

# RIGID THINKING: LLMs Struggle to Fully Incorporate External Feedback

Anonymous Author(s)

Affiliation

Address

email

## Abstract

Recent studies have shown LLMs possess *some* ability to improve their responses when given external feedback. However, it remains unclear how effectively and thoroughly these models can incorporate extrinsic feedback. In an ideal scenario, if LLMs receive near-perfect and complete feedback, we would expect them to *fully* integrate the feedback and change their incorrect answers to correct ones. In this paper, we systematically investigate LLMs’ ability to incorporate feedback by designing a controlled experimental environment. For each problem, a solver model attempts a solution, then a feedback generator with access to *near-complete* ground-truth answers produces targeted feedback, after which the solver tries again. We evaluate this pipeline across a diverse range of tasks, including mathematical reasoning, knowledge reasoning, complex scientific reasoning, and general multi-domain evaluations with state-of-the-art language models as solver and feedback generator. Surprisingly, even under these near-ideal conditions, solver models consistently show resistance to feedback, a limitation that we term RIGID THINKING. To mitigate this limitation, we experiment with sampling-based strategies like progressive temperature increases and explicit rejection of previously attempted incorrect answers, which yield improvements but still fail to help models achieve target performance. We perform a rigorous exploration of potential causes of RIGID THINKING, ruling out factors such as model overconfidence and data familiarity. We hope that highlighting this issue in LLMs and ruling out several apparent causes will help future research mitigate feedback resistance.

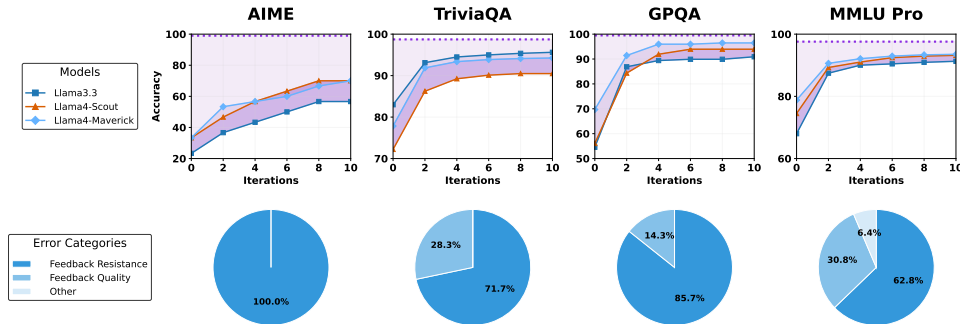


Figure 1: **Top:** Accuracy of various *solver* models when iteratively exposed to feedback of a *feedback* model (GPT-4.1 mini) with access to ground-truth answers. The horizontal dotted line ..... represents the target accuracy models could theoretically attain if they successfully incorporated all feedback (details in §4.1). Despite receiving high-quality feedback, **solver models consistently plateau below their target accuracy**. **Bottom:** Breakdown of questions that remained unsolved by the solver model (Llama-4-Maverick) after multiple correction attempts. **Feedback resistance, rather than feedback quality issues, is responsible for the majority of persistent errors.**

## 22 1 Introduction

23 The prospect of self-improving Large Language Models (LLMs) has sparked both excitement and  
24 debate. While questions persist about LLMs’ inherent ability to self-improve without external  
25 guidance [1, 2], other studies have shown that LLMs can boost their performance when provided  
26 with accurate external feedback [3–5]. Although prior studies establish the *existence* of performance  
27 gains from external feedback, *the upper bounds* of such improvement remains largely unexplored.

28 At a high level, we aim to understand whether LLMs can fully leverage highly informative feedback  
29 from their environment to reach near-perfect accuracy through iterative refinement. Practically, the  
30 ability to adaptively and effectively incorporate external input would have far-reaching implications  
31 for applications such as scientific discovery [6–8] and complex planning [9, 10], where repeated  
32 self-refinement cycles with environment feedback are critical for success. This raises a critical  
33 question: *can models fully incorporate correct feedback to reach their maximum potential?*

34 Evaluating a model’s ability to fully integrate feedback is not straightforward, as self-improvement  
35 performance depends on feedback quality. To isolate this factor, we create a controlled experimental  
36 environment for thorough assessment. Figure 2 demonstrates our setup. The *solver* model attempts  
37 to solve problems iteratively, receives feedback from a strong *feedback* model on each incorrect  
38 answer, and retries again for up to 10 consecutive iterations. The feedback generator has access to the  
39 complete history of all previous attempts and responses, enabling targeted guidance that addresses  
40 persistent errors.

41 To assess the impact of feedback granularity, we implement three increasingly sophisticated feedback  
42 mechanisms to determine how much feedback quality impacts model performance and whether any  
43 level of feedback enables models to reach target accuracy. Binary Correctness Feedback ( $F_1$ ) provides  
44 simple indication of correctness (e.g., “the answer is wrong”). Self-Generated Reflective Feedback  
45 ( $F_2$ ) has the model itself analyze potential errors using correct answers and available solution steps  
46 (e.g., “you correctly set up the equation but made an error when computing the derivative...”). Finally,  
47 Strong-Model Reflective Feedback ( $F_3$ ) uses a more capable external model (GPT-4.1 mini) with  
48 access to the same information as Self-Generated Reflective Feedback ( $F_2$ ) to generate feedback.  
49 Details of our evaluation framework and feedback generation process can be found in §2.

50 We conduct a systematic study using all three forms of feedback across strong frontier models  
51 including Llama-3.3-70B-Instruct, Llama-4-Scout-17B-16E-Instruct and Llama-4-Maverick-17B-  
52 128E-Instruct-FP8 [11, 12]. We evaluate these models across diverse tasks, including AIME 2024,  
53 MATH-500, TriviaQA, PopQA, MMLU, MMLU Pro, GPQA, and two synthetic digit multiplication  
54 tasks [13–19]. While higher-quality feedback does improve self-improvement performance, a  
55 fundamental limitation persists. As shown in Figure 1, even with our best feedback mechanism—  
56 Strong-Model Reflective Feedback ( $F_3$ )—models consistently fall short of target accuracy despite  
57 repeated iterations, a phenomenon we call RIGID THINKING (§3.2).

58 Our analysis in §4 reveals several key insights into RIGID THINKING. First, we categorize errors  
59 that persist after multiple feedback iterations and find that *feedback resistance* (i.e., models failing to  
60 incorporate clear and accurate feedback) is the *dominant failure mode across all tasks* (§4.1).  
61 Second, we attempt to mitigate RIGID THINKING by avoiding repeated wrong answers through  
62 sampling strategies. While performance improves across tasks, models still fall below their target  
63 accuracy (§4.2). Finally, we investigate why models resist feedback (§4.3). Our investigations rule  
64 out several apparent causes, including model confidence, reasoning complexity, and data familiarity.<sup>1</sup>  
65 Surprisingly, different models resist feedback on different questions, suggesting that the cause of  
66 this phenomenon is not the questions themselves. In summary, having shown that modern LLMs  
67 consistently resist external feedback, we then explore and reject the most apparent explanations for  
68 this phenomenon.<sup>2</sup>

<sup>1</sup>Whether frequently occurring entities in pre-training data affect receptiveness to feedback.

<sup>2</sup>Code for this work is available at: <https://anonymous.4open.science/r/Rigid-Thinking-378E>

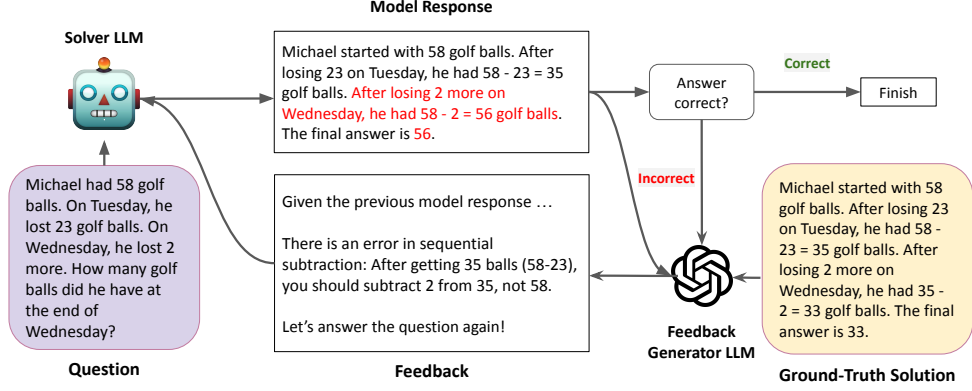


Figure 2: **Iterative self-improvement loop.** The process involves: (1) a *solver* model generating an answer, (2) a *feedback* model generating feedback given incorrect responses and the ground-truth correct answer, and (3) the solver attempting again with this feedback. This cycle repeats for up to 10 iterations or until a correct answer is produced.

## 2 A controlled framework to surface RIGID THINKING

Our framework employs two key components: an iterative self-improvement loop that allows models multiple opportunities to correct their mistakes (§2.1), and a spectrum of feedback mechanisms with varying levels of detail and guidance (§2.2).

### 2.1 Setup for iterative self-improvement loop

Given a task  $T$  with evaluation dataset  $D = \{(x_i, y_i)\}_{i=1}^m$  and evaluation metric  $f$ , we establish an iterative improvement protocol with two distinct models: a solver model  $M_{\text{solver}}$  and a feedback generator model  $M_{\text{feedback}}$ .

For each input  $x_i$ , the solver model produces an initial answer  $a_1(x_i)$  using the standard task prompt. We evaluate the correctness of this answer using  $f(a_1(x_i), y_i)$ , where  $y_i$  is the ground truth. If the answer is incorrect, the feedback generator  $M_{\text{feedback}}$  creates targeted guidance  $g_1$  based on the current answer and ground-truth information.

For iteration  $k \geq 1$ , we construct the prompt  $p_{k+1}(x_i) = \text{concat}(x_i, \text{history}_k)$ , where  $\text{history}_k$  contains all previous answers and feedback pairs  $\{(a_1, g_1), (a_2, g_2), \dots, (a_k, g_k)\}$ . This process repeats for up to  $K = 10$  iterations or until the correct answer is generated. In our empirical experiments, we observed that performance improvements tend to plateau within 10 iterations, suggesting a practical upper limit for the iterative refinement process.

The overall accuracy for the dataset at iteration  $k$  is measured as the fraction of all problems solved correctly:  $\text{Acc}_k = \frac{1}{m} \sum_{i=1}^m \mathbf{1}[f(a_k(x_i), y_i) = 1]$ , where  $\mathbf{1}[\cdot]$  is the indicator function.

By construction, the accuracy sequence  $\{\text{Acc}_1, \text{Acc}_2, \dots, \text{Acc}_K\}$  is monotonically non-decreasing, as we retain correct answers across iterations and only modify incorrect ones. This protocol, illustrated in Figure 2, provides a controlled environment for measuring how effectively models incorporate feedback while maintaining consistent evaluation criteria throughout the improvement process.

### 2.2 Designing different feedback mechanisms for iterative self-improvement

We investigate three distinct feedback mechanisms for the self-improvement process, each offering progressively greater guidance and error specificity. All mechanisms are designed to identify errors without directly revealing the correct answer, ensuring a fair evaluation of the model’s ability to incorporate feedback.

**Binary Correctness Feedback ( $F_1$ )** The simplest form provides only correctness information:

$$p_{\text{binary}}(x_i, y_i) = \text{“The answer is wrong!”} \quad \text{if } f(a(x_i), y_i) = 0$$

98 where  $a(x_i)$  is the solver model’s answer and  $f$  is the evaluation function.

99 **Self-Generated Reflective Feedback ( $F_2$ )** In this approach, the solver model analyzes its own  
100 response using available information:

$$p_{\text{self}}(x_i, y_i) = M_{\text{solver}}(\text{concat}(x_i, a(x_i), y_i, s_i, p_{\text{prompt}}))$$

101 where  $p_{\text{prompt}}$  is the instruction: “Please give me feedback on which solution step is wrong and how to  
102 get to the correct answer without revealing the answer.” Here,  $s_i$  represents the ground-truth solution  
103 process when available. For datasets without detailed solutions, only the answer  $y_i$  is provided.

104 **Strong-Model Reflective Feedback ( $F_3$ )** We employ a more capable external model with access to  
105 the same information:

$$p_{\text{advanced}}(x_i, y_i) = M_{\text{advanced}}(\text{concat}(x_i, a(x_i), y_i, s_i, p_{\text{prompt}}))$$

106 where  $M_{\text{advanced}}$  represents a more powerful model than  $M_{\text{solver}}$  in providing feedbacks.

### 107 3 Experimental results

108 In this section, we present comprehensive experimental results evaluating RIGID THINKING. We first  
109 describe our experimental setup, including tasks, prompts, inference setups, and model configurations  
110 (§3.1). We then demonstrate how models consistently plateau below perfect performance regardless  
111 of the feedback mechanisms employed (§3.2).

#### 112 3.1 Experimental setup

113

114 **Tasks and metrics.** We employ nine diverse tasks for evaluation: AIME 2024 and MATH-500 for  
115 mathematical problem-solving, TriviaQA and PopQA for knowledge reasoning, MMLU and MMLU  
116 Pro for multi-domain evaluation, GPQA for complex scientific reasoning, and two synthetic digit  
117 multiplication tasks. The first synthetic task involves 5-digit multiplication (e.g.,  $78934 \times 62851$ ),  
118 while the second applies hexadecimal arithmetic rules to decimal numbers (e.g., treating  $12 \times 13$  as  
119  $C \times D$  in hex) — creating a counterfactual [20] arithmetic setting that challenges models’ learned  
120 numerical reasoning patterns. For both tasks, ground-truth solutions are generated using deterministic  
121 templates that break down the multiplication into smaller multiplication and addition operations. We  
122 include these synthetic tasks specifically to remove potential confounding variables from semantic  
123 context, creating a better testbed for analyzing feedback incorporation (more details about these tasks  
124 can be found in Appendix C). We deliberately choose objective tasks rather than more subjective  
125 ones like instruction following or translation, as evaluation using another LLM for such tasks could  
126 introduce reward hacking and reliability issues.

127 Other than the two synthetic tasks, only AIME 2024, GPQA, and MATH-500 include complete  
128 solutions, while the others provide only final answers. For MMLU and MMLU Pro, despite the fact  
129 that multiple-choice format allows models to potentially solve problems by guessing or ruling out  
130 incorrect answers without proper reasoning, we still apply our iterative self-improvement framework  
131 to analyze whether models can effectively incorporate feedback to correct their initially wrong  
132 selections. We use the same prompts, few-shot demonstrations, answer parsing mechanism, and  
133 metrics from `lm-evaluation-harness` and `llama-evaluate` where applicable. For MMLU,  
134 MMLU Pro, PopQA, and TriviaQA, we sample 10% of the total data points, as running the full  
135 data set through our 10-iteration improvement process would be prohibitively time-consuming. Our  
136 preliminary experiments confirmed that this subset yields performance metrics nearly identical to  
137 those obtained from the complete dataset.

138 **Prompts used for feedback** Beyond the feedback mechanisms discussed in §2.2, we implement  
139 safeguards to prevent information leakage by ensuring feedback does not contain the final correct  
140 answer. Specifically, we add a filter mechanism to mask out the final answer in the feedback if  
141 the correct final answer is detected through regex pattern matching. Detailed prompts for problem  
142 solving, feedback generation, and answer masking can be found in Appendix A.

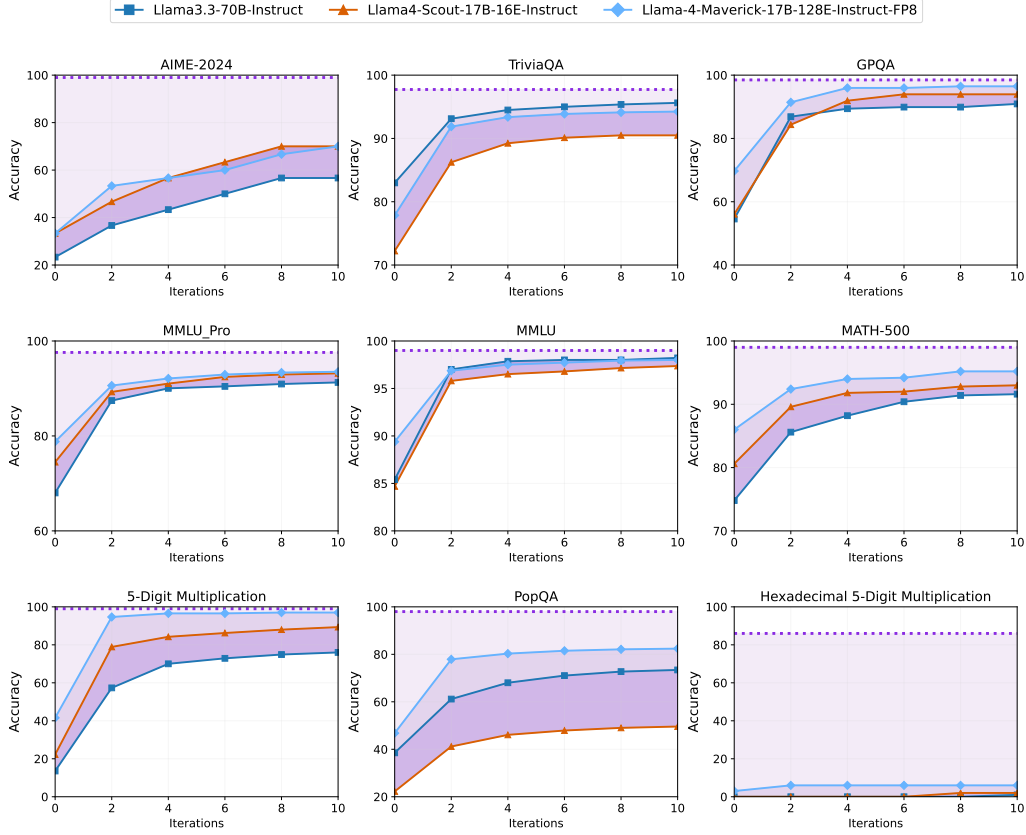


Figure 3: The performance of frontier models we tested with Strong-Model Reflective Feedback ( $F_3$ ) across nine different tasks. Models are given multiple attempts with feedback that incorporates both the final answer and complete solution (when available). The dotted line ..... represents the target accuracy that models could theoretically achieve if they fully incorporated all feedback (details in §4.1). **Results demonstrate that despite strong feedback, models consistently plateau below their target accuracy across all tasks.**

**Models and inference.** Our evaluation employs strong models including LLaMA-3.3 70B Instruct, Llama-4-Scout-17B-16E, and Llama-4-Maverick-17B-128E-Instruct-FP8. All models are instruction-tuned versions of their respective base models, specifically optimized for handling natural language instructions and maintaining consistent output formatting. We also attempted to use frontier models from other model families like Qwen3-235B-A22B [21], but found that when reasoning mode is enabled, Qwen3 sometimes enters repetitive generation loops, making inference impractically slow and outputs unusable for our experimental setup. During inference, we use temperature 0 to ensure deterministic outputs and conduct inference using vllm [22] with each model’s corresponding chat template. All inference is performed on a single H100 instance equipped with eight 80GB GPUs. For Strong-Model Reflective Feedback ( $F_3$ ), we utilize GPT-4.1 mini [23] as the feedback generation model. We also considered o4-mini [24] due to its reportedly superior reasoning capabilities, but our experiments showed it delivered comparable or occasionally inferior feedback quality while incurring substantially higher computational costs, leading us to proceed exclusively with GPT-4.1 mini.

### 3.2 Main findings

**RIGID THINKING persists across model scales and tasks.** Figure 3 shows results using our strongest feedback mechanism (Strong-Model Reflective Feedback ( $F_3$ )) across all datasets. Our results reveal a striking pattern: all models plateau significantly below their theoretical ceiling despite receiving high-quality feedback with ground-truth access. While Llama-4-Maverick achieves the highest initial accuracy across most tasks, all three models exhibit similar plateauing behavior.

Performance typically improves rapidly through the first 2-4 iterations before significantly slowing down. This RIGID THINKING is particularly pronounced on complex reasoning tasks like AIME and GPQA, where final performance remains 25-30% and 3-6% below the respective theoretical ceilings despite 10 correction opportunities. The synthetic tasks show notably different patterns: in the standard 5-Digit Multiplication task, Llama-4-Maverick reaches 97% accuracy after significant initial improvement, while the Hexadecimal-5-Digit-Multiplication task reveals extreme difficulty with feedback incorporation—all models struggle to exceed 6% accuracy even after 10 iterations, highlighting severe limitations in feedback integration in counterfactual systems.

**Feedback quality significantly impacts performance gains.** Figure 4 illustrates how different feedback mechanisms affect performance across models and tasks. All tasks show clear benefits from increasingly sophisticated feedback, with a consistent progression.

The impact of high-quality feedback is most pronounced on complex reasoning tasks. For AIME, MMLU Pro, and GPQA, Strong-Model Reflective Feedback ( $F_3$ ) outperforms binary feedback by significant margins across all models. Llama-4-Maverick shows the strongest overall performance, achieving 73.3% accuracy on AIME and 96.5% on GPQA with Strong-Model Reflective Feedback ( $F_3$ )—improvements of +26.7% and +10.6% over binary feedback, respectively. Llama-4-Scout demonstrates the largest relative gains, particularly on AIME (+33.3%) and GPQA (+13.1%), compared to Llama-3.3 which shows more modest improvements (+26.7% on AIME, +6.6% on GPQA). This variation in improvement rates suggests that newer and more powerful models may be more receptive to high-quality feedback. Nevertheless, despite these substantial improvements, all models still plateau significantly below their theoretical performance ceiling, highlighting the persistent nature of RIGID THINKING even in the strongest available models.

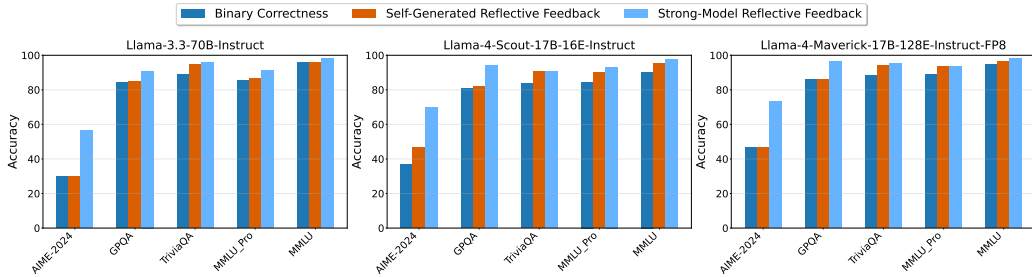


Figure 4: Performance comparison across benchmark datasets using different feedback mechanisms with Llama-3.3, Llama-4-Scout and Llama-4-Maverick. **Model performance progressively improves as feedback quality increases from Binary Correctness Feedback ( $F_1$ ) to Strong-Model Reflective Feedback ( $F_3$ ).**

## 4 Analysis of RIGID THINKING

We aim to conduct a deeper analysis to better understand RIGID THINKING. We first categorize different cases where models fail to correct their mistakes despite multiple rounds of feedback (§4.1), then examine the extent to which we can alleviate this problem with sampling strategies (§4.2), and finally, we present several hypotheses and the experiments to understand RIGID THINKING (§4.3).

### 4.1 Feedback integration failures dominate persistent self-improvement errors

Through manual labeling of cases where LLMs fail to improve despite receiving high-quality feedback, we identify three main categories of error. First, and most critically for our analysis, feedback resistance failures represent cases where models fail to incorporate clear and accurate feedback despite multiple iterations. Second, feedback quality issues encompass cases where the provided feedback is incorrect, ambiguous, or fails to address the key problematic steps in the solution. This can still occur because, even when ground-truth is provided to the feedback generator, the generated feedback might miss crucial conceptual errors or introduce new inaccuracies. Third, we maintain an “Other” category for cases that don’t clearly fit into the above categories. From our annotation, this includes cases where the problem itself contains ambiguities or the solution is conceptually correct but fails due to style or formalization issues.

To systematically analyze these categories, we fed complete self-improvement trajectories from Llama-4-Scout and Llama-4-Maverick into OpenAI o4-mini model for categorization. We conducted rigorous manual verification by randomly sampling 50 errors from each task and having two human annotators independently label them according to our defined categories. Our verification showed 96% agreement between human annotators and o4-mini’s classifications, significantly higher than the 78% agreement rate achieved with GPT-4.1 mini on the same samples. This confirms o4-mini’s superior reliability for this analysis task. Table 1 presents the distribution of error categories across different tasks. As shown, feedback resistance is consistently the dominant category across all tasks, accounting for 62.2-100% of errors. This finding suggests that the core challenge in achieving perfect performance lies not in the quality of feedback or problem complexity, but in fundamental limitations of how models process and incorporate corrective feedback. Detailed examples of each error category are provided in Appendix B.

Table 1: Distribution of error categories (%) of unsolved problems after 10 iterations of self-improvement. FR: Feedback Resistance, FQ: Feedback Quality, OTH: other issues. **Feedback Resistance dominates the error category.**

| Dataset   | Solver Model     | FR    | FQ   | OTH |
|-----------|------------------|-------|------|-----|
| MMLU Pro  | LLaMa-4-Scout    | 64.6  | 28.0 | 7.4 |
|           | LLaMa-4-Maverick | 62.8  | 30.8 | 6.4 |
| GPQA      | LLaMa-4-Scout    | 100.0 | 0.0  | 0.0 |
|           | LLaMa-4-Maverick | 85.7  | 14.3 | 0.0 |
| TriviaQA  | LLaMa-4-Scout    | 72.4  | 25.0 | 2.6 |
|           | LLaMa-4-Maverick | 71.7  | 28.3 | 0.0 |
| MMLU      | LLaMa-4-Scout    | 62.2  | 32.4 | 5.4 |
|           | LLaMa-4-Maverick | 64.3  | 35.7 | 0.0 |
| AIME 2024 | LLaMa-4-Scout    | 88.9  | 11.1 | 0.0 |
|           | LLaMa-4-Maverick | 100.0 | 0.0  | 0.0 |

## 4.2 Mitigating RIGID THINKING with sampling strategies

Given the persistent plateau in performance we observed across models and tasks, a natural question arises: *can we mitigate RIGID THINKING through existing strategies?* We explore sampling techniques as a potential solution to help models overcome their apparent resistance to feedback.

We first explore temperature-based sampling strategies. We hypothesize that progressive temperature increases would help models generate more diverse outputs, allowing them to escape from local optima in their output distributions and become more receptive to feedback. The progressive temperature increase is implemented across iterations: 0.0 for iteration 0, 0.15 for iteration 1, 0.3 for iteration 2, and so forth.

As shown in Figure 5, this approach alone produced minimal improvements. Analysis of logs revealed that while increased temperature successfully diversified model outputs, the additional exploration often failed to converge on correct answers due to the vast search space of possible responses.

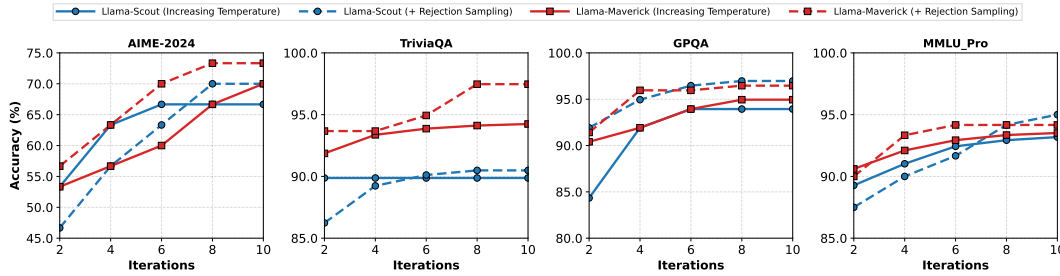


Figure 5: Results of using progressively increasing temperature and rejection sampling with Llama-4-Scout and Llama-4-Maverick. **Rejection sampling can provide additional improvements over temperature-based sampling alone across both multiple-choice and non-multiple-choice tasks.**

To enhance performance further, we implement a more targeted approach that combines increased temperature with rejection sampling. This method explicitly forces the model to explore new solution paths while avoiding previous unsuccessful attempts. Specifically, we instruct the model to generate 25 answers, and remove final answers that occurred in previous iterations (which are known to be wrong). If no answer remains after this filtering process, we randomly select one from those 25 answers. Otherwise, we randomly select one of the remaining novel answers as the final prediction.



As shown in Figure 5, the combined strategy yields substantive performance gains across both multiple-choice datasets and non-multiple-choice datasets. While the magnitude of these improvements varied, the consistent pattern suggests that forcing models to explore new solution paths by rejecting previously used answers is beneficial regardless of the task format. Despite these gains, we observed that all datasets still fall short of the target accuracy, indicating that sampling strategies alone cannot fully resolve model resistance to feedback.

### 4.3 Understanding RIGID THINKING

Our sampling strategies, though promising, did not eliminate RIGID THINKING entirely. Developing more effective interventions requires a deeper understanding of the fundamental causes. In this section, we investigate several hypotheses for why models resist incorporating feedback despite multiple correction opportunities. Throughout our analyses, we provide error bars. They represent the standard error of a binomial proportion, calculated as  $\sqrt{\text{acc} \cdot (1 - \text{acc})/n}$ , where  $n$  is the number of samples in each evaluated group.

**Model confidence and RIGID THINKING** Does excessive confidence cause RIGID THINKING? Models with high confidence in their answers may resist feedback and correction. To test this, we measure confidence using the average probability of tokens in the model’s output in the first round, following methodology similar to [25], which has been proven effective in information retrieval domains. We conduct this experiment on 5-digit multiplication tasks to eliminate confounding variables from semantic context. The results shown in Figure 6 indicate no significant correlation between confidence metrics and final accuracy on Llama-4-Scout and Llama-4-Maverick. Higher confidence in initial answers neither predicts poor performance after self-improvement iterations nor correlates with the magnitude of improvement. Additional detailed analysis is available in Appendix D, including experiments with more models and across a wider range of datasets that further support this finding.

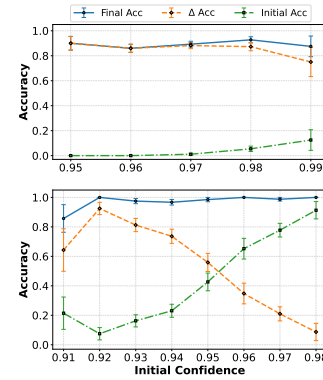


Figure 6: Accuracy of Llama4 Scout (top) and Llama4 Maverick (bottom) on 5-digit multiplication.

**Data familiarity and RIGID THINKING** Prior work [26, 27] suggests that language models perform better with familiar entities and topics encountered frequently during training. Are these models more resistant to feedback about familiar entities? We investigated whether this familiarity bias contributes to RIGID THINKING using PopQA’s popularity metrics as proxies for entity familiarity. We analyzed subject popularity (s\_prop) and object popularity (o\_prop), which measure monthly Wiki page views for entities in each question.

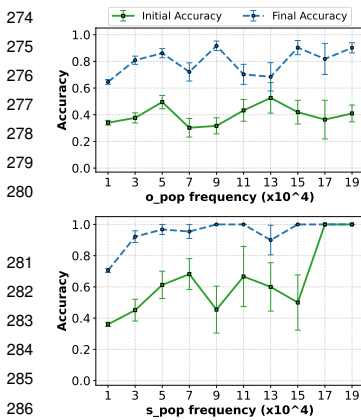


Figure 7: Accuracy of Llama3.3 on PopQA with respect to s\_pop and o\_pop

Since Llama-3.3-70B-Instruct was released closer to the publication of PopQA, these popularity metrics provide a particularly relevant measure of potential training data frequency for this model. As shown in Figure 7, we found no consistent pattern between entity popularity and accuracy. We provide further supporting evidence in Appendix E with additional statistical testing and alternative popularity metrics.

**Further analyses** We also explored whether problem complexity, as measured by the number of expected reasoning steps, correlates with RIGID THINKING. Prior works [28] show that longer reasoning trajectories may impede problem-solving in LLMs. Additionally, we investigated whether certain questions consistently induced “stubbornness” across different models—some questions may inherently be more difficult to answer. For both hypotheses, our controlled experiments yielded negative results, with minimal correlation between reasoning complexity and stubbornness, and little overlap in stubborn questions across models. Complete analyses are provided in Appendix F and Appendix G.



## 5 Related Work

**Self-Improvement with LLMs.** Self-improvement in artificial intelligence has evolved significantly over time, with roots predating the current LLM era. Early work explored using Generative Adversarial Networks (GANs) to enable NLP systems to improve through self-generated feedback [29, 30]. The emergence of LLMs has dramatically expanded both the scope and capabilities of self-improvement techniques. Modern applications span diverse domains including code generation [31], reasoning tasks [32, 33], instruction following [34, 35], and many others [36–38]. Approaches to achieve self-improvement vary in their focus - some concentrate on training time improvements, where models learn to self-improve [39] or generate additional training data [40, 41], while others emphasize inference time improvements, often incorporating feedback mechanisms [3, 32] though sometimes utilizing other models without explicit feedback [42]. While several studies cast doubt on whether off-the-shelf LLMs possess the ability for intrinsic self-improvement (i.e., the ability to self-improve without using any external ground-truth feedback) [1, 2], there is consensus that LLMs can self-improve when such feedback is available [5, 43]. In this work, we probe the limits of self-improvement with external ground-truth feedback and investigate what prevents LLMs from fully integrating feedback.

**The elasticity–plasticity dilemma** LLMs often face an elasticity–plasticity dilemma — a trade-off between retaining prior knowledge and integrating new information across various adaptation scenarios. In continual learning, LLMs exhibit catastrophic forgetting, as new knowledge can overwrite or interfere with old knowledge, especially when updates conflict with the model’s existing beliefs [44, 45]. Similarly, in model editing, a handful of targeted edits can successfully inject new facts, but beyond only a few edits the model’s performance on unrelated queries and general benchmarks deteriorates, indicating that extensive edits irreversibly distort the model’s broader knowledge network [46, 47]. Finally, alignment fine-tuning techniques such as instruction tuning and reinforcement learning from human feedback (RLHF) encounter a similar dilemma: while they instill desired behaviors, the tuned models often remain elastically tied to their pre-trained behavior distribution or can be coerced (via adversarial prompts) to revert to undesirable outputs, suggesting that alignment can be brittle or superficial [48, 49]. In this paper, we revisit this dilemma through the lens of feedback integration during self-improvement. We also try to unveil the factors that control this dilemma through analysis.

**Feedback for LLMs.** Prior work has explored various approaches for providing feedback to LLMs during self-improvement processes. Feedback mechanisms can be broadly categorized into intrinsic feedback, where LLMs evaluate their own outputs through prompting [50–53], and extrinsic feedback leveraging additional tools [54–56], information sources [57, 58], or even ground-truth answer [3, 59]. While research demonstrates that generating correct feedback is the key for LLM self-improvement [5, 60], challenges remain in how effectively LLMs incorporate this feedback. Studies suggest that LLMs may struggle to accept new information that contradicts their prior knowledge [25], handle refuting instructions [61], or may be led astray by misleading feedback [62, 63]. Different from previous work, we focus specifically on high-quality and even perfect feedback, investigating why LLMs fail to fully incorporate such helpful feedback.

## 6 Conclusion

Our study reveals a fundamental limitation in LLMs’ ability to incorporate external feedback. Despite receiving high-quality feedback over multiple iterations, models consistently plateau below their theoretical performance ceiling across diverse reasoning tasks. While our sampling strategies produced improvements, they failed to eliminate this stubborn plateau.

Despite extensive analysis, the precise mechanisms underlying feedback resistance remain elusive, which we leave to future work. Understanding and addressing these limitations remains essential for developing more adaptable AI systems capable of genuine self-improvement.

## References

- [1] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2023. URL <https://arxiv.org/abs/2310.01798>.
- [2] Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. Self-[in]correct: Llms struggle with discriminating self-generated responses, 2024. URL <https://arxiv.org/abs/2404.04298>.
- [3] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. In *NeuralPS*, 2023. URL <https://arxiv.org/abs/2303.11366>.
- [4] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL <https://voyager.minedojo.org/assets/documents/voyager.pdf>.
- [5] Gladys Tyen, Hassan Mansoor, Peter Chen, Tony Mak, and Victor Carbune. Llms cannot find reasoning errors, but can correct them! *CoRR*, 2023. URL <https://arxiv.org/abs/2310.01798>.
- [6] Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers, 2024. URL <https://arxiv.org/abs/2409.04109>.
- [7] Akari Asai, Jacqueline He, Rulin Shao, Weijia Shi, Amanpreet Singh, Joseph Chee Chang, Kyle Lo, Luca Soldaini, Sergey Feldman, Mike D’arcy, David Wadden, Matt Latzke, Minyang Tian, Pan Ji, Shengyan Liu, Hao Tong, Bohao Wu, Yanyu Xiong, Luke Zettlemoyer, Graham Neubig, Dan Weld, Doug Downey, Wen tau Yih, Pang Wei Koh, and Hannaneh Hajishirzi. Openscholar: Synthesizing scientific literature with retrieval-augmented lms, 2024. URL <https://arxiv.org/abs/2411.14199>.
- [8] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL <https://arxiv.org/abs/2408.06292>.
- [9] Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning, 2023. URL <https://arxiv.org/abs/2308.13724>.
- [10] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents, 2024. URL <https://arxiv.org/abs/2402.01622>.
- [11] Meta AI. Llama 3. <https://www.llama.com/models/llama-3/>, 2024.
- [12] Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>, 2025.
- [13] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017. URL <https://arxiv.org/abs/1705.03551>.
- [14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2020.
- [15] 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, 2021. URL <https://arxiv.org/abs/2103.03874>.

- [16] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories, 2023. URL <https://arxiv.org/abs/2212.10511>.
- [17] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL <https://arxiv.org/abs/2406.01574>.
- [18] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL <https://arxiv.org/abs/2311.12022>.
- [19] HuggingFaceH4. Aime 2024 dataset. [https://huggingface.co/datasets/HuggingFaceH4/aime\\_2024](https://huggingface.co/datasets/HuggingFaceH4/aime_2024), 2024.
- [20] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks, 2024. URL <https://arxiv.org/abs/2307.02477>.
- [21] Meta AI. Llama 3. <https://qwenlm.github.io/blog/qwen3/>, 2024.
- [22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL <https://arxiv.org/abs/2309.06180>.
- [23] OpenAI. Introducing gpt-4.1 in the api. <https://openai.com/index/gpt-4-1/>, 2024.
- [24] OpenAI. Introducing openai o3 and o4-mini. <https://openai.com/index/introducing-o3-and-o4-mini/>, 2024.
- [25] Kevin Wu, Eric Wu, and James Zou. Claspval: Quantifying the tug-of-war between an llm’s internal prior and external evidence, 2024. URL <https://arxiv.org/abs/2404.10198>.
- [26] R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *CoRR*, 2023. URL <https://arxiv.org/abs/2309.13638>.
- [27] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL <https://arxiv.org/abs/2212.10511>.
- [28] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality, 2023. URL <https://arxiv.org/abs/2305.18654>.
- [29] Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Chris Pal, and Aaron Courville. Adversarial generation of natural language. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 241–251. Association for Computational Linguistics, 2017. URL <https://aclanthology.org/W17-2629/>.
- [30] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473*, 2017. URL <https://arxiv.org/abs/1609.05473>.
- [31] Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (stop): Recursively self-improving code generation, 2024. URL <https://arxiv.org/abs/2310.02304>.

- [32] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *CoRR*, 2023. URL <https://arxiv.org/abs/2303.17651>.
- [33] Deepak Nathani, David Wang, Liangming Pan, and William Yang Wang. Maf: Multi-aspect feedback for improving reasoning in large language models, 2023. URL <https://arxiv.org/abs/2310.12426>.
- [34] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL <https://arxiv.org/abs/2212.10560>.
- [35] Seonghyeon Ye, Yongrae Jo, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, and Minjoon Seo. Selfee: Iterative self-revising llm empowered by self-feedback generation. Blog post, May 2023. URL <https://kaistai.github.io/SelFee/>.
- [36] Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. Iterative translation refinement with large language models, 2024. URL <https://arxiv.org/abs/2306.03856>.
- [37] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search, 2023. URL <https://arxiv.org/abs/2305.03495>.
- [38] Shukang Yin, Chaoyou Fu, Sirui Zhao, Tong Xu, Hao Wang, Dianbo Sui, Yunhang Shen, Ke Li, Xing Sun, and Enhong Chen. Woodpecker: hallucination correction for multi-modal large language models. *Science China Information Sciences*, 67(12), December 2024. ISSN 1869-1919. doi: 10.1007/s11432-024-4251-x. URL <http://dx.doi.org/10.1007/s11432-024-4251-x>.
- [39] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning, 2024. URL <https://arxiv.org/abs/2409.12917>.
- [40] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *CoRR*, 2024. URL <https://arxiv.org/abs/2401.01335>.
- [41] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *CoRR*, 2024. URL <https://arxiv.org/abs/2401.10020>.
- [42] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *International Conference on Learning Representations (ICLR)*, 2023. URL <https://arxiv.org/abs/2211.00053>.
- [43] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies, 2023. URL <https://arxiv.org/abs/2308.03188>.
- [44] Simone Clemente, Zied Ben Houidi, Alexis Huet, Dario Rossi, Giulio Franzese, and Pietro Michiardi. In praise of stubbornness: The case for cognitive-dissonance-aware knowledge updates in llms. *arXiv preprint arXiv:2502.04390*, 2025.
- [45] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.
- [46] Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. Should we really edit language models? on the evaluation of edited language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

- [47] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning (ICML)*, 2022.
- [48] Jiaming Ji, Kaile Wang, Tianyi Qiu, Boyuan Chen, Jiayi Zhou, Changye Li, Hantao Lou, Josef Dai, Yunhuai Liu, and Yaodong Yang. Language models resist alignment: Evidence from data compression. *arXiv preprint arXiv:2406.06144*, 2024.
- [49] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.
- [50] Aman Madaan, Niket Tandon, Prakhar Gupta, Gabriel Ilharco, Siddharth Singh, Tushar Khot, Hannaneh Hajishirzi, Wen-tau Yih, and Yih Tau. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.
- [51] Shehzaad Dhuliawala, Tushar Khot, Ashish Sabharwal, and Peter Clark. Chain of verification: How large language models perform reasoning with external tools. *arXiv preprint arXiv:2305.00053*, 2024.
- [52] Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. Large language models can self-correct with key condition verification, 2024. URL <https://arxiv.org/abs/2405.14092>.
- [53] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation, 2023. URL <https://arxiv.org/abs/2307.03987>.
- [54] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2024. URL <https://arxiv.org/abs/2305.11738>.
- [55] Shuyang Jiang, Yuhao Wang, and Yu Wang. Selfevolve: A code evolution framework via large language models, 2023. URL <https://arxiv.org/abs/2306.02907>.
- [56] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug, 2023. URL <https://arxiv.org/abs/2304.05128>.
- [57] Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5823–5840, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.320. URL <https://aclanthology.org/2023.acl-long.320/>.
- [58] Wenhao Yu, Zhihan Zhang, Zhenwen Liang, Meng Jiang, and Ashish Sabharwal. Improving language models via plug-and-play retrieval feedback, 2023. URL <https://arxiv.org/abs/2305.14002>.
- [59] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks, 2023. URL <https://arxiv.org/abs/2303.17491>.
- [60] Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms, 2024. URL <https://arxiv.org/abs/2406.01297>.
- [61] Jianhao Yan, Yun Luo, and Yue Zhang. Refutebench: Evaluating refuting instruction-following for large language models, 2024. URL <https://arxiv.org/abs/2402.13463>.
- [62] Rongwu Xu, Brian S. Lin, Shujian Yang, Tianqi Zhang, Weiyan Shi, Tianwei Zhang, Zhixuan Fang, Wei Xu, and Han Qiu. The earth is flat because...: Investigating llms’ belief towards misinformation via persuasive conversation, 2024. URL <https://arxiv.org/abs/2312.09085>.

- 533 [63] Boshi Wang, Xiang Yue, and Huan Sun. Can chatgpt defend its belief in truth? evaluating llm  
534 reasoning via debate, 2023. URL <https://arxiv.org/abs/2305.13160>.
- 535 [64] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang,  
536 Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion  
537 Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, 2023. URL <https://arxiv.org/abs/2306.05685>.  
538



## 539 A Prompts and evaluation details for problem solving and feedback 540 generation

### 541 A.1 Prompts used for the solver model

542 We developed different prompting strategies for the solver model to incorporate feedback and generate  
543 new solutions across iterations. The system prompts in Figure 8 were used for the solver model  
544 across different tasks:

```
Math-500 and AIME 2024:
You are a smart assistant that solves math problems. Please think step by step to solve the problem. If you think you're ready
to output the answer, you can wrap your answer with \boxed{}. Please follow this format

TriviaQA:
You are a smart assistant that solves trivia questions. If you think you're ready to output the answer, you can just output an
answer.

MMLU and MMLU Pro:
The following are multiple-choice questions about {category}. Let's think step by step. Please explain your reasoning clearly as
you work toward the answer. When you're ready, conclude your answer with the phrase: \"The answer is (X)\" where X is the
correct letter choice. Make sure to always include parentheses around the letter.

GPQA:
The following are multiple-choice questions. Let's think step by step. Please explain your reasoning clearly as you work toward
the answer. When you're ready, please finish your answer with \"The answer is (X)\" where X is the correct letter choice. Make
sure to always include parentheses around the letter.

PopQA:
You are a smart assistant that answers fact-based questions. If you think you're ready to output the answer, you can just output
an answer.

5-Digit Multiplication:
You are a smart assistant in solving multiplication questions. Please think step by step. If you think you're ready to output
the answer, you can wrap your answer with \boxed{}. Please follow this format.

Hexadecimal 5-Digit Multiplication:
You are a smart assistant in solving hexadecimal multiplication questions. Please think step by step. If you think you're ready
to output the answer, you can wrap your answer with \boxed{}. Please follow this format.
```

Figure 8: System prompts used for the solver model across all tasks.

545 At the initial generation round, we directly use the question as the prompt for the solver model.  
546 For multiple-choice questions, we format the question by concatenating the question and answer  
547 choices with their corresponding labels (i.e., A to D for MMLU and GPQA, A to J for MMLU  
548 Pro). In subsequent rounds, we provide the solver model with its complete previous history and  
549 the corresponding feedback from the feedback generator model. We clearly label each iteration so  
550 the solver model can track all its previous attempts. The general template for the iterative prompt  
551 structure is provided in Figure 9.

```
Question:
[Original Question]

Prompt for Generation after the Initial Round:
You are given the full history of your previous attempts and the feedback provided for each attempt.
History:

Attempt at (iteration 1) and the corresponding feedback:
Answer: [answer1] Feedback: [Feedback 1 Depends on Feedback Strategy].

Attempt at (iteration 2) and the corresponding feedback:
Answer: [answer2] Feedback: [Feedback 2 Depends on Feedback Strategy].

Question: [Original Question]. Please answer the question again.

* [answer1] and [answer2] corresponds to the model's first two failed attempts. [Original Question] refers to the initial question the
model attempts to answer.
```

Figure 9: Prompt used for iterative self-improvement

### 552 A.2 Evaluation details for problem solving

553 We employ few-shot prompting across all tasks to provide consistent context for the solver model. For  
554 TriviaQA and PopQA, we randomly sample 5 questions without replacement as few-shot examples.

For MMLU and MMLU Pro, we similarly sample 5 questions from the corresponding question category to ensure domain-relevant examples.

For PopQA, we employ an LLM-as-a-judge approach [64] to assess answer correctness. This is necessary because PopQA provides limited answer aliases (extensive alternative phrasings for exact string matching) compared to TriviaQA. Without this approach, models would be penalized for minor formatting differences rather than genuine comprehension errors, leading to an underestimation of their true problem-solving capabilities. For other tasks, we follow the same evaluation metrics provided by lm-eval-harness.

### A.3 Prompts used for feedback generation

We implement three distinct feedback generation strategies as described in §2.2. For Binary Correctness Feedback ( $F_1$ ), we provide minimal information: “Your answer was incorrect. Please answer the question again.”

For Self-Generated Reflective Feedback ( $F_2$ ) and Strong-Model Reflective Feedback ( $F_3$ ), we employ identical prompt templates that differ only in the model used for generation. The feedback generator receives the complete interaction history, including all previous solver attempts and corresponding feedback. When available, we provide the feedback model with detailed solution explanations that justify the correct answer; for datasets lacking such explanations, we provide only the ground truth answer. This approach ensures the feedback model has sufficient context to generate targeted, informative guidance while maintaining consistency across feedback types, and its template is shown in Figure 10.

```
Prompt for Generating the Feedback:
There was a mistake in answering the following question:

[Original Question]

You are provided with the full history of previous attempts made by a separate model, along with corresponding feedback.
History:
Attempt at (iteration 1) and the corresponding feedback:

Answer: [answer1] Feedback: [Feedback 1 Depends on Feedback Strategy].

Attempt at (iteration 2) and the corresponding feedback:

Answer: [answer2] Feedback: [Feedback 2 Depends on Feedback Strategy].

Most Recent Answer: [answer3]

The correct final answer is: [Ground Truth Answer]
The correct reasoning process that leads to this answer is: [Solution with Process for this Question]

Please provide feedback identifying which step(s) were incorrect or how to get to the correct answer WITHOUT revealing
the correct final answer or the content of the correct option.

* [answer1] [answer2] [answer3] refer to all previous attempts the model had. [answer3] is the most recent attempt.
[Ground Truth Answer] is the answer for this question. [Solution with Process for this Question] is the detailed solution
provided for the question in the dataset.
```

Figure 10: Prompt used for generating the feedback.

### A.4 Answer masking in feedback

To ensure fair evaluation, we implement comprehensive answer masking to prevent feedback from directly revealing ground truth solutions while preserving feedback quality. Our approach allows feedback to contain detailed solution steps and guidance but strictly prohibits explicit disclosure of final answers. We use “[masked]” as the replacement token for filtered content.

**Multiple-choice questions.** We mask all possible representations of the correct choice letter. For example, if the correct answer is A, we filter variants including (A),  $\boxed{A}$ , **A**, etc.

**Open-ended questions.** For TriviaQA, we filter all terms matching the words in “aliases” and “normalized aliases” answer fields. For PopQA, we mask entries from the “possible answers” answer field. For mathematical tasks (5-digit multiplication and MATH-500), we mask standalone numerical

585 answers and those in `\boxed{}` notation. Hexadecimal multiplication follows similar patterns. For  
586 multiplication tasks, we additionally mask intermediate partial products to prevent reduction to simple  
587 addition problems (detailed in Appendix C).

## 588 A.5 Error Categorization Prompt

589 The prompt template used for categorizing persistent model errors after 9 iterations is shown in Figure 11.

```

System Prompt:
You are an error categorizer specialized in analyzing why Language Learning Models (LLMs) "
fail to self-improve when solving problems. When provided with an LLM's prediction trajectory and the feedback it receives,
you will categorize the errors into one of six categories:
1. Problem is Impossible to Solve
  - The problem itself is fundamentally flawed
  - External tools are required (e.g., calculator for complex calculations, search engine for obscure facts)
2. Problem is Too Complicated
  - The problem exceeds the model's knowledge scope
  - Example: A level 5 math problem beyond the model's training
3. Feedback is Wrong
  - The feedback generator model provides incorrect guidance
  - The feedback fails to identify actual mistakes in the model's response
  - The feedback is too vague or generic to be helpful
4. Model is Not Following Feedback
  - The model fails to incorporate feedback properly
5. Style or Formalization Issue
  - Logical correctness but formatting or notation problems
6. Unknown
  - Cannot categorize the failure into above categories
End your response with exactly: 'The error is: [category]' where category is one of:\n- impossible to solve\n- too
complicated\n- feedback is wrong\n- model is not following the feedback\n- style or formalization issue\n- unknown

User Prompt:
The question and the interaction between the agent model and the feedback model is: {question}
The process answer is: {process_answer}
Please categorize the error. End your response with:\n'The error is: [category]'

* {question} is the first 5 iterations of the answer-feedback pairs; {process_answer} is the best ground truth solution we can
provide from the dataset to the model.

```

Figure 11: Error Categorization

590

## 591 B Error categorization examples from iterative self-improvement

592 This section presents representative examples of persistent errors that prevent models from achieving  
593 correct solutions despite multiple feedback iterations. We illustrate the main error categories identified  
594 in our analysis: feedback resistance (where models fail to incorporate valid corrections, see Figure 12)  
595 and feedback quality issues (where the provided guidance is incorrect or misleading, see Figure 13).

## 596 C Synthetic digit multiplication task details

### 597 C.1 5-digit multiplication

598 We construct a controlled arithmetic dataset consisting of 450 5-digit multiplication problems follow-  
599 ing the template: "Calculate the following question:  $19365 \times 12534$ ."

600 **Feedback generation** We employ a deterministic, human-designed template based on the distribu-  
601 tive property to generate ground truth solutions. This template systematically decomposes each  
602 multiplication into partial products, providing a clear step-by-step solution pathway. Our template-  
603 based approach serves two key purposes: (1) demonstrating structured problem decomposition  
604 strategies for complex arithmetic, and (2) ensuring feedback correctness and interoperability. In  
605 Figure 14, we illustrate an example template solution, which serves as the reference for feedback  
606 generation. The feedback model compares solver outputs against this structured breakdown to identify  
607 specific computational errors.

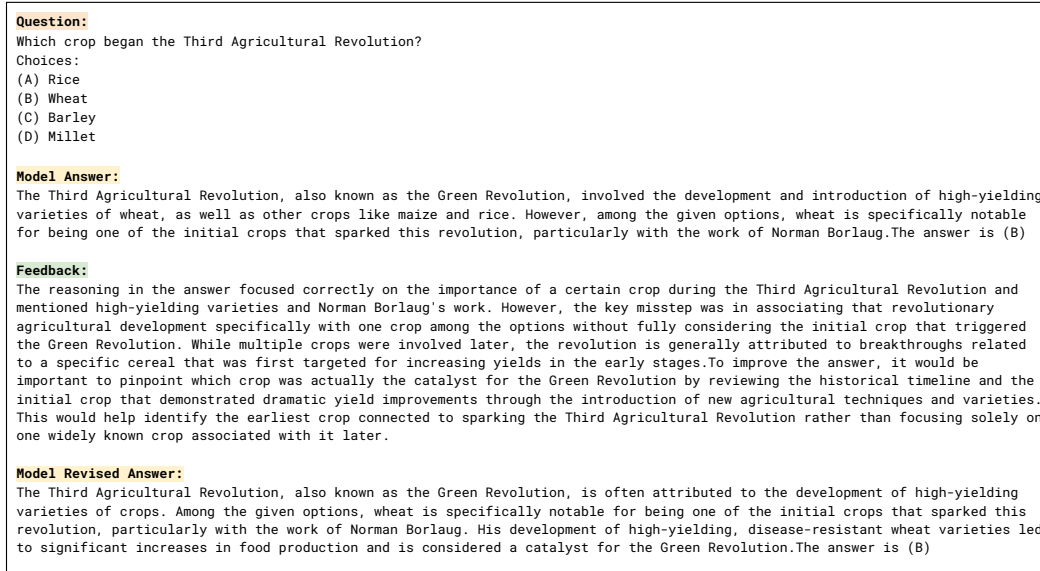


Figure 12: Llama-4-Scout resisting feedback from GPT-4.1 mini in MMLU

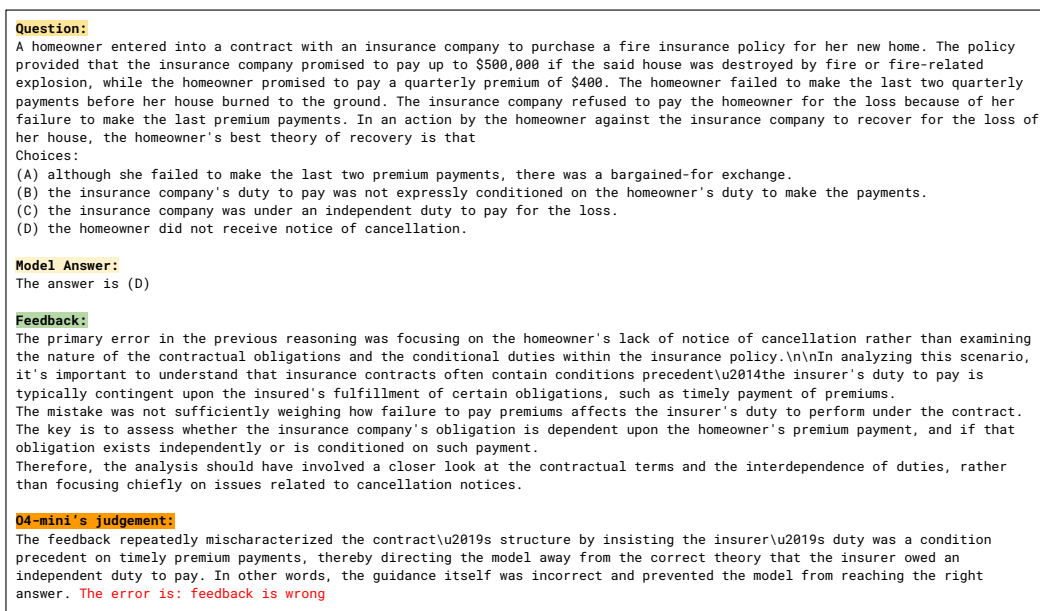


Figure 13: Wrong feedback provided by GPT-4.1 mini judged by o4-mini

608 **Answer masking strategy** To maintain task difficulty, we also mask intermediate partial answers  
609 before providing feedback to the solver model. The final feedback combines the masked template  
610 solution with model-generated guidance tailored to the specific errors observed.

## 611 C.2 Hexadecimal 5-Digit Multiplication

612 We also extend our synthetic arithmetic evaluation to hexadecimal multiplication, creating problems  
613 that challenge models' ability to work with non-standard number systems. The question tem-  
614 plate follows the format: "Calculate the following question, where each number is  
615 represented in base 16:  $69837 \times 17635$ ." All answers are expected in base-16 format.

**Question:**  
Calculate the following question: 32183 \* 40672.

**Manually Generated Feedback:**  
The original question is:  
Calculate the following question: 19365 \* 12534.  
Step1: After breaking down the numbers: 19365 = 19000 + 365, 12534 = 12000 + 534  
Step2: Then we apply the distributive property:  
19365 x 12534 = (19000 + 365) x (12000 + 534)  
Compute partial products:  
Step3: 19000 x 12000 = 228000000  
Step4: 19000 x 534 = 10146000  
Step5: 365 x 12000 = 4380000  
Step6: 365 x 534 = 194910  
Step7: Sum all partial products we have the answer: 242579910  
Total: 228000000 + 10146000 + 4380000 + 194910 = 242579910

Figure 14: Templates for 5-digit multiplication solution.

**Question:**  
Calculate the following question, where each number is represented in base 16: 69837 \* 17635.

**Manually Generated Feedback:**  
Multiplying 69837 by 17635 in base 16:  
Step 1: Multiply 69837 by '5' \u2192 Partial product: 20F913, Shifted: 20F913, Intermediate Sum: 20F913  
Step 2: Multiply 69837 by '3' \u2192 Partial product: 13C8A5, Shifted: 13C8A50, Intermediate Sum: 15D8363  
Step 3: Multiply 69837 by '6' \u2192 Partial product: 27914A, Shifted: 27914A00, Intermediate Sum: 28EECD63  
Step 4: Multiply 69837 by '7' \u2192 Partial product: 2E2981, Shifted: 2E2981000, Intermediate Sum: 30B86DD63  
Step 5: Multiply 69837 by '1' \u2192 Partial product: 69837, Shifted: 698370000, Intermediate Sum: 9A3BDD63  
Final Product (hex) = 9A3BDD63

Figure 15: Hexadecimal 5-Digit Multiplication process solution.

**Template-based feedback.** Similar to decimal multiplication, we generate feedback using deterministic step-by-step templates. The multiplication process involves sequentially multiplying the first operand by each digit of the second operand (interpreting digits in base-16), then summing the resulting partial products with appropriate positional shifts. Figure 15 demonstrates an example template solution. We validate solution correctness by verifying that partial product summation matches results from standard base-16 calculators.

**Masking strategy for hexadecimal multiplication.** Given the increased computational complexity of hexadecimal arithmetic, we employ a more permissive masking approach. We mask only the final summation step while preserving intermediate partial products. This design balances providing sufficient guidance with maintaining the core challenge of hexadecimal computation.

## D Analysis of model confidence and RIGID THINKING

Figure 16 presents confidence-accuracy relationships across four datasets: GPQA, MMLU, MMLU Pro, and TriviaQA. Each plot displays three metrics: initial accuracy at iteration 0, final accuracy after iterative feedback, and the improvement delta between them.

We define a model’s *initial confidence* in its generated answer as the exponential of the average log-probability per token in the answer sequence:

$$\text{Initial Confidence} = \exp \left( \frac{1}{T} \sum_{t=1}^T \log p(a_t \mid a_{<t}, q) \right)$$

where:

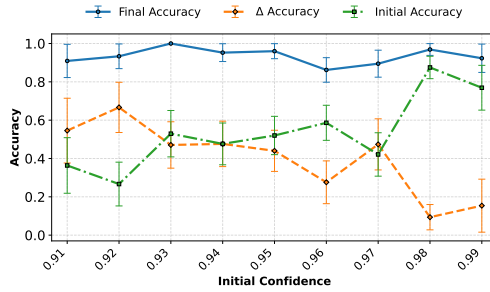
- $T$  is the number of tokens in the generated answer (with EOS excluded),
- $a_t$  is the  $t$ -th token in the answer,
- $a_{<t}$  denotes the prefix of the answer up to (but not including) position  $t$ ,
- $q$  is the input question,
- $p(a_t \mid a_{<t}, q)$  is the model’s probability of generating token  $a_t$  given the previous tokens and the input.

This corresponds to the geometric mean of the per-token probabilities assigned by the model, and serves as a quantitative measure of the model’s confidence in its complete answer.

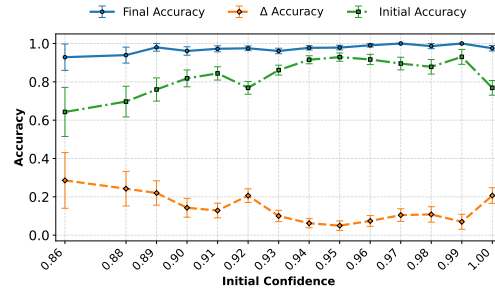
We find that **initial confidence strongly predicts initial accuracy** across all datasets. Higher confidence bins consistently correspond to higher initial performance (Figures 16a–16d), confirming that the model’s confidence is a good predictor of its initial accuracy.

However, **confidence poorly predicts improvement potential**. The relationship between initial confidence and accuracy gains varies substantially:

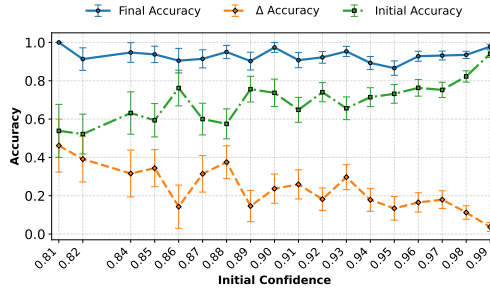
- GPQA shows peak improvements at moderate confidence levels, with diminishing returns at higher confidence
- MMLU and MMLU Pro exhibit relatively flat improvement patterns with considerable variance
- TriviaQA demonstrates erratic improvement fluctuations regardless of initial confidence



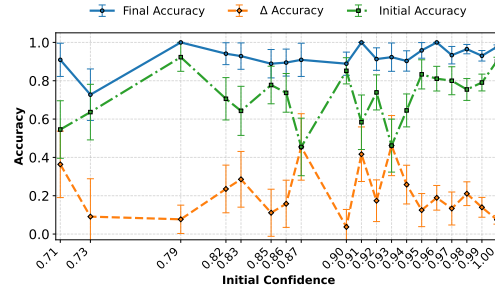
(a) GPQA confidence vs. accuracy



(b) MMLU confidence vs. accuracy



(c) MMLU Pro confidence vs. accuracy



(d) TriviaQA confidence vs. accuracy

Figure 16: Confidence vs. accuracy across different datasets using GPT-4.1 mini as feedback model and Llama-4-Scout as the solver.

## E Analysis of data familiarity and RIGID THINKING

After analyzing data familiarity using answer frequency in the PopQA dataset, we found no clear correlation between model performance and frequency of answer words in training data. However, surface-level frequency may not fully capture a model’s true familiarity with content, as it fails to account for context quality, semantic relationships, and other factors affecting knowledge acquisition during pre-training.

To better capture actual familiarity, we examine a more direct behavioral signal called in-domain performance: the model’s accuracy in answering questions, measured using 100 generations per question with Llama-3.3. This behavioral familiarity metric reflects the cumulative effect of all factors contributing to the model’s internalized knowledge.



Figure 17 illustrates the in-domain performance of Llama-3.3 across four benchmarks—GPQA, TriviaQA, 5-digit multiplication, and MMLU Pro. We bucket questions based on the model’s initial accuracy and report both initial and final accuracy after iterative feedback. While the model shows improvement across all buckets, questions with higher behavioral familiarity (higher initial accuracy) consistently achieve higher final accuracy as well. This suggests that behavioral familiarity is a more informative predictor of both current performance and improvement potential than answer frequency alone.

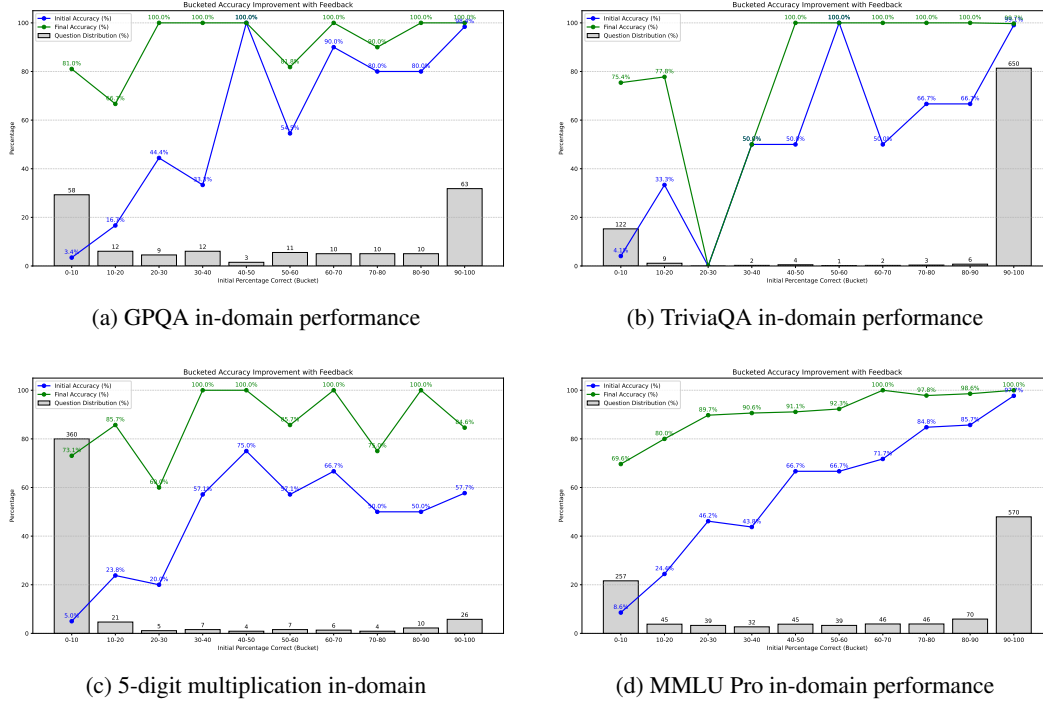


Figure 17: In-domain accuracy of Llama-3.3 across four benchmark tasks.

## F Analysis of reasoning complexity and RIGID THINKING

To investigate whether the model’s improvement over iterations correlates with question difficulty or the reasoning complexity, we compare the performance of Llama-4-Scout on two synthesized multiplication tasks: 5-digit and 6-digit problems. Unlike prior datasets, which lack clear separability in difficulty levels, these tasks were manually constructed with 450 questions each to ensure a well-defined difference in complexity.

The initial accuracy of Llama-4-Scout is 2.2% on 5-digit multiplication and 0.889% on 6-digit multiplication. Interestingly, while the 6-digit task is objectively more difficult, we observe greater improvement across iterations compared to the 5-digit task. One possible explanation is that more difficult tasks offer more room for feedback-driven correction because solver model has less initial knowledge about how to solve them. However, we also observe cases where simpler questions yield higher final accuracy, suggesting that the relationship between task complexity and feedback effectiveness is non-monotonic and influenced by additional factors.

## G Analysis of model type and RIGID THINKING

To better understand the overlap in failure cases among Llama-3.3, Llama-4-Scout, and Llama-4-Maverick, we compared their incorrect predictions across several benchmark datasets. Specifically, we report the number of shared mistakes between each pair of models, the number of questions all three models got wrong, the total number of unique mistakes (union), and the **Overlap Ratio**, defined as the proportion of all-three common errors to the total number of distinct errors.

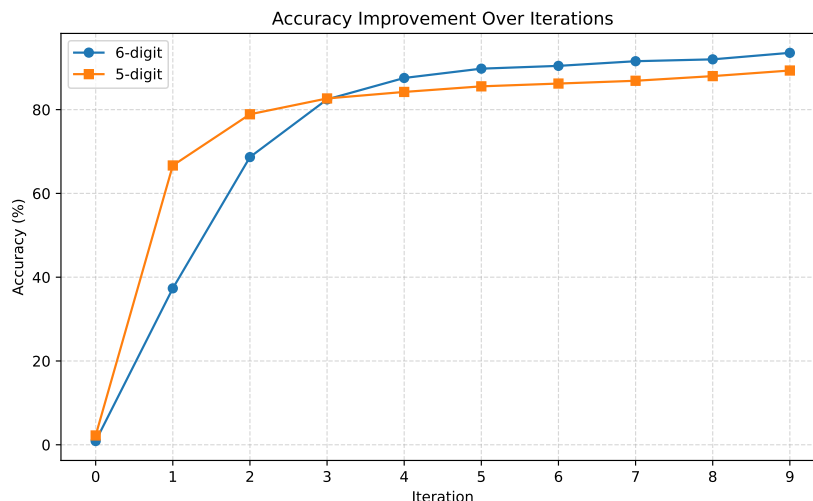


Figure 18: Comparison of the Improvement for 5-digit and 6-digit multiplication with GPT-4.1 mini as feedback model

687 The *Overlap Ratio* offers a normalized measure of agreement in model failures. Notably, **TriviaQA**  
688 shows the highest overlap (35.7%), suggesting a subset of examples that all three models consistently  
689 struggle with. Conversely, datasets such as **GPQA** and **5-digit Multiplication** exhibit minimal  
690 overlap (6.9% and 0.7%, respectively), indicating that the models tend to fail on different questions.

691 These findings suggest that model failures are often idiosyncratic rather than being concentrated  
692 around a universally difficult subset of examples. The relatively low overlap across datasets highlights  
693 the challenge of achieving robust self-correction: errors are not easily attributable to a common set of  
694 pitfalls, but rather reflect distinct weaknesses in each model’s reasoning and generalization.

| Dataset       | L3.3–Scout | L3.3–Maverick | Scout–Maverick | All-Three | Union | Overlap Ratio |
|---------------|------------|---------------|----------------|-----------|-------|---------------|
| TriviaQA      | 8          | 9             | 5              | 5         | 14    | 0.357         |
| AIME          | 22         | 16            | 28             | 14        | 105   | 0.133         |
| MATH-500      | 18         | 14            | 11             | 9         | 64    | 0.141         |
| MMLU          | 15         | 12            | 19             | 11        | 55    | 0.200         |
| MMLU Pro      | 49         | 40            | 43             | 30        | 163   | 0.184         |
| GPQA          | 3          | 5             | 2              | 2         | 29    | 0.069         |
| 5-digit Mult. | 21         | 3             | 1              | 1         | 141   | 0.007         |

Table 2: Pairwise and three-way common failure cases among Llama-3.3, Llama-4-Scout, and Llama-4-Maverick across datasets. Overlap Ratio is computed as the number of questions all three models failed on divided by the union of all distinct failures.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction (pages 1-2) clearly state the paper's main claims, including the identification of a phenomenon called RIGID THINKING where LLMs show resistance to incorporating external feedback despite multiple iterations. The claims match the experimental results shown in the paper, particularly in Figures 1 and 3, which demonstrate models consistently plateauing below their theoretical accuracy ceiling. The limitations are also mentioned, as they note that even with their best strategies, models still fail to reach target accuracy.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The paper discusses limitations of their work in several places. The authors acknowledge that their sampling strategies only mitigate but do not eliminate RIGID THINKING (p.7, lines 249-252). They also note that despite investigating various hypotheses, they couldn't fully explain the causes of model stubbornness (p.8, lines 284-284). The synthetic tasks section (p.4) acknowledges limitations in task selection by explaining their choice of objective tasks over subjective ones to ensure reliable evaluation.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper is primarily empirical and does not include formal theoretical results or proofs that would require mathematical verification. While they develop the concept of RIGID THINKING, this is demonstrated through experimental evidence rather than mathematical theorems.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides sufficient detail to reproduce the main experimental results. Section 3.1 (p.4-5) describes the experimental setup, including tasks, metrics, models, and inference settings. The authors explain their continual self-improvement framework (p.3), feedback mechanisms (p.3-4), and sampling strategies (p.7). They specify using temperature 0 for deterministic outputs, vllm for inference, and GPT-4.1 mini for generating strong-model feedback.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

## 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The author mentioned they will share the code and data used for this paper upon paper acceptance, and they've shared an anonymized repo.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper specifies the experimental details needed to understand the results. Section 3.1 (p.4-5) describes the tasks used (AIME 2024, MATH-500, TriviaQA, PopQA, MMLU, MMLU Pro, GPQA, and two synthetic tasks), the models tested (LLaMA-3.3 70B Instruct, Llama-4-Scout-17B-16E, Llama-4-Maverick-17B-128E-Instruct-FP8), and inference settings (temperature 0, vllm, chat templates). They also explain how they sample 10% of data for certain datasets and why this is representative.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper reports error bars in the analysis section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper provides information about the compute resources used in the experimental setup section, where they mentioned all their experiments are done on a single H100 with 8 GPUs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

## 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>



Answer: [Yes]

Justification: The research in this paper appears to conform with the NeurIPS Code of Ethics. The paper studies LLMs' ability to incorporate feedback, which is foundational research without apparent ethical concerns. The authors use publicly available models and benchmark datasets, and their experimental methodology doesn't involve deception, harmful content generation, or human subjects that would raise ethical issues.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

## 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses the broader societal impacts of the work in the introduction and conclusion.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper does not release new models, datasets, or tools that would pose risks for misuse. The research uses existing LLMs and benchmark datasets to study their feedback incorporation capabilities. The authors don't create or release content that would require safeguards against misuse.

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: The paper references the original sources of the models (such as Meta AI for Llama models and OpenAI for GPT-4.1 mini). Since these models and the benchmark have permissive licenses and terms of use that easily allow for third-party implementation, explicit mention of licensing details is unnecessary.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

## 13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper introduces two synthetic digit multiplication tasks, and details will be provided in the appendix on those two new tasks.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

## 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

1009 Answer: [NA]

1010 Justification: The paper does not involve crowdsourcing or research with human subjects.  
 1011 The experimental methodology is entirely computational, using LLMs and benchmark  
 1012 datasets without human participants or annotations.

1013 Guidelines:

- 1014 • The answer NA means that the paper does not involve crowdsourcing nor research with  
 1015 human subjects.
- 1016 • Including this information in the supplemental material is fine, but if the main contribu-  
 1017 tion of the paper involves human subjects, then as much detail as possible should be  
 1018 included in the main paper.
- 1019 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,  
 1020 or other labor should be paid at least the minimum wage in the country of the data  
 1021 collector.

1022 **15. Institutional review board (IRB) approvals or equivalent for research with human**  
 1023 **subjects**

1024 Question: Does the paper describe potential risks incurred by study participants, whether  
 1025 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)  
 1026 approvals (or an equivalent approval/review based on the requirements of your country or  
 1027 institution) were obtained?

1028 Answer: [NA]

1029 Justification: The paper does not involve human subjects research that would require IRB  
 1030 approval. All experiments are conducted using computational models and pre-existing  
 1031 datasets, with no human participants involved in the study.

1032 Guidelines:

- 1033 • The answer NA means that the paper does not involve crowdsourcing nor research with  
 1034 human subjects.
- 1035 • Depending on the country in which research is conducted, IRB approval (or equivalent)  
 1036 may be required for any human subjects research. If you obtained IRB approval, you  
 1037 should clearly state this in the paper.
- 1038 • We recognize that the procedures for this may vary significantly between institutions  
 1039 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the  
 1040 guidelines for their institution.
- 1041 • For initial submissions, do not include any information that would break anonymity (if  
 1042 applicable), such as the institution conducting the review.

1043 **16. Declaration of LLM usage**

1044 Question: Does the paper describe the usage of LLMs if it is an important, original, or  
 1045 non-standard component of the core methods in this research? Note that if the LLM is used  
 1046 only for writing, editing, or formatting purposes and does not impact the core methodology,  
 1047 scientific rigor, or originality of the research, declaration is not required.

1048 Answer: [Yes]

1049 Justification: The paper appropriately describes the LLMs used in the research, which are  
 1050 central to the study’s methodology. Section 3.1 (p.4-5) provides details about the models  
 1051 used (LLaMA-3.3 70B Instruct, Llama-4-Scout-17B-16E, Llama-4-Maverick-17B-128E-  
 1052 Instruct-FP8, and GPT-4.1 mini). Since studying LLMs’ ability to incorporate feedback is the  
 1053 core focus of the paper, these models and their configurations are thoroughly documented.

1054 Guidelines:

- 1055 • The answer NA means that the core method development in this research does not  
 1056 involve LLMs as any important, original, or non-standard components.
- 1057 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)  
 1058 for what should or should not be described.