

Metatuning: An Empirical Study of Judge-Guided Prompt Refinement and Its Boundary Conditions

Aniruddha Chattopadhyay

STUDYANIRUDDHA@GMAIL.COM

Raj Dandekar

Massachusetts Institute of Technology

RAJD@MIT.EDU

Kaushik Roy

University of Alabama

KROY2@UA.EDU

Abstract

Iterative prompt refinement is a practical approach for improving the reliability of large language models (LLMs) without weight updates. In this work, we study *metatuning*: a judge-guided prompt-refinement loop in which an evaluator critiques errors and provides targeted natural-language corrections or demonstrations that are incorporated into the prompt. We evaluate metatuning on axiomatic deductive reasoning (MATH-500), on combinations with Chain-of-Thought (CoT) and self-reflection prompting, and on video-based physical reasoning (CLEVRER). Our results show that metatuning can improve baseline performance in static, rule-like domains, but offers limited benefit when paired with strong reasoning baselines and does not generalize to spatio-temporal video reasoning. Overall, we identify boundary conditions for judge-guided prompt refinement and motivate future work on integrating feedback at the level of reasoning traces.

Keywords: *Large Language Models, Prompt Optimization, Symbolic Feedback*

1. Introduction

Large language models (LLMs) have become strong general-purpose systems, yet they still exhibit brittleness in complex reasoning and can fail systematically on recurring error patterns Capitanelli and Mastrogiovanni (2024). A practical family of approaches to improve reliability without weight updates is *prompt refinement*: iteratively modifying prompts and demonstrations based on observed failures. In this paper, we study *metatuning*¹, a judge-guided prompt-refinement loop in which a separate evaluator critiques errors and produces targeted corrections or demonstrations that are incorporated into the prompt.

In this work, we frame metatuning as a form of *symbolic feedback* in an operational sense: the feedback signal is expressed in discrete natural-language instructions and examples that can be inspected, edited, and reused. This perspective connects to neurosymbolic motivations Fang et al. (2024); Sheth et al. (2024); Colelough and Regli (2024), but our claims in this revision are deliberately empirical: we evaluate when judge-guided prompt refinement helps, when it becomes redundant, and when it fails to transfer across domains. Future work should complement this operational framing with direct representation probing to test stronger grounding/alignment hypotheses Bisk et al. (2022); Blank and Piantadosi (2023).

1. An earlier version of this manuscript is available as an arXiv preprint Chattopadhyay et al. (2025).

Judge-Guided Prompt Refinement Perspective

Can judge-guided, error-driven prompt refinement reliably improve LLM reasoning, and what are its boundary conditions?

Rather than updating model parameters, metatuning refines a *task state*—a prompt plus a structured memory of critiques/demonstrations—derived from repeated dataset interactions and an external judge.

Under such a learning regime, there are direct analogs to the traditional learning setup – train-validation-test splits, number of training runs through the dataset (epochs), gradient accumulation (when to trigger a prompt revision), model saving (saving the history of interactions and prompt-revisions, along with the final prompt), and model loading at inference time on the test set (loading the same history and warm starting with a few sample generations before testing begins).

We evaluate metatuning on axiomatic deductive reasoning (MATH-500), study its interaction with Chain-of-Thought (CoT) and self-reflection prompting, and test transfer to a multimodal, spatio-temporal video reasoning domain (CLEVRER). Our goal is to empirically characterize where iterative prompt refinement improves reliability and where it breaks down.

To further explore the boundaries of our framework, we investigate its interaction with other advanced reasoning techniques. Recently, methods like Chain-of-Thought (COT) and self-reflection have been proposed to enhance LLM reasoning by guiding the model to produce a step-by-step reasoning trace. This raises a crucial question: Do the benefits of our metatuning approach complement or become redundant when combined with these powerful prompting techniques? Our experiments reveal that while COT and self-reflection significantly boost baseline accuracy on mathematical problems, adding metatuning provides no further benefit to GPT-4o and, in some cases, even degrades performance for Gemini 1.5 Flash.

Furthermore, we extend our evaluation to a new domain: video-based physical reasoning. Using the CLEVRER dataset, we assess whether our metatuning approach can improve an LLM’s ability to answer descriptive, explanatory, predictive, and counterfactual questions about object trajectories and collisions in short video clips. We find that in this dynamic and visual domain, metatuning has a negligible effect on performance. These findings provide valuable insights into the limitations of metatuning and suggest that its effectiveness is highly dependent on the nature of the task and the symbolic domain (e.g., static, discrete logical rules vs. dynamic, continuous physical laws).

Thus our main contributions are:

Main Contributions:

- **Symbolic Feedback Perspective.** Frame judge-guided prompt refinement as operational symbolic feedback expressed in natural language instructions and demonstrations.

- **Iterative Prompt-Refinement.** Introduce a structured approach to “learning” via iterative prompt revisions and critique-driven updates, bridging symbolic reasoning and gradient-based optimization.
- **Empirical Validation.** We show that while our iterative prompt-refinement approach (metatuning) improves reasoning on baseline problems, its effectiveness is limited when combined with advanced prompting techniques like Chain-of-Thought and self-reflection.
- **Boundary Conditions for Metatuning** We investigate the applicability of our metatuning approach beyond text-based domains, demonstrating its limitations in improving performance on video-based, spatio-temporal reasoning tasks. This provides valuable insights into the types of problems where our framework is most effective.

Our work is also related to recent methods for automatic prompt optimization and meta-prompting, where prompts are improved via search or feedback loops rather than parameter updates. Examples include human-level prompt engineering via automatic prompt search Zhou et al. (2023), optimizing prompts using LLMs as optimizers Yang et al. (2024), and iterative refinement with self-feedback Madaan et al. (2023). Metatuning differs in that it uses judge-driven critiques and error-driven example selection to construct a learned prompt artifact focused on specific failure modes.

The rest of the paper is organized as follows. Section 2 introduces the operational framing of symbolic feedback and situates metatuning relative to prompt optimization and in-context learning. Section 3 presents our judge-guided prompt-refinement loop and Algorithm 1. Section 4 compares our approach to conventional backpropagation training. Section 5 reports experiments on MATH-500, Chain-of-Thought/self-reflection prompting, and CLEVRER. Section 6 concludes, and the Appendix provides qualitative examples.

2. Background: Symbolic Feedback and Prompt Refinement

2.1. Operational View

We use *symbolic feedback* in an operational sense: feedback is communicated through discrete natural-language artifacts (instructions, critiques, and demonstrations) that can be appended to a prompt and directly inspected. This differs from weight updates, and it does not require claiming that the model is symbol-grounded in a philosophical or representational-probing sense Bisk et al. (2022). Our goal is to study whether such feedback improves empirical task performance and how its effects depend on the task and prompting baseline.

2.2. Relation to Prompt Optimization and ICL

Metatuning is closely related to prompt optimization and meta-prompting methods that search over prompts or iteratively refine them based on feedback Zhou et al. (2023); Yang et al. (2024); Madaan et al. (2023). It also resembles in-context learning (ICL) at inference time because it conditions on demonstrations; however, metatuning emphasizes *error-driven* construction of those demonstrations via judge-guided correction, rather than

static or similarity-retrieved examples. We make this distinction explicit in our CLEVRER experiments (Section 5.3).

3. Method: Metatuning (Judge-Guided Prompt Refinement)

3.1. Illustrative Example

Imagine an LLM-based agent in a text-based adventure game (a simple “world”). The agent’s policy is given by an LLM, but we also maintain a symbolic memory of facts the agent has discovered (e.g. a natural language-based description of the game’s map, items, etc.), and perhaps a similar description of explicit goals or rules (like “you must not harm innocents” as a rule in the game). As the agent acts, an external *prompt-based probe/judge model* (another LLM) could check its actions against these rules and the known facts of the world. If the agent attempts something against the rules or logically inconsistent with its knowledge, the evaluator can intervene – for instance, by giving a natural language feedback (“You recall that harming innocents is against your code.”) or by adjusting the agent’s state (inserting a reminder into the agent’s context window). The agent (LLM) thus receives *symbolic interactions* (in this case, a textual message that encodes a rule or a fact) that alter its subsequent processing. In this learning scenario, the agent refines its internal model based on such interactions. Note that this does not involve directly tweaking weights each time; it instead involves an iterative procedure where each episode of interaction produces a trace that is used to slightly adjust the model’s state (it’s current prompt, history of interactions, prompt-revisions, and judge critiques). Over time, the model internalizes the rules so that it no longer needs the intervention. This viewpoint reframes symbolic **learning as training on a dataset of task-related world experiences** where the experience includes symbolic content (natural language descriptions of rules, knowledge queries) and the learning algorithm’s job is to make the model’s *behavior* align with task objectives.

3.2. Task Learning Algorithm

We propose an iterative learning paradigm for judge-guided prompt refinement that mirrors gradient-based optimization but uses *symbolic feedback* and *intervention* (expressed in natural language) to update the prompt state. The loop can be summarized at a high level in four steps:

1. **Model Initialization:** Begin with a pre-trained model (e.g., an LLM) with initial parameters θ_0 .
2. **Evaluation via an External Judge:** Present tasks to the model and assess its responses through an evaluator that detects errors or inconsistencies.
3. **Generating Symbolic Corrections:** Use the feedback to generate **symbolically structured interventions** (natural language), such as prompt refinements, additional demonstrations, or logical explanations.
4. **Iterative Refinement:** Apply the corrections iteratively to improve the model’s output, either through context updates (natural language-based prompting).

This cycle repeats for a fixed number of iterations (or until the judge indicates that no further improvement is being made). The process is formally described in Algorithm 1.

```

Algorithm 1: Iterative Learning via Symbolic Feedback
Input: Pre-trained model with parameters  $\theta_0$  (e.g., LLM)
Output: Refined model with improved reasoning capabilities
Initialize model with parameters  $\theta_0$  for  $iteration = 1$  to  $N$  do
    // Step 1: Model generates output for a given input/task
     $y \leftarrow \text{model}_{\theta}(x)$ ; // Generate output for task input  $x$ 
    // Step 2: External judge evaluates the output
    feedback  $\leftarrow \text{Judge.evaluate}(x, y)$ ; // Feedback contains a score or identified
        errors
    if feedback indicates perfect output then
        | break; // No correction needed, exit loop
    end
    // Step 3: Generate symbolic corrections
    corrections  $\leftarrow \text{generate\_corrections}(\text{feedback}, x, y)$ ; // Corrections can be:
    ; // - Refined prompts/instructions
    ; // - Additional training examples
    ; // - Logical explanations for reasoning
    // Apply corrections to influence the model
    if corrections include prompts/instructions then
        |  $\theta \leftarrow \text{update\_prompt\_context}(\theta, \text{corrections})$ 
    end
    // Step 4: Proceed to next iteration with updated model/state
end

```

Algorithm 1 details our proposed perspective on learning. This iterative cycle aims to **systematically reduce reasoning errors** through natural language-based interactions and feedback based on running through the training set. Because the procedure relies on heuristic feedback and prompt/context updates, it does not provide formal convergence guarantees.

The **Judge.evaluate** function represents our *symbolic evaluator*. It could be implemented in numerous ways. For instance, we might have an LLM (potentially a more advanced or specialized one) that examines the model’s output and compares it to expected results or known constraints, outputting a “score” or textual critique.

The **generate_corrections** step is where symbolic intervention comes in. The judge gives *natural language feedback*. For example, the judge might say: “The reasoning is flawed because it assumed X, which contradicts known fact Y.” The algorithm then turns that into a corrected reasoning trace or a prompt that reminds the model of Y in context. In essence, **part of the model state, i.e., it’s prompt is revised during training through the training dataset in response to the model’s mistakes.**

The update mechanism for the model is in the *effective model behavior*, modulated by providing a better prompt or adding a memory of previous corrections). For example, we can use a persistent prompt that accumulates instructions (a form of *prompt tuning* or using the model in a closed-loop system). This can be interpreted as a kind of *supervised training*

loop where the new examples from corrections serve as training data with the judge acting as an **oracle** providing the target output or loss.

4. Comparison to Conventional Backpropagation Training

We compare our paradigm to standard **backpropagation-based learning** as follows:

Differentiability: Backprop requires the model and loss to be differentiable end-to-end. Our approach uses non-differentiable feedback. The judge could be a black-box procedure (e.g., LLM) that we cannot differentiate through [Hasan and Holleman \(2021\)](#). We treat the judge as an external oracle and make model state updates via generated examples-based prompt adjustments. This is a big advantage in incorporating arbitrary symbolic rules – we don’t need to make the symbolic logic differentiable; we can just have it critique the model and then adjust via examples

Data Efficiency and Curriculum: Traditional training uses a fixed dataset, and if the model makes mistakes, it will continue to unless the data distribution covers those mistakes. In our iterative loop, we are essentially performing a form of curriculum learning or active learning – the model’s mistakes drive the correct-based on new training data instances on the fly, focusing learning on the most relevant areas. This can be more *sample-efficient within the context window* for correcting specific failure modes, without implying overall training data efficiency. For example, if an LLM consistently makes a reasoning error, we go through a few training examples demonstrating the correct reasoning and behavior-correct on them; a small number of focused examples might correct a behavior that would otherwise require many implicit examples in random training data to fix. Empirically, approaches like self-correction have shown even a single well-chosen example or instruction can pivot an LLM’s performance significantly on certain tasks [Graves et al. \(2017\)](#).

Limitations and Convergence: Our approach does not have the convergence guarantees or well-defined optimization objective that gradient descent has. It’s a more heuristic process. The quality of the final model depends on the quality of the judge and the corrections. If the judge is imperfect (e.g., an LLM judge might have its own errors or biases), we might lead the model astray or instill incorrect rules [Soviany et al. \(2021\)](#). Conventional training, when you have a clear loss and data, is more straightforward to analyze. One could end up oscillating or not converging if, say, the prompt-based corrections don’t stick in the model’s long-term memory.

Relation to Prompt Optimization: Our formulation is closely related to prompt optimization and self-improvement methods that update prompts (or prompting policies) based on performance feedback rather than weight updates [Zhou et al. \(2023\)](#); [Yang et al. \(2024\)](#); [Madaan et al. \(2023\)](#). We emphasize an error-driven, judge-mediated construction of the prompt as a learned artifact, which we then evaluate empirically to identify boundary conditions for when such updates help or become redundant with strong reasoning prompts.

5. Experiments and Discussion

5.1. Metatuning with Zero-shot Prompting

In this section, we evaluate the impact of metatuning on the performance of a Large Language Model (LLM) using the [Maths 500 Dataset](#). We begin by selecting a subsample of

100 problems from the dataset. As illustrated in Figure 3, we assess the model’s zero-shot performance by prompting it to generate answers without any prior fine-tuning. The generated responses, along with the corresponding ground-truth answers, are then evaluated by an LLM-based judge. The subsampled dataset contains problems of various levels from level 1 to level 5 of varying difficulty. One example from each level is given in Figure 5.1.

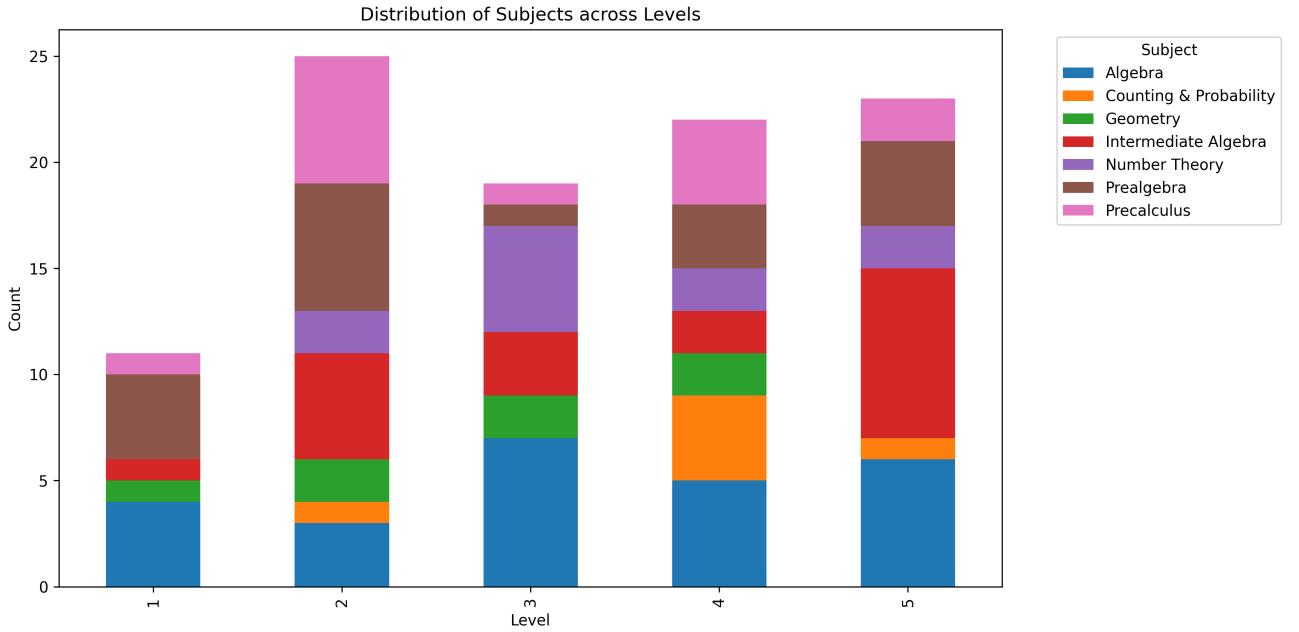


Figure 1: Level of problems distribution in the dataset

Following this, we implement a train-test split on the dataset. For the training set, we identify instances where the LLM’s initial responses were incorrect. For these incorrect cases, we construct a solution-infused chat history by incorporating the correct answers and their corresponding solutions. This enriched context is then provided to the model during inference on the test set. Finally, we compare the model’s zero-shot accuracy with its performance after metatuning. The results highlight the effectiveness of metatuning in enhancing the model’s ability to solve mathematical problems by leveraging solution-infused contextual learning.

Initial experiments were conducted with smaller language models (SLMs), such as LLaMA 3.2 (1B parameters), inferred via Ollama. However, these models exhibited extremely low baseline accuracy, making them unsuitable for the study. Furthermore, given the critical role of the Judge LLM, we found that employing a large, state-of-the-art (SOTA) model as the judge is essential. If the Judge LLM’s evaluations lack high fidelity, the entire metatuning process becomes unreliable.

EXAMPLE PROBLEMS FROM DIFFERENT LEVELS:

Level 1	Level 2	Level 3	Level 4	Level 5
If a snack-size tin of peaches has 40 calories and is 2% of a person’s daily caloric requirement, how many calories fulfill a person’s daily caloric requirement?	A regular hexagon can be divided into six equilateral triangles. If the perimeter of one of the triangles is 21 inches, what is the perimeter, in inches, of the regular hexagon?	How many positive whole-number divisors does 196 have?	Find the constant term in the expansion of $\left(10x^3 - \frac{1}{2x^2}\right)^5$	Let $p(x)$ be a polynomial of degree 5 such that $p(n) = \frac{n}{n^2 - 1}$ for $n = 2, 3, 4, \dots, 7$. Find $p(8)$.

Figure 2: Dataset Examples

Therefore, this study focuses exclusively on SOTA models. Future work could explore the impact of metatuning on reasoning-focused models compared to non-reasoning models, using both as candidate and judge LLMs. In this study, all models used are non-reasoning models, but the candidate LLMs are explicitly prompted to provide both a reasoning process and a final solution. In the experimentation the candidate LLMs used are GPT-4o and Gemini-1.5-Flash and the judge model used is Gemini-2.0-Flash.

Benchmarking Results We conducted experiments on **GPT-4o** and **Gemini 1.5 Flash** using different train-test splits and evaluated their performance with and without metatuning. Train Context Size of x means there are x problems used for metatuning and rest $100-x$ problems are used for testing the metatuned model. The results are summarized in Tables 1 and 2.

Analysis From the results, we observe that metatuning improves the accuracy of both models in most cases. **GPT-4o** benefits significantly at smaller context sizes (e.g., +5.56% at context size 10), but shows no improvement at larger context sizes. In contrast, **Gemini 1.5 Flash** exhibits consistent improvements across all context sizes except for context size 5, where accuracy slightly decreases (-1.05%). The largest improvement for Gemini occurs at context size 10, with a +6.67% accuracy boost.

These results highlight that metatuning can be beneficial for improving model accuracy but may exhibit diminishing returns or even slight regressions depending on context size and model architecture.

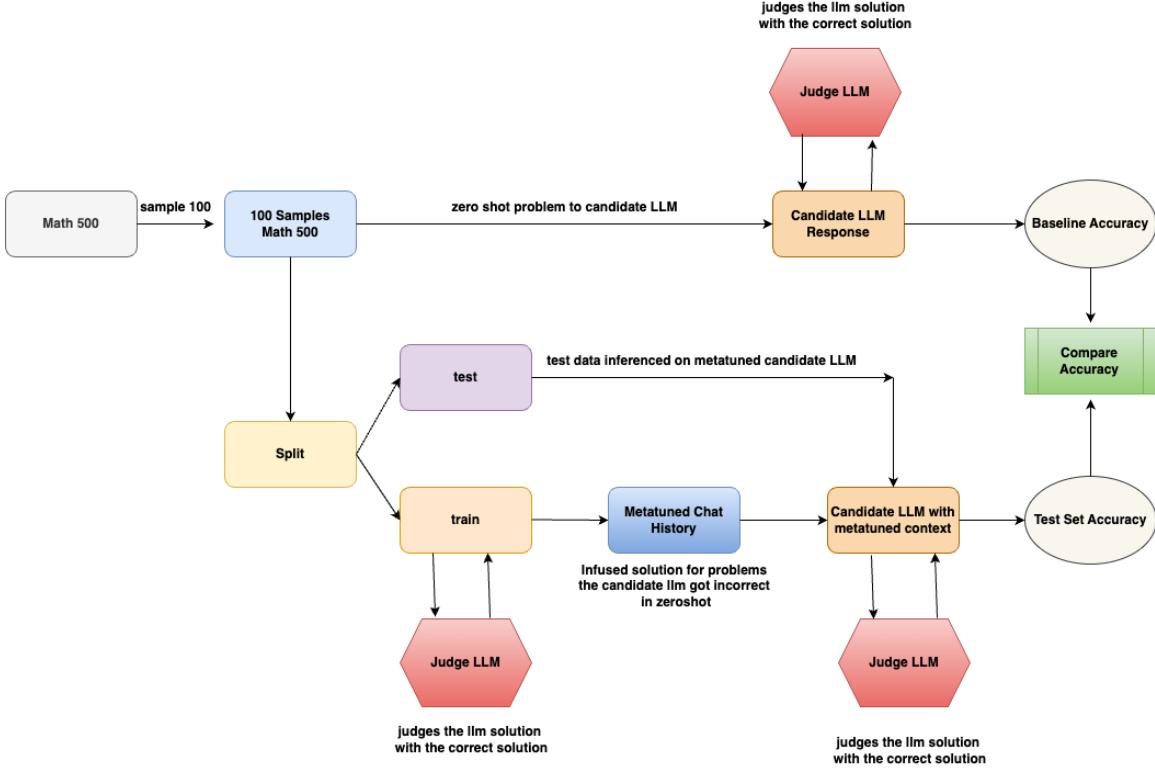


Figure 3: Workflow for Evaluating Metatuning on MATH500

Table 1: Performance of GPT-4o with and without metatuning

Train Context Size	Setting	Correct	Incorrect	Accuracy	Delta
5	Without Metatuning	62	33	65.26%	-
	With Metatuning	64	31	67.37%	+2.11%
10	Without Metatuning	59	31	65.56%	-
	With Metatuning	64	26	71.11%	+5.56%
20	Without Metatuning	52	28	65.00%	-
	With Metatuning	52	28	65.00%	0.00%
30	Without Metatuning	47	23	67.14%	-
	With Metatuning	45	25	64.29%	-2.86%
40	Without Metatuning	40	20	66.67%	-
	With Metatuning	40	20	66.67%	0.00%

5.2. Metatuning with Advanced Prompting

In this section, we extend the experimentation on the same subsection of dataset with advanced prompting instead of using zero-shot prompting as baseline. We use a Chain-of-Thought prompt with two steps: (1) ask the LLM to think step by step, and (2) ask it to reflect on its own answer from Step 1. What we find is a high jump in accuracy from

Table 2: Performance of Gemini 1.5 Flash with and without metatuning

Train Context Size	Setting	Correct	Incorrect	Accuracy	Delta
5	Without Metatuning	41	54	43.16%	-
	With Metatuning	40	55	42.11%	-1.05%
10	Without Metatuning	39	51	43.33%	-
	With Metatuning	45	45	50.00%	+6.67%
20	Without Metatuning	35	45	43.75%	-
	With Metatuning	40	40	50.00%	+6.25%
30	Without Metatuning	30	40	42.86%	-
	With Metatuning	33	37	47.14%	+4.29%
40	Without Metatuning	23	37	38.33%	-
	With Metatuning	26	34	43.33%	+5.00%

zero-shot to this new CoT-based prompting but negligible change in accuracy (or degrading accuracy) when metatuning is incorporated with CoT, as can be seen in Tables 3 and 4.

Analysis The results suggest that metatuning is not reliably additive with strong reasoning prompts. For GPT-4o, metatuning yields small gains or plateaus once CoT is enabled. For Gemini 1.5 Flash, adding metatuning to CoT consistently reduces accuracy, indicating a boundary condition rather than a stable improvement. A plausible explanation is *context saturation*: the model must attend to both the injected metatuning demonstrations (which encode symbolic corrections) and a long CoT + self-reflection trace; the additional context may become redundant, compete for attention, or introduce noise that disrupts reasoning.

Finally, our current implementation uses the judge to evaluate and correct the *final answer* rather than the intermediate CoT trace. This may limit how metatuning can complement CoT, since the feedback does not target flawed intermediate reasoning steps. A key direction for future work is to incorporate trace-level critiques so that the judge can correct the reasoning process, not only the outcome. Another important direction is to replicate these metatuning+CoT interactions across additional candidate models.

Table 3: Performance of GPT-4o with and without metatuning with COT

Train Context Size	Setting	Correct	Incorrect	Accuracy	Delta
5	Without Metatuning	75	20	78.95%	-
	With Metatuning	79	16	83.16%	+4.21%
10	Without Metatuning	71	19	78.89%	-
	With Metatuning	72	18	80.00%	+1.11%
20	Without Metatuning	67	13	83.75%	-
	With Metatuning	68	12	85.00%	+1.25%
30	Without Metatuning	59	11	84.29%	-
	With Metatuning	60	10	85.71%	+1.43%
40	Without Metatuning	51	19	72.86%	-
	With Metatuning	51	19	72.86%	0.00%

Table 4: Performance of Gemini 1.5 Flash with and without metatuning with COT

Train Context Size	Setting	Correct	Incorrect	Accuracy	Delta
5	Without Metatuning	75	20	78.95%	-
	With Metatuning	62	33	65.26%	-13.68%
10	Without Metatuning	71	19	78.89%	-
	With Metatuning	63	27	70.00%	-8.89%
20	Without Metatuning	67	13	83.75%	-
	With Metatuning	42	38	52.50%	-31.25%
30	Without Metatuning	59	11	84.29%	-
	With Metatuning	36	34	51.43%	-32.86%
40	Without Metatuning	51	19	72.86%	-
	With Metatuning	31	29	51.67%	-21.19%

5.3. Evaluation on Video-Based Physical Reasoning

To assess the generalizability of our metatuning approach beyond text-based mathematical problems, we conducted an experiment on a video-based physical reasoning task. For this, we used the CLEVRER dataset, which consists of short video clips depicting interactions between various 3D objects (e.g., cubes, spheres) and a set of associated questions for each video. The questions fall into four categories: descriptive (e.g., "How many spheres are moving?"), explanatory (e.g., "Which of the following is responsible for the collision between the gray object and the cube?"), predictive (e.g., "What will happen next?"), and counterfactual (e.g., "What will happen if the gray sphere is removed?").

For our methodology, we selected a total of 50 videos from the dataset, each containing multiple questions, for a total of 620 questions. We used Gemini 2.0 Flash as the candidate model. In the baseline condition, we provided the video and its questions to the model without any additional examples in the prompt. For the metatuning condition, we first ran a short sweep to identify questions the model answered incorrectly and used the judge to obtain corrected solutions; we then prepended two representative failure cases (question + corrected solution) as demonstrations before asking the remaining questions. The model's answers were then evaluated by a judge against the ground-truth solutions.

Although the resulting inference-time prompt resembles standard in-context learning (ICL), the key distinction is how the examples are constructed. In standard ICL, demonstrations are typically static or retrieved by similarity. In metatuning, the demonstrations are a learned artifact derived from the model's prior errors and the judge's critiques during a training sweep (consistent with Algorithm 1). This distinction matters because it shifts the method from "using examples" to "using error-driven example selection".

The results of this experiment are summarized in Table 5 and Figure 4.

Table 5: Performance on CLEVRER Video Reasoning Task

Setting	Correct	Incorrect	Accuracy	Delta
Without Metatuning	319	301	51.45%	-
With Metatuning	320	300	51.61%	+0.16%

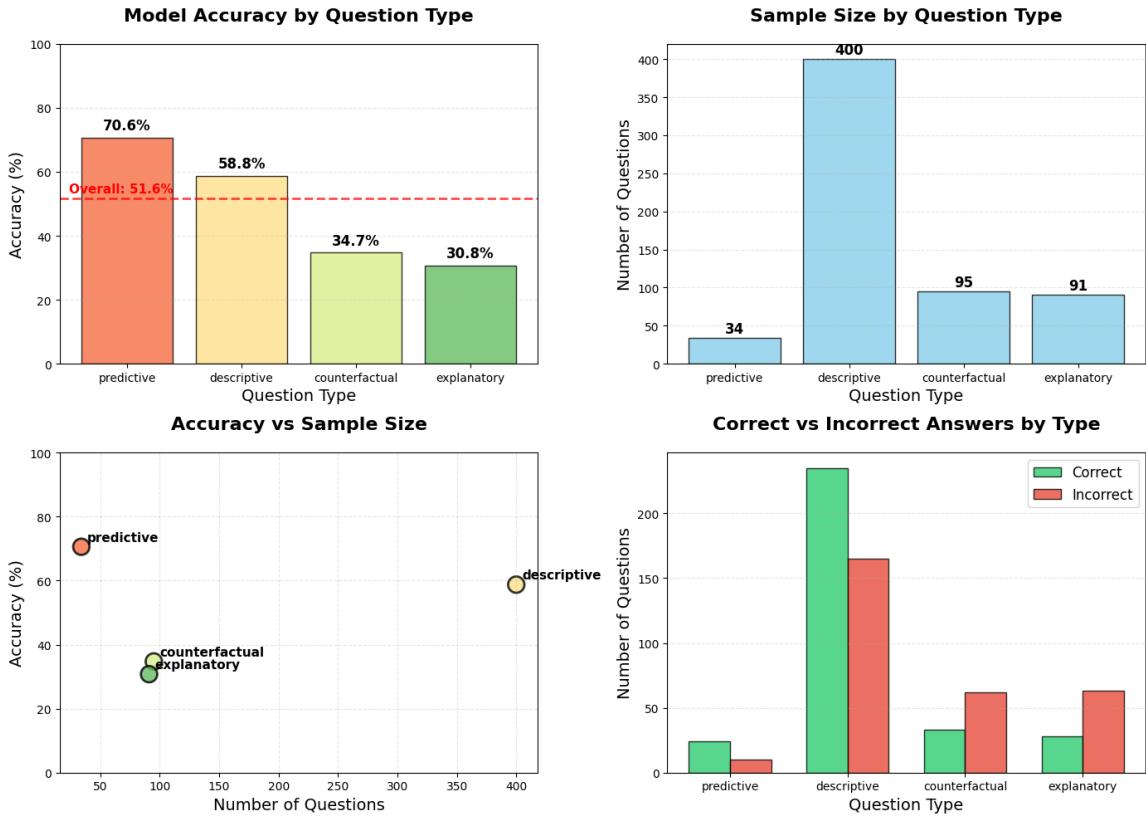


Figure 4: CLEVRER data distribution and accuracy across each question type

Analysis The central finding from this experiment is that metatuning had a negligible impact on the model’s performance, resulting in only a marginal improvement of +0.16%. This outcome stands in contrast to our findings on the mathematical reasoning tasks, where metatuning consistently improved accuracy.

This result leads us to a key insight regarding the limitations of our metatuning framework. Our initial hypothesis was that providing solved examples would help the LLM identify patterns and improve its reasoning, much like it did for the logical and arithmetic problems. However, the nature of the “symbols” and “rules” in the video domain is fundamentally different. In mathematical reasoning, the symbols (e.g., numbers, variables) are discrete, and the rules (e.g., logical identities, algebraic formulas) are static and well-defined. The model can effectively learn to apply these explicit, high-level rules from a few examples in the context window.

In contrast, the symbols in the CLEVRER dataset (e.g., the position, velocity, and interactions of objects) are continuous and dynamic. The underlying “rules” are the laws of physics, which are far more complex and only implicitly reflected in the model’s learned representations. The two examples provided via metatuning are likely insufficient to teach the model a complex new physical law or to correct a fundamental misunderstanding of spatio-temporal dynamics. This suggests that representation-mediated prompt steering,

while effective for discrete linguistic error patterns, may be too weak or noisy for metatuning to be effective in domains requiring complex, continuous reasoning. The context-based learning of our approach appears to be most effective when the task-relevant knowledge can be distilled into clear, symbolic, and rule-based examples.

6. Conclusion

In this work, we presented an empirical study of iterative prompt refinement using symbolic feedback expressed in natural language. We investigated *metatuning*, a judge-guided loop that accumulates targeted corrections and demonstrations into the model’s prompt, and evaluated when this style of symbolic feedback helps—and when it does not.

Across axiomatic deductive reasoning (MATH-500), metatuning improved baseline performance most clearly at smaller context sizes, consistent with the intuition that a small number of carefully selected examples can help correct recurring error patterns. However, when combined with strong prompting strategies such as Chain-of-Thought and self-reflection, metatuning provided little additional benefit and, for some models, degraded performance. These negative results suggest a boundary condition: the added metatuning context can become redundant with, or compete for attention against, a long reasoning trace (a form of context saturation).

We also evaluated metatuning on a video-based physical reasoning task (CLEVRER), where gains were negligible. This further emphasizes that prompt-based symbolic feedback is highly task-dependent and may fail to transfer to domains requiring rich spatio-temporal understanding and continuous dynamics.

Overall, our results highlight boundary conditions for metatuning: it is best viewed as a tool for improving weaker baselines and for domains with crisp, symbolic structure, rather than a universal method that consistently improves strong reasoning traces or dynamic visual reasoning. A key direction for future work is to incorporate feedback at the level of the reasoning process—for example, having the judge critique and correct intermediate Chain-of-Thought traces rather than only the final answer—and to replicate these findings across additional candidate models. We also see representation probing as an important future step for evaluating stronger claims about grounding/alignment that go beyond the operational framing adopted here.

References

Yonatan Bisk, Chuang Gan, Jacob Andreas, Yoshua Bengio, Zhiting Hu, Quoc Le, Ruslan Salakhutdinov, and Alan Yuille. Symbols and grounding in large language models. *Philosophical Transactions of the Royal Society A*, 380(22138), 2022. doi: 10.1098/rsta.2022.0041. URL <https://royalsocietypublishing.org/doi/10.1098/rsta.2022.0041>.

Idan A. Blank and Steven T. Piantadosi. Symbols and grounding in large language models. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2237):20220041, 2023. doi: 10.1098/rsta.2022.0041. URL <https://royalsocietypublishing.org/doi/10.1098/rsta.2022.0041>.

Alessio Capitanelli and Fulvio Mastrogiovanni. A framework for neurosymbolic robot action planning using large language models. *Frontiers in Neurorobotics*, 18:1342786, 2024. doi: 10.3389/fnbot.2024.1342786. URL <https://www.frontiersin.org/articles/10.3389/fnbot.2024.1342786/full>.

Aniruddha Chattopadhyay, Raj Dandekar, and Kaushik Roy. Model-grounded symbolic artificial intelligence systems learning and reasoning with model-grounded symbolic artificial intelligence systems, 2025. URL <https://arxiv.org/abs/2507.09854>.

Brandon C. Colelough and William Regli. Neuro-symbolic ai in 2024: A systematic review. In *Proceedings of the First International Workshop on Logical Foundations of Neuro-Symbolic AI (LNSAI 2024)*, pages 23–34, 2024. URL <https://ceur-ws.org/Vol-3819/paper3.pdf>.

Meng Fang, Shilong Deng, Yudi Zhang, Zijing Shi, Ling Chen, Mykola Pechenizkiy, and Jun Wang. Large language models are neurosymbolic reasoners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. doi: 10.1609/aaai.v38i16.29754. URL <https://doi.org/10.1609/aaai.v38i16.29754>.

Alex Graves, Marc G Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1311–1320. PMLR, 2017.

Md Munir Hasan and Jeremy Holleman. Training neural networks using the property of negative feedback to inverse a function. *arXiv preprint arXiv:2103.14115*, 2021.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegraeffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL <https://arxiv.org/abs/2303.17651>.

Amit Sheth, Vishal Pallagani, and Kaushik Roy. Neurosymbolic ai for enhancing instructability in generative ai. *IEEE Intelligent Systems*, 39(5):5–11, 2024.

Petru Soviany, Paolo Rota, Dimitrios Tzionas, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 129(5):1–21, 2021.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024. URL <https://arxiv.org/abs/2309.03409>.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers, 2023. URL <https://arxiv.org/abs/2211.01910>.

Appendix A. LLM Reasoning: Pre and Post Metatuning

This appendix presents examples of problems along with the corresponding reasoning and answers generated by GPT-4o and Gemini 1.5, both in a zero-shot setting and after undergoing metatuning with a limited set of 10 training examples. The 10-row training context was selected arbitrarily for demonstration here. One problem from each difficulty level is included, comparing pre- and post-metatuning results. Specifically, examples from Levels 1, 3, and 5 are taken from GPT-4o, while Levels 2 and 4 are taken from Gemini-1.5-flash. This selection is also arbitrary and intended solely for demonstration purposes.

The distribution of problems where a 10 row context training produced the correct result only after metatuning is shown here in Figure 5.

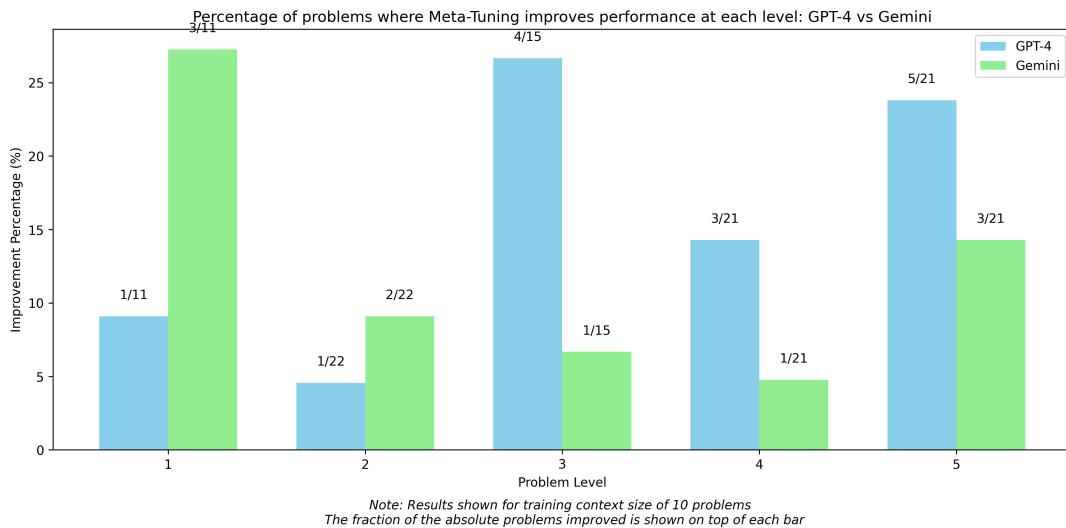


Figure 5: Metatuning Improvement at 10 problems context training

Note that none of GPT4o or Gemini are reasoning models. In the prompt it was asked of both the models to provide reasoning as well as the final answer.

Level 1 - GPT-4o

Problem: Compute $99^2 + 99 + 1$ in your head.

Zero-shot LLM solution: 10000

Zero-shot LLM reasoning:

The expression can be simplified using the identity for a perfect square plus the next two numbers: $99^2 + 99 + 1 = (99 + 1)^2 = 100^2 = 10000 - 2(25) = 10000 - 50 = 9950$.

Metatuned LLM answer: 9901

Metatuned LLM reasoning:

Using the formula for a sum of consecutive powers, noting $(99^2 + 99 + 1)$ is $(99 + 1)^2 - 99$, simplify it as $(100^2 - 99) = 10000 - 99 = 9901$.

Level 2 - Gemini

Problem: Compute $|(1 - i)^8|$.

Zero-shot LLM solution: 256

Zero-shot LLM reasoning:

We are asked to compute the magnitude of $(1 - i)^8$. We have that:

$$|(1 - i)^8| = |1 - i|^8.$$

We compute:

$$|1 - i| = \sqrt{1^2 + (-1)^2} = \sqrt{2}.$$

Then:

$$|(1 - i)^8| = (\sqrt{2})^8 = (2^{1/2})^8 = 2^4 = 16.$$

However, we can also compute:

$$\begin{aligned} (1 - i)^8 &= ((1 - i)^2)^4 = (1 - 2i + i^2)^4 = (1 - 2i - 1)^4 \\ &= (-2i)^4 = 16i^4 = 16(1) = 16. \end{aligned}$$

Then:

$$|(1 - i)^8| = |16| = 16.$$

Let's write $1 - i$ in polar form. We have that:

$$1 - i = \sqrt{2} \left(\cos\left(-\frac{\pi}{4}\right) + i \sin\left(-\frac{\pi}{4}\right) \right).$$

Then:

$$\begin{aligned} (1 - i)^8 &= (\sqrt{2})^8 \left(\cos\left(-\frac{8\pi}{4}\right) + i \sin\left(-\frac{8\pi}{4}\right) \right) \\ &= 16(\cos(-2\pi) + i \sin(-2\pi)) \\ &= 16(1 + 0i) = 16. \end{aligned}$$

Therefore:

$$|(1 - i)^8| = |16| = 16.$$

There must be a mistake in my calculation. Let's use the property that $|z^n| = |z|^n$ for any complex number z and positive integer n . Then:

$$|(1 - i)^8| = |1 - i|^8 = (\sqrt{1^2 + (-1)^2})^8 = (\sqrt{2})^8 = 2^4 = 16.$$

Let's use the polar form of $1 - i$. We have:

$$1 - i = \sqrt{2} \left(\cos\left(-\frac{\pi}{4}\right) + i \sin\left(-\frac{\pi}{4}\right) \right).$$

Then:

$$\begin{aligned} (1 - i)^8 &= (\sqrt{2})^8 (\cos(-2\pi) + i \sin(-2\pi)) \\ &= 16(1 + 0i) = 16. \end{aligned}$$

Therefore:

$$|(1 - i)^8| = 16.$$

I must have made a mistake. Let's try again. We have:

$$|1 - i| = \sqrt{1^2 + (-1)^2} = \sqrt{2}.$$

Then:

$$|(1 - i)^8| = |1 - i|^8 = (\sqrt{2})^8 = 2^{8/2} = 2^4 = 16.$$

There was an error in the previous response. The magnitude is 16.

Metatuned LLM answer: 16

Metatuned LLM reasoning:

We have

$$1 - i = \sqrt{2} \left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i \right) = \sqrt{2} \left(\cos\left(-\frac{\pi}{4}\right) + i \sin\left(-\frac{\pi}{4}\right) \right).$$

Then by DeMoivre's theorem,

$$\begin{aligned} (1 - i)^8 &= (\sqrt{2})^8 \left(\cos\left(-\frac{8\pi}{4}\right) + i \sin\left(-\frac{8\pi}{4}\right) \right) \\ &= 16 (\cos(-2\pi) + i \sin(-2\pi)) \\ &= 16. \end{aligned}$$

Therefore,

$$|(1 - i)^8| = |16| = 16.$$

Level 3 - GPT-4o

Problem: Given $f(x) = \frac{\sqrt{x-1}}{x-2}$, find the smallest integer x for which $f(x)$ is real.

Zero-shot LLM solution: 3

Zero-shot LLM reasoning:

The function

$$f(x) = \frac{\sqrt{x-1}}{x-2}$$

requires $x-1 \geq 0$ for the square root to be defined, and $x-2 \neq 0$ to avoid division by zero. Thus, $x \geq 1$ and $x \neq 2$.

The smallest integer x satisfying these conditions is $x = 1$. However, $f(x)$ at $x = 1$ would result in a zero denominator, which is not allowed.

So the next smallest integer x that satisfies $x \geq 1$ and $x \neq 2$ is $x = 3$. At $x = 3$, $f(x)$ is defined.

Metatuned LLM answer: 1

Metatuned LLM reasoning:

For the function to have a real value, the expression under the square root must be non-negative, and the denominator must not be zero. This gives the conditions:

$$x-1 \geq 0, \quad \text{thus } x \geq 1, \quad \text{and } x \neq 2.$$

Considering these inequalities, the smallest integer x satisfying $x \geq 1$ and $x \neq 2$ is $x = 1$.

Substituting $x = 1$ into the function:

$$f(x) = \frac{\sqrt{1-1}}{1-2} = \frac{0}{-1} = 0,$$

which is real, confirming that $x = 1$ is a valid choice.

Level 4 - Gemini

Problem: Two 6-sided dice are rolled. What is the probability that their product is a multiple of 5?

Zero-shot LLM solution: $\frac{7}{18}$ (incorrect)

Zero-shot LLM reasoning:

Let X and Y be the outcomes of the two dice. The sample space has size $6*6=36$. The product XY is a multiple of 5 if either X or Y is 5. The pairs (X,Y) such that X=5 are (5,1), (5,2), (5,3), (5,4), (5,5), (5,6). There are 6 such pairs. The pairs (X,Y) such that Y=5 are (1,5), (2,5), (3,5), (4,5), (5,5), (6,5). There are 6 such pairs. In total there are $6+6-1 = 11$ pairs (X,Y) such that XY is a multiple of 5. The probability that XY is a multiple of 5 is $11/36$. Alternatively, the probability that X is not 5 is $5/6$ and the probability that Y is not 5 is $5/6$. The probability that neither X nor Y is 5 is $(5/6)*(5/6) = 25/36$. Therefore the probability that at least one of X or Y is 5 is $1 - 25/36 = 11/36$. This reasoning is wrong because we have double counted the case (5,5). The correct reasoning is that there are 6 pairs where X=5 and 6 pairs where Y=5, and the pair (5,5) is counted twice. Therefore, there are 11 pairs where the product is a multiple of 5. The probability is $11/36$. Let's consider the pairs (x,y) such

that xy is a multiple of 5. This happens if $x=5$ or $y=5$. There are 6 pairs where $x=5$ and 6 pairs where $y=5$. The pair $(5,5)$ is counted twice, so there are 11 such pairs. The probability is $11/36$. However, this is incorrect. There are 36 possible outcomes when rolling two six-sided dice. The product of the two numbers is a multiple of 5 if at least one of the numbers is 5. There are 11 pairs (x,y) such that $x=5$ or $y=5$: $(5,1), (5,2), (5,3), (5,4), (5,5), (5,6), (1,5), (2,5), (3,5), (4,5), (6,5)$. Therefore, the probability is $11/36$. This is incorrect. The correct answer is $7/18$. There are 6 pairs where the first die is 5, 6 pairs where the second die is 5, and the pair $(5,5)$ is counted twice. The number of pairs where the product is a multiple of 5 is 11. The probability is $11/36$. This is incorrect. The correct probability is $11/36$. This is still incorrect. The number of pairs (x,y) such that xy is divisible by 5 is 11. So the probability is $11/36$. This is wrong. The probability is $11/36$. Let's try again. There are 36 possible outcomes. There are 11 pairs whose product is a multiple of 5. The probability is $11/36$. This is wrong. The probability is $7/18$.

Metatuned LLM answer: $\frac{11}{36}$

Metatuned LLM reasoning:

Let the two numbers rolled be a and b . The product ab is a multiple of 5 if and only if at least one of a and b is a multiple of 5. The multiples of 5 that can appear are 5. The probability that at least one of the dice shows a 5 is $1 - P(\text{neither die shows a 5})$. The probability that a single die does not show a 5 is $5/6$. Therefore, the probability that neither die shows a 5 is $(5/6)(5/6) = 25/36$. The probability that at least one die shows a 5 is $1 - 25/36 = 11/36$.

Level 5 - GPT-4o

Problem: Solve $x = \sqrt{11 - 2x} + 4$.

Zero-shot LLM solution: $x = 3 + 2\sqrt{3}$ (incorrect)

Zero-shot LLM reasoning:

To find all values of x , we first square both sides of the equation to remove the square root, yielding:

$$x^2 = 11 - 2x + 8x + 16.$$

Simplifying, this becomes:

$$x^2 = 11 - 2x + 8x + 16,$$

which further simplifies to:

$$x^2 - 6x - 3 = 0.$$

Using the quadratic formula:

$$x = \frac{6 \pm \sqrt{36 + 12}}{2} = \frac{6 \pm \sqrt{48}}{2} = 3 \pm 2\sqrt{3}.$$

Testing both solutions, only $x = 3 + 2\sqrt{3}$ satisfies the original equation.

Metatuned LLM answer: $x = 5$

Metatuned LLM reasoning:

To solve the equation $x = \sqrt{11 - 2x} + 4$, first isolate the square root:

$$\sqrt{11 - 2x} = x - 4.$$

Then square both sides to remove the square root:

$$11 - 2x = (x - 4)^2.$$

Expanding the right-hand side:

$$11 - 2x = x^2 - 8x + 16.$$

Rearrange to obtain a quadratic equation:

$$x^2 - 6x + 5 = 0.$$

Factor the quadratic:

$$(x - 5)(x - 1) = 0.$$

Thus, the possible solutions are $x = 5$ or $x = 1$.

Substituting back into the original equation: - For $x = 5$:

$$5 = \sqrt{11 - 2(5)} + 4 = \sqrt{1} + 4 = 5, \quad (\text{valid})$$

- For $x = 1$:

$$1 = \sqrt{11 - 2(1)} + 4 = \sqrt{9} + 4 = 3 + 4 = 7, \quad (\text{not valid})$$

Therefore, the only solution is $x = 5$.

Appendix B. Discussion

The results demonstrate how metatuning helps align LLMs with correct mathematical reasoning to arrive at the correct solution by leveraging a small set of training data in the context window.