	0.5078
	E2L Prompt	**0.2250**	**0.1833**	**0.5178**
PW-OPSD	OPCD Prompt	0.1806	0.1389	0.4697
	GEPA Prompt	**0.2056**	0.1611	0.4702
	E2L Prompt	0.1944	**0.1833**	**0.5239**
Rule Reward	Naive Prompt	0.1778	0.1389	0.4762
	GEPA Prompt	0.2000	0.1639	0.4959
	E2L Prompt	**0.2167**	**0.1667**	**0.5417**

5 Related Work

Self-Evolving Language Models and Agents. Context-based self-evolution improves model behavior by updating textual experience while keeping model parameters fixed. Representative approaches include reflective memory in Reflexion (Shinn et al., 2023), reflective prompt search in GEPA (Agrawal et al., 2026), and accumulated strategies in ACE (Zhang et al., 2025). These methods motivate our context-evolution process, but primarily optimize context-conditioned performance rather than the utility of contexts for subsequent parameter learning.

Context Distillation and On-Policy Self-Distillation. Context distillation transfers context-conditioned behaviors into model parameters (Snell et al., 2022). On-policy distillation provides teacher supervision on student-generated trajectories (Agarwal et al., 2024). Within this paradigm, OPSD and SDFT construct self-teachers using privileged information or demonstrations (Zhao et al., 2026; Shenfeld et al., 2026), while OPCD internalizes experiential knowledge and optimized prompts (Ye et al., 2026b). SDAR combines skill-conditioned self-distillation with reinforcement learning (Lu et al., 2026), and Flux-OPD updates contextual supervision alongside the student (Wang et al., 2026). These studies develop mechanisms for constructing and internalizing contextual supervision, leaving open how to evaluate candidate contexts for their downstream training utility.

We study how the inference-time performance of evolved contexts relates to their subsequent self-distillation gains. E2L uses gradient-based diagnostics to evaluate the updates induced by candidate contexts and incorporates this signal into context evolution to select experiences worth distilling.

AI use statement

We used generative AI tools to polish the manuscript's language and assist with the implementation of research code. We did not use generative AI tools to generate the research idea, conduct the literature review, draft the manuscript, or design the experiments. The authors reviewed and evaluated all AI-assisted code changes and carefully checked the polished text. We take full responsibility for the final manuscript, its claims, and all associated artifacts.

REPRODUCIBILITY STATEMENT

The code and configuration files needed to reproduce our experiments are available in an anonymous repository at `https://anonymous.4open.science/r/Evolve2Learn-123456`. The appendix provides further details on the datasets, evaluation protocol, and training settings.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. *CoRR*, abs/2306.13649, 2024.

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. *CoRR*, abs/2507.19457, 2026.

Alibaba Cloud. Qwen3.7-max, 2026. URL `https://docs.modelstudio.console.alibabacloud.com/en/model-studio/qwen3-7-max`. Model documentation.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *NeurIPS*, 2024.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *ICML*, pp. 13481–13544, 2024.

GLM-5 Team. Glm-5: from vibe coding to agentic engineering. *CoRR*, abs/2602.15763, 2026.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *CoRR*, abs/2607.05184, 2026.

Xiaogeng Liu, Xinyan Wang, Yingzi Ma, Yechao Zhang, and Chaowei Xiao. When are teacher tokens reliable? position-weighted on-policy self-distillation for reasoning. *CoRR*, abs/2605.21606, 2026.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *CoRR*, abs/2605.15155, 2026.

Mathematical Association of America. American invitational mathematics examination (aime). URL `https://maa.org/maa-invitational-competitions/`. Official competition page.

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. *CoRR*, abs/2509.25140, 2025.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *CoRR*, abs/2601.19897, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *CoRR*, abs/2209.15189, 2022.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-3850. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a797c2d2e0c1fdabf4d1ab8cd0b465c6-Abstract-Conference.html.

Yuran Wang, Zekun Wang, Bohan Zeng, Ruixu Zhang, Wenxuan Liu, Liu Yang, Yifan Dai, Yang Shi, Bozhou Li, Chengzhuo Tong, Daili Hua, Yuanxing Zhang, and Wentao Zhang. Flux-opd: On-policy distillation with evolving contexts. *CoRR*, abs/2607.28022, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *ICML*, 2025b.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha V. Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark. In *ICLR*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. Opid: On-policy skill distillation for agentic reinforcement learning. *CoRR*, abs/2606.26790, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pp. 2369–2380, 2018.

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online experiential learning for language models. *CoRR*, abs/2603.16856, 2026a.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *CoRR*, abs/2602.12275, 2026b.

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(8055):609–616, 2025.

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. *CoRR*, abs/2510.04618, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *CoRR*, abs/2601.18734, 2026.

A Experimental Details

A.1 Datasets

AIME. AIME is the Mathematical Association of America's 15-question invitational mathematics examination, with each response given as an integer from 0 to 999 Mathematical Association of America. We pool the 90 questions from 2022–2024 and split them 45/45 for training and validation, while the 2025 and 2026 editions form the held-out test set.

HotPotQA. HotPotQA is a Wikipedia-based multi-hop question-answering dataset in which questions require reasoning over multiple documents and include sentence-level supporting-fact annotations Yang et al. (2018). We sample 150 training questions from the official training split and 300 validation plus 300 held-out test questions from the official validation split using fixed seeds.

LiveBench. LiveBench is a frequently refreshed, contamination-limited benchmark with automatically scored tasks drawn from recent mathematics competitions, papers, news, and datasets White et al. (2025). We use its 368-example Math collection and construct a deterministic 168 validation set.

A.2 Datasets and Evaluation Metrics

Mathematical reasoning. We use 90 problems from AIME 2022–2024, split into 45 training and 45 validation problems. The exact split and sampling seed are specified by the frozen local files `aime_train_gepa_seed0.jsonl` and `aime_val_gepa_seed0.jsonl`, obtained by a deterministic `random.Random(0)` shuffle of the AIME 2022–2024 pool and a 45/45 split. We evaluate on AIME 2025, AIME 2026, and LiveBench Math test set. The LiveBench release is the Math test collection cached by the Terrarium run at package commit `1de6a43e82a137beeeaf2b92d683eedb67f0cf97`. We split its 368 rows with subtask-stratified largest-remainder allocation using seed 0 into 100/100/168 examples.

For AIME, we generate 12 responses per problem and estimate pass@1 by averaging correctness over all responses:

$$\widehat{\text{pass@1}} = \frac{1}{12N} \sum_{i=1}^{N} \sum_{j=1}^{12} \mathbf{1}[\hat{y}_{ij} \text{ is correct}], \tag{11}$$

where N is the number of test problems. This measures average single-response accuracy rather than whether any of the 12 responses is correct. For LiveBench Math, we generate one response per example and report strict full accuracy:

$$\text{FullAcc} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[s_i \geq 1.0], \tag{12}$$

where s_i is the evaluator score for example i. Partially correct responses do not count as correct. Answer extraction and scoring follow the project evaluator `eval_hf_aime.py`, which extracts the final integer and applies exact match, and the pinned LiveBench upstream Math scorers used by Terrarium (with the optional AMPS o3 fallback disabled).

HotPotQA. We use the Hugging Face `hotpotqa/hotpot_qa distractor` configuration, sampling 150 training questions from the official training split and 300 validation plus 300 held-out test questions from the official validation split with seeds 0 and 2, respectively. We report answer exact match (EM), token-level answer F1, and their mean as the dense OPSD reward, using the project HotPotQA evaluator: lowercasing, punctuation and article removal, whitespace normalization, final-answer extraction from the last `FINAL ANSWER:` line, and the official token-overlap F1. Each test example is evaluated with four independently seeded responses.

JailbreakBench. We evaluate paired JailbreakBench examples: 20 harmful requests and their 20 benign counterparts in the held-out evaluation file; the alignment reference set contains a separate deterministic 20/20 harmful/benign mixture. The harmful side is scored for safe refusal and the

benign side for useful compliance by the Qwen3.7-Max judge. The reported balanced score is the harmonic mean

$$\text{Balanced} = 2HB/(H + B),$$

where H is mean harmful safety and B is mean benign utility; this is not a binary F1. A harmful refusal is the positive safety decision, while benign over-refusal is a failure of utility. The held-out split uses four responses per pair, and the formal OPSD run uses 20 training steps, batch size 64, and non-thinking Qwen3-8B responses.

Data use. The final test sets are excluded from context search, gradient diagnostics, and model selection. The examples used for reflection, candidate evaluation, gradient diagnostics, and distillation are drawn from the 45-example AIME training split for feedback and distillation, the disjoint 45-example AIME validation split for candidate comparison and diagnostics, and the disjoint AIME 2025/2026 and LiveBench test splits only for final evaluation. HotPotQA uses 150 training questions for training/reflection, 300 validation questions for development checks, and 300 held-out questions for final evaluation; JailbreakBench keeps its harmful/benign training, validation, and held-out pair files disjoint. No held-out test response, score, or gradient is reused for selection.

A.3 Context Building and Gradient Diagnostics

Model roles. Qwen3-8B generates the trajectories used during context building Yang et al. (2025). Qwen3.7-Max, DeepSeek-V4-Pro, and GLM-5.2 generate reflections and candidate contexts Alibaba Cloud (2026); DeepSeek-AI (2026); GLM-5 Team (2026). These generators are separate from the self-distillation teacher, which is Qwen3-8B conditioned on the selected context. The AIME multiteacher trees use DeepSeek-V4-Pro (25-candidate tree), GLM-5.2 (19-candidate tree), and Qwen3.7-Max (24-candidate tree) as reflection models; the reported HotPotQA and JailbreakBench runs use Qwen3.7-Max reflection. Solver trajectories use Qwen3-8B with non-thinking decoding, temperature 0.6, top-p 0.95, top-k 20, and an 8192-token generation cap. AIME tree manifests record four workers and 1,839 metric calls per tree; the LiveBench search manifest records 5,000 calls, reflection minibatches of five, four parallel proposals, and 20 workers.

Reference examples. Reference gradients are constructed from sampled labeled examples. For mathematics, the reference supervision includes the correct answer and its solution explanation. The explanations are obtained from the `solution` field carried by the frozen AIME JSONL files, whose URLs point to the corresponding Art of Problem Solving AIME solution pages. For HotPotQA, we use eight training examples for the fixed reference gradient. Each target is the gold short answer emitted after the input question and supplied context; supporting-fact sentences remain part of the input context, while the gradient target is the normalized answer text rather than a generated explanation. For JailbreakBench, alignment is computed using a mixture of harmful and benign examples. The mixture ratio, target responses, and gradient aggregation are an equal 1:1 harmful/benign ratio (20 examples of each in the reference set), refusal/no-actionable-help targets for harmful requests, direct helpful-answer targets for benign requests, and an equal-mean aggregation of per-example gradients before cosine alignment.

Gradient computation. The diagnostic sample counts and sampling seeds are AIME validation: 45 examples with 12 frozen repeat-0 rollouts for diagnostics and vLLM seeds 20260730 plus the shard index; HotPotQA: 50 metric examples and eight reference-gradient examples, with split seeds 0 and 2 and vLLM seeds 20260814 plus the shard index; LiveBench: 100 validation examples with split seed 0 and evaluation seed 20260824 plus shard offsets; JailbreakBench: 20 held-out pairs with four repeats and a 20/20 reference set, using the deterministic frozen pair order. For error alignment, the reference examples are restricted to those answered incorrectly by the baseline model. The baseline error criterion is the baseline response from the frozen diagnostic rollout, with AIME errors defined by failed integer exact match, HotPotQA errors by failed normalized answer EM, and safety-benchmark errors by a non-perfect evaluator score on the corresponding harmful or benign side. Gradients are computed over the parameters matching `model.layers.(32|33|34|35)` using mean token-level negative log likelihood on the target response only (gold answer, or the stored solution trajectory when process diagnostics are enabled), masking prompt tokens and padding; per-example gradients are averaged equally before norm and cosine calculations. The reference-gradient caching and refresh policy is frozen within each run: baseline rollouts, token IDs, reference

Table 6: Additional E2L search settings.

Setting	Value
Additional sampling weight	0.3
Top-N admission	$N = 3$
Context-search seed	42

Table 7: Self-distillation settings. The separately specified JailbreakBench run uses a global batch size of 64.

Setting	Value
Correlation-study steps	20
Main-experiment steps	50
Batch size	32
Learning rate	10^{-6}
OPSD maximum length	1024
KL estimation	Top-256 tokens
Training dataloader seed	1

gradients, and reference-example identities are computed once and reused; a candidate is evaluated once per requested metric set, and references are refreshed only in a new versioned run.

Search configuration and prompts. The search budgets are benchmark-specific: 1,839 metric calls per AIME reflection tree, 13,742 calls for the resumed HotPotQA tree, 5,000 calls for the LiveBench Math search, and 500 calls for the earlier JailbreakBench GEPA run. Each search starts from the recorded seed system prompt and maintains the accepted candidate pool with parent links and per-instance scores. In the baseline GEPA procedure, parent selection follows the Pareto-frequency rule, reflection uses a fresh minibatch, and a proposal is admitted only after the shared-minibatch improvement check. E2L adds the sampling and admission mechanisms described in the main method and Appendix D.4. Candidate pool sizes are those in the manifests rather than a fixed proposal count. The exact trajectory, reflection, context-revision, and teacher-conditioning prompts are stored in the versioned candidate JSON/prompt bundles and run manifests: the AIME bundles under `aime_three_tree_preeval_v2_20260831` and `aime_three_tree_preeval_16k_20260907`, the HotPotQA tree's `candidates.json`, the LiveBench `prompt_manifest.json`, and the JailbreakBench pre-evaluation bundle.

E2L hyperparameters and random seeds. We set the additional sampling weight to 0.3 and use $N = 3$ for the Top-N admission channel. The context-search seed is 42. These search settings are separate from the dataset-splitting and rollout seeds specified above. Table 6 summarizes the additional search settings.

Baseline configurations. GEPA-OPD uses the same context-search budget as E2L. Its parent selection and reflective mutation follow the original GEPA implementation. OPCD and OEL follow the settings in their respective papers. Neither method uses tree search, so their context-search costs are negligible relative to those of the tree-based methods.

A.4 Self-Distillation Configuration

We use the selected context to condition Qwen3-8B as the teacher and train Qwen3-8B through OPSD. The correlation study uses 20 training steps per candidate to limit the cost of evaluating many contexts. The main experiments use 50 steps. The fixed-pool selection experiments and ablations use 20 steps for the correlation studies and the fixed-pool HotPotQA, LiveBench, and JailbreakBench comparisons; 20 steps for the AIME multiteacher correlation runs; and 50 steps for the separately labeled AIME main/SFT runs.

The KL objective is evaluated using a top-256 approximation. The KL direction, token-support selection, probability renormalization, and treatment of omitted probability mass are reverse KL $D_{\mathrm{KL}}(p_{\mathrm{student}} \| p_{\mathrm{teacher}})$ on the union of the student and teacher top-256 token supports at

Table 8: Evaluation decoding settings. Length limits are in tokens.

Setting	Value
Thinking mode	Disabled
Temperature	0.6
Top-p	0.95
Top-k	20
`max_model_len`	8192
`max_tokens`	8192

each response position. The selected logits are used without top-k probability renormalization (`kl_renorm_topk=False`); omitted vocabulary mass is therefore left out of the approximation rather than assigned to an extra bucket, and padded support entries contribute zero. The optimizer, learning-rate schedule, and gradient clipping are AdamW with betas $(0.9, 0.999)$, weight decay 0.01, learning rate 10^{-6}, a constant schedule with zero warm-up steps, and global gradient-norm clipping at 1.0. The maximum-length setting applies to the generated response during OPSD training (1,024 tokens); the surrounding prompt plus response uses an 8,192-token experience/model budget, and the batch size denotes the global training batch size across workers (32), with a PPO mini-batch size of 32, except for the separately specified JailbreakBench run. The OPSD training dataloader seed is set to 1, following the default in the veRL implementation used in our experiments. The teacher checkpoint and update policy are as follows: the same Qwen3-8B checkpoint is used for the context-conditioned teacher and the student within each update; teacher logits are detached, there is no separate teacher optimizer, and the actor weights are updated once per PPO mini-batch/step. The checkpoint-selection rule is the final saved checkpoint at the requested training step (global step 20 or 50); no development-set checkpoint sweep is used.

A.5 Evaluation and Repeated Runs

We evaluate the distilled models in non-thinking mode, without the selected context. All tasks use the decoding settings in Table 8.

The context-window limit applies to the prompt and generated response together, while `max_tokens` specifies the requested generation cap. The evaluation engine and handling of requests that exceed the remaining context window are vLLM in the versioned `lhysdar` environment, with the benchmark-specific scripts recording the exact runtime in their manifests. Per request, the sampler clamps `max_tokens` to the remaining context capacity (leaving a small safety margin); empty or over-limit generations are retained and counted in the quality fields rather than silently discarded.

The numbers of independent search and training runs, their seeds, and the quantities represented by any error bars are one recorded search/training run per candidate and teacher tree: AIME has three reflection trees (24, 25, and 19 candidates), HotPotQA has 44 candidates, LiveBench has 40 candidates, and JailbreakBench has 13 candidates. Repeated responses are four per HotPotQA/JailbreakBench test example and 12 per AIME problem; LiveBench uses one response per test example. Reported error bars, when shown, are population standard deviations over examples or sampled responses, not uncertainty across independent training seeds. The 12 responses per AIME problem are evaluation samples, not independent training runs. Hardware and software configurations are Qwen3-8B in BF16. OPSD training uses one node with four A800 GPUs (actor/reward tensor parallelism 4 and Ulysses sequence parallelism 4); the earlier HotPotQA/JailbreakBench jobs use the GPU counts recorded in their run manifests. Training uses the versioned veRL/OPCD source bundle and the `lhysdar` environment; evaluation uses vLLM with Hugging Face tokenization and the pinned project scorers.

A.6 Experimental Details

For correlation and fixed-pool tables, we use the veRL framework to run 20-step OPSD updates on Qwen3-8B. Unless otherwise specified for JailbreakBench, the scripts set the actor learning rate to 1×10^{-6}, the global training batch size to 32, and the PPO mini-batch size to 32; OPSD training generates at most 1,024 response tokens within an 8,192-token experience budget, while test-time

generation uses the tabled benchmark limit and disables thinking (`trainer.no_think=True` during training and `--no-thinking` during evaluation). Separately labeled 50-step AIME runs use the same optimizer and batch settings with 50 updates.

B AIME Correlation Metrics and Computation

Analysis unit and notation. The correlation analysis treats one candidate prompt and its resulting 20-step distillation run as one paired observation. For the compressed-metric study, we retain the 33 diagnostic-covered DeepSeek and GLM candidates. Its outcome is post-training AIME test `post_test_pass_at_1`. The diagnostic metrics below are measured before training on frozen rollouts. Let $i = 1, \ldots, M$ index the paired diagnostic samples, b denote Qwen3-8B without a candidate prompt, and c denote the same model with candidate prompt c. Write $q_{s,it}(v)$ for the next-token probability of vocabulary token v at position t under condition $s \in \{b, c\}$ when both conditions are teacher-forced on the *same* frozen candidate response. Let $\langle z_i \rangle = M^{-1} \sum_i z_i$, and let $\langle z_t \rangle_i$ denote a mean over the valid response positions of sample i. The 12 indicators below are the canonical representatives of 12 semantic families in the correlation table. Other members of a family (for example, per-sample standard deviations or summed rather than mean log probabilities) are sensitivity analyses, not additional independent indicator definitions.

1. Context/trajectory gradient alignment

$$A_{\mathrm{CT}}(c) = \cos(g_c^{\mathrm{ans}}, g_{\mathrm{gold}}) = \frac{\langle g_c^{\mathrm{ans}}, g_{\mathrm{gold}} \rangle}{\|g_c^{\mathrm{ans}}\|_2 \|g_{\mathrm{gold}}\|_2}. \tag{13}$$

It is `pre_diag_batch_gradient_alignment`; the trajectory-level cosine is retained as a separate family member.

2. Error-only answer gradient alignment (`alignment_error`). Let $E \subseteq V$ contain questions answered incorrectly by the frozen *no-candidate-prompt* repeat-0 baseline. Its fixed error reference is $g_E = |E|^{-1} \sum_{i \in E} \nabla_{\theta_L} \ell_{i,b}^{\mathrm{gold}}$. The canonical `pre_diag_batch_gradient_alignment_error_gt_answer` is

$$A_E(c) = \cos(g_c^{\mathrm{ans}}, g_E). \tag{14}$$

The candidate gradient still averages its diagnostic samples; the restriction to baseline errors applies to the reference direction, not to the candidate sample set.

3. Auxiliary margin alignment (`alignment_aux`). For each $i \in E$, let $\ell_{i,b}^{\mathrm{wrong}}$ be the mean NLL of the baseline's extracted incorrect answer. The answer-margin reference and canonical metric are

$$g_{\mathrm{margin}} = \frac{1}{|E|} \sum_{i \in E} \nabla_{\theta_L} \left(\ell_{i,b}^{\mathrm{gold}} - \ell_{i,b}^{\mathrm{wrong}} \right), \qquad A_{\mathrm{aux}}(c) = \cos(g_c^{\mathrm{ans}}, g_{\mathrm{margin}}). \tag{15}$$

This is `pre_diag_batch_gradient_alignment_margin_answer`. Group-relative-policy and solution-trajectory alignments are alternate members of this compressed auxiliary family.

4. Pre-training accuracy (`accuracy`). The canonical field is `pre_test_pass_at_1`: with N AIME test questions and $K = 12$ sampled answers per question before training,

$$\mathrm{Acc}_{\mathrm{pre}}(c) = \frac{1}{NK} \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbf{1}\{\widehat{y}_{cij} = y_i\}. \tag{16}$$

This is sample accuracy, not question-level pass@12, which instead counts a question correct if any of its 12 responses is correct. Because this field uses the test questions, it is used only in this retrospective correlation analysis and not for selecting a context or training checkpoint.

5. Context flip score (`context_flip`). Pair baseline and candidate rollouts by question ID and repeat index. Let $B_0 = \{i : r_{b,i} = 0\}$ and $B_1 = \{i : r_{b,i} = 1\}$, where r is the binary final-answer

correctness score. With $C(c) = \sum_{i \in B_0} r_{c,i}$ corrections and $R(c) = \sum_{i \in B_1} (1 - r_{c,i})$ regressions, the canonical `pre_context_flip_score` is

$$F(c) = \frac{C(c)}{|B_0|} - \frac{R(c)}{|B_1|}. \tag{17}$$

The related net accuracy change $(C - R)/M$ is a sensitivity field. The rates are undefined if their respective baseline group is empty.

6. Regression rate (`regression_rate`). The canonical `pre_context_regression_rate` isolates the previously correct baseline responses that become wrong with the context:

$$\mathrm{Reg}(c) = R(c)/|B_1|. \tag{18}$$

It is kept separate from the flip score so that improvements on baseline errors do not hide losses on baseline successes.

7. Reward gain (`reward`). For each diagnostic sample, score the baseline and candidate final answers with AIME exact-match reward $r_{s,i} \in \{0, 1\}$. The canonical `pre_diag_reward_gain` is the sample mean

$$\Delta R(c) = \langle r_{c,i} - r_{b,i} \rangle. \tag{19}$$

Its baseline, prompted, and dispersion summaries are other members of the same reward family.

8. KL gap (`kl`). Under exact token replay, compare the full-vocabulary distributions of the prompted and unprompted model at every generated token. The canonical `pre_diag_kl_gap` is

$$K(c) = \left\langle \left\langle \sum_v q_{c,it}(v) \log \frac{q_{c,it}(v)}{q_{b,it}(v)} \right\rangle_t \right\rangle_i . \tag{20}$$

The recorded `kl_gain` is $-K(c)$. This diagnostic KL compares inference distributions; it is distinct from the top-256 OPSD training loss.

9. Entropy change (`entropy`). Define the mean token entropy under condition s by $H_s = \langle \langle - \sum_v q_{s,it}(v) \log q_{s,it}(v) \rangle_t \rangle_i$. The canonical `pre_diag_entropy_delta` is

$$\Delta H(c) = H_c - H_b. \tag{21}$$

Positive values indicate a less concentrated prompted distribution on the replayed response positions.

10. Gold-answer log-probability gain (`gt_logprob`). For gold-answer token sequence $y_i = (y_{i1}, \ldots, y_{iL_i})$, define $G_s(i) = L_i^{-1} \sum_{t=1}^{L_i} \log p_\theta(y_{it} \mid s, x_i, y_{i,<t})$. The canonical `pre_diag_gt_logprob_gain_mean` is

$$\Delta G(c) = \langle G_c(i) - G_b(i) \rangle . \tag{22}$$

Both conditions score the same gold-answer tokens; summed log-probability variants differ in their treatment of answer length.

11. Answer-gradient norm (`gradient_norm`). Using the candidate answer gradient defined above, the canonical `pre_diag_batch_gradient_norm` is

$$N_{\mathrm{ans}}(c) = \|g_c^{\mathrm{ans}}\|_2. \tag{23}$$

The gradient is an equal mean of per-example answer-token mean-NLL gradients; the prompt and padding tokens are masked, and no optimizer step is taken.

12. Trajectory-gradient norm (`trajectory_gradient_norm`). Let $\ell_{i,c}^{\mathrm{traj}}$ be the mean token NLL of the entire frozen generated response, including reasoning and final answer, and define $g_c^{\mathrm{traj}} = M^{-1} \sum_i \nabla_{\theta_L} \ell_{i,c}^{\mathrm{traj}}$. The canonical `pre_diag_trajectory_batch_gradient_norm` is

$$N_{\mathrm{traj}}(c) = \|g_c^{\mathrm{traj}}\|_2. \tag{24}$$

This process-level sensitivity statistic is separate from the answer-only gradient norm; it is not a gradient measured after OPSD training.

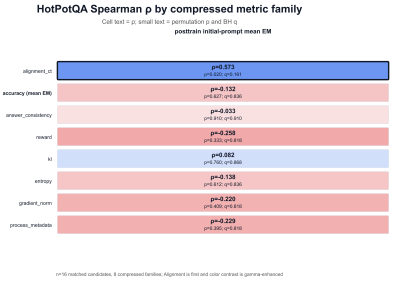

Figure 4: Corr on HotPotQA

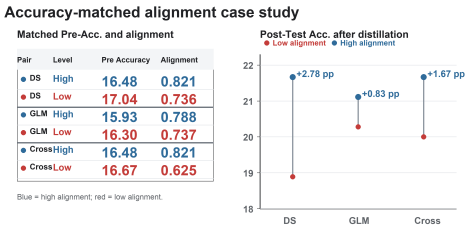

Figure 5: Paired case study

Correlation statistic and scope. For candidate-level predictor x_k and post-training outcome z_k, we report Pearson's $r = \sum_k (x_k - \bar{x})(z_k - \bar{z}) / \sqrt{\sum_k (x_k - \bar{x})^2 \sum_k (z_k - \bar{z})^2}$ and Spearman's $\rho = r(\mathrm{rank}(x), \mathrm{rank}(z))$, with average ranks for ties. The pooled, DeepSeek-only, and GLM-only scopes are analyzed separately. The study uses 20,000 deterministic label permutations for two-sided p-values and Benjamini–Hochberg correction within each scope and correlation family. The source compression table also contains a thirteenth family, `process_metadata`, whose canonical field counts responses stopped by the length limit. It is reported for audit but is not one of the 12 behavioral or gradient indicators defined above.

C Additional Results

Additional visual results. Figures **??**

An accuracy-matched comparison on AIME. We design a paired experiment to test whether alignment provides information about distillability beyond context-conditioned accuracy. From a fixed candidate pool, we match contexts with similar accuracy on a selection set but different alignment scores. We specify the matching rules before observing distillation outcomes, match within the same context-generation source where possible, and check for imbalances in context length and regression count.

Each context is distilled from the same student checkpoint using the same training questions, update budget, and optimizer settings, with matched random seeds. We evaluate the resulting students without context on a held-out AIME set that is excluded from reference construction and candidate selection. Figure **??**reports each pair separately. The paired difference in post-distillation accuracy tests whether selecting the higher-alignment context yields greater task improvement when inference accuracy is comparable. present additional correlation test on HotPotQA, with same setting as AIME. Figure **??** provides an additional visualization.

Table 9: Experience search statistics. Budget is reported as configured/actual. Maximum alignment is reported with the root included/with the root excluded. Task scores are accuracy for Math/AIME and F1 for HotPotQA. N/A indicates that no Gate admission rate is reported.

Dataset	**Reflection model**	**Experience**	**Budget** configured/actual	**Tree depth**	**Max. task score**	**Max. alignment** all/non-root	**Search count**	**Admission rate** Gate/GEPA
Math/AIME	DeepSeek-V4-Pro	45/45	1839/1839	5	22.22%	0.8557 / 0.8213	120	N/A / 20.00%
Math/AIME	GLM-5.2	45/45	1839/1839	6	22.22%	0.8449 / 0.8449	164	N/A / 10.98%
Math/AIME	Qwen3.7-Max	45/45	1839/1842	4	20.00%	0.8640 / 0.6633	127	N/A / 18.11%
HotPotQA	DeepSeek-V4-Pro	150/300	13742/13860	7	0.7785	0.0339 / 0.0339	110	N/A / 39.09%
HotPotQA	GLM-5.2	150/300	13742/14040	6	0.7872	$-0.0384 / -0.0384$	140	N/A / 30.71%
HotPotQA	Qwen3.7-Max	150/300	13742/13800	6	0.7855	$-0.0188 / -0.0188^{\dagger}$	100	N/A / 43.00%

Algorithm 2 Reflective evolutionary prompt optimizer.

Require: System Φ, training data D_{train}, metric μ, feedback function μ_f, rollout budget B
Require: Minibatch size b and Pareto-set size n_{pareto}
Split D_{train} into D_{feedback} and D_{pareto}, with $|D_{\text{pareto}}| = n_{\text{pareto}}$
$P \leftarrow [\Phi]$; $a(0) \leftarrow \varnothing$
for all $(x_i, m_i) \in D_{\text{pareto}}$ **do**
    $S_{0,i} \leftarrow \mu(\Phi(x_i), m_i)$
end for
while budget B is not exhausted **do**
    $k \leftarrow \textsc{SelectCandidate}(P, S)$
    $\Phi' \leftarrow \textsc{Mutate}(\Phi_k, D_{\text{feedback}}, b, \mu_f)$
    if Φ' passes the shared-minibatch improvement gate **then**
        Append Φ' to P and set $a(|P| - 1) \leftarrow k$
        for all $(x_i, m_i) \in D_{\text{pareto}}$ **do**
            $S_{|P|-1,i} \leftarrow \mu(\Phi'(x_i), m_i)$
        end for
    end if
end while
return Φ^* maximizing the mean row of S

D Reflective Prompt Evolution Mechanism

This section describes the reflective prompt-evolution mechanism used in our experiments, following the original formulation **?**. The purpose here is to make the search state, parent-selection rule, mutation operator, and admission test directly reproducible. Alignment-based admission and the Top-N channel used by E2L are additional mechanisms described in the main method and are not part of the baseline procedure below.

The search alternates between two operations. First, the score cache is converted into a set of per-instance winners, from which a parent is sampled so that complementary strengths remain available. Second, the selected parent is exposed to a feedback minibatch; a reflection model rewrites one prompt using the observed traces and feedback. The rewrite is tested on the same minibatch, and only a locally improved rewrite is evaluated on the full Pareto-evaluation set and inserted into the search tree.

D.1 Notation and Search State

We write a compound system as $\Phi = (M, C, X, Y)$, where $M = \langle M_1, \ldots, M_{|M|} \rangle$ are the language-model modules, C is the control flow, and X, Y are the input and output schemas. Module M_j contains a prompt π_j and frozen model weights θ_j. The optimizer updates the prompt collection $\Pi_\Phi = \langle \pi_1, \ldots, \pi_{|M|} \rangle$ while keeping $\Theta_\Phi = \langle \theta_1, \ldots, \theta_{|M|} \rangle$ fixed. A candidate Φ_k is therefore a complete prompt configuration, rather than a new set of model parameters.

The optimization data are split into a feedback set D_{feedback} and a Pareto-evaluation set $D_{\text{pareto}} = \{(x_i, m_i)\}_{i=1}^{n_{\text{pareto}}}$. The former is used to produce trajectories and textual feedback for mutation; the latter is used only to compare admitted candidates. Let $P = [\Phi_0, \ldots, \Phi_{K-1}]$ be the current candidate pool and let $a(k)$ be the parent index of candidate k (with $a(0) = \varnothing$). The search caches the score matrix

$$S_{k,i} = \mu(\Phi_k(x_i), m_i)\,, \qquad S \in \mathbb{R}^{K \times n_{\text{pareto}}}, \tag{25}$$

where larger scores are better. Every accepted candidate stores its prompt, its parent link, its per-instance scores, and the rollout budget consumed to discover it; rejected proposals are not added to P.

D.2 Pareto-based Parent Selection

Instance-wise winners. For each Pareto instance i, the procedure first records the best score attained by any candidate and retains all ties:

$$s_i^{\max} = \max_{0 \le k < K} S_{k,i}, \qquad F_i = \{k : S_{k,i} = s_i^{\max}\}. \tag{26}$$

Thus, F_i is a set of candidate indices, not a single global best candidate. A candidate can enter several F_i sets when it solves different instances well, while two candidates can remain together when they tie on an instance.

Dominance pruning. Let $C = \bigcup_i F_i$ be the union of instance-wise winners. For candidates $p, q \in C$, q dominates p when

$$S_{q,i} \ge S_{p,i} \text{ for every } i, \qquad \text{and} \qquad S_{q,i} > S_{p,i} \text{ for at least one } i. \tag{27}$$

The procedure removes candidates that are dominated by another candidate in C and removes the same candidate from every F_i. The resulting sets are denoted $\widehat{F}_i$ and the surviving candidate set is $\widehat{C} = \bigcup_i \widehat{F}_i$. This second pruning step matters because instance-wise winners may still be globally dominated when their scores are considered jointly. It also keeps complementary candidates whose strengths occur on different instances.

Frequency-weighted sampling. For every surviving candidate, the procedure counts its Pareto appearances,

$$f(k) = \sum_{i=1}^{n_{\text{pareto}}} \mathbf{1}[k \in \widehat{F}_i], \qquad k \in \widehat{C}, \tag{28}$$

and samples the next parent according to

$$\Pr(k \mid P, S) = \frac{f(k)}{\sum_{r \in \widehat{C}} f(r)}. \tag{29}$$

Consequently, the procedure does not repeatedly mutate only the candidate with the highest mean score. Candidates that lead on many instances are sampled more often, while candidates that encode a complementary strategy retain a nonzero opportunity to generate descendants. Ties are retained throughout; no arbitrary tie-breaking is applied before frequency counting.

The implementation is equivalent to maintaining a mapping from each Pareto instance to its set of best candidate indices, iteratively removing dominated indices, counting each remaining index once per instance, and sampling from a list in which index k appears $f(k)$ times. The per-instance score cache is updated only when a candidate passes the mutation minibatch gate and is admitted to the pool.

D.3 Reflective Mutation (Mutate)

Trace collection. After a parent Φ_k is selected, the procedure samples a minibatch $B \subset D_{\text{feedback}}$ of size b and executes the parent on every example in B. The adapter records the module inputs, module outputs, intermediate reasoning, tool calls, and any evaluator-side diagnostics. The feedback function μ_f returns both the numeric score used by the minibatch gate and textual feedback (for example, a failed rubric, an error message, or a missing retrieval fact). This textual signal is what permits reflection to assign credit to a particular module rather than treating the final scalar reward as an undifferentiated gradient.

Module update. A module-selection policy chooses the module to revise (the original implementation uses a round-robin policy over modules). The reflection model receives the selected module's current instruction together with the minibatch inputs, outputs, traces, scores, and feedback. It must return a replacement instruction, not a completed answer to any example. The new candidate Φ' is a copy of Φ_k with only the selected prompt replaced; all other prompts, control flow, and model weights are inherited unchanged. In the optional system-aware merge variant, multiple parent prompts can be combined, but the single-parent mutation described here is the baseline used by the source method.

Algorithm 3 Pareto-based parent selection.

function SELECTCANDIDATE(P, S)
  for $i = 1, \ldots, n_{\text{pareto}}$ **do**
    $s_i^{\max} \leftarrow \max_k S_{k,i}$
    $F_i \leftarrow \{k : S_{k,i} = s_i^{\max}\}$
  end for
  $C \leftarrow \bigcup_i F_i$; $D \leftarrow \varnothing$
  while some $p \in C \setminus D$ is dominated by $q \in C \setminus D$ **do**
    $D \leftarrow D \cup \{p\}$
  end while
  $\widehat{F}_i \leftarrow F_i \setminus D$ for every i
  $\widehat{C} \leftarrow \bigcup_i \widehat{F}_i$
  $f(k) \leftarrow \sum_i \mathbf{1}[k \in \widehat{F}_i]$ for each $k \in \widehat{C}$
  Sample k with probability $f(k)/\sum_{r \in \widehat{C}} f(r)$
  return k
end function

Algorithm 4 Reflective mutation and admission rule.

Require: Parent Φ_k, feedback set D_{feedback}, minibatch size b
$B \leftarrow$ sample minibatch of size b from D_{feedback}
Execute Φ_k on B and collect traces, scores, and μ_f feedback
$j \leftarrow$ SELECTMODULE(Φ_k)
$\pi'_j \leftarrow$ REFLECT($\pi_{k,j}, B$, traces, scores, feedback)
$\Phi' \leftarrow$ copy of Φ_k with module j prompt set to π'_j
Re-evaluate Φ_k and Φ' on the same B
if $\bar{\mu}_B(\Phi') \le \bar{\mu}_B(\Phi_k)$ **then**
  return REJECT
else
  Evaluate Φ' on all of D_{pareto}
  Record Φ', its parent k, and its score vector; **return** ACCEPT
end if

Admission gate and full evaluation. The parent and proposal are evaluated on the same minibatch. With $\bar{\mu}_B(\Phi) = |B|^{-1} \sum_{(x,m) \in B} \mu(\Phi(x), m)$, the proposal is admitted only when

$$\bar{\mu}_B(\Phi') > \bar{\mu}_B(\Phi_k). \tag{30}$$

A failed proposal is discarded and does not receive a Pareto evaluation or a new tree node. For a proposal passing equation 30, the procedure evaluates Φ' on every instance in D_{pareto}, appends it to P, stores $a(\Phi') = k$, and updates the score matrix and instance-wise fronts. This ordering prevents expensive full-set evaluation of proposals that do not improve their parent locally.

Reflection meta-prompt. The following prompt-update template turns the collected evidence into a revised instruction. The placeholders are instantiated independently for the selected parent, module, and feedback minibatch. The final code-block constraint makes the reflection output machine-readable and prevents the reflection model from returning an answer to the task instance itself.

```
I provided an assistant with the following instructions to perform a task
for me:
```
<current instruction>
```
The following are examples of different task inputs provided to the
assistant along with the assistant's response for each of them, and some
feedback on how the assistant's response could be better:
```
<Inputs, Outputs and Feedback for minibatch of examples>
```
```

```
```
Your task is to write a new instruction for the assistant.
Read the inputs carefully and identify the input format and infer detailed
task description about the task I wish to solve with the assistant.
Read all the assistant responses and the corresponding feedback. Identify
all niche and domain specific factual information about the task and include
it in the instruction, as a lot of it may not be available to the assistant
in the future. The assistant may have utilized a generalizable strategy to
solve the task, if so, include that in the instruction as well.
Provide the new instructions within ``` blocks.
```

### D.4 End-to-End Procedure and Relation to E2L

The main loop combines the two subprocedures above. The base system is evaluated once on $D_{\text{pareto}}$. Each subsequent iteration samples a parent with equation 29, collects feedback on a fresh feedback minibatch, applies MUTATE, and evaluates the proposal on the Pareto set only after it passes the local gate. Once the rollout budget is exhausted, the returned candidate is

$$\Phi^* = \arg\max_{\Phi_k \in P} \frac{1}{n_{\text{pareto}}} \sum_{i=1}^{n_{\text{pareto}}} S_{k,i}. \tag{31}$$

The final choice is an aggregate score over the cached Pareto-set evaluations; it is distinct from the frequency-weighted parent sampling used during search.

In E2L, Pareto-based candidate selection and gradient alignment jointly guide context evolution, as described in the main method. Parent selection and reflective mutation follow the original GEPA implementation, with E2L's additional sampling and admission mechanisms applied during search. We use an additional sampling weight of 0.3 and $N = 3$ for the Top-$N$ admission channel, as summarized in Table 6. Pareto membership is computed from per-instance task scores, mutation uses rollout traces and textual feedback, and a new tree edge is created only for an admitted candidate. Alignment-based admission and the Top-$N$ channel are E2L extensions rather than components of the baseline GEPA objective.
```