Chain-of-Experience for Continual LLM Improvement

Anonymous ACL submission

Abstract

Humans continuously learn from experience, whereas conventional large language model (LLM) evaluations ignore the models' ability to improve through inference-time interaction. In this paper, we study how LLMs learn from iterative experience at test time, a setting we refer to as Chain-of-Experience (CoE), analyzing how models improve across repeated attempts with feedback. Through iterative interactions with self or environmental feedback, models accumulate experiential traces that inform future problem solving. This forms a continual improvement loop beyond zero-shot inference. We instantiate CoE with diverse feedback mechanisms, including model self-feedback and environmental signals such as correctness or public coding test pass rates, and evaluate across math, coding, and knowledge domains using 7 LLMs, including GPT-5, Gemini-2.5 Pro, and Claude-4.5 Sonnet. Our study show that by leveraging iterative experience consistently outperforms feedback-free baselines, achieving substantial performance gains with self feedback alone, alongside a 5.6% overall improvement and 19% lower API cost across tasks and models. We further observe a positive correlation between the LLM base ability and its improvement capacity, and show that models can still improve under weak or spurious feedback, with different feedback contributing to distinct improvement aspects and most gains emerging early in the experience iterations.

1 Introduction

Humans naturally and continuously learn from their experiences, each success or failure contributes to an evolving understanding that informs future decisions. Contemporary machine learning systems — modern large language models (LLMs) in particular (Achiam et al., 2023; Team et al., 2023; Grattafiori et al., 2024; Liu et al., 2024a; Yang et al., 2025) — behave quite differently: once trained, they are deployed in a fixed state, treating every inference as an isolated event and ignoring the rich feedback embedded in the problem solving process itself. In contrast, learning should be a continuous process, LLMs should be able to improve not only during training but also at test-time by iteratively interacting with their environment — processing feedback, engaging in reflection to constantly update their understanding, or shortly, improving from their own knowledge and experience (Silver and Sutton, 2025; Snell et al., 2025).

Inspired by successes in reinforcement learning, search-based methods explore a vast range of solutions. Heuristic strategies (Yao et al., 2023; Hao et al., 2023), including majority voting (Wang et al., 2022, 2024a) aim to identify higher-quality answers leveraging verifiers from candidates parallelly generated by LMs. Nevertheless, the model experience from these methods is still temporary, which is usually consolidated into a *single answer* and then discarded, leaving models to restart each problem without accumulated insight. An alternative paradigm is self-refinement, where models are enabled to iteratively improve through self-critique and correction (Madaan et al., 2023; Shinn et al., 2023), internalizing feedback within a single context. Extensions such as self-debugging (Chen et al., 2023) and Reflexion (Shinn et al., 2023) incorporate external signals like code execution signals, but they remain fragmented and shallow in the use of iterative experience.

Building on these existing attempts, we present a comprehensive analysis of LLMs learning from iterative experience, enabling models to improve during inference through extended environmental interactions, feedback processing, and self-reflection. We conceptualize this setting as Chain-of-Experience (CoE), wherein models engage in iterative problem-solving with feedback. Through this lens, we investigate a central question: *"How can LLMs evolve from accumulated experience*

with interactions and feedback to improve during test time?" In this paper, we use the CoE framework to systematically explore prolonged environment interactions across four types of feedback: none, model feedback, executor feedback for code tasks, and correctness feedback for general tasks. This analysis offers new insights into how LLMs evolve through iterative experience and demonstrates their meta-capacity to leverage feedback while generalizing across accumulated experience.

To evaluate CoE-based algorithms, we prompt seven state-of-the-art models, including GPT-5, o3, Gemini-2.5 Pro, and Claude-4.5 Sonnet, across math, coding, and knowledge domains. Extensive experiments over four feedback types—drawn from existing algorithms and our designed paradigm (*i.e.*, none, executor, model, and correctness feedback)—show that CoE with feedback consistently yields notable improvements (Figure 1). In particular, simple CoE methods substantially outperform test-time scaling approaches that rely on experience from other tasks (Suzgun et al., 2025; Zhang et al., 2025b), achieving an average of 7-9% gain over these algorithms with just self feedback (average 62.9% to 71.0%). Moreover, models leveraging iterative experience and feedback also reason more efficiently, delivering a 5.6% overall improvement with 19% lower API cost across all tasks and models. We further observe a clear positive correlation between learning gains from feedback and base ability, with an average Pearson correlation of +0.5 across five benchmarks, indicating that stronger models evolve more effectively from experience. Finally, our analyses reveal deeper behavioral insights: models remain robust under spurious or weak feedback, most gains occur early in the iterations, and distinct improvement trajectories emerge under different feedback under the CoE framework.

2 Related Work

Training-free Test-time Strategies. Large language models can reason without additional training via various test-time prompting strategies. Chain-of-Thought (CoT) elicits step-by-step reasoning and improves arithmetic, commonsense, and symbolic tasks (Wei et al., 2022), spawning a family of "Chain-of-X" methods (Yu et al., 2023; Li et al., 2023; Huang et al., 2023; Chia et al., 2023). Representative variants include contrastive CoT (Chia et al., 2023), least-to-most prompting (Zhou et al., 2022), task-specialized forms such as Chain-of-Explanation, Chain-of-Note, and Chain-of-Knowledge (Huang et al., 2023; Yu et al., 2023; Li et al., 2023), and Tree-of-Thought (ToT), which explores multiple reasoning paths via search and outperforms CoT on planning tasks (Yao et al., 2023). With the emergence of large reasoning models such as OpenAI's *o* series (Jaech et al., 2024) and DeepSeek R1 (Guo et al., 2025), verifier-based methods that select among parallel generations have regained attention, including step-level (Lightman et al., 2023; Wang et al., 2024b; Zhang et al., 2024) and output-level (Zheng et al., 2023a; Cai et al., 2024; Liu et al., 2025) verification. While effective as post-processing using external feedback (Liu et al., 2024b; Wang et al., 2025; Tu et al., 2025; Wang et al., 2024c), these methods lack the iterative loop for model evolving. Our CoE is also training-free, but differs by using feedback to drive iterative self-evolving during inference.

Learning from Experiences. Learning from experience underlies both human intelligence and AI systems. Reinforcement learning formalizes experience through policy gradients and actor–critic methods (Williams, 1992; Schulman et al., 2017), achieving success in games, robotics, and control (Silver et al., 2016), and more recently via post-training on online generations to improve alignment and reasoning (Bai et al., 2022; Guan et al., 2024; Shao et al., 2024; Yu et al., 2025). Beyond training, experience can accumulate during inference. For cross-task experience, Dynamic Cheat-Sheet (DC) (Suzgun et al., 2025), Agentic Context Engineering (ACE) (Zhang et al., 2025b), and related approaches (Zheng et al., 2023b; Wang et al., 2024d; Zhao et al., 2024) maintain persistent inference-time memories that distill reusable strategies, while agentic scaffolds enable collective experience sharing across agents (Tang et al., 2025; Chen et al., 2025a; Ouyang et al., 2025). For same-task experience, Reflexion (Shinn et al., 2023), Self-Refine (Madaan et al., 2023), Self-Debug (Chen et al., 2023), S* (Li et al., 2025), Iteration-of-Thought (Radha et al., 2024), and Re-Veal (Jin et al., 2025) iteratively refine outputs using self-feedback or execution signals, while recent pipelines further exploit offline model experience for agent improvement (Zhang et al., 2025a; Chen et al., 2025b). In contrast to prior work, we present a unified framework that treats a model's entire solving history as experience and systematically studies diverse feedback signals to enable contin-

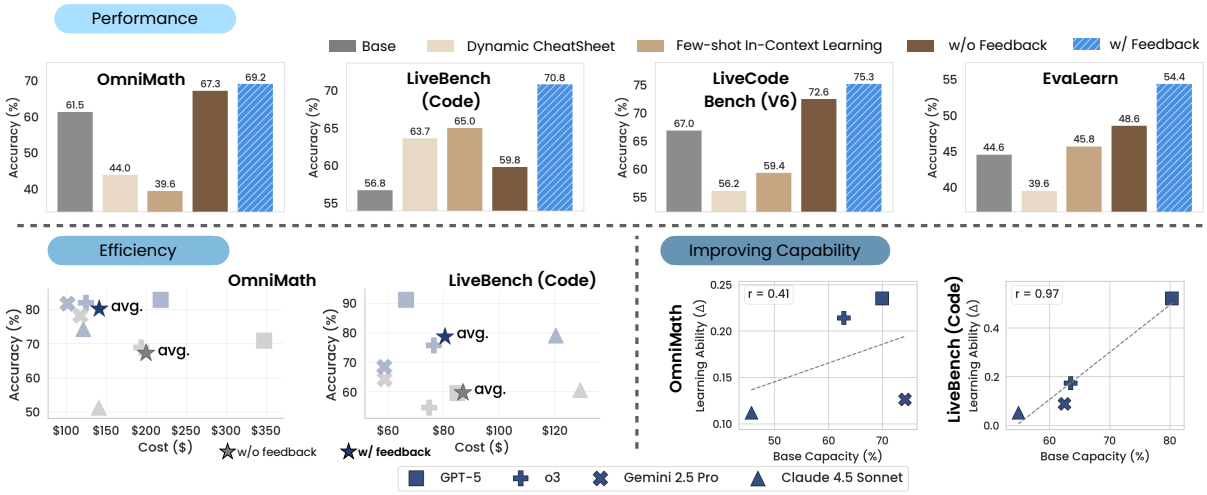

Figure 1: Summarized results on four benchmarks across math, code, and knowledge over four LLMs. Iterative problem solving under CoE provides three benefits: *upper:* by incorporating **feedback** in CoE, the average performance across four LLMs outperforms other test-time augmentations; *lower left:* models are capable of achieving the better performance with lower API cost with **feedback**; *lower right:* LLMs that perform better on the task display better improving capabilities through CoE with moderate to strong Pearson correlation. We present more explanations regarding baselines in Section 4.1.

ual improvement at test time.

3 Model Improvement via CoE

In this section, we first provide general concepts of the iterative problem solving setup, termed *Chain-of-Experience* (CoE), followed by a detailed discussion of four diverse feedback types to enhance model experience. Finally, we provide explanations on how we scale up the iteration of experience to probe model learning performance at test-time.

3.1 Overview

In the traditional question-answering setting (Sutskever et al., 2014; Raffel et al., 2020; Roberts et al., 2020; Brown et al., 2020), when given a question Q, large language models (LLMs) will generate a plausible response A sampled from the conditional distribution $P(A \mid Q)$. To extend this paradigm into the era of experience (Silver and Sutton, 2025), we incorporate an environment feedback variable F to represent the observable consequence or evaluation of responses when grounded in an interactive environment. Formally, the feedback f is sampled from the conditional distribution $f \sim P'(F \mid Q, A)$, where P' is modeled by an environment that can be instantiated as a coding execution environment, an internal world model of the agent, or even a real-world environment, providing generative (Ouyang et al., 2022; Mahan et al., 2024) or environment feedback (Shao et al., 2024; Madaan et al., 2023) for the given sequence.

In this paper, we investigate a paradigm that extends the single-turn formulation into a sequential decision process, where each response a_i at step i depends on the full history and corresponding environmental feedback of prior attempts ; we refer to this setting as *Chain-of-Experience* (CoE). The generative process is defined:

$$a_t \sim P(a_t \mid Q, e_0, e_1, \ldots, e_{t-1}) \quad (1)$$

where e_i is the i^{th} experience consists of (a_i, f_i).

3.2 Feedback Spectrum

To fully characterize how different forms of experience shape the model's iterative evolution, we categorize feedback along a spectrum of richness — from completely implicit to strongly explicit signals. Each feedback type corresponds to a specific instantiation of the environment's response, which in turn influences the next-step action via the CoE generative process. In general, the model can be updated according to

$$a_t \sim P(a_t \mid Q, (a_0, f_0), \ldots, (a_{t-1}, f_{t-1})),$$

where the function f_i is the feedback we employed at the $i-$th iteration. Below, we describe these four feedback types used in this study, together with their formal definitions.

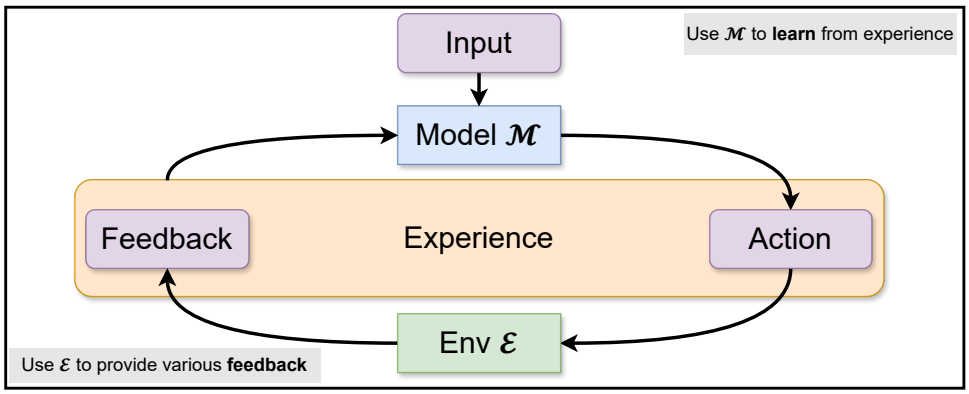

Figure 2: An overview of our studies iterative improvement loop for LMs. The model $\mathcal{M}$ learns by repeatedly interacting with the environment $\mathcal{E}$ (*e.g.*, model simulation or coding environment). In each cycle, $\mathcal{M}$ generates actions conditioned on the input and accumulated experience, then receives feedback from $\mathcal{E}$ to form a new experience. We investigate four specific feedback in this study: none, execution, model, and correctness feedback.

No feedback. The environment provides no evaluation or signal, *i.e.*, $f_i = \varnothing$. The experience reduces to $e_i = (a_i, \varnothing)$, the next action is sampled from

$$a_t \sim P(a_t \mid Q, a_0, a_1, \ldots, a_{t-1}),$$

meaning any improvement must arise from reflection on prior attempts, without external guidance.

Execution feedback. For tasks grounded in executable or interactive environments (*e.g.*, coding tasks with interpreters or unit tests), feedback is generated by running the model's response a_i inside the environment $\mathcal{E}$:

$$f_i \sim P'(F \mid Q, a_i) = \mathcal{E}(Q, a_i),$$

where f_i may include execution traces, error messages, runtime logs, or test-case outcomes.

Model feedback. A (possibly identical) auxiliary language model $\mathcal{M}_{\text{fb}}$ acts as a judge or critic:

$$f_i = \mathcal{M}_{\text{fb}}(Q, a_i),$$

where f_i may include textual critiques, preference scores, or structured evaluations. This enables refinement even in the absence of an external environment, relying purely on linguistic or preference-based signals.

Correctness feedback. When a domain-specific verifier is available, the environment supplies binary correctness signals:

$$f_i = \mathbf{1}\{a_i \text{ is correct}\} \in \{0, 1\}.$$

Such oracle-like information provides explicit fine-grained evaluation of success and failure. Although this type of feedback is often unrealistic in real-world settings, where ground-truth verification is costly or unavailable, still, we include it as a high-signal reference setting to approximate an upper bound on the benefits of iterative refinement.

3.3 Experience Scaling

Unlike prior approaches that scale along the token dimension (Wei et al., 2022; Wang et al., 2022), the reasoning-step dimension (Kojima et al., 2022), or rely on limited refinement steps (Madaan et al., 2023; Suzgun et al., 2025), we focus on a more realistic scaling paradigm that explicitly increases turns of environment feedback. Specifically, we scale the number of model–environment interactions, allowing language models to accumulate and evolve from experience. In this work, we attribute the underlying "improving algorithm" to models' inherent in-context learning capability, and systematically study how performance scales with both interaction quantity and feedback richness. We conduct scaling experiments under all feedback types within the CoE framework, enabling continual experience accumulation and evolution.

4 Experiments

In this section, we first outline experimental configurations, including benchmarks and models, then we provide an overview of baselines under the CoE context. Finally, we present comprehensive analysis on scaling up iterative experience with diverse feedback, highlighting three key findings.

4.1 Experiment Setup

Datasets. We focus on three different tasks: math, coding, and knowledge. More specific, we select two benchmarks for each task: AIME 2025 (Balunović et al., 2025), OmniMath (Gao

et al., 2024), LiveCodebench (V6) (Jain et al., 2024), LiveBench (Code) (White et al., 2024), EvaLearn (Dou et al., 2025), and GPQA Diamond (Rein et al., 2024). Detailed descriptions are in Appendix C.

Baselines. As for baselines, we examine model skills in either (1) utilizing different levels of built-in reasoning or (2) leveraging experience from previously solved problems. For controlling reasoning depth, OpenAI and Claude models can be tuned to produce varying amounts of reasoning tokens. For methods that absorb experience from prior examples, we adopt few-shot in-context learning (ICL) (Brown et al., 2020) as a standard baseline. For a more sophisticated approach, we select Dynamic CheatSheet (Suzgun et al., 2025) and Agentic Context Engineering (ACE) (Zhang et al., 2025b), which maintain a continually updated external memory of reusable strategies. Although follow-up works (Ouyang et al., 2025; Cai et al., 2025) introduce finer-grained refinements in a similar processing loop, we use these two as the representative baseline. We select the most $k \in [1, 5, 8, 12, 15, 20]$ relevant solutions for ICL, DC, and ACE to form their context, we present baseline details in Appendix D.

Models. We focus on the latest language models with inherent reasoning abilities from various developers to probe their improving capabilities during test-time: GPT-5 (OpenAI, 2025a), GPT-5-mini (OpenAI, 2025a), o4-mini (OpenAI, 2025b), o3 (OpenAI, 2025b), o3-mini (OpenAI, 2025b), Gemini-2.5 Pro (Comanici et al., 2025), and Claude 4.5 Sonnet (Anthropic, 2025). We run all experiments for three times and report the mean and standard deviation statistics. We present detailed prompting configurations in Appendix A.

4.2 Scaling with Experience

By incorporating experiences of varying levels into generations, we observe three notable findings spanning task performance, efficiency, and improving capability.

> **Findings 1**: Performance: The Chain-of-Experience setting with feedback boosts reasoning LLMs on various tasks.

LLMs equipped with feedback consistently outperform almost all baselines and settings. As shown in Figure 3, across six benchmarks, the Chain-of-Experience (CoE) paradigm—where models iteratively absorb and reuse prior feedback—delivers substantial performance gains (full results in Appendix G). With self feedback or executor/correctness feedback (as an upper bound), seven modern reasoning models achieve average improvements of 5.6% and 11.1% over their no-feedback counterparts, underscoring the value of explicit outcome-based signals for model refinement. For coding-centric tasks such as LiveBench (Code) and LiveCodeBench (V6), programmatic executor feedback derived from public test-case verification drives sharp accuracy gains of 8.6% on average (from 66.4% to 75.0%), while self-judgement feedback still provides a 7.0% lift. This indicates that models can internalize both fine-grained and abstract feedback cues into subsequent reasoning. On non-coding tasks (*e.g.*, AIME 2025, OmniMath, GPQA Diamond), the same trend holds: correctness feedback establishes an upper bound, and self feedback—though noisier—continues to foster improvement (*e.g.*, 75.1% $>$ 67.1% $>$ 62.5% w/o feedback).

In comparison, although ICL, DC, and ACE remain widely used, none demonstrates reliable scaling under advanced reasoning models. Averaged across six benchmarks in Table 2, ICL, ACE, and DC achieve only 62.1%, 64.0%, and 62.7% respectively, all trailing a simple without baseline (66.8%). In contrast, incorporating explicit feedback yields consistent gains: results with model self feedback reaches 71.0% (+7-9% over ICL/ACE/DC), while the best feedback signal further improves performance to 79.3%. Overall, these results proves that feedback-driven CoE acts as a more general and effective test-time scaling framework, enabling LLMs to autonomously improve across domains over no-feedback inference and other baselines.

> **Findings 2**: Efficiency: LLMs with feedback strikes a balance between performance and API calling cost.

As shown in Figure 4, incorporating feedback into the Chain-of-Experience framework not only enhances model performance but also improves efficiency by lowering the total API calling cost required to achieve strong results — as models with feedback tend to perform better with lower API costs (at the upper left of sub-figures). We present full results are in Appendix G. Across these tasks, most LLMs using feedback-based variants (blue objects) consistently dominate the no feedback base-

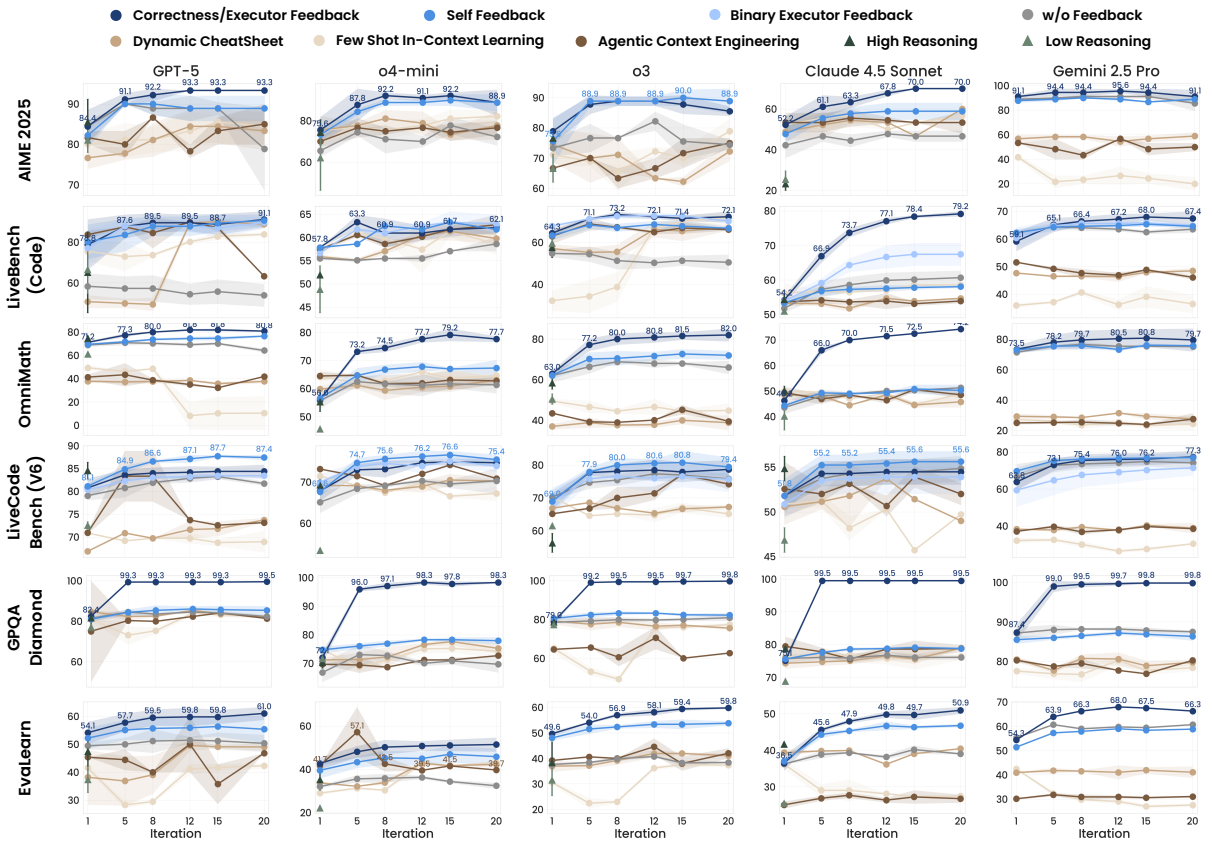

Figure 3: Results of five state-of-the-art LLMs on six benchmarks using different generation techniques. Models under CoE with different levels of feedback (correctness/executor, self, and binary executor feedback) generally perform better than the baseline strategies (no feedback, DC, ICL, and the ones with different reasoning efforts). Results are averaged over 3 runs and we shade the standard deviation with a lighter color and plot bars showing model performance under different reasoning efforts. The full results are in Appendix G.

lines (gray square), achieving higher accuracy at lower total price (except Gemini 2.5 Pro). We present the full results in Figure 12, Appendix G. For the two coding tasks, self feedback emerges as a cost-effective compromise: it captures the majority of the performance gain of executor (*i.e.*, 73.4% vs. 75.0%) while incurring 13.4% fewer API calls (*e.g.*, \$70.7 vs. \$81.6), and moreover, requires 20% less cost than the no-feedback counterpart. Similarly, on AIME 2025 and EvaLearn, self feedback provides more informative input with substantially lower API cost with 47.3% and 7.0% reductions across all seven LLMs, even compared with the no-feedback solution (*e.g.*, \$8.8 vs. \$4.6 on AIME 25 and \$325.3 vs. \$302.4 on EvaLearn), while still yielding an average 4.4% and 6.9% accuracy improvement, respectively. One exception is Gemini 2.5 Pro: its self-feedback produces much more output (*i.e.*, incurring higher costs) but yields smaller gains, suggesting Gemini may be less effective than other mainstream LLMs at judging and improving its own responses. These observations indicate that large models are capable of performing meaningful self-evaluation and internal calibration without the need for expensive external evaluators.

Findings 3: Improving Capability: LLMs that perform better on the task shows higher learning gain during test-time.

We calculate the improving capability of a model $\mathcal{M}$ using $\Delta_{\mathcal{M}} = \frac{S_{\text{max}} - S_{\text{base}}}{1 - S_{\text{base}}}$, where S_{base} denotes the model's initial zero-shot performance without feedback and S_{max} represents its peak accuracy achieved under our CoE setting with model self feedback. All numbers are averaged across three runs. In Figure 5, we observe a clear positive trend between base performance and improving capability across benchmarks. For instance, models on both coding tasks *e.g.*, LiveBench (Code) with $r = 0.97$) and LiveCodeBench (V6) ($r = 0.83$) show strong correlations, indicating that models with stronger initial reasoning ability tend to learn

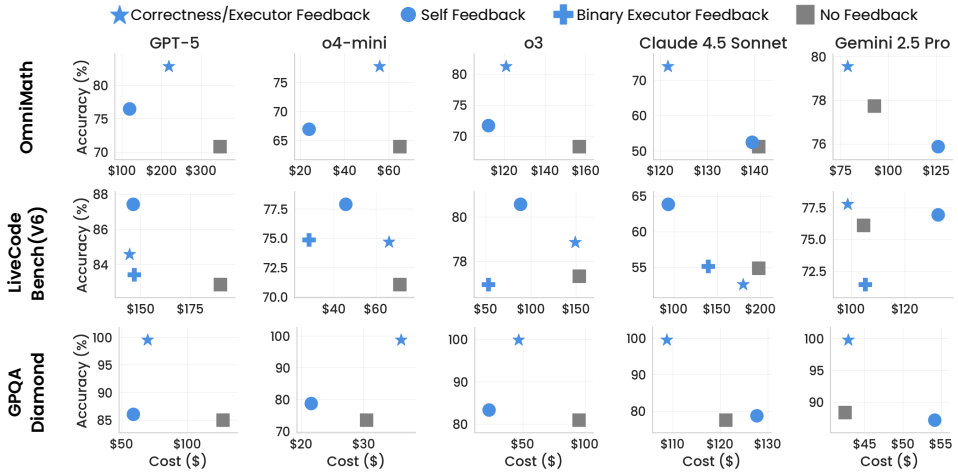

Figure 4: Total cost of each model over task completion vs. its best performance within 20 iterations. LLMs with feedback generally achieve higher scores with fewer costs (at the upper left), while iterative experience without feedback generally falls behind (at lower right). We provide full results over six benchmarks in Appendix G.

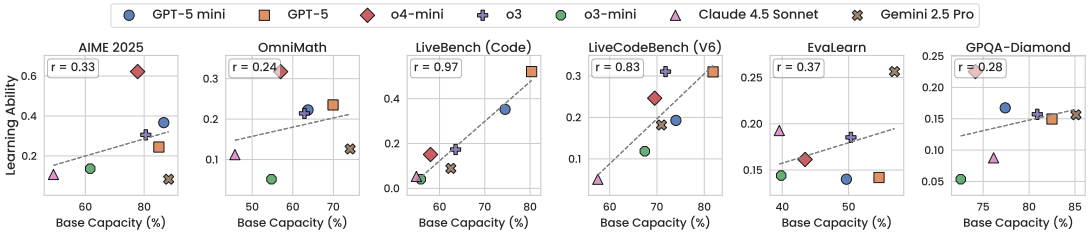

Figure 5: Zero-shot performance of models (Base Capacity) and the learning gain show positive Pearson correlations (r), indicating that better-performing LLMs possess greater improving capability. Scores are averaged across 3 runs.

more effectively from feedback. Although tasks like AIME 2025 ($r = 0.33$) and OmniMath ($r = 0.24$) show relatively weaker correlations, the overall trend remains consistent, with an average task-level Pearson correlation of 0.50. These findings suggest that learning from experience is an emergent property that scales with model capacity — larger and more capable LLMs not only start from a higher baseline but are also inherently better at digesting feedback, adapting to new information, and improving through iterative interactions.

5 Further Discussion and Conclusion

To further investigate LLMs' learning capacity under CoE, we conduct four ablations on the iterative problem-solving setup, including spurious feedback and improvement pattern analysis. We additionally study model behavior under varying feedback strengths (Appendix E.3) and extended experience iterations (Appendix E.2).

Learning from Spurious Feedback. To investigate the robustness of LLMs under spurious feedback, we design an experiment where models receive exclusively *incorrect* or *correct* feedback (*e.g.*, always stating "the answer is incorrect" or vice versa). We report the best model performance over 20 iterations in Table 1. This setup evaluates whether models can recover from (or even benefit under) misleading feedback signals. We find that spurious feedback generally degrades performance by average 7.6% on AIME 2025 and 2.6% on GPQA-Diamond. Yet, stronger models such as GPT-5 mini exhibit greater robustness, with only minor drops of 2.5% and 0.6%, compared to o4-mini's larger declines of 12.8% and 4.6%.

To further enhance reliability, we introduce *Selective Majority Voting* (SelMV-n), which aggregates final answers via majority voting among the first n valid attempts. This method mitigates randomness and reinforces consistent reasoning behaviors even when feedback is misleading. Interestingly, on GPQA-Diamond, SelMV with incorrect feedback surpasses model feedback by 0.9% (79.4% $\rightarrow$ 80.3%), while accuracies after SelMV improve by average 1.2% and 2.3% on AIME 2025 and GPQA-Diamond, respectively. These results underscore the robustness of reasoning LLMs, being capable of learning stable reasoning trajecto-

Figure 6: Percentages of different reasons for LLMs' improvement patterns from 6,630 incorrect to correct response pairs. We employ GPT-5 to conduct this automatic analysis.

ries despite adversarial or spurious signals. To demistify the role of different feedback, we provide more results about spurious and different levels of feedback in Appendix E.1 and E.3.

Analysis of Improvement Patterns. To better understand reasons for the improvement of LLMs through iterative feedback, we design the experiments to analyze the "why" behind each flip from incorrect to correct answer of LLMs. We collect 6,630 examples across all tasks and five models (*i.e.*, GPT-5, GPT-5 mini, o4-mini, o3, o3-mini) and leverage the latest GPT-5 model to analyze the cause. Full prompt for this task is available in Appendix H. We define four factors behind the improvement of LLMs: Feedback Fidelity for improving from trail feedback, Self Reflection for referring to self-reflection, Specification Recall for correcting based on the question and/or format requirements, Random for model improving from other reasons. We present the detailed criteria of these factors in Appendix E.4.

In Figure 6, we have several observations regarding the reasons of models' iterative improving. (1) LLMs can effectively leverage feedback for improvement. Across the five feedback settings, 47.7% of all improvements are attributed to feedback-related reasons, showing that large language models can meaningfully interpret and act on feedback signals to refine their subsequent outputs. (2) Coding feedback often drives specification-oriented refinement. In coding-related tasks, an average of 30.0% of improvements originate from specification recall, reflecting that coding tasks are highly format- and syntax-sensitive, encouraging models to focus on precise compliance with

Feedback	AIME 2025		GPQA Diamond	
	GPT-5 mini	o4 mini	GPT-5 mini	o4 mini
Self	93.3	91.1	79.9	78.8
SelMV Self	91.1	88.9	80.4	79.5
All *Correct*	90.0	73.3	79.3	75.8
SelMV *Correct*	**93.3**	73.3	79.3	76.3
Incorrect	91.7	83.3	79.3	72.7
SelMV *Incorrect*	89.7	**86.7**	**82.8**	**77.8**

Table 1: The best performance over 20 iterations under constant *correct*" or *incorrect*" feedback (*e.g.*, "the answer is correct"). Selective majority voting (SelMV) helps LLMs maintain performance. Results are averaged over 3 runs with best scores **emphasized**.

structural and formatting requirements. (3) Model-generated feedback elicits stronger learning effects. Improvements under model feedback show a higher feedback-related proportion compared to other feedback sources (*e.g.*, 58.7% > 41.1%), suggesting that self-generated feedback tends to be more detailed and contextually aligned, thereby enhancing models to leverage it for improvement.

Conclusion We present a comprehensive analysis of Chain-of-Experience (CoE), showing that LLMs can improve during inference through iterative feedback and accumulated experience. Across math, coding, and knowledge tasks, methods in CoE consistently enhances performance and efficiency, demonstrating the effectiveness of feedback-driven test-time learning. Our analysis also reveals a positive correlation between model ability and improvement capacity, and shows that most gains emerge early, even under weak or spurious feedback. Finally, we analyze various model improvement patterns during this iterative process.

6 Limitations

Our evaluation focuses primarily on math, knowledge, and coding benchmarks. While these domains offer controlled settings to probe iterative improvement, there are interaction-intensive scenarios where experience unfolds over long horizons (Jimenez et al., 2023; Yao et al., 2024), to which the CoE paradigm should naturally extend. In addition, we do not update model parameters in this study. This choice isolates the Chain-of-Experience mechanism as a test-time paradigm, but it also means that the observed improvements arise from contextual reuse of experience rather than true learning; incorporating weight updates to internalize experience remains an important next step toward training models with persistent the "learning-from-experience" abilities.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2025. Introducing Claude Sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5. Accessed: 2025-11-17.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, and 1 others. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.

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

Silin Chen, Shaoxin Lin, Xiaodong Gu, Yuling Shi, Heng Lian, Longfei Yun, Dong Chen, Weiguo Sun, Lin Cao, and Qianxiang Wang. 2025a. Swe-exp: Experience-driven software issue resolution. *arXiv preprint arXiv:2507.23361*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Sara Cao, Yuanhao Xiong, Haibo Tong, and 1 others. 2025b. Scaling agent learning via experience synthesis. *arXiv preprint arXiv:2511.03773*.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, and 1 others. 2025c. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*.

Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. 2023. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Shihan Dou, Ming Zhang, Chenhao Huang, Jiayi Chen, Feng Chen, Shichun Liu, Yan Liu, Chenxiao Liu, Cheng Zhong, Zongzhang Zhang, and 1 others. 2025. Evalearn: Quantifying the learning capability and efficiency of llms via sequential problem solving. *arXiv preprint arXiv:2506.02672*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Melody Y Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Helyar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, and 1 others. 2024. Deliberative alignment: Reasoning enables safer language models. *arXiv preprint arXiv:2412.16339*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in

llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Fan Huang, Haewoon Kwak, and Jisun An. 2023. Chain of explanation: New prompting method to generate quality natural language explanation for implicit hate speech. In *Companion proceedings of the ACM Web conference 2023*, pages 90–93.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Yiyang Jin, Kunzhao Xu, Hang Li, Xueting Han, Yanmin Zhou, Cheng Li, and Jing Bai. 2025. Reveal: Self-evolving code agents via iterative generation-verification. *arXiv preprint arXiv:2506.11442*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E Gonzalez, and Ion Stoica. 2025. S*: Test time scaling for code generation. *arXiv preprint arXiv:2502.14382*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024b. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, and 1 others. 2025. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024. Generative reward models. *arXiv preprint arXiv:2410.12832*.

OpenAI. 2024. New embedding models and API updates. https://openai.com/index/new-embedding-models-and-api-updates/. Accessed: 2025-11-17.

OpenAI. 2025a. Introducing GPT-5. https://openai.com/index/introducing-gpt-5/. Accessed: 2025-11-17.

OpenAI. 2025b. Introducing OpenAI o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/. Accessed: 2025-11-17.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, and 1 others. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*.

Santosh Kumar Radha, Yasamin Nouri Jelyani, Ara Ghukasyan, and Oktay Goktas. 2024. Iteration of thought: Leveraging inner dialogue for autonomous large language model reasoning. *arXiv preprint arXiv:2409.12618*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and 1 others. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

David Silver and Richard S Sutton. 2025. Welcome to the era of experience. *Google AI*, 1.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

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

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, and 1 others. 2025. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Haoqin Tu, Weitao Feng, Hardy Chen, Hui Liu, Xianfeng Tang, and Cihang Xie. 2025. Vilbench: A suite for vision-language process reward modeling. *arXiv preprint arXiv:2503.20271*.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024a. Boosting language models reasoning with chain-of-knowledge prompting. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4958–4981.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, and 1 others. 2025. Visualprm: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024c. Helpsteer 2: Open-source dataset for training top-performing reward models. *Advances in Neural Information Processing Systems*, 37:1474–1501.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024d. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, and 1 others. 2024. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. tau -bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *arXiv preprint arXiv:2311.09210*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025a. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, and 1 others. 2025b. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023b. Synapse: Trajectory-as-exemplar prompting with memory for computer control. *arXiv preprint arXiv:2306.07863*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Detailed Experimental Setup

Different Reasoning Efforts. For OpenAI models, we employ the default 'low' and 'high' in the reasoning_effort parameter to twitch models' reasoning level. For Claude 4.5 Sonnet, we disable the thinking mode and set the thinking budget to 10,000 as the low and high reasoning variants, respectively.

Decoding Parameters. For OpenAI models except GPT-OSS-120B, we use the default decoding parameter with temperature set to 1.0. For Gemini, Claude, and GPT-OSS-120B models, we use a temperature of 0.2 for decoding.

B Averaged Results

We present the averaged best scores over 20 iterations of different methods in Table 2. From the table, we can clearly draw the conclusion that feedback helps LLMs perform better on all six benchmarks, while existing self-improving algorithms (*i.e.*, ACE, DC) do not perform decently on this testing suite.

C Evaluated Benchmarks

We present detailed descriptions of our evaluated benchmarks below:

- AIME 2025 (Balunović et al., 2025) is a challenging and universal math benchmark consists of 30 cases from AIME in 2025.
- OmniMath (Gao et al., 2024) is a universal olympiad level mathematic benchmark consists of a total of 4,428 problems, and we sample 200 examples across all difficulties for evaluation.
- LiveCodebench (V6) (Jain et al., 2024) is an evolving benchmark for challenging code generation. We select its latest version (V6) alone with a total of 175 samples.
- LiveBench (Code) (White et al., 2024) is originated from a living benchmark spans across six aspects. We select the coding aspect, comprising 128 samples for evaluation.
- EvaLearn (Dou et al., 2025) is the first benchmark that evaluate the experience learning abilities of language models, which includes total 648 examples.
- GPQA Diamond (Rein et al., 2024) is a challenging multiple-choice question set in biology, chemistry, and physics, authored by PhD-level experts. It consists of 198 examples in total.

D Baselines

Built-in Reasoning: A native test-time scaling mechanism implemented in OpenAI models allows the reasoning effort to be adjusted between low and high. We treat this as a built-in and straightforward scaling baseline and present further details in Appendix A

Few-shot In-Context Learning (ICL). In few-shot ICL, each demonstration is a previously solved question–answer pair. As the model processes tasks sequentially, all past pairs are stored in an experience buffer. For a new question, we retrieve the k most similar past questions using embeddings from OpenAI's `text-embedding-3-large` model (OpenAI, 2024), and include their original question–answer formats as demonstrations, followed by the new question. When fewer than k examples are available, all prior examples are used. We evaluate ICL with $k \in [1, 5, 8, 12, 15, 20]$.

Dynamic CheatSheet (DC) (Suzgun et al., 2025). Dynamic CheatSheet (DC) is a test-time learning method that maintains an adaptive external memory of reusable strategies or code snippets distilled from prior solutions. As new problems are solved, DC summarizes high-level strategies from selected past tasks—using ground-truth answers for clean experience curation—and stores them as structured cheatsheets. Past tasks are retrieved using the same similarity-based retrieval as few-shot ICL; when fewer than k tasks exist, all available examples are used. Consistent with ICL, cheatsheets are synthesized from the most recent $k \in [1, 5, 8, 12, 15, 20]$ relevant solutions.

Agentic Context Engineering (ACE) (Zhang et al., 2025b). Agentic Context Engineering (ACE) is a context adaptation framework that treats prompts as evolving playbooks rather than static demonstrations. It incrementally distills reusable strategies and domain insights through a generate–reflect–curate process, producing localized context updates that preserve prior knowledge and avoid monolithic rewrites. Similar to few-shot ICL, ACE retrieves relevant past trajectories and integrates insights from the most recent $k \in [1, 5, 8, 12, 15, 20]$ trajectories into its evolv-

Method	AIME 2025	LiveCodeBench (V6)	LiveBench (Code)	OmniMath	GPQA Diamond	EvaLearn
ICL	71.83%	62.50%	65.46%	53.12%	78.45%	40.99%
ACE	71.98%	66.94%	69.38%	50.33%	76.58%	42.54%
DC	73.33%	63.59%	68.58%	48.64%	79.56%	42.68%
w/o Feedback	77.78%	72.57%	60.16%	65.17%	80.02%	44.91%
Reasoning-high	69.05%	70.63%	55.46%	61.81%	76.21%	39.58%
Reasoning-low	60.48%	61.03%	55.38%	50.60%	72.92%	29.34%
Binary-Executor	–	72.90%	71.65%	–	–	–
Self	82.22%	**75.69%**	69.94%	67.52%	81.03%	51.73%
Correctness/Executor	**89.05%**	74.50%	**75.78%**	**79.61%**	**99.52%**	**57.05%**

Table 2: Average performance comparison (%) across different LLMs on different datasets. For baselines, ICL, ACE, DC stands for few-shot in-context learning, agentic context engineering, and dynamic cheatsheet, respectively.

ing playbook.

E Full Discussions

In this section, we present the full version of different discussions in

E.1 LLM with All Spurious "Correct" Feedback

In Figure 7, we further present model performance under two extreme conditions: receiving uniformly "correct" feedback (*e.g.*, the answer is correct) and the SelMV-augmented results. We observe that although performance initially drops after exposure to such incorrect feedback, which suggests temporary confusion in adapting to inconsistent supervision. The models quickly recover and even improve as they adapt to the underlying pattern. Interestingly, both GPT-5 mini and o4-mini exhibit larger gains when exposed to entirely incorrect feedback, as such feedback compels the models to re-evaluate their reasoning and verify their outputs. In contrast, consistently positive feedback tends to induce overconfidence, misleading the models into accepting their initial responses without critical reassessment. This observation suggests that, paradoxically, constructive noise (in the form of seemingly negative feedback) can sometimes stimulate deeper reasoning and enhance robustness in iterative test-time learning.

E.2 Extended Rounds of Iterations

To further probe the learning capacity of LLMs, we extend the number of experience iterations from 20 to 50, as shown in Figure 8. Across both AIME 25 and OmniMath, we observe that most performance gains occur within the first 20 iterations, while later stages yield only marginal improvements (*e.g.*, average 16.7% $>$ 2.2% on AIME 25; 21.2% $>$ 3.5% on OmniMath). This trend consistently holds across different models, suggesting that LLMs quickly internalize and consolidate the useful feedback signals in the early stages, after which learning saturates. These results highlight that the majority of test-time learning under CoE happens rapidly — shorter adaptation loops in our CoE are sufficient for most reasoning tasks, with less significant returns from prolonged experience accumulation.

E.3 Feedback Strength

For experience with model feedback, one intuitive exploration is to design CoE with different feedback providers. Specifically, we cluster GPT-5 and GPT-5 mini as a pair and allow each to serve as a feedback generator for the other on a mathematical and a coding task, as illustrated in Figure 9. Interestingly, external model feedback proves to be highly effective on these two benchmarks, where it even slightly surpasses correctness-based/executor feedback. For instance, GPT-5 mini achieves a peak accuracy of 94.4% with model feedback, compared to 93.0% with correctness feedback; similarly, GPT-5 reaches 93.3% vs. 92.2%, showing that high-quality model-generated judging signals can substitute explicit correctness supervision. On the more challenging OmniMath benchmark, GPT-5 continues to help its mini variant outperform its self-feedback baseline. However, because GPT-5 mini starts with only 62.5% zero-shot accuracy, it is unable to provide sufficiently reliable feedback to improve GPT-5, resulting in inferior performance relative to correctness feedback. Moreover, both GPT-5 and GPT-5 mini underperform their AIME 2025 results ($\geq$60% vs. $\geq$80%), and correctness feedback remains clearly superior: GPT-5 gets 82.8% with correctness feedback but only 74.5% with external model feedback. Overall, these find-

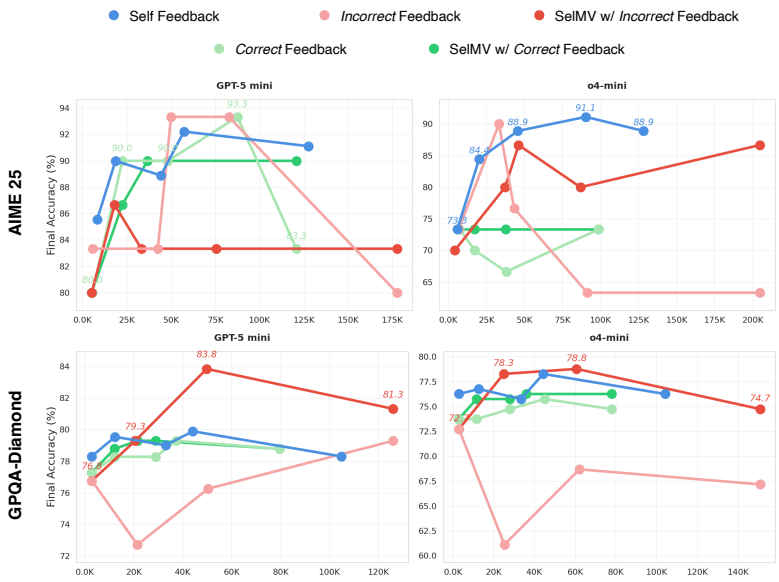

Figure 7: Model performance using constant "incorrect" and "correct" feedback. By leveraging the selective majority voting, LLMs show decent performance when facing spurious feedback on math tasks.

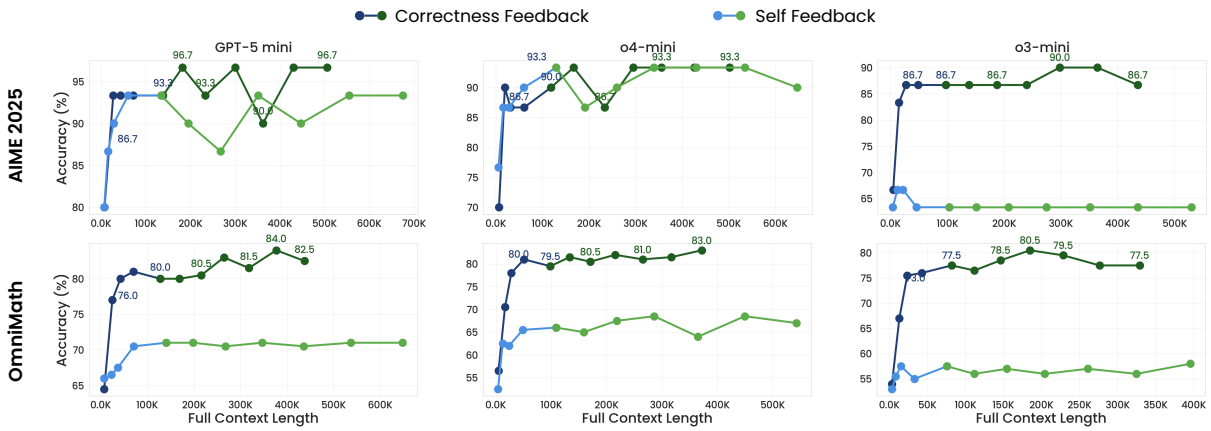

Figure 8: Performance of GPT-5 mini, o4-mini, and o3-mini with extended iterations of experience to 50 on math tasks. We mark performance points within the first 20 iterations of experience in blue, and those from iterations 20 to 50 in green.

ings highlight a consistent trend: feedback quality correlates with the verifier's base ability on the task, and model-generated feedback becomes competitive with correctness supervision only when the verifier is sufficiently strong, suggesting a practical threshold for deploying model-as-judge in iterative experience frameworks.

E.4 Analysis of Improvement Patterns

We present the detailed criteria that we used for classifying the

- Feedback Fidelity: Improvements directly driven by external feedback, where the model explicitly incorporates provided guidance or corrections into its next response.
- Self Reflection: Improvements arising from the model's own reasoning, identifying and correcting errors with little reliance on external feedback.
- Specification Recall: Adjustments motivated by task instructions or formatting requirements, as the model re-aligns with the original question or output schema.
- Random: Changes with no identifiable cause, typically minor rewording or stylistic variations unrelated to feedback or specification.

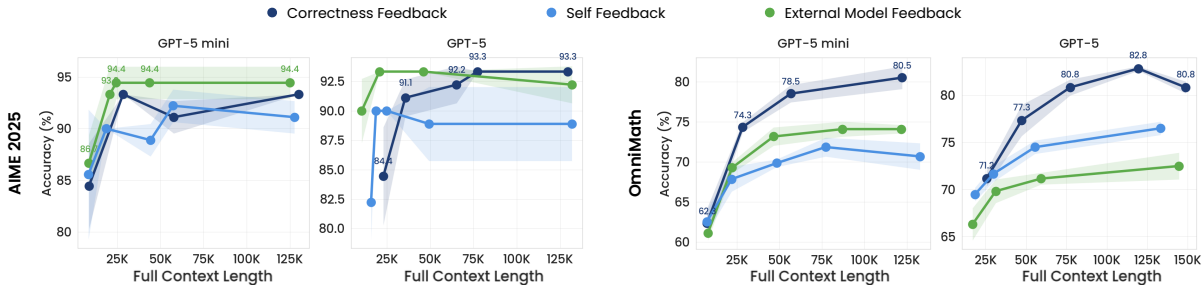

Figure 9: Model performance of GPT-5 and GPT-5 mini with external model feedback on two math tasks.

F BrowseComp-Plus

We also report model performance on BrowseComp-Plus (Chen et al., 2025c). It is a benchmark to evaluate deep research systems, isolating the effect of the retriever with a local database. It is sourced from the BrowseComp (Wei et al., 2025) and we sample 200 examples to accelerate the evaluation.

In Figure 10, we observe that unlike coding and math tasks, BrowseComp-Plus requires knowledge beyond the scope of the models' training data. Consequently, for most models, incorporating self feedback leads to a performance decline compared to the no-feedback setting, highlighting the limitation of relying solely on self feedback in out-of-distribution knowledge scenarios.

G Full Results of Performance and Efficiency

We present full results of model performance (Figure 3) and API costs (Figure 12) regarding seven reasoning LLMs over six benchmarks.

H Prompt for Improvement Pattern Analysis

DUAL-AXIS LIFT/CHANGE ATTRIBUTION JUDGE
You are a **dual-axis attribution judge**. Your job is to identify both **why** the model changed and **what** specifically changed between two consecutive attempts on the same problem. Do **not** decide if the solution is correct overall — correctness labels are provided. Instead, attribute the observed change along two orthogonal dimensions:
* **Change Driver (WHY)** – the motivation or trigger behind the change. * **Change Manifestation (WHAT)** – the concrete locus or type of modification made.
—
I. Change Driver (WHY the change occurred)
These categories capture *the source or motivation* of the update in the second attempt.
1. **Feedback Fidelity** – The model directly *uses* the provided feedback to modify its output. *Signals:* Edits match failing test or critique location; added clause mirrors feedback.
2. **Self-Reflection / Internal Reasoning** – The model self-identifies an error or improvement without (using) explicit feedback or tests a different approach to see if it performs better. *Signals:* "I realized…," "previously I miscalculated…," or internally consistent reformulation not prompted by feedback or when there's no feedback.
3. **Specification Recall / Compliance Awareness** – The model remembers or re-aligns with task instructions or formatting requirements. *Signals:* Adds "FINAL ANSWER:" wrapper, adheres to requested schema, restores omitted step explicitly mentioned in prompt.

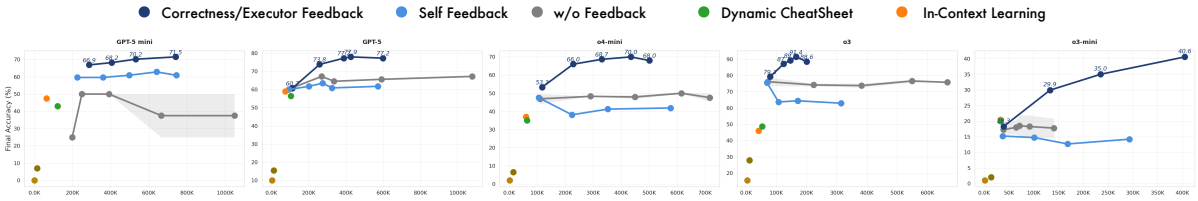

Figure 10: On BrowseComp-Plus, self-feedback models fall behind as the task requires external search-based knowledge.

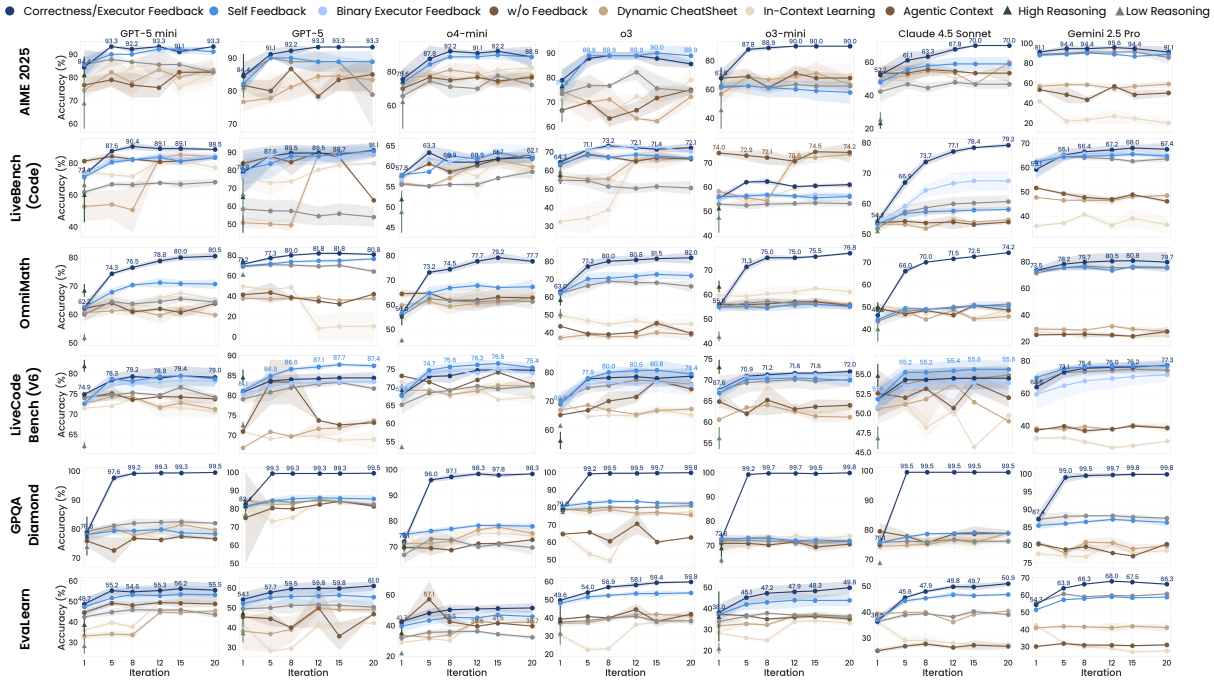

Figure 11: Full results of seven state-of-the-art LLMs on six benchmarks incorporating different generation techniques. Model accuracies with different levels of feedback (correctness/executor, self, and binary executor feedback) generally perform better than the baseline strategies.

4. **Random / Drift / Unknown Driver** – The motivation cannot be inferred; the change appears stochastic or stylistic. *Signals:* Superficial rewording, minor ordering changes, no logical link to feedback or instruction.

—

II. Change Manifestation (WHAT changed)
These categories describe *the technical or structural form* of the change between attempts.

A. **Structural Plan / Algorithm Revision** – A new high-level approach or reformulation (e.g., brute force → DP, heuristic → formula). *Signals:* Rewritten main structure, new helper functions, change in complexity or data representation.*

B. **Local Step Soundness & Invariant Fix** – Correction of a local logic, variable, arithmetic, or invariant violation while keeping the overall plan. *Signals:* Fixed off-by-one in loop, corrected variable update, repaired algebraic derivation.*

C. **Edge / Boundary Condition Handling** – Added or fixed guard for extreme/special cases (e.g., empty, zero, overflow, tie). *Signals:* ‘if n == 0‘, ‘<=‘ ‘<‘, added epsilon, handled ‘len == 1‘.*

D. **Output / Format Compliance** – Adjusted presentation or output schema without changing algorithmic content. *Signals:* Added “FINAL ANSWER:,” fixed JSON/CSV layout, printing only required token.*

E. **Other / Ambiguous Change** – Cannot clearly assign to A–E or insufficient evidence. *Signals:* Stylistic edits, reordering, or unrelated cleanup.*

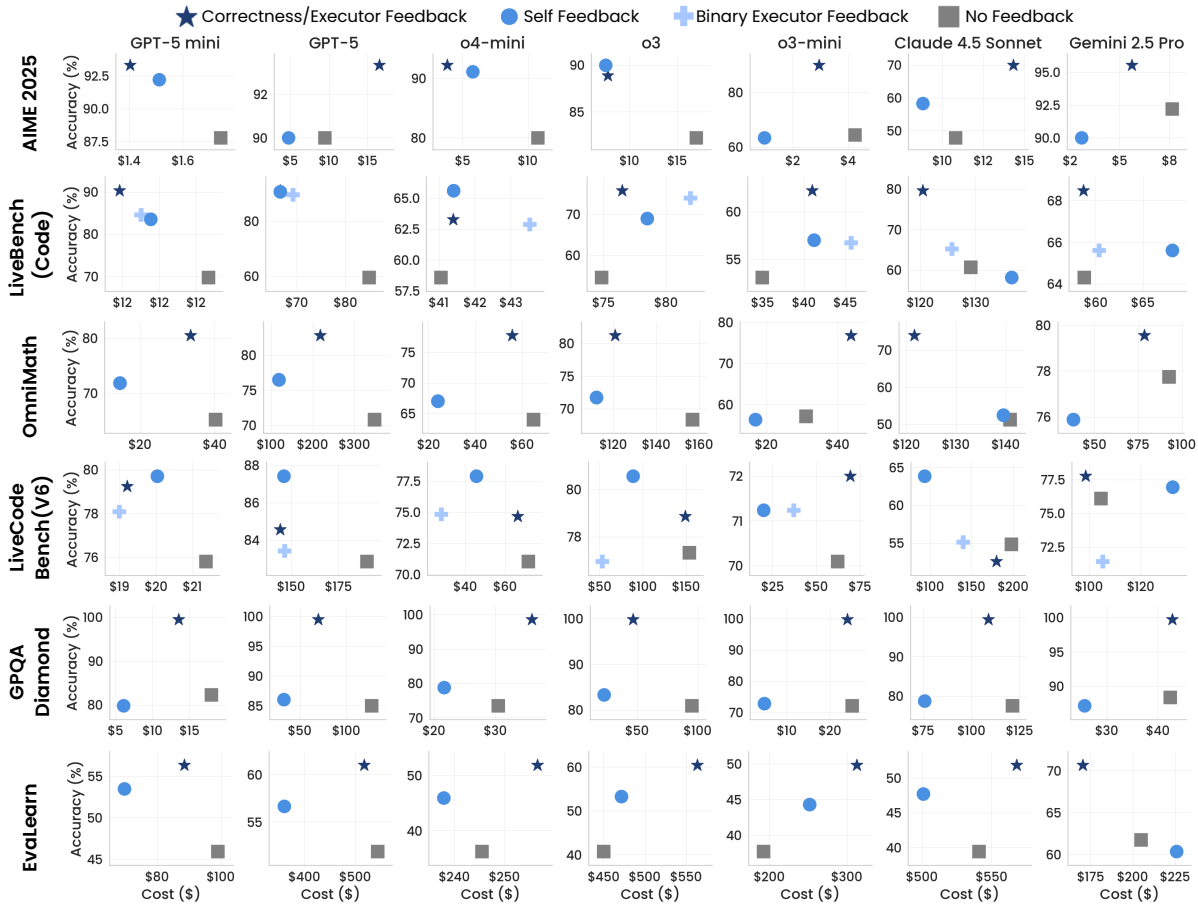

Figure 12: Full results of total API cost (in dollar) vs. best model performance over 20 iterations. LLMs with detailed feedback (*i.e.*, self feedback) achieves decent results with fewer costs, while CoE without feedback generally falls behind (at lower right).

— ... ...
VII. Constraints
* Do **not** recompute correctness or logic. * Focus on identifying the **causal link (WHY)** and the **technical locus (WHAT)** of the change. * Be concise: cite only minimal evidence sufficient to justify your judgment. * If code is long, highlight the 1–2 lines most diagnostic of change. * If feedback is visible but not followed, set ‘"visible": true, "operationalized": false‘. * Always describe the *learning pattern* observed in Attempt_t+1.

I Declaration of AI Tool Usage

During the preparation of this manuscript, we used OpenAI’s GPT-5 model for minor language refinement and smoothing of the writing. The AI tool was not used for generating original content, conducting data analysis, or formulating core scientific ideas. All conceptual development, experimentation, and interpretation were conducted independently without reliance on AI tools.