REVISITING LLMS AS CONTEXT ENGINEERS

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models (LLMs) are used in more and more new tasks, context engineering becomes a fundamental method to efficiently steer LLMs by constructing instruction prompts or metadata without intensive training. A growing number of LLM context engineering (LCE) approaches automate the task by casting the LLM itself as the engineer that draws generalizable instructions or experiences from offline trials as context. Although these methods resemble how humans learn from experiences, it remains unclear **when and how** LLMs can engineer effective context from data. To answer the question, we first establish a benchmark composed of datasets with different reasoning patterns. We compared different LCE methods with different LLM families in 8B size, showing that there is no single winner for all tasks, even compared with simple baselines like chain-of-thought. Even when scaling model capabilities via more parameters or generation iterations, LCEs do not consistently boost existing performance. Across all datasets and models, the performance gains of LCEs are negatively correlated ($r = -0.5, p \leq 0.05$) with the diversity of the failure modes of reasoning trajectories. Investigating how LCEs work, we find that (1) the improvement in instruction adherence can explain some success of LCEs; (2) one major bottleneck is whether LLMs can extract effective experiences out of natively diverse failures. Through this study, we uncover the limited capabilities of LCEs in abstracting principled contexts from diverse failure modes and understand how LCEs work, providing guidelines for future work.

1 INTRODUCTION

Recent advances in LLMs have demonstrated strong generalization ability across a wide range of real-world applications. In parallel with these developments, one line of work adapts LLMs to downstream tasks through training-based methods such as Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022), Direct Preference Optimization (DPO) (Rafailov et al., 2023), and Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which update the model parameters to align with task-specific objectives. However, these training-based methods often demand large amounts of preference data (Ouyang et al., 2022) or hundreds of thousands of rollouts (Shao et al., 2024). The inefficiency of these methods motivated a complementary line of work that seeks to adapt LLMs in a training-free manner by engineering the prompt or context of LLMs rather than altering their intrinsic capabilities.

Early efforts in this direction mainly rely on human heuristics to engineer the instructions (Agrawal et al., 2026; Cai et al., 2025a; Khattab et al., 2024; Opsahl-Ong et al., 2024; Yuksekgonul et al., 2025; Zhang et al., 2026) that elicit LLM reasoning capabilities (Wang et al., 2023; Wei et al., 2022; Yao et al., 2023a) or execute a better workflow (Hu et al., 2025; Li et al., 2026; Shinn et al., 2023; Xu et al., 2025a; Yao et al., 2023b; Zhang et al., 2025a; Zhu et al., 2025). In-Context Learning (ICL) (Brown et al., 2020) enables few-shot learning via in-context examples. Chain-of-Thought (CoT) prompting and its variants (Wei et al., 2022; Yao et al., 2023a) instruct LLMs to generate intermediate reasoning steps. Later, agent frameworks (Yao et al., 2023b) rely on the reasoning to take actions, allowing LLMs to interact with external tools and environments. However, such methods are designed to accomplish tasks in a single attempt but cannot learn from accumulated human trials recorded in offline datasets. For example, CoT may avoid a mistake in one attempt through reasoning, but the same error may recur in subsequent trials.

Given a set of training examples, **LLM as Context Engineer (LCE)**, or automatic context engineering, was proposed to learn context from data without parameter training (Cai et al., 2025a; Hu et al.,

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.

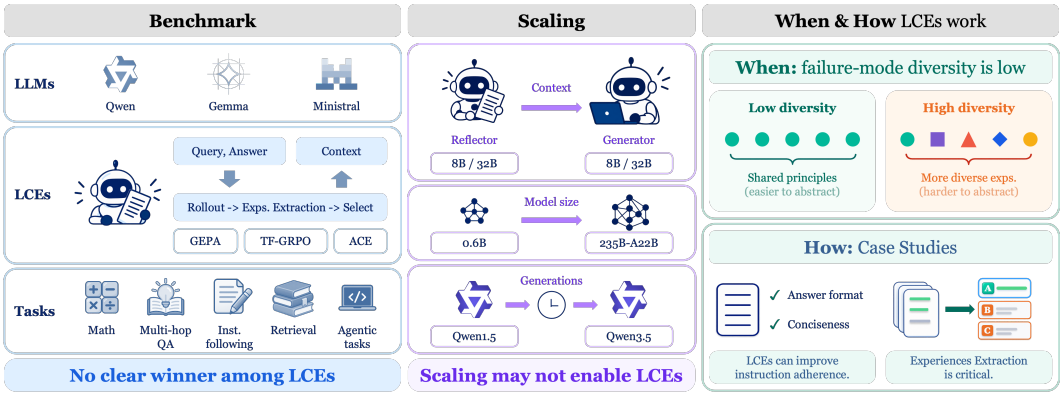

Figure 1: Overview of our work. We investigate whether LCEs can engineer effective context and further examine the mechanisms underlying their successes and failures.

2025; Xu et al., 2025b; Yang et al., 2024; Zhang et al., 2026) The main principle is to cast the LLM itself as a context engineer that operates on textual space rather than parameter space (Yang et al., 2024). Multi-prompt Instruction Proposal Optimizer v2 (MIPROv2) (Opsahl-Ong et al., 2024) and Genetic-Pareto (GEPA) (Agrawal et al., 2026) propose effective sampling strategies for searching over candidate prompts. Training-Free Group Relative Policy Optimization (TF-GRPO) (Cai et al., 2025a) extracts semantic advantages by comparing agent rollouts, while Agentic Context Engineering (ACE) (Zhang et al., 2026) curates an evolving context and uses metadata to track the contributions of each insight.

Despite their success, these methods are typically evaluated using different LLMs and distinct benchmarks, leaving their relative advantages and broader applicability unclear. For example, ACE (Zhang et al., 2026) reports improvements with DeepSeek-V3.1-671B (DeepSeek-AI, 2024) on agentic (Trivedi et al., 2024) and financial tasks (Loukas et al., 2022; Wang et al., 2025), whereas GEPA (Agrawal et al., 2026) reports improvements with Qwen3-8B (Qwen Team, 2025) and GPT-4.1-mini (OpenAI, 2025) on math (Hendrycks et al., 2021) and multi-hop reasoning (Yang et al., 2018; Jiang et al., 2020). These inconsistencies in evaluation protocols motivate us to conduct a comprehensive benchmark of LCEs across LLM families and diverse tasks under comparable evaluation settings. As summarized in Fig. 1, our major contributions in this paper are as follows. (1) We conduct a comprehensive comparison between LCE methods in a unified benchmark, using various LLMs, LCE methods, and task types. (2) We also evaluate the effect of scaling model parameter size and model generations. (3) We conduct new ablation studies and case studies to show when and how LCEs can actually work. Our key findings are as follows:

- **LLMs can** engineer contexts via task-specific training, but no single method can achieve the best performance in all datasets at test time.
- **Scaling** helps but cannot reverse existing failures of LCEs. Even scaling to newer generations and more parameters remains insufficient to make LCEs more effective across different settings.
- **When&How:** Through aggregate analysis, we find that LCEs work better when facing less diverse failure modes of reasoning. Case studies show that the ability to abstract principled contexts from heterogeneous failures is the major bottleneck and that the success of some LCE methods is attributed to the improvement in adherence to (format) instruction rather than reasoning.

2 Related Work

AI Agents AI Agent frameworks can be viewed as test-time scaling methods that allocate additional inference-time computation to organize the operation of LLMs during task execution, without modifying the model parameters. CoT prompting (Wei et al., 2022) is a widely adopted starting point for eliciting the reasoning capabilities of LLMs by generating intermediate steps before the final answer, achieving substantial improvements across diverse downstream applications. Building on CoT, self-consistency (Wang et al., 2023) samples multiple reasoning chains from the same prompt and selects the final answer by majority voting to improve reasoning reliability. Beyond purely internal reasoning, ReAct (Yao et al., 2023b) interleaves reasoning traces with task-specific

actions such as tool calls, allowing the model to ground its reasoning in external observations rather than relying solely on parametric knowledge. Another research line extends reasoning into more structured trajectories. Tree-of-Thoughts (Yao et al., 2023a) frames reasoning as a search over a tree of intermediate thoughts, allowing the model to explore multiple reasoning paths. Graph-of-Thoughts (Besta et al., 2024) further extends this paradigm by representing intermediate thoughts as nodes in a reasoning graph, enabling more flexible aggregation, recombination, and refinement of partial solutions. Despite this progress, these frameworks rely on predefined workflows and offer limited flexibility in adapting to specific tasks. Our study therefore focuses on analyzing methods that automatically engineer contexts in a training-free manner.

LLM Context Engineers Similar to training-based methods, LCEs follow the same optimization loop but rely on textual feedback and updates rather than gradient-based parameter modifications. Automatic Prompt Engineer (Zhou et al., 2023) pioneers this paradigm by searching over LLM-generated instruction candidates to maximize task accuracy. Optimization by Prompting (Yang et al., 2024) casts the LLM itself as an iterative optimizer that refines prompts based on past trajectory. Deep Language Networks (Sordoni et al., 2023) extend this idea to stacked LLMs by jointly optimizing prompts via variational inference. Reflexion (Shinn et al., 2023) introduces a self-reflection feedback loop where the model verbally reflects on task outcomes and conditions subsequent attempts on these self-generated critiques. Building on these advancements, subsequent methods move beyond instance-specific reflection toward optimizing a reusable system prompt that encodes a general strategy for the target task. DSPy (Khattab et al., 2024) provides a unified toolkit inspired by PyTorch abstractions for declaratively programming LLM pipelines. Within this framework, MIPROv2 (Opsahl-Ong et al., 2024) and GEPA (Agrawal et al., 2026) propose different sampling strategies to optimize the instructions. Inspired by backpropagation in neural networks, TextGrad (Yuksekgonul et al., 2025) uses natural language to propagate textual gradients. Recently, another line of work maintains an evolving external context, such as abstract experiences or reusable skills (Cai et al., 2025b; Wu et al., 2026; Li et al., 2026). TF-GRPO (Cai et al., 2025a) and ReasoningBank (Ouyang et al., 2026) contrast successful and failed task executions to extract reusable insights. ACE (Zhang et al., 2026) curates an evolving playbook through iterative bullet-level updates. Despite the success in the reported benchmarks, the inconsistent evaluation makes the actual effectiveness of LCEs unclear. Thus, this work aims to uncover and understand the limitations of LCEs.

3 PRELIMINARIES

When prompting LLM to answer a question, the input contains not only a query but also a context. The **context** could be an instruction or essential information, e.g., definitions of terminology, for answering the query. Distinct from manual context engineering, **LCEs** generate context automatically by iteratively updating the context based on information gained from data. In this paper, we conceptualize the paradigm as a way to learn context π (a sequence of tokens) from a dataset $D = \{(x, y)\}$ where (x, y) is a pair of query and response to the LLM. We are interested in investigating whether LLMs can learn a generalizable π (prompt, experiences, or metadata) from D. Context engineering can be formulated as $\pi^* = \arg\max_\pi V(\pi; D)$, where V quantifies the performance of π. A general context engineering method can be formulated as

$$\pi = \phi_V(\{E(x, t, y)|t \sim \text{LLM}(x, \pi^-), (x, y) \sim D\}|\pi^- \sim \Pi). \tag{1}$$

The formulation includes three steps that transform samples into an informative context π. **Step R (Rollout):** a trajectory t is unrolled to answer the query x based on a sampled context π^-. **Step E (Extraction):** $E(\cdot)$ processes the triplet (x, t, y) to generate an intermediate artifact, e.g., experience or feedback. **Step ϕ (Selection):** $\phi_V(\cdot)$ leverages the evaluation V to transform the intermediate artifact to produce the final context by selecting the most important information.

Existing context engineering methods instantiate the Eq. (1) in different ways. Below, we review four representative methods with open-source implementations.

TextGrad (Yuksekgonul et al., 2025) propagates textual "gradients" (reflection) through a structured pipeline to update prompts iteratively. The method integrates agentic reflection (Yang et al., 2024) in a back-propagation form, enabling optimization of hierarchical context. For TextGrad, Π is the delta distribution on the prompt from the last iteration. E generates feedback on the prompt and ϕ_V update the prompt based on the feedback.

GEPA (Agrawal et al., 2026) realizes a sample-efficient prompt optimizer for compound AI systems. GEPA samples execution trajectories and uses an LLM to reflect on them in natural language to diagnose failures. To avoid local optima, GEPA samples new prompt candidates from a Pareto frontier of high-performing generated prompts in training. We can formulate GEPA in Eq. (1), where Π is a distribution characterized by the Pareto frontier in the search tree, E conducts reflective prompt mutation based on the feedback on t, and ϕ_V selects the prompt.

TF-GRPO (Cai et al., 2025a) instantiate the agentic reinforcement learning. In contrast to GEPA's iterative rewriting of the system prompt, TF-GRPO maintains an external experience pool, instantiating the policy as a frozen LLM paired with a variable experiential context. Mirroring vanilla GRPO (Shao et al., 2024), TF-GRPO generates grouped rollouts in training but replaces the numerical group-relative advantage with semantic reflection on trajectory differences. TF-GRPO instantiates Eq. (1) as follows: E extracts the experiences via reflecting on t, and ϕ selects experiences into the pool as context.

ACE (Zhang et al., 2026) focuses on augmenting external information in context – a structured playbook containing items, each carrying *metadata* (a unique identifier and counters tracking helpfulness) and *content* (a short reusable strategy or failure mode). ACE applies incremental delta updates iteratively and only consolidates the playbook by merging semantically similar entries via embedding-based de-duplication, without actively deleting past bullets. ACE is similar to TF-GRPO in Eq. (1), except that the context π is composed of metadata and more structured experiences.

As summarized in Table 1, these methods cover a wide spectrum of principles. In terms of the form of π, TextGrad, and GEPA were designed for engineering the system prompts, while TF-GRPO and ACE were designed for distilling experiences that augment the context at test time.

Table 1: **Comparison of evaluated context engineering methods.** *Sys* denotes the system prompt; *Exp* denotes experience (e.g., a pool of distilled insights or a playbook injected into context).

Method	Context π	Main Strategy
TextGrad	Sys	Textual gradient backpropagation
GEPA	Sys	Textual reflection + Mutation sampling
TF-GRPO	Exp	Group-relative textual advantage
ACE	Exp + metadata	Textual reflection

In our study, we also include two simple baselines that do not require iterative engineering. (1) Chain-of-Thought **(CoT)** (Wei et al., 2022) uses the prompt *"Solving the problem step by step, {Question or Other Input Context}"* to elicit reasoning before the final prediction, while for AppWorld, we use the initial prompt from the ACE implementation to specify the basic constraints and rules. (2) In-Context Learning **(ICL)** (Brown et al., 2020) guides LLMs using successful rollouts. Different from Brown et al. (2020), we enrich the context with reasoning demonstrations to illustrate how to complete a task step-by-step. We first generate five trajectories for each training sample to construct a demonstration pool. For each test sample, we then randomly select three successful trajectories from this pool as in-context demonstrations.

4 Can LLMs Engineer Context?

We reviewed the evaluation protocols used in different publications of the representative methods and observed minimal overlap across them (see details in Table 4). The disjoint datasets leave the general effectiveness of context engineering unclear. To understand the effectiveness of LCEs, we establish a unified benchmark across different LLM families in 8B size. We include diverse tasks: mathematical reasoning (MATH500, Lightman et al. (2024)), multi-Hop reasoning (HotpotQA, Yang et al. (2018)), instructions following (IFBench, Pyatkin et al. (2025)), retrieval (MiddleMatch, Leviathan et al. (2025)) and Agentic Tasks (AppWorld, Trivedi et al. (2024)). Additional experimental setup is provided in Section B.1.

LCEs generally improve performance in training. In Fig. 2, we compare how different LCEs can improve the context *before* and *after* engineering, namely, during training. We report their validation performance and use the same Qwen3-8B (Qwen3) model (Qwen Team, 2025). The validation set was used to select engineered context during training. Our results show that TF-GRPO, GEPA, and ACE consistently improve validation performance across the four datasets. On MiddleMatch and IFBench, all LCEs improve validation performance, with maximum gains exceeding 20 points for ACE on IFBench and 10 points for GEPA. On MATH500 and HotpotQA, TF-GRPO, GEPA, and

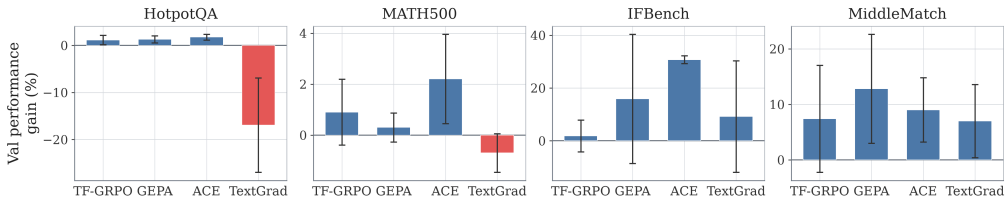

Figure 2: Validation performance gains after context engineering in 4 datasets.

Table 2: Performance of LCEs across LLM families. No method consistently improves the context. **Bold** and $(\cdot)$ denote the best performance and performance changes relative to the initial, respectively.

Model	Dataset	CoT	ICL	TF-GRPO	GEPA	ACE
Qwen3	MiddleMatch	31.27 ± 2.10	$\mathbf{46.07 \pm 4.80}$	37.73 ± 8.57 (+6.47)	42.33 ± 3.80 (+12.53)	25.20 ± 3.96 (+4.73)
	IFBench	32.35 ± 0.82	30.54 ± 0.94	32.21 ± 1.58 (-0.14)	$\mathbf{33.30 \pm 3.64}$ (-0.37)	32.17 ± 1.28 (+0.64)
	MATH500	$\mathbf{83.20 \pm 2.51}$	79.80 ± 1.04	82.10 ± 2.43 (-1.10)	80.60 ± 1.43 (-0.20)	63.80 ± 2.93 (-7.30)
	HotpotQA	$\mathbf{38.00 \pm 0.75}$	37.60 ± 0.83	$\mathbf{38.00 \pm 0.97}$ (0.00)	32.80 ± 0.99 (-1.54)	33.67 ± 0.97 (-3.26)
	AppWorld	21.55 ± 1.95	23.10 ± 3.89	$\mathbf{23.69 \pm 2.08}$ (+2.14)	23.33 ± 1.54 (+0.48)	18.93 ± 1.29 (-3.45)
Gemma4	MiddleMatch	79.33 ± 1.93	80.47 ± 2.32	90.13 ± 3.43 (+10.80)	$\mathbf{93.33 \pm 2.91}$ (+7.40)	87.47 ± 6.16 (+0.73)
	IFBench	39.76 ± 2.22	34.63 ± 2.17	41.63 ± 1.21 (+1.87)	$\mathbf{46.70 \pm 4.54}$ (-1.22)	42.09 ± 1.43 (+1.13)
	MATH500	67.70 ± 1.04	$\mathbf{77.90 \pm 3.54}$	73.80 ± 4.12 (+6.10)	71.20 ± 1.92 (+3.50)	69.40 ± 1.14 (+1.60)
	HotpotQA	$\mathbf{33.80 \pm 0.80}$	32.60 ± 3.27	33.67 ± 1.29 (-0.13)	29.47 ± 1.45 (+0.73)	31.67 ± 0.78 (-0.20)
	AppWorld	25.24 ± 2.25	26.67 ± 2.50	$\mathbf{29.05 \pm 3.13}$ (+3.81)	28.81 ± 0.68 (+1.67)	26.07 ± 4.40 (-0.36)
Min3	MiddleMatch	62.00 ± 3.44	49.53 ± 3.91	$\mathbf{71.40 \pm 6.46}$ (+9.40)	59.60 ± 4.64 (+22.20)	38.93 ± 8.69 (-1.07)
	IFBench	28.50 ± 2.19	26.46 ± 1.87	28.57 ± 1.88 (+0.07)	$\mathbf{36.70 \pm 1.84}$ (+0.82)	31.21 ± 1.90 (+0.27)
	MATH500	87.50 ± 1.06	68.10 ± 2.41	$\mathbf{88.30 \pm 1.15}$ (+0.80)	85.10 ± 1.43 (-1.30)	80.10 ± 3.90 (-3.00)
	HotpotQA	23.07 ± 1.26	27.07 ± 0.95	24.13 ± 1.91 (+1.07)	31.00 ± 1.60 (+2.80)	$\mathbf{31.27 \pm 1.38}$ (+3.13)
	AppWorld	26.43 ± 1.96	28.45 ± 2.12	27.14 ± 2.75 (+0.71)	24.29 ± 7.52 (-1.90)	$\mathbf{30.60 \pm 1.08}$ (+4.40)

ACE achieve only marginal validation gains, while TextGrad fails to produce effective contexts and even leads to a substantial performance decline of more than 15 points on HotpotQA.

No free lunch for LCEs at test time. Beyond validation, we compare these methods at test time, where we only use three methods (TF-GRPO, GEPA, ACE) that stably improve validation performance. To evaluate LCEs in multi-step agentic settings, we additionally include AppWorld, a benchmark requiring LLMs to complete long-horizon tasks to further assess whether LCEs remain effective when task completion requires sustained interaction rather than a single response. We include three LLM families in our evaluation: Qwen3-8B (Qwen3), Gemma-4-E4B-it (Gemma4) (Gemma Team, 2026), and Ministral-3-8B-Instruct-2512-BF16 (Min3) (Liu et al., 2026). Table 2 reports the performance on different datasets across three LLM families and the performance changes relative to each LCE's initial, with CoT and ICL included as reference baselines.
(1) Limited advantages over simple baselines. Although these LCEs can improve the validation performance, these gains do not consistently transfer to the test set and even cause degradation relative to simple baselines. For instance, all LCEs fail to achieve CoT prompting on MATH500 with Qwen3. We notice there are some wins of LCEs over CoT or ICL for specific models on HotpotQA and MATH500. But the wins occur only when a model's baseline performance is lower than that of other models on the same task. For example, on HotpotQA, Min3 has the lowest CoT and ICL scores among the three models (23.07 and 27.07, respectively), while ACE raises its score by 3.13 compared to its initial. Yet, LCEs yield no performance gains for the other LLMs. Thus, we conjecture that such gains are more likely when the model's capabilities are not fully exploited.
(2) No clear winner among LCEs across datasets or models. For example, TF-GRPO performs best on AppWorld with Qwen3, while ACE leads on Min3. We also observe that LCEs substantially improve Gemma4 on MATH500 but provide limited performance gains for Qwen3 and Min3. On HotpotQA, LCEs improve Min3 more than the other models. The limitation suggests that there is no free lunch for LCEs – different models/datasets may need different LCE algorithms.

The limited generalizability of LCE methods encourages us to further investigate the working mechanism in the later sections.

LCE with Scaling. After testing LCE methods with 8B-scale models, we further investigate whether scaling up models can improve the effectiveness of LCEs.
(1) Bigger Reflector Can Cause More Gain but Also More Loss. In Fig. 3, we investigate whether

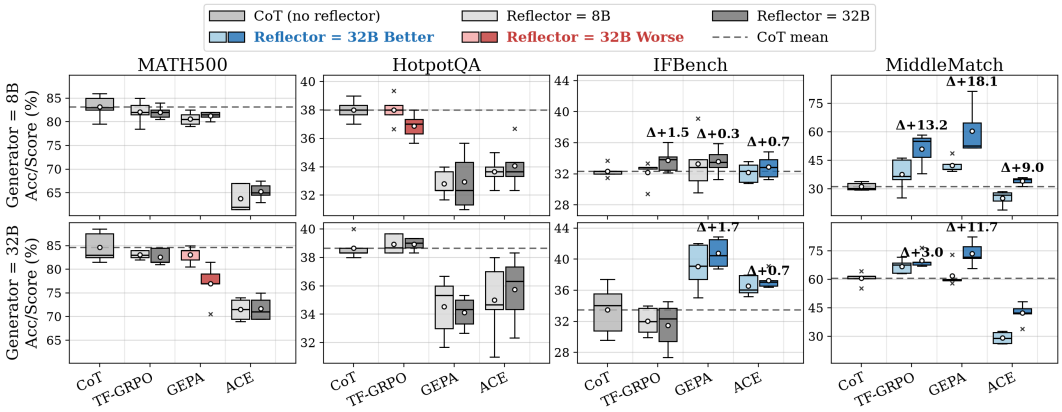

Figure 3: Ablation on the scale of the Reflector and Generator across four datasets. Scaling the reflector cannot rescue failures but further enhances existing improvements.

larger LLMs can engineer better context for smaller models or vice versa. We define the Reflector as the model that generates reflections to engineer the context, and the Generator as the model that executes tasks using the optimized context. Then, we conduct experiments on four datasets using GEPA, TF-GRPO, and ACE, varying the sizes of the Reflector and Generator between 8B and 32B. We observe that although larger LLMs engineer more effective context in most settings on IFBench and MiddleMatch, they can even perform worse than smaller models on MATH500 and HotpotQA. *(2) Bigger/Newer Models Do Not Necessarily Enable LCE Better Than Smaller/Older Ones.* In addition to scaling the reflector and generator separately, we jointly scale the self-reflecting model from Qwen3-0.6B to Qwen3-235B-A22B across four datasets (Qwen Team, 2025). Fig. 4 shows that model scaling primarily strengthens the underlying task solver, while the effectiveness of LCEs remain inconsistent. On MiddleMatch, both GEPA and TF-GRPO outperform CoT across model sizes, with particularly large gains at 235B. On IFBench, both methods match CoT at the 235B scale but fail to engineer useful context at smaller scales. In contrast, neither method yields reliable gains on MATH500 or HotpotQA: improvements at some smaller scales disappear or even reverse as the model grows. Thus, more parameters are insufficient to make LCEs consistently effective.

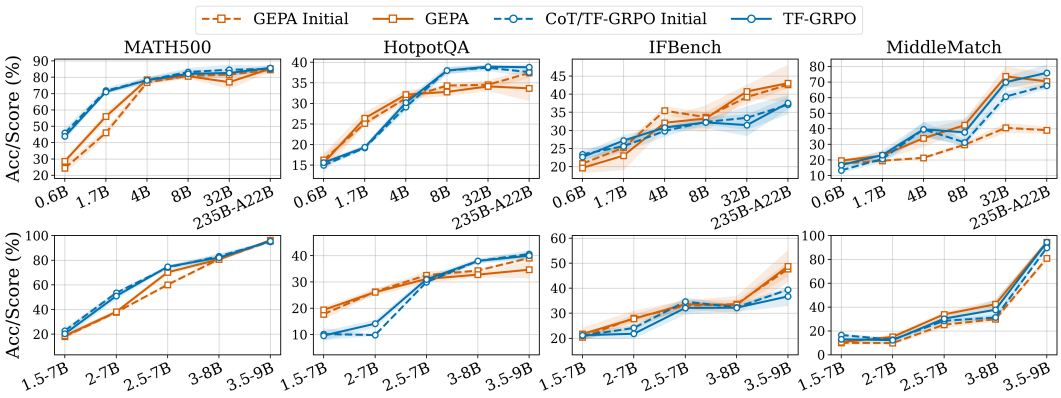

Figure 4: Scaling experiments. **Top:** Parameter-size scaling of self-reflection from 0.6B to 235B-A22B (Qwen3) across different datasets. **Bottom:** Comparison of model generations from Qwen1.5 to Qwen3.5. CoT and GEPA initial are the initials for TF-GRPO and GEPA, respectively.

We further investigate whether newer LLM generations lead to better LCE behavior. Specifically, we evaluate GEPA and TF-GRPO across five successive Qwen generations of comparable size, from Qwen1.5-7B to Qwen3.5-9B (Bai et al., 2023), on four datasets. As shown in Fig. 4, although baseline performance generally improves across generations, the gains from LCEs still exhibit no consistent trend. For example, GEPA shows smaller gains on HotpotQA as the LLM generation advances, while it continues to show substantial gains on MiddleMatch. Overall, these results provide no clear evidence that newer LLM generations systematically lead to better LCE behavior.

5 When & How Can LLMs Engineer Effective Context?

In addition to the aggregated Table 2, we will conduct qualitative and quantitative analysis to uncover the failure/success mechanisms of the LCEs.

When Do LCEs Work? In Table 2, which method is effective is not simple. We notice the largest gain (best LCE versus best baseline) in MiddleMatch, and the smallest gain in MATH500. In Ministral, LCEs are more useful than in Qwen3. Thus, we suppose both the data and the model play roles here. The intuition is that the LCEs learn from the model's failed rollouts and therefore the rollouts hinge not just on data but also on models. Specifically, we hypothesize that *LCEs are more effective when rollouts exhibit homogeneous failure modes*. To examine this hypothesis, we analyze the correlation between failure-mode diversity in the initial rollouts and performance gains *across all experimental settings* (model, dataset, methods). We characterize failure-mode diversity through the following steps:

(1) Embedding Rollout Failure-modes. For each failed reasoning rollout, we diagnose its cause by using an LLM. We instruct an LLM to summarize the cause mode, grounded in specific evidence and expressed in generalizable terms without instance-specific details (see exact prompt at Fig. 14). We then encode each failure mode as a vector $\mathbf{x}$ using the embedding model `all-mpnet-base-v2` (Song et al., 2020). The embedding model is widely used for short text; for example, ACE uses it to detect duplicate items in the playbook.

(2) Measuring Diversity. We randomly subsample N extracted failure-mode text samples and use the Vendi Score (Friedman & Dieng, 2023) to quantify failure-mode diversity. This score has been used to quantify pattern diversity, such as the diversity of generated images (Stein et al., 2023). The score ranges from 1 to N and can be interpreted as the effective number of distinct samples. The lower scores indicate more homogeneous failure modes. Detailed experimental setup is in Section B.2.

In Fig. 5, we present the performance gains versus Vendi scores and the mean/standard deviations of them on the axes. Performance gain is defined as the performance score difference between after and before LCE. We can see that different datasets present different diversity and the variance comes from models and LCE initial prompts. MiddleMatch is the least diverse task, and AppWorld is the most diverse given that its task is to handle interactions with an app. Our major findings are:

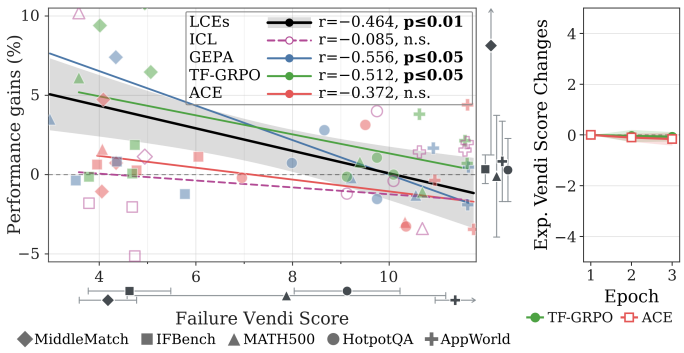

Figure 5: **Left:** Correlation between LCE performance gains and the *Vendi Score of failure modes*. Markers and error bars show means and standard deviations on both axes for LCEs. The significant negative correlation indicates that LCEs tend to achieve larger performance gains when failure modes are less diverse. **Right:** Changes in the *Vendi Score of experiences* across epochs. The small changes imply that LCEs struggle to distill experiences into shared principles.

(1) Less Diverse Failure Modes, Better The LCEs Work. The left figure clearly shows the negative correlation between diversity and performance gains across all LCE methods. GEPA and TF-GRPO present the statistical significance, but ACE does not. We conjecture that differences in context updates may explain this pattern. GEPA rewrites prompts, while TF-GRPO modifies or deletes selected experiences. These updates may lose useful knowledge when failure modes are diverse. ACE's incremental accumulation may better preserve such knowledge, making its performance gains less dependent on failure-mode diversity.

(2) In contrast, the ICL gains turn out to be irrelevant to the diversity. That means the relation hinges on the LLM operations, for example, reflection on failures, which are missing from ICL.

(3) In Fig. 5 (**Right**), we show how the Vendi Score of experiences (e.g., distilled insights or a bulleted playbook) extracted by ACE and TF-GRPO from training rollouts changes over epochs across different models and datasets. The Vendi Score changes very little across iterations. That means the LLMs are not able to reduce the diversity of experiences by iteratively optimizing context, which explains why LCEs do not perform well on highly diverse tasks.

Failure-mode Diversity Mainly Comes from Data. To identify the main sources of variation in failure-mode diversity across settings, we decompose the variance of Vendi Scores into dataset, model, and LCE main effects, together with residual variance. In Fig. 6, we show the variance decomposition proportions. A larger variance proportion for a factor indicates greater differences in failure-mode diversity across its categories. The calculation step of the variance decomposition is provided in Appendix B.3. In the figure, the small variance proportion attributed to the model suggests that failure-mode diversity varies relatively little across models, whereas the large proportion attributed to the dataset indicates substantial variation across datasets. This suggests that failure-mode diversity is driven primarily by the dataset, with some datasets naturally eliciting a broader range of failure modes. For example, the high similarity in MiddleMatch queries contributes to its low failure diversity: In MiddleMatch, questions are generated using a fixed template: *"Here's a list of names:* `\n\n{sequence of names}\n\n` *What is the single name that appears right between* `<name_1>` *and* `<name_2>`*?"* Such homogeneous data patterns make the reflection complexity lower and therefore lower failure mode diversity.

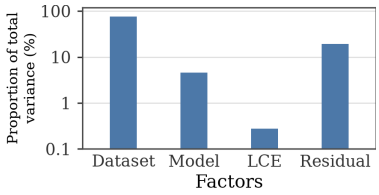

Figure 6: Ratio of total variance in Vendi Score of failure modes.

How Do LCEs Work? We have seen that LCEs work well on low-diversity datasets but it also happens, though not frequently, that LCEs can sometimes improve performance on high-diversity tasks. For example, in MATH500 with Gemma4 and HotpotQA with Min3, LCEs exhibit high variance in both performance gains and failure-mode Vendi Scores (Fig. 5). Here, we conduct case studies to understand how LCEs conquer these difficult tasks.

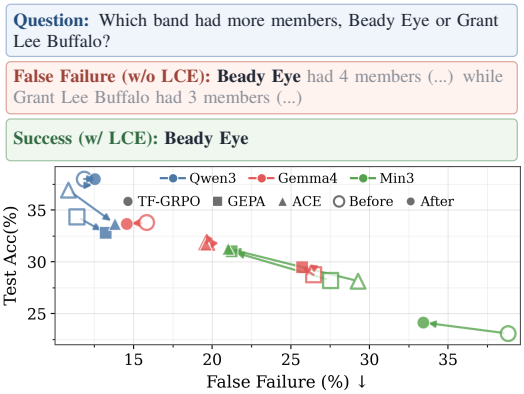

Figure 7: LCE behavior on HotpotQA. **Top:** Example of a verbose prediction from Min3 leading to failure. **Bottom:** LCEs reduce the false failure rate in Ministral alongside performance gains.

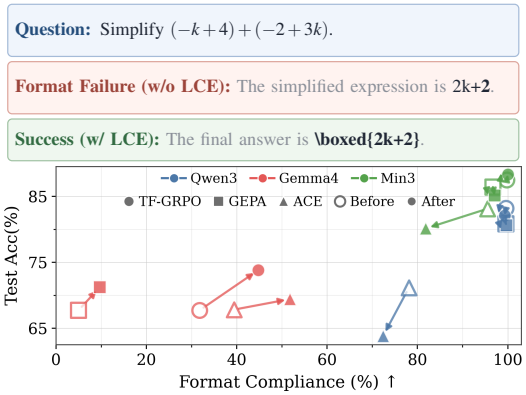

Figure 8: LCE behavior on MATH500. **Top:** Example of a format failure from Gemma4. **Bottom:** LCEs improve compliance with the required format in Gemma4 while improving performance.

Case Study: LCEs improve adherence to the task instructions. In multiple samples, we notice that the experiences extracted in LCEs are related to the output format or instruction details, less relevant to problem solution ideas. **(1)** In *HotpotQA*, by investigating the generated samples, we find that Min3 tends to generate verbose answers for problems, which sometimes cannot be identified as an answer by the evaluation. One example is given in Fig. 7 (Top), where we show that both answers contained the correct answer (bolded) but LCE makes the model produce the recognizable answer. For HotpotQA, the evaluation relies on exact matching of answers, and thus only the LCE answer is considered correct. Beyond the example, we found that the improvement of LCEs on HotpotQA is related to the reduction of such "False Failure". During optimization, LCEs repeatedly identify this verbosity failure and reinforce conciseness through instructions such as *"Answer only what the question asks. Avoid extraneous details."* Concretely, in Fig. 7 (Bottom), we show that the accuracy increase is related to the reduced False Failure rate, i.e., the proportion of predictions marked as incorrect by Exact Match that, however, actually contain the ground-truth answer. **(2)** For *MATH500*, we observe that Gemma4 frequently fails to follow the format requirement, such as "`Put the final answer in \boxed{...}`", as illustrated in Fig. 8 (Top). Thus, in Fig. 8 (Bottom), we report format compliance before and after optimization across models. We found that LCEs reinforce this instruction in the optimized context, improving format adherence and contributing to Gemma4's

Table 3: Performance under different experience utilization and retrieval settings. **R**, **E**, and ϕ denote training rollouts, knowledge extraction, and experience selection, respectively. k denotes the number of examples or experiences used, with var. indicating that the number varies depending on the optimized experiences. **Bold** and underlined values indicate the best and second-best results. Q, G, and M denote Qwen3, Gemma4, and Min3, respectively.

Method	R	E	ϕ	k	MiddleMatch^{Q}	MATH500^{G}	MATH500^{M}	IFBench^{G}
					ICL > LCE > CoT		LCE ≥ CoT > ICL	
CoT				0	$31.27_{\pm 2.10}$	$67.70_{\pm 1.04}$	$\underline{87.50}_{\pm 1.06}$	$39.76_{\pm 2.22}$
ICL	✓			3	$\mathbf{46.07}_{\pm 4.80}$	$\mathbf{77.90}_{\pm 3.54}$	$68.10_{\pm 2.41}$	$34.63_{\pm 2.17}$
GRPO-ICL RAG	✓			3	$\underline{44.00}_{\pm 3.46}$	$\underline{74.10}_{\pm 2.25}$	$26.50_{\pm 4.23}$	$37.41_{\pm 1.44}$
All-Exps Random	✓	✓		3	$32.00_{\pm 6.00}$	$69.70_{\pm 0.91}$	$87.40_{\pm 0.55}$	$\underline{40.92}_{\pm 2.07}$
Final-Exps	✓	✓	✓	var.	$37.73_{\pm 8.57}$	$73.80_{\pm 4.12}$	$\mathbf{88.30}_{\pm 1.15}$	$\mathbf{41.63}_{\pm 1.21}$

performance gains greatly. Details on calculating the false failure rate and format compliance are in Section B.4.

Case Study: TF-GRPO. To further investigate method-level failures, we conduct an additional ablation study with TF-GRPO, whose framework follows the general principles in Eq. (1) with clearly separated workflow stages and achieves the best performance in more settings in Table 2 across LCEs. In Table 3, we compare the ablations under two representative performance regimes based on whether LCE beats baselines: ICL > LCE > CoT and LCE ≥ CoT > ICL. Here, we add two inference variants that use a fixed number of experiences or in-context examples: (1) GRPO-ICL RAG (retrieval-augmented generation) retrieves in-context demonstrations used during TF-GRPO training, excluding query samples that can be solved in all rollouts. (2) All-Exps Random randomly samples experiences from all experiences produced by TF-GRPO (Final-Exps). The RAG variants retrieve items based on semantic similarity to the retrieval query using `Qwen3-Embedding-0.6B` (Zhang et al., 2025b), which supports encoding long in-context demonstrations. Beyond the inference variants considered above, we also evaluate several additional variants. As these experiments offer limited new insights, we report the full results in Section E.

Experience extraction (E) plays the most critical role. Across all four cases, either catastrophic performance loss (left two columns) or gain (right two columns) starts from the All-Exps Random, where Experience Extraction was first introduced to the ablation. That means when LLMs did not extract useful experiences from rollouts, the performance will not be improved over simply ICL of rollouts. Instead, when useful experiences are extracted (i.e., gains via **E**), the shorter experiences improve performance. We conjecture it is due to the smaller context length. The conjecture can be supported by our observation in Section F, that lengthy contexts can diminish the benefits of useful knowledge in ICL.

6 Conclusion

In this paper, we study LCEs across diverse tasks and LLM families. No method consistently outperforms even a simple CoT or ICL baseline, and larger or newer models do not reliably improve LCE effectiveness. We therefore conduct further experiments to investigate when and how LCEs are effective. We find that LCEs work better when facing less diverse failure modes, while such diversity varies substantially across datasets. Extracted experiences show no consistent reduction in diversity over training iterations, suggesting that current LCEs struggle to distill shared principles. Our ablations also show that converting training rollouts into experiences can discard useful information. Together, these findings identify limited extraction capability as a key bottleneck in turning diverse failures into broadly useful context.

Guidelines for future studies. Our work provides insights into the effectiveness and working mechanisms of LCEs. Yet, it is still open how to improve the LCEs. Our insights suggest that it is important to test methods under diverse models and datasets to fully demonstrate their generalization. LCE method development should pay more attention to overcoming the bottleneck in extracting useful experiences from diverse failure modes of reasoning. Moreover, future work may also explore incorporating knowledge on demand to reduce irrelevant content and control context length as the training data grows.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing, limited coding assistance, and synthetic data generation. Writing assistance was restricted to revising author-written text, with all final decisions made by the authors. We did not use AI to draft manuscript sections or construct mathematical proofs. Our study focuses on the analysis of existing LCEs and does not introduce new methods or theoretical results. For coding, the authors implemented the core experimental components and used AI to assist with routine extensions. For example, after implementing the Qwen3-8B experiments on HotpotQA, we used AI to adapt shell scripts for additional LLM families on HotpotQA. AI-assisted code underwent both automated checks assisted by AI and manual review by the authors. For synthetic data generation, we used AI to generate problems and reference reasonings based on existing problems and their reasonings, constructing the MATH-Sync dataset described in the Section F. These synthetic examples served as high-quality in-context demonstrations with reference reasonings. To reduce potential information leakage, we removed synthetic examples that were highly semantically similar to the original problems or shared their ground-truth answers. To assess answer correctness in the synthetic dataset, we asked the generating model to solve each synthetic problem repeatedly and checked whether its answers can consistently match the proposed reference answer. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including its text, claims, code, and datasets.

ETHICS STATEMENT

This work does not raise specific ethical concerns. Our contributions focus on evaluating the effectiveness of LLM Context Engineers and investigating the mechanisms underlying their successes and failures. All experiments are conducted on publicly available benchmarks using open-source models and do not involve human subjects, sensitive data, or privacy-related information. We are not aware of any potential conflicts of interest.

REPRODUCIBILITY STATEMENT

This work does not propose new methods or mathematical theory, but instead focuses on systematically evaluating the effectiveness of existing LLM Context Engineers and analyzing their mechanisms. We provide detailed experimental settings, including the configurations and prompts used for different tasks, as well as the procedures used in our analysis, in Section B.1. All evaluated LCE methods are based on their official implementations, and all datasets used in our experiments are publicly available. These details are provided to facilitate reproduction of our experimental results and analyses. Code will be released upon acceptance.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Art of Problem Solving. Aime problems and solutions, 2025. URL `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

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

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025b.

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

DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Prompt repetition improves non-reasoning llms. *arXiv preprint arXiv:2512.14982*, 2025.

Hanchen Li, Runyuan He, Qizheng Zhang, Changxiu Ji, Qiuyang Mang, Xiaokun Chen, Lakshya A Agrawal, Wei-Liang Liao, Eric Yang, Alvin Cheung, James Zou, Kunle Olukotun, Ion Stoica, and Joseph E. Gonzalez. Combee: Scaling parallel prompt learning for self-improving LLM agents. In *Third Conference on Language Modeling*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Georgios Paliouras. FiNER: Financial numeric entity recognition for XBRL tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022.

OpenAI. Introducing gpt-4.1 in the api, 2025. URL https://openai.com/index/gpt-4-1/.

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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

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

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems*, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/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.

Li Siyan, Vethavikashini Chithrra Raghuram, Omar Khattab, Julia Hirschberg, and Zhou Yu. PAPILLON: Privacy preservation from Internet-based and local language model ensembles. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, 2020.

Alessandro Sordoni, Eric Yuan, Marc-Alexandre Côté, Matheus Pereira, Adam Trischler, Ziang Xiao, Arian Hosseini, Friederike Niedtner, and Nicolas Le Roux. Joint prompt optimization of stacked llms using variational inference. In *Advances in Neural Information Processing Systems*, 2023.

George Stein, Jesse Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Ross, Valentin Villecroze, Zhaoyan Liu, Anthony L Caterini, Eric Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Dannong Wang, Jaisal Patel, Daochen Zha, Steve Y Yang, and Xiao-Yang Liu. Finlora: Benchmarking lora methods for fine-tuning llms on financial datasets. *arXiv preprint arXiv:2505.19819*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

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 Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. WebWalker: Benchmarking LLMs in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Yujiang Wu, Shanshan Zhong, Yubin Kim, and Chenyan Xiong. What generative search engines like and how to optimize web content cooperatively. In *The Fourteenth International Conference on Learning Representations*, 2026.

Guowei Xu, Mert Yuksekgonul, Carlos Guestrin, and James Zou. metatextgrad: Automatically optimizing language model optimizers. In *Advances in Neural Information Processing Systems*, 2025a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, 2025b.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *Advances in Neural Information Processing Systems*, 2025.

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.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025a.

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. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

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 *The Eleventh International Conference on Learning Representations*, 2023.

Runchuan Zhu, Bowen Jiang, Lingrui Mei, Fangkai Yang, Lu Wang, Haoxiang Gao, Fengshuo Bai, Pu Zhao, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. AdaptFlow: Adaptive workflow optimization via meta-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

A Evaluation Protocols in Representative Methods

Table 4: Reported benchmarks of evaluated context engineering methods.

Method	Reported Benchmarks
TextGrad	BBH, GSM8K (Suzgun et al., 2023; Cobbe et al., 2021)
GEPA	HotpotQA, IFBench, HoVer, PUPA, AIME25, LiveBench-Math (Yang et al., 2018; Jiang et al., 2020; Pyatkin et al., 2025; Siyan et al., 2025; White et al., 2025)
TF-GRPO	AIME24, AIME25, WebWalkerQA (Art of Problem Solving, 2025; Wu et al., 2025)
ACE	AppWorld, FiNER, Formula (Trivedi et al., 2024; Loukas et al., 2022; Wang et al., 2025)

B Experimental Setup

B.1 Experimental details for Benchmark and Scaling Experiments

We use Qwen3-8B (Qwen Team, 2025), Gemma-4-E4B-it (Gemma Team, 2026), and Ministral-3-8B-Instruct-2512-BF16 (Liu et al., 2026) for the main benchmark. Our scaling experiments examine both model size, from Qwen3-0.6B to Qwen3-235B-A22B, and model generation, from Qwen1.5-7B to Qwen3.5-9B. For each task, we use the same configuration across all models and LCEs in both the main benchmark and the scaling experiments. For single-step experiments, we set the maximum output length to $16{,}384$ tokens and the sampling temperature to $T = 0.7$. For AppWorld, we use the same sampling temperature and a maximum output length of $4{,}096$ tokens per step.

Methods. We adopt the official codebase of each method in our single-step experiment. GEPA (Agrawal et al., 2026) is officially supported in the DSPy package (Khattab et al., 2024), and we set `auto="heavy"` to maximize performance. For TextGrad (Yuksekgonul et al., 2025), we use the official package with 3 epochs and a batch size of 5. For TF-GRPO (Cai et al., 2025a), we follow the official setting with 3 epochs and a group size of 5, using the original prompt templates for the critic and summarizer; to align with other methods and maximize performance, we run validation after each epoch and select the best experience pool. For ACE (Zhang et al., 2026), we use the official implementation with a batch size of 5 and 3 epochs with bullet de-duplication. For AppWorld, we use the ACE and GEPA implementations provided in the official ACE codebase and adapt TF-GRPO to the same framework with the original prompts. We set the length limits for per-step execution outputs and the total message history to $10{,}000$ and $200{,}000$ characters, respectively. For ACE on AppWorld, we use the default settings: a batch size of 1 and a maximum of 5 Reflector refinement rounds. Each method is run with 5 random seeds ($\{0, 1, 2, 3, 4\}$). All models are served with vLLM on NVIDIA GPUs.

Benchmarks and Datasets. In Table 4, we show that existing methods are mostly evaluated on disjoint datasets. To understand the landscape, we select a variety of datasets with different reasoning patterns: *Reasoning* to find the solution (MATH500 (Lightman et al., 2024), HotpotQA (Yang et al., 2018)); *Retrieval* from long context (HotpotQA (Yang et al., 2018), MiddleMatch (Leviathan et al., 2025)); *Following Instructions* with unseen constraints (IFBench (Pyatkin et al., 2025)) and *Agentic Tasks* requiring multi-step planning and execution (AppWorld (Trivedi et al., 2024)). In these tasks, training-based methods like RLHF/RLVF have been shown to significantly improve LLMs' capabilities (Ouyang et al., 2022; Schulman et al., 2017; Rafailov et al., 2023). But little work was done to examine the LCEs in these tasks in a unified setting.

MATH500 (Lightman et al., 2024) is a 500-problem subset of competition-level mathematics drawn from the MATH benchmark (Hendrycks et al., 2021). Following prior settings, we use a 100/200/200 split for training/validation/test, and adopt Qwen-Verify to grade the results.

HotpotQA (Yang et al., 2018) is a multi-hop question-answering dataset in which the model must aggregate evidence across multiple documents. To unify the training framework across methods, we follow prior settings (Yuksekgonul et al., 2025) and provide the gold and distractor passages directly in the context, thereby removing the retrieval stage. Following GEPA (Agrawal et al., 2026), we use a 150/300/300 split and adopt exact match as the evaluation metric.

IFBench (Pyatkin et al., 2025) is an instruction-following dataset designed to evaluate whether LLMs can comply with user instructions rather than merely answer the underlying question. Following GEPA (Agrawal et al., 2026), we use a 150/300/294 split and adopt the official IFBench (Pyatkin et al., 2025) metric.

MiddleMatch (Leviathan et al., 2025) is a basic retrieval benchmark in which the model must identify the target name positioned between two given anchor names. We generate the data following the procedure of prior work, and use a 100/100/300 split with exact match as the evaluation metric.
AppWorld (Trivedi et al., 2024) is a benchmark of stateful tasks spanning multiple applications, where agents interpret user requests and interact with applications through APIs. We use the standard 90/57/168 training/development/test-normal split and adopt the official task goal completion as the evaluation metric.

Prompt Templates We make minor adaptations to each method's prompt to follow its original design while keeping the core instructions consistent across methods. Since the resulting prompts differ across methods and are therefore not directly comparable, we report each template's pre-optimization performance separately. Here we use MATH500 as a running example to introduce the fully initial prompt template of different methods.

```
Variables: Solve the following problem step by step. %%% will add specific prompt according
     to the datasets

Fixed: Put your final answer within \\boxed{{}}. %%% will be changed to other required
     format instructions according to the datasets
```

Figure 9: Prompt Template of TextGrad

```
Solve the following problem step by step. Put your final answer within \\boxed{{}}. %%%
     will be changed to task prompt with format instructions
{problem}

When solving problems, you MUST first carefully read and understand the helpful
     instructions and experiences:
{experiences}
```

Figure 10: Prompt Template of TF-GRPO.

```
Module: dspy.ChainOfThought(CoTSignature)

class CoTSignature(dspy.Signature):
    question = dspy.InputField()
    answer = dspy.OutputField(desc="Put your final answer within \\boxed{{}}.") %%% desc
     will be changed to format instructions

Variables: Given the fields `question`, produce the fields `answer`. %%% will be changed to
     task prompt
```

Figure 11: Prompt Template of GEPA.

```
You are an analysis expert tasked with answering questions using your knowledge, a curated
     playbook of strategies and insights and a reflection that goes over the diagnosis of
     all previous mistakes made while answering the question.

**Instructions:**
- Read the playbook carefully and apply relevant strategies, formulas, and insights
- Pay attention to common mistakes listed in the playbook and avoid them
- Show your reasoning step-by-step
- Be concise but thorough in your analysis
- If the playbook contains relevant code snippets or formulas, use them appropriately
- Double-check your calculations and logic before providing the final answer

Your output should be a json object, which contains the following fields:
- reasoning: your chain of thought / reasoning / thinking process, detailed analysis and
     calculations
- bullet_ids: each line in the playbook has a bullet_id. all bulletpoints in the playbook
     that's relevant, helpful for you to answer this question, you should include their
     bullet_id in this list
- final_answer: put your final answer within \\boxed{{}}. %%% will be changed to task
     prompt with format instructions


**Playbook:**
{}

**Reflection:**
{}

**Question:**
{}
```

```
**Answer in this exact JSON format:**
{{
  "reasoning": "[Your chain of thought / reasoning / thinking process, detailed analysis
     and calculations]",
  "bullet_ids": ["calc-00001", "fin-00002"],
  "final_answer": "[Your concise final answer here]"
}}

---
```

Figure 12: Prompt Template of ACE.

GEPA and TextGrad update the entire prompt by default, so we keep the format instructions fixed and optimize only the task prompt. For TextGrad and TF-GRPO, we append the task prompt and format requirements to the CoT prompt. Since we use the CoT module in GEPA, we replace the CoT.instructions in the DSPy module with the task prompt. ACE provides its own CoT template, so we only replace its final answer block with the task prompt and format requirements, as illustrated in our MATH500 example. We list the task-specific prompts and format requirements for each dataset in Fig. 13. For HotpotQA, we use the default template and append additional format instructions following GEPA's setup. For IFBench, we use the same instructions as GEPA. For MiddleMatch, we use the prompt from prior work (Leviathan et al., 2025) and add output-format requirements to extract the answer.

```
HotpotQA:
Task prompt: [None]
Format instructions: Directly provide the answer inside <answer> and </answer>, without
     detailed illustrations, for example, <answer> Beijing </answer>.

MATH500:
Task prompt: [None]
Format instructions: Put your final answer within \\boxed{{}}.

IFBench:
Task prompt: Respond to the query. Ensure the response is correct and adheres to the given
     constraints.
Format instructions: [None]

MiddleMatch:
Task prompt: [None]
Format instructions: Put the exact final name within \boxed{}, for example, \boxed{Michael
     Jackson}.
```

Figure 13: Task prompt and format instructions for different tasks.

For AppWorld, we adopt the initial prompts for GEPA and ACE from the ACE implementation and use GEPA's initial prompt for TF-GRPO.

B.2 Experimental details for Failure-Mode Extraction

Configurations We use Qwen3.8-27B (Qwen Team, 2026) to uniformly label the failure modes. For each experimental setting (model, dataset, method), we collect failed training trajectories generated using the initial context across five runs with different random seeds. We then combine these trajectories into a single set. Since the Vendi Score depends on sample size, we randomly sample the same number of trajectories, N, for each experimental setting to calculate the score. We repeat this sampling and calculation 100 times for each setting and report the mean score.

Prompt Template for Failure Mode Extraction The prompt templates used to extract failure modes for different tasks are shown in Fig. 14.

```
Diagnose the failure in this validated error trajectory.

Describe the smallest reusable failure mechanism that caused the error.

Requirements:
- Return exactly one short, specific sentence.
- Identify the failed operation, relation, constraint, evidence use or inference step.
- Use semantic roles instead of instance-specific names, dates, values, or positions.
- Do not add background, advice, headings, bullets, or generic evaluation.
- Do not state only the outcome, such as "incorrect answer" or "reasoning failure".
- Base the diagnosis only on the supplied trajectory and evaluator evidence.
- Failure may be caused by output that violates the task instructions, such as an
     unexpected format.
```

```
Question:
{question}

Task instruction used by the generator:
{task_instructions}

Input context: %%%Only provide for HotpotQA
{input_context}

Gold supporting-facts passages for only diagnosing evidence use: %%%Only provide for
    HotpotQA
{supporting_facts_text}

Reference answer: %%%Only provide for IFBench
{ground_truth}

Complete model reasoning or raw output:
{raw_output}

Extracted prediction following the task instructions:
{extracted_prediction}

Score:
{score}

Evaluator feedback: %%% Only provide for IFBench
{feedback}

Return only JSON with exactly one field:
{"failure_mechanism": "One short sentence describing the reusable failure mechanism."}
```

(a) Prompt for single-step experiments.

```
Diagnose the failure in this validated error trajectory.

Describe the smallest reusable failure mechanism that caused the error.

Requirements:
- Return exactly one short, specific sentence.
- Identify the failed operation, relation, constraint, evidence use or inference step.
- Use semantic roles instead of instance-specific names, dates, values, or positions.
- Do not add background, advice, headings, bullets, or generic evaluation.
- Do not state only the outcome, such as "incorrect answer" or "reasoning failure".
- Base the diagnosis only on the supplied trajectory and evaluator evidence.
- Failure may be caused by output that violates the task instructions, such as an
    unexpected format.

Task context:
{task_context}

Task instructions:
{method_specific_initial_instructions}

Message history:
{history_messages}

Ground truth code:
{ground_truth_code}

Test report:
{test_report}

Feedback:
{feedback}

Return only JSON with exactly one field:
{"failure_mechanism": "One short sentence describing the reusable failure mechanism."}
```

(b) Prompt for AppWorld.

Figure 14: Failure mode extraction prompt for different tasks.

Since LCEs often rely on the generator model for self-reflection, we also use each generator model to label the failure modes in its own rollouts. As shown in Fig. 15, the statistically significant negative correlation between Vendi Score and performance gains further suggests that LCEs construct more effective contexts when failure modes are less diverse.

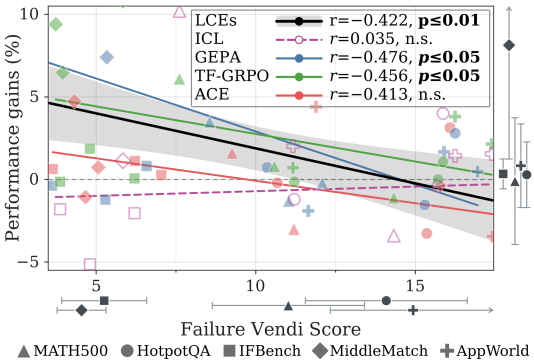

Figure 15: Correlation between LCE performance gains and the *Vendi Score of failure modes* that are labeled by generator models themselves. Markers and error bars show means and standard deviations on both axes for LCEs. The significant negative correlation indicates that LCEs tend to achieve larger performance gains when failure modes are less diverse.

B.3 Variance Decomposition of Vendi Scores

We denote the Vendi Score for dataset i, model j, and LCE method k by v_{ijk}. Let $\mathcal{Q} = \{\mathrm{D}, \mathrm{M}, \mathrm{L}\}$ denote the set of three factors, with n_{D}, n_{M}, and n_{L} denoting the numbers of datasets, models, and LCE methods, respectively. The total number of experimental settings is $n_{\mathrm{DML}} = n_{\mathrm{D}} n_{\mathrm{M}} n_{\mathrm{L}}$. For each factor $q \in \mathcal{Q}$, let μ_a^q denote the mean Vendi Score for a specific element in q indexed by a, averaged over the other two factors. With the overall mean $\mu = \sum_{i,j,k} v_{ijk} / n_{\mathrm{DML}}$, the contribution of each main effect to the total variance is $\sigma_q^2 = \sum_{a=1}^{n_q} \left(\mu_a^q - \mu\right)^2 / n_q, \quad q \in \mathcal{Q}$. The residual is $\sigma_{\mathrm{res}}^2 = \sum_{i,j,k} \left(v_{ijk} - \mu_i^{\mathrm{D}} - \mu_j^{\mathrm{M}} - \mu_k^{\mathrm{L}} + 2\mu\right)^2 / n_{\mathrm{DML}}$, which captures interactions among the factors. These components sum to the total variance: $\sigma_{\mathrm{total}}^2 = \sum_{q \in \mathcal{Q}} \sigma_q^2 + \sigma_{\mathrm{res}}^2$. We report the variance for each factor and residual as a proportion of the total variance, $\sigma_q^2 / \sigma_{\mathrm{total}}^2$, where $q \in \mathcal{Q} \cup \{\mathrm{res}\}$. Here, a larger variance proportion indicates greater differences in failure-mode diversity across the levels of the corresponding factor.

B.4 Experimental Details for Case Studies

When calculating the false failure rate, we directly measure the proportion of predictions that contain the ground-truth answer in the final response but are marked as incorrect. We remove some refusal responses, e.g., cases where the model cannot find the correct answer but repeats text containing the ground-truth answer, from false failure predictions by matching phrases such as *unable to determine* to avoid additional noise. For calculating the format compliance rate on MATH500, we measure the proportion of final answers from which our Qwen-Verify can successfully identify the `\boxed{}`.

C Pass@K and Pass^K Evaluation

LCEs Can Introduce New (But Possibly Bad) Knowledge. It was debatable whether the (none) improvement at Pass@1 actually reveals the gain of new knowledge/capability or just better exploits existing knowledge/capability (Yue et al., 2025). Obviously, the former is a more valuable gain than simply reshaping the output distribution, and is expected with a lot of data. To uncover the type of gain, we report Pass@K or Best@K (on IFBench) for $\mathrm{K} \in [1, 128]$ in Fig. 16. Pass@K is the fraction of problems solved by at least one of K generated samples and Best@K represents the expected maximum score across K generated samples. All Pass@K or Best@K results are computed from the same set of 128 generated samples per problem using the unbiased estimator $\text{Best@K} = \mathbb{E}_{\text{problems}}\left[\sum_{i=\mathrm{K}}^{128} \left(\binom{i-1}{\mathrm{K}-1} \Big/ \binom{128}{\mathrm{K}}\right) s_{(i)}\right]$, where $s_{(i)}$ is the i-th smallest score among the 128 generated samples. Pass@K is the special case of Best@K with binary scores $s_i \in \{0, 1\}$. The comparison between Pass@K and Pass@1 can indicate how the context impacts the performance. (1) For performance gain at Pass@1, if the gain is maintained with larger K, we say there is new

knowledge introduced. Otherwise, the learning is less valuable. (2) For performance loss at Pass@1, it is also worth examining the curve. If misinformation is introduced, the curve of learned context should be globally lower than the baseline.

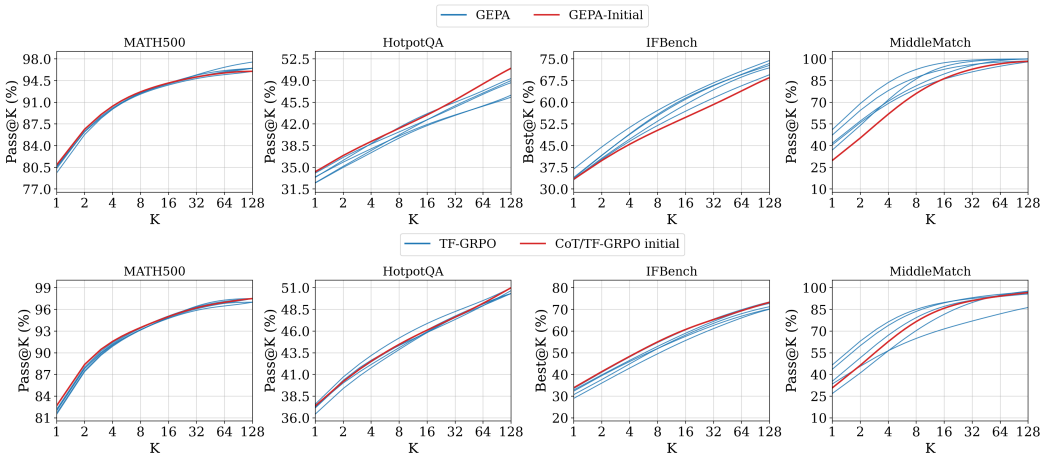

Figure 16: Pass@K or Best@K on four benchmarks. **Top row**: GEPA. **Bottom row**: TF-GRPO. Each figure compares the different optimized contexts by GEPA/TF-GRPO versus their initial performance.

In Fig. 16, we observe that LCEs can introduce both beneficial and harmful knowledge. On HotpotQA, GEPA's context underperforms its initial across all attempts, with the gap persisting up to Pass@128, indicating that LCEs can inject harmful knowledge that significantly biases the model and is difficult to repair by additional sampling. On MiddleMatch, both TF-GRPO and GEPA improve over their initial performance from Pass@1 to Pass@K in most attempts. Moreover, on IFBench and MATH500, some of GEPA's contexts match or slightly fall below their initials at Pass@1 but exceed them at larger K. These results suggest that the LCEs can also introduce useful knowledge hinges in the data and trial. Besides Pass@K, we also report Pass^K, which evaluates the model's ability to consistently produce correct answers across K attempts. Given 128 sampled responses for each problem, Pass^K is computed as Pass^K $= \mathbb{E}_{\text{problems}}\left[\binom{c}{\text{K}}/\binom{128}{\text{K}}\right]$, where c is the number of correct answers in the 128 attempts. For IFBench, we compute the expected score of K attempts as the Pass^K. As shown in Fig. 17, GEPA and TF-GRPO fail to achieve their initial performance from Pass^1 to Pass^K, indicating that the harmful knowledge will undermine the reliability of LLMs to generate correct answers.

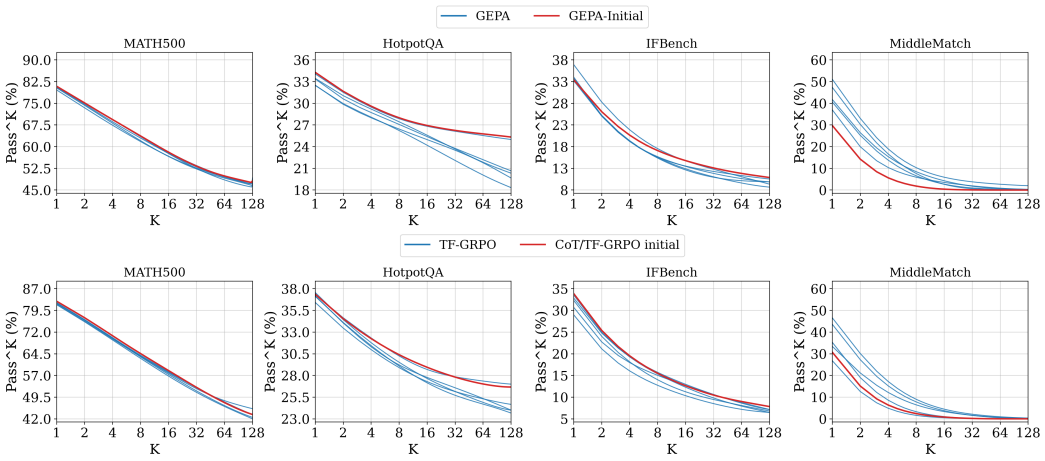

Figure 17: Pass^K Evaluation on four benchmarks. **Top row**: GEPA. **Bottom row**: TF-GRPO. Each figure compares the different optimized contexts by GEPA/TF-GRPO versus their initial performance.

Table 5: Performance under different experience utilization and retrieval settings. **R**, **E**, and ϕ denote training rollouts, knowledge extraction, and experience selection, respectively. k denotes the number of examples or experiences used, with var. indicating that the number varies depending on the optimized experiences. **Bold** and underlined values indicate the best and second-best results. Q, G, and M denote Qwen3, Gemma4, and Min3, respectively.

Method	R	E	ϕ	k	MiddleMatch^{Q}	MATH500^{G}	MATH500^{M}	IFBench^{G}
					ICL > LCE > CoT		LCE ≥ CoT > ICL	
CoT				0	$31.27_{\pm 2.10}$	$67.70_{\pm 1.04}$	$87.50_{\pm 1.06}$	$39.76_{\pm 2.22}$
ICL	✓			3	$\mathbf{46.07}_{\pm 4.80}$	$\mathbf{77.90}_{\pm 3.54}$	$68.10_{\pm 2.41}$	$34.63_{\pm 2.17}$
GRPO-ICL RAG	✓			3	$\underline{44.00}_{\pm 3.46}$	$\underline{74.10}_{\pm 2.25}$	$26.50_{\pm 4.23}$	$37.41_{\pm 1.44}$
All-Exps Random	✓	✓		3	$32.00_{\pm 6.00}$	$69.70_{\pm 0.91}$	$87.40_{\pm 0.55}$	$\underline{40.92}_{\pm 2.07}$
All-Exps RAG	✓	✓		3	$39.07_{\pm 11.87}$	$68.30_{\pm 2.89}$	$\mathbf{88.50}_{\pm 1.27}$	$38.78_{\pm 1.67}$
Final-Exps	✓	✓	✓	var.	$37.73_{\pm 8.57}$	$73.80_{\pm 4.12}$	$\underline{88.30}_{\pm 1.15}$	$\mathbf{41.63}_{\pm 1.21}$
Final-Exps RAG	✓	✓	✓	3	$38.53_{\pm 5.62}$	$68.80_{\pm 1.04}$	$87.90_{\pm 1.88}$	$40.00_{\pm 2.11}$

D Additional Observations in Scaling Experiments

Good Context Boosts Smaller Model to Match Larger Ones. On MiddleMatch, we observe a more consistent trend than other benchmarks: (1) Both methods with a 32B Reflector boost the performance of the 8B LLM, and GEPA even pushes the 8B LLM (42.33 → 60.40) to match the 32B CoT baseline (60.67), though with larger variance (12.93 on 8B vs. 3.32 on 32B). (2) Both methods with an 8B Reflector enhance the 32B LLM compared to their own initial prompts, while only TF-GRPO with an 8B Reflector outperforms the best CoT baseline.

Smaller Model can unlock the capability of Larger Model. On IFBench, GEPA generally benefits from scaling up the reflector (Fig. 3), whereas TF-GRPO under the 32B generator slightly underperforms. Remarkably, while the 8B and 32B models show similar CoT performance (32.35 vs. 33.47), using an 8B reflector alone is sufficient to unlock the 32B model's potential: GEPA pushes the 32B model to 39.08.

E Case Study on TFGRPO with more variants

We add two inference variants that use a fixed number of experiences or in-context examples: (1) All-Exps RAG retrieves experiences from the full set of extractor-proposed experiences accumulated during TF-GRPO training. (2) Final-Exps RAG retrieves experiences from the final experience pool produced by TF-GRPO (Final-Exps). The full results are shown in Table 5.

F Which Is Better Context: More High-Quality Experiences Or Fewer Experiences?

In-Context Demonstrations Can Mislead LLMs Into Making Errors. In settings where LCE ≥ CoT > ICL, replacing random demonstrations with retrieved training demonstrations reduces accuracy sharply from 68.10% to 26.50%. Our analysis identifies two main causes: (1) lengthy context impairs reasoning, and (2) demonstrations can distract models from the target problem. Training demonstrations are selected from problems with both successful and failed attempts. These problems are challenging for the model and may require more extensive reasoning to distinguish correct solutions from plausible alternatives, making GRPO-ICL demonstrations substantially longer than ICL demonstrations. To examine the effect of context length, we reduce the number of retrieved demonstrations to match the total context length of ICL, as shown in Table 6. This restores accuracy to approximately 68.10%, suggesting that excessive context length may contribute to the performance degradation, potentially by making it harder for the model to focus on critical reasoning steps. Meanwhile, we observe cases where Min3 answers a demonstration question instead of the target problem. Thus, we report the ratio of responses that answer an in-context demonstration rather than the target problem. As shown in Table 7, we find that Min3 is difficult to distinguish demonstrations

from the target problem, leading to these failures. LCEs may mitigate this issue by extracting knowledge from demonstrations, leaving only target question to answer in the context.

Table 6: Performance under different ICL settings with varying prompt lengths.

Method	k	Avg. Prompt Tokens	Accuracy
ICL	3	3,859	68.10
GRPO-ICL RAG	1	3,807	68.50
GRPO-ICL RAG	3	11,112	26.50

Table 7: Ratio of the predictions that incorrectly match the in-context demonstration answer.

Method	k	Answer Any ICL
ICL	3	15.10%
GRPO-ICL RAG	1	14.90%
GRPO-ICL RAG	3	64.30%

Controlled Study With Gold ICL. Although the above results suggest that lengthy context may be harmful, another possible explanation for the failures is that GRPO-ICL demonstrations are of lower quality. To separate the effects of context length and quality, we conduct an additional controlled study using ICL. Since directly evaluating the "quality" of an evolving prompt is difficult and inherently subjective, we therefore construct MATH-Sync by paraphrasing each MATH500 question conditioned on its ground-truth reasoning trace. We validate correctness via majority voting and remove duplicates (cosine similarity > 0.9 or identical final answers), yielding 318 (synthetic, original) pairs. For each test problem, we use its paired synthetic paraphrase as a *gold ICL demonstration*, which shares the underlying solution process with the test problem by construction and thus provides a controlled high-quality context. We compare three baselines: (i) the original CoT prompt with no demonstrations, (ii) ICL with irrelevant demonstrations sampled from unrelated problems, and (iii) gold ICL using the matched synthetic paraphrase. As shown in Fig. 18, gold ICL substantially outperforms both other baselines, confirming that high-quality demonstrations are genuinely valuable when they are concise.

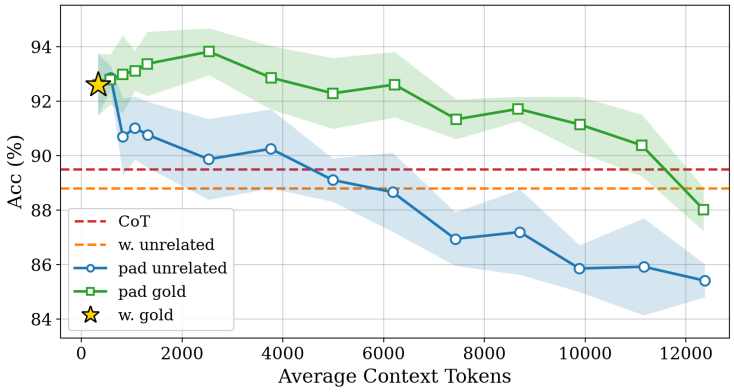

Figure 18: Ablation on context length. As high-quality context grows longer, performance eventually drops below both baselines.

Long Gold Context Is Worse Than Concise Context. We then progressively pad the gold ICL prompt in two ways: **(a)** by repeating the gold demonstration itself, holding quality constant while increasing length; and **(b)** by inserting additional ICL demonstrations sampled from unrelated problems, increasing length while diluting quality. In the early regime, both padding strategies further improve performance, consistent with the intuition that "more demonstrations help." However, as the prompt continues to grow, both curves eventually fall below all three baselines, including the original CoT prompt with no demonstrations at all. Crucially, this degradation occurs even under the repeated-gold setting, where every additional token comes from a high-quality, perfectly aligned demonstration. Length alone is therefore sufficient to drive performance below baseline. Together with the gold ICL result, this shows that the value of an injected experience is not monotonic in

either quality or quantity: high-quality information improves performance up to a point, after which length-induced degradation overwhelms the benefit.