

MuTIS: Enhancing Reasoning Efficiency through Multi-Turn Intervention Sampling in Reinforcement Learning

Anonymous EMNLP submission

Abstract

Long chain-of-thought (CoT) reasoning have recently attracted significant attention, with models such as DeepSeek-R1 achieving remarkable performance across various reasoning benchmarks. However, a common challenge to these models is the "overthinking" problem, leading to excessive intermediate steps and diminished inference efficiency. While numerous efforts have targeted reduction in generated tokens, these frequently encounter an inherent trade-off: enhancements in efficiency often come at the cost of degradation in performance. To overcome such challenges, we introduce the **Multi-Turn Intervention Sampling Framework** (MuTIS). Our framework leverages multi-turn interventions within rollouts to produce high-quality, concise reasoning chains. It fine-tunes reasoning models through reinforcement learning, demonstrably surpassing the previously described accuracy-efficiency trade-off. Through extensive experiments on challenging mathematical reasoning benchmarks, our approach achieves a substantial **11.3%** improvement in accuracy while concurrently reducing token utilization by an average of **60.1%**. Code, data, and models will be fully open-sourced.

1 Introduction

The advent of DeepSeek-R1 (Guo et al., 2025) in early 2025 marked a new avenue for efficiently training large language models (LLMs) through reinforcement learning with verifiable rewards (RLVR). For models with large number of parameters (e.g. DeepSeek-R1 671B), long chain-of-thought (CoT) (Wei et al., 2022; Chen et al., 2025) has proven particularly effective in enhancing reasoning capabilities. This improvement is attributed to the capacity of long CoT models for deep reasoning, extensive exploration, self-verification and reflection, particularly the "aha moment" described in R1-Zero.

	Accuracy (% , \uparrow)	#Token (\downarrow)
Qwen2.5-Math-1.5B	22.7	659.65
R1-Distill-Qwen-1.5B	38.8 \uparrow	2471.43 \uparrow
MuTIS(Our Method)	45.1 \uparrow	1217.47 \downarrow

Table 1: To facilitate a fair comparison of token efficiency, we selected problems that **all three models generate final answers correctly** and analyzed their respective token consumption. We calculated the **average accuracy** across five mathematical datasets.

However, recent research indicates that long Chain-of-Thought (CoT) does not show a clear positive correlation with higher accuracy. o1-like models (e.g., deepseek-r1, o1-preview) do not show any advantage over non-o1-like models on the critique abilities (He et al., 2025; Wu et al., 2025). Redundant reflection words often do not positively contribute to the correction of reasoning trajectories. These limitations become particularly pronounced when applying the long CoT paradigm to smaller models (≤ 3 B parameters), a phenomenon termed the *Small Model Learnability Gap* (Li et al., 2025).

We investigate this phenomenon by testing DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025) on five mathematical reasoning benchmarks¹. Our analysis, averaging results across five datasets, revealed that in **46.6%** of the cases (See Figure 4 for details), the distilled model’s outputs became trapped in unproductive loops. In such instances, the model typically cycled through the same reasoning points repeatedly, unable to generate a final answer, thereby simultaneously wasting token resources and failing to provide correct solutions. Furthermore, as shown in Table 1, after distillation from R1, the Qwen2.5-math base model’s token consumption increased **fourfold** even on problems it solved correctly. These observations align with recent research (Ivgi et al., 2024; Sui et al., 2025)

¹Math500, AMC23, OlympiadBench, Minerva, AIME24. See Section 4.1 for details.

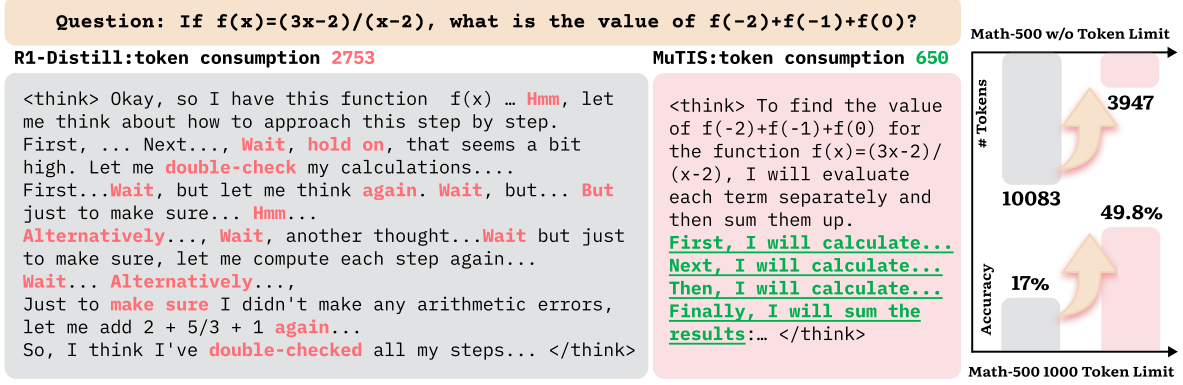


Figure 1: Overview of MuTIS(Our Method). 1) left: A comparative illustration of reasoning chains for the baseline reasoning model versus MuTIS. 2) right: Performance metrics on mathematical reasoning datasets, showcasing MuTIS’s significant reduction in token cost alongside enhancements in accuracy.

and underscore two severe drawbacks inherent in such reasoning models:

- 1) **Repetition.** For complex tasks that reasoning model fails to answer correctly, its output can devolve into unproductive, repetitive sequences.
- 2) **Overthinking.** For comparatively simpler reasoning tasks, these models tend to generate verbose and redundant thinking processes, resulting in significant computational overhead.

We ask the research question: *How do we improve the performance of reasoning models while reducing the unnecessary tokens?* To this end, we introduce a new framework: **Multi-Turn Intervention Sampling (MuTIS)**. As shown in the figure 1, through multi-turn intervention sampling in reinforcement learning, we **curtail the tendencies for repetition and overthinking**, thereby guiding the model towards more concise and effective reasoning. Experimental evaluations demonstrate that, our method achieves **impressive performance gains** over the original model in the challenging math reasoning datasets.

As depicted in Figure 1, a performance improvement of 32.8% was observed under 1K token limits. Furthermore, our approach enhances both accuracy and efficiency in evaluations conducted without a token limit, translating to significant gains in both the computational cost-effectiveness and overall performance of reasoning models. Furthermore, our method possesses excellent **scalability** and **transferability**, as evidenced by its superior performance on larger-parameter models (e.g., 7B) and in out-of-domain (OOD) tasks.

2 Related Work

Efficient Reasoning aims to optimize inference cost for long chain-of-thought (CoT) LLMs while preserving reasoning capabilities, offering practical benefits such as reduced computational costs and improved responsiveness for real-world applications. To address these challenges, researchers have focused on improving token efficiency in long CoT LLMs. TALE-EP (Han et al., 2024) use the LLM itself to estimate a token budget and incorporates it into the prompt to guide more token-efficient responses. RouteLLM (Ong et al., 2024) trains a query router to dispatch incoming queries to suitable LLMs based on complexity. Cheng and Van Durme (2024), Xu et al. (2025), and Geiping et al. (2025) compress textual reasoning steps into fewer latent representations to shorten response lengths. KimiTeam et al. (2025), Sheng et al. (2024), Yeo et al. (2025), and (Aggarwal and Welleck, 2025) integrate a length reward into RL framework. Despite the success of approaches like L1, current methods have yet to leverage the advantages of Multi-turn intervening reasoning.

3 Method

3.1 Multi-turn Markov Decision Process

We employ the Markov Decision Process (MDP) and construct a quadruple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ to model the multi-turn intervention process. In contrast to a single-turn rollout, characterized by a single state transition sequence (s_0, a_0, R_1, s_T) , a multi-turn intervention ex-

Algorithm 1: Multi-Turn Intervention Sampling (MuTIS) RL Training

Inputs: LLM policy $\pi_\theta(y|x)$, intervention prompt \mathcal{IP} , max turns \mathcal{T} , max response length per turn Len_{max} , auxiliary agent Au , Initial Reasoning Task \mathcal{I}

Outputs: Updated policy $\pi_{\theta'}$

```

1  $\mathcal{S}_0 \leftarrow \mathcal{I}$ ; Let  $H_0 \leftarrow \mathcal{S}_0$ ; Let  $\mathcal{H}_{full} \leftarrow \square$ ;
2 for  $t = 1$  to  $\mathcal{T}$  do
3   Let current input  $x_t \leftarrow H_{t-1}$ ;
4   Generate response segment  $\tau'_t$ ;
5   Infer action  $a_t$  from  $\tau'_t$ ;
6   if  $a_t = \text{'provide\_final\_answer'}$  then
7     Extract final answer  $Res$  from  $\tau'_t$ ;
8     Append  $\tau_t$  to  $\mathcal{H}_{full}$ ;
9     Break For loop;
10  else if  $a_t = \text{'ask\_Au'}$  then
11    Extract question  $Q_t$  from  $\tau'_t$ ;
12    Truncate  $\tau'_t$  at the position of  $Q_t$  to get  $\tau_t$ ;
13    Obtain answer  $C_t^{Au}$  from  $Au$ ;
14     $H_t \leftarrow H_{t-1} \oplus \tau_t \oplus Q_t \oplus A_t^{Au}$ ;
15    Append  $(\tau_t \oplus Q_t \oplus C_t^{Au})$  to  $\mathcal{H}_{full}$ ;
16  else
17     $\tau_t \leftarrow \tau'_t$ ;
18     $H_t \leftarrow H_{t-1} \oplus \tau_t \oplus \mathcal{IP}$ ;
19    Append  $(\tau_t \oplus \mathcal{IP})$  to  $\mathcal{H}_{full}$ ;
20   $\mathcal{S}_t \leftarrow H_t$ ;
21  $\mathcal{S}_{final} \leftarrow \bigoplus_{segment \in \mathcal{H}_{full}} segment$ ;
22 Calculate reward  $R_{final}$  and  $\mathcal{L}_{MuTIS}(\theta)$ ;
23  $\theta' \leftarrow f(\theta, \mathcal{L}_{MuTIS}(\theta), R_{final})$ ;
24 return  $\pi_{\theta'}$ 

```

tends a task over a sequence of transitions: $(s_0, a_0, R_1, s_1, a_1, \dots, R_T, s_T)$.

\mathcal{S} : The state representation comprises: 1) H_t : The current dialogue history. 2) C_t : Spontaneous Communication content with auxiliary agents. 3) \mathcal{IP} : Inter-turn prompt perturbations. Besides, \mathcal{H}_{full} denotes the aggregate of all textual components, including model outputs, information from auxiliary agent.

\mathcal{A} : The set of possible actions includes: 1) Continue the thinking process and performing further reasoning. 2) *provide_final_answer*: Provide a quick response with final answer Res . 3) *ask_Au*: Communicate with auxiliary agent Au .

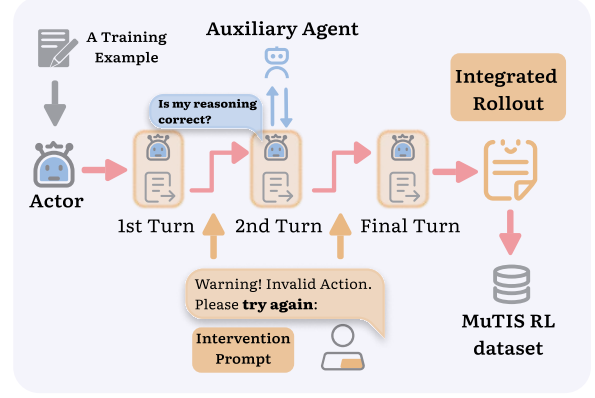


Figure 2: Framework of MuTIS. 1)The LLM initiates a rollout based on the provided mathematical task. 2)If the LLM’s response **exceeds the predefined maximum length**, the rollout is truncated, and an **Intervention Prompt (IP)** is inserted. 3)The truncated response, combined with the IP, forms the input for the LLM to **continue its rollout** in the subsequent turn. 4) During iterative generation, the LLM has the option to **consult an auxiliary agent** to verify its reasoning chain. 5)After multi-turn rollout finished, all outputs from the individual turns are **merged**. 5) The **integrated rollout** is then used to train the model via **reinforcement learning**.

\mathcal{P} : In our multi-turn intervention framework, the model’s state transition matrix \mathcal{P} is perturbed. This perturbation encourages the model to deviate from its initially most probable reasoning trajectory, thereby facilitating transitions to alternative states in subsequent rollouts.

\mathcal{R} : We utilize a rule-based final answer reward, denoted as \mathcal{R}_{acc} , which is focused on the accuracy of the final answer. This approach leads to a sparse reward signal, defined as follows:

$$\mathcal{R}_{final} = \begin{cases} 1 & \text{correct answer within limited turn,} \\ 0 & \text{otherwise.} \end{cases}$$

The sparse nature of \mathcal{R}_{acc} dictates that, at any given time t , the initial $T - t - 1$ terms in the sequence of subsequent immediate rewards (i.e., $r_t, r_{t+1}, \dots, r_{T-1}$) are zero, assuming the final reward occurs at time T

$$G_t = R_{t+1} + \dots + \gamma^{T-t-1} R_T \quad (R_i = 0, i \neq T).$$

3.2 Reinforcement Learning with Multi-Turn Intervention Sampling

The overall pipeline of MuTIS is illustrated in Figure 2. the core of our method is to guide the reasoning model to generate **efficient and concise reasoning chains** for RL training.

Reinforcement Learning. We employ the Proximal Policy Optimization (PPO) algorithm for training and adopt a rule-based accuracy reward, which is granted solely based on the correctness of the final answer. Thus, a reward of 1 is received if the model outputs the correct final answer within the predefined turn limit. Conversely, the reward is 0 if the model either fails to respond within this limit or provides an incorrect answer.

Multi-turn Rollout. To achieve concise reasoning, we enforce a maximum response length of 2000 tokens for each rollout. If a model’s output surpasses this limit, its rollout is forcibly terminated, and the model receives the intervention prompt: **"Warning! Your previous action is invalid. Please try again:"**. Following this intervention, the model is allowed to continue its response, effectively resuming from the point of interruption. This iterative process repeats until the model provides a final answer or the dialogue exceeds a predefined maximum number of turns.

We design a **flexible multi-turn termination logic**. When the model outputs predefined terminal response tokens, such as “final answer“, boxed, or the <answer> tags. MuTIS promptly detects these and truncates the rollout at that position, whereupon the Multi-Turn Intervention also ceases. This design facilitates the capture of fine-grained model states and prevents superfluous rollout turns after an answer has been generated.

Communication Mechanism. We supplement the maximum response length constraint with an "Auxiliary Agent". When the model encounters difficulties during training and requires assistance, it can communicate with this auxiliary agent to receive guidance. We have implemented a detection mechanism for special tokens. During a rollout, if the LLM utilizes an **<ask> tag**, our system detects this and invokes **gpt-4o mini** to act as an Auxiliary Agent. This Auxiliary Agent addresses the specific query embedded by the LLM within the <ask> tag and returns a corresponding answer. The provided answer is then encapsulated in <communicate> tags and inserted into the ongoing rollout. Moreover, inspired by the design of Search-R1 (Jin et al., 2025), any external information, such as the Auxiliary Agent’s response, is **masked** during the loss calculation. This masking prevents interference with the optimization of the LLM’s parameters throughout the RL training phase.

Prompt design. To evaluate our experiment’s sensitivity to prompt formulation, we designed and tested five distinct prompt variants. Specific details of the prompt designs are provided in the Appendix B.

We introduce **two variants** of our method: "multi-turn intervention with ask" (**MuTIS-Ask**), which allows the model to request help from the Auxiliary Agent, and "multi-turn intervention without ask" (**MuTIS**), which does not include this feature.

4 Experiment

4.1 Experiment Setup

MuTIS is implemented using the veRL (Sheng et al., 2024) reinforcement learning framework. For the multi-turn generation design, we reference the codebase from Search-R1 (Jin et al., 2025).

Training Dataset. Our primary training data was derived from the "default" partition of the OpenR1-Math-220k (Face, 2025) dataset, which initially comprised over 90,000 samples. We applied several filtering criteria to refine this dataset. A detailed description of the filtering process is provided in the Appendix A.

Additionally, inspired by recent studies suggesting great benefits of smaller datasets for model training (Muennighoff et al., 2025; Ye et al., 2025). However, these researches mainly focus on Supervised Fine-Tuning (SFT). We aimed to investigate the impact of RL training across different data scales. Therefore, we conducted a separate set of experiments using 817 data points from the LIMO (Ye et al., 2025) as training data, with 10% of these reserved for validation. Our experiments demonstrated that training on both datasets yielded **comparably strong performance**. However, in terms of training dynamics, the LIMO dataset showed a **faster convergence rate** than open-r1. Consequently, models derived from training on each of these datasets were selected for subsequent experiments. Further experimental details are provided in Appendix A.

Evaluation. We assessed our method and baseline models on five math reasoning benchmarks. To further assess model generalization, we also included evaluations on out-of-domain multiple-choice question datasets. The dataset versions used were aligned with those available in the LIMO

	Accuracy (% , \uparrow)					# Tokens (\downarrow)				
	MATH500	AMC23	Olympiad	Minerva	AIME24	MATH500	AMC23	Olympiad	Minerva	AIME24
DeepSeek-R1-Distill-Qwen-1.5B										
R1-Distill (2025)	69.4	55.0	28.9	23.9	16.7	10083	15927	20686	14410	25549
Be Concise (2024)	69.8	47.5	32.2	23.5	16.7	8818	17773	18535	9639	26130
Fixed Budget (2024)	69.8	52.5	30.2	22.8	16.7	9753	17648	20518	13075	24376
MuTIS (Ours)	74.6	62.5	40.2	29.4	30.0	3060	5847	8248	2586	13640
MuTIS-Ask (Ours)	76.8	62.5	37.2	29.0	20.0	3947	8411	10505	1856	17429
DeepScaleR-1.5B-Preview										
DeepScaleR (2025)	78.8	65.0	45.0	34.2	30.0	6586	9335	13015	11810	19051
Be Concise (2024)	78.4	72.5	45.2	30.5	26.7	5861	9183	11848	9691	17471
Fixed Budget (2024)	81.0	67.5	43.1	32.0	36.7	5351	8298	12686	10148	18898
L1-Exact (2025)	83.8	67.5	46.2	39.0	26.7	2243	2236	3162	3489	2771
MuTIS (Ours)	84.8	70.0	48.9	35.7	26.7	3564	5809	8667	5647	14892
MuTIS-Ask (Ours)	84.6	75.0	46.8	30.9	36.7	2703	6298	6280	3594	12556

Table 2: This Table provides a visual comparison of **accuracy and efficiency** between MuTIS and baseline methods on mathematical reasoning benchmarks. All evaluations were conducted using an **identical framework** and consistently aligned hyperparameters to ensure a fair comparison.

repository. We use **greedy decoding** for all evaluations, which introduces no randomness in the outputs. Consequently, the same answer is obtained regardless of the random seed, ensuring that all reported data correspond to results from a **single sampling pass**. Further evaluation details can be found in Appendix A.

Our experiments are conducted on R1-Distill Models (Guo et al., 2025) and DeepScaleR-1.5B-Preview (Luo et al., 2025).

Baseline. We compared our method against two types of baselines: one comprising approaches based on test-time prompt optimization, and the other concurrent related methods.

1. **Be Concise** (Renze and Guven, 2024): It appends the phrase "be concise" to the base prompt.
2. **Fixed Budget** (Nayab et al., 2024): Prompt the model to "limit the answer length to [SOME NUMBER] words." We adopted a similar approach, augmenting the base prompt by adding the following instruction: "The final answer is output before the maximum number of tokens (max_tokens) is used:"
3. **L1-Exact** (Aggarwal and Welleck, 2025) applies a reinforcement-learning objective that combines a correctness reward with an exact-length penalty to optimize both performance and efficiency.

4.2 Evaluation Results

As indicated in Table 2, MuTIS simultaneously improves accuracy while significantly reducing token

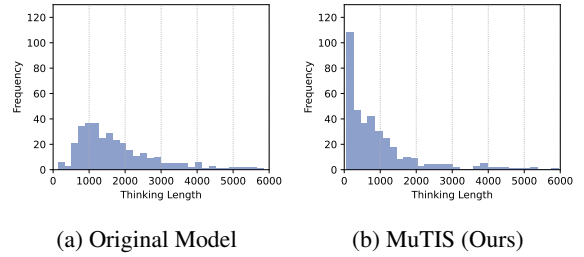


Figure 3: (a) and (b) present an analysis of "thinking length" distributions on Math-500, specifically for **correctly answered problems**. (a) shows the distribution for the R1-distill (baseline) model, while (b) depicts the distribution for the same model after MuTIS RL training

consumption. Furthermore, when evaluated on five mathematical reasoning datasets, the model consistently exhibited performance enhancements to a notable degree across all of them. In the Olympiad-bench dataset, our approach achieves a substantial 11.3% improvement in accuracy while concurrently reducing token utilization by an average of 60.1%.

Enhanced Response Succinctness for Correct Solutions. As shown in the table 1. For problems where both the baseline and our model provided correct answers, our method demonstrated a remarkable capability for response compression, yielding more concise yet accurate solutions.

Refinement of Thinking Phase. Recent efforts to enhance the efficiency of reasoning models have largely focused on optimizing their Thinking Phase. Muennighoff et al. (2025) employ test-time scaling to allocate predefined token budgets, while Ma et al. (2025) directly bypass the thinking process via sim-

	Accuracy (% , \uparrow)			#Token (\downarrow)		
	MATH500	AMC23	Olympiad	MATH500	AMC23	Olympiad
DeepSeek-R1-Distill-Qwen-7B						
R1-Distill	86.4	67.5	44.3	5053	9178	12061
MuTIS (Ours)	87.4	77.5	54.1	2377	3296	5966

Table 3: MuTIS demonstrates **superior scalability on 7B models**, with the figure presenting a comparison of accuracy and efficiency between the DeepSeek-R1-Distill-Qwen-7B and MuTIS.

	Accuracy (% , \uparrow)			#Token (\downarrow)		
	MMLU	GPQA	R-Bench	MMLU	GPQA	R-Bench
DeepSeek-R1-Distill-Qwen-1.5B						
R1-Distill	33.6	33.8	27.9	13895	27880	31043
MuTIS (Ours)	39.2	33.3	29.7	13677	22553	28139
DeepScaleR-1.5B-Preview						
DeepScaleR	48.2	36.4	30.2	12108	22922	25653
MuTIS (Ours)	43.7	36.9	36.6	9534	11172	17836

Table 4: MuTIS exhibits stability when applied to both out-of-domain (OOD) data and across diverse data formats.

ple prompting. In line with these research directions, we analyzed the behavioral changes within the thinking phase of MuTIS.

As depicted in Figure 3, MuTIS exhibits a substantially reduced thinking length compared to the original model. For 93.2% of tasks, MuTIS completes the Thinking Phase using fewer than 1000 tokens, achieving an accuracy of 82.4% on these tasks. This performance strongly demonstrates MuTIS’s capability for concise and accurate reasoning.

Strong scalability with large-parameter models.

To validate the scalability of our method, we extended the MuTIS RL Training Pipeline to larger models(DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025)). As presented in the table 3, the experimental results demonstrate that MuTIS achieves similarly significant improvements on these larger-parameter models: reasoning efficiency is enhanced, token consumption is markedly reduced, and dataset accuracy is increased. For instance, on OlympiadBench, MuTIS boosted performance by 9.8% while decreasing token consumption by 50%. These findings illustrate the superior and scalable performance of MuTIS across models of varying parameter sizes. For more detailed experimental details, please refer to Appendix A

Out-of-Domain Transferability. The enhancements in reasoning capabilities fostered by MuTIS also demonstrate generalization to different task formats and transferability to out-of-domain tasks.

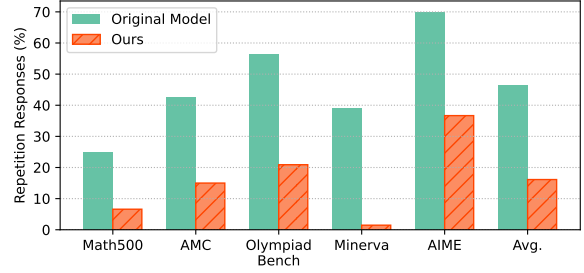


Figure 4: This figure illustrates the frequency of the “Repetition” phenomenon across various datasets. Our method is represented by the right-hand hatched bars, while the original reasoning model is represented by the left-hand bars.

Despite being trained exclusively on mathematical generation tasks, models exhibited strong generalization performance on multiple-choice question (MCQ) datasets. We test our method on MMLUPro (Hendrycks et al., 2021a,b) (Math MCQs), GPQA (Rein et al., 2023) (graduate-level multi-disciplinary MCQs) and R-Bench (Li et al., 2024) (graduate-level multi-disciplinary MCQs) As shown in the table 4 This was evidenced by slightly improvements in accuracy and reductions in token consumption on these out of domain and format tasks. Moreover, its training regimen, primarily focused on mathematical reasoning tasks, does not significantly compromise performance on other tasks and, in some instances, even enhances it. It indicates a more efficient and fundamentally improved reasoning process that transcends the specific training task format.

4.3 Reasoning Under Token Constraints

We conducted evaluations under varying **maximum generation token constraints**, forcing the model to complete its reasoning and generate a response within the token limit. As shown in Figure 5, comparative analysis across multiple datasets reveals that MuTIS significantly outperforms the original reasoning model.

The original model’s accuracy typically commences its improvement only after the token exceeds approximately 500. This behavior suggests the existence of a significant “**Effective Token Threshold**”—a point that must be reached for the model to complete its reasoning process and generate an answer. Consequently, most tasks require a substantial token budget, often considerably surpassing this 500-token baseline, for successful

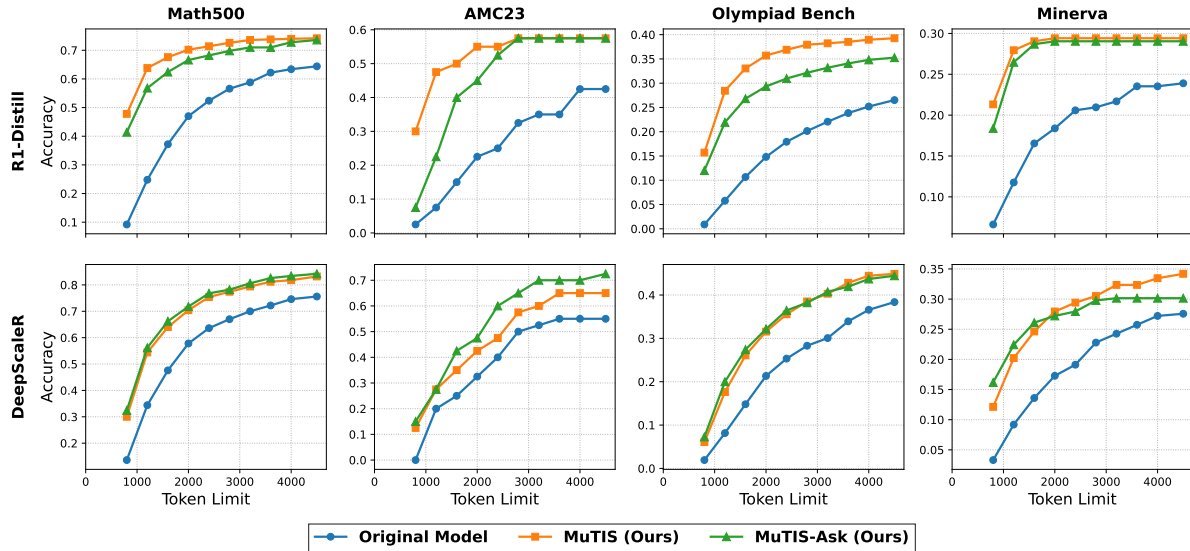


Figure 5: The figure compares the **accuracy** of MuTIS with two baseline models: R1-Distill (first row) and DeepScaleR (second row). All evaluations presented were conducted under **identical token limit settings**.

execution. In stark contrast, MuTIS demonstrates significant performance gains even with highly restricted token budgets. For instance, on the Math-500 dataset, MuTIS achieves over 40% accuracy using only 800 tokens.

4.4 Mitigation of Repetition Issues

As shown in the Figure 4, MuTIS substantially mitigates the incidence of "ineffective loops"—a phenomenon where models generate excessively long, non-productive responses when failing to solve a problem. Consequently, the proportion of responses truncated due to exceeding the default maximum token limit (typically 32,768 tokens in standard evaluations) was markedly reduced from 46.6% to 16.1 %. This provides strong evidence that our method effectively mitigates the "Repetition" problem across most scenarios. Further details regarding the evolution of response length after MuTIS training are provided in the Appendix.

4.5 Ablation Study

Pivotal Role of Multi-turn. Our method’s core philosophy is to utilize **Multi-Turn** Interventions to influence the model’s reasoning trajectory, thereby steering reinforcement learning (RL) optimization towards more effective and efficient policy space regions.

To assess the specific contribution of our method’s multi-turn interaction, we conducted a controlled

	Accuracy (% , \uparrow)			#Tokens (\downarrow)		
	MATH 500	AMC23	Olympiad	MATH 500	AMC23	Olympiad
R1-Distill	69.4	55.0	28.9	10083	15927	20686
Single-turn	67.0	52.3	30.2	1483	4072	4688
3-turn (Ours)	76.8	62.5	37.2	3947	8411	10505
5-turn	67.8	45.0	32.0	665	1580	2096

Table 5: Our ablation studies ensured a consistent total length across varied experimental configurations, fixing the overall token limit at 6000. This was achieved through setups such as 3 turns with a 2000-token limit each (3×2000), a single 6000-token turn (1×6000), and 5 turns with a 1200-token limit each (5×1200)

ablation study. To ensure fairness and isolate the iterative impact, the single-turn baseline also received an Intervention Prompt (IP) post-interaction. This design enables precise analysis of the multi-turn engagement’s pivotal role in the observed performance benefits.

As shown in the Figure 5, While a single-turn setting significantly reduces token consumption, it **slightly reduces accuracy**. A 3-turn setup substantially boosts accuracy compared to the single-turn approach, though token consumption increases. Conversely, further increasing the number of turns to five can again lower token consumption, but this often leads to a decline in accuracy. Our analysis indicates that with a 5-turn setup constrained by a tight 1200-token per-turn limit, the model experiences **excessive intervention**, which adversely impacts its performance.

This ablation study across different turn configura-

Keyword	Original Model	MuTIS	MuTIS-Ask
wait	8.73	0.97	3.15
hold on	0.18	0.00	0.01
but	10.92	3.63	5.00
not sure	0.21	0.06	0.46
maybe	3.51	0.71	1.43
double-check	0.07	0.10	0.10
think again	0.09	0.01	0.17
alternatively	2.03	0.84	1.08
another idea	0.11	0.02	0.08
another approach	0.04	0.03	0.04

Table 6: This table illustrates the difference in the frequencies of the reflection words between the original R1-Distill-1.5B model and the two variants of MuTIS. The frequencies are counted as the **average times of occurrence every 1000 tokens** in responses.

tions demonstrates that the 3-turn design ultimately chosen for MuTIS achieves **an optimal balance** between accuracy and token consumption. It appears to exert an "appropriate level of intervention" on the model's rollouts, thereby fostering both effective and efficient reasoning.

5 Discussion

5.1 Reflection Words in Reasoning models

Do small inference models really need tons of reflection words?

Research on DeepSeek-R1-Zero (Guo et al., 2025) have shown that reflection words like "Wait" are important markers of self-verification in reasoning models. However, as shown in Table 6, our experimental results on smaller models show that such self-reflection words, including "Wait", **decrease** significantly during the MuTIS training process. Concurrently, the model's reasoning becomes more concise, and its performance under limited token conditions improves. This suggests that these reflection words are substantially redundant. While existing research has documented "Superficial Reflection" behavior (Liu et al., 2025) in base models like Qwen2.5-Instruct, our experiments reveal that reasoning models exhibit a form of self-verification that can be characterized as **"Ineffective Noise."**

5.2 Behavior Analysis in the Reinforcement Learning Process

After MuTIS intervenes to guide models toward generating concise reasoning chains, it primarily employs RL to optimize LLM parameters.

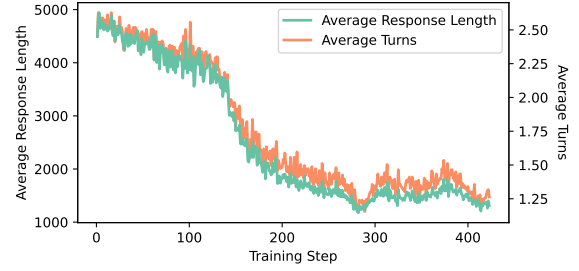


Figure 6: This figure illustrates the progression of both response length and the number of response turns for the deepscaler-1.5B model during MuTIS RL training

Consequently, we further analyzed the behavioral changes exhibited by the models during this RL process. As depicted in figure 6, the average response length of models undergoing MuTIS's RL process **steadily decreases**, from an initial 5000 tokens to approximately 1500 tokens. Concurrently, the average number of multi-turn iterations drops from an original 2.5 to around 1.25. This indicates that while original models struggle under strict token constraints, models trained with MuTIS learn to provide concise answers within a minimal number of turns.

A recent study posited that RL does not fundamentally expand a model's capability boundaries (Yue et al., 2025) but rather increases the probability of accessing pre-existing correct states within its search space. This implies that RL predominantly helps models solidify their conviction in effective reasoning paths. Our experimental findings with MuTIS support this perspective: RL's role in making responses increasingly concise demonstrates its efficacy in enabling rapid convergence within the model's search space. This process embodies the model **shifting from self-doubt to firm conviction**.

6 Conclusion

We introduce a novel Multi-Turn Intervention Sampling (MuTIS) approach for RL training. This method innovatively employs multi-turn rollouts and incorporates dual guidance – from an Intervention Prompt and an Auxiliary Agent – to steer models toward generating **high-quality, concise reasoning chains**. Our experiments indicate simultaneous improvements in both accuracy and efficiency.

Limitations

We demonstrate that training small reasoning models with multi-turn intervening sampling achieves effective reasoning. While computational constraints prevented us from exploring the full potential of the method on larger models (e.g., 32B models), future work will focus on extending our approach for enhanced generalization and wider applicability.

During training, our method’s response length can significantly fluctuate before ultimately stabilizing. This suggests that effective KL divergence constraints could be important for achieving more stable training dynamics in our Multi-turn Intervention process.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

AMC. American Mathematics Competitions. 2025a. American Invitational Mathematics Examination (AIME)). https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination. Accessed: 2025-05-19.

AMC. American Mathematics Competitions. 2025b. American Mathematics Competitions (AMC). <https://maa.org/student-programs/amc/>. Accessed: 2025-05-19.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.

Hugging Face. 2025. [Open r1: A fully open reproduction of deepseek-r1](#).

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. [The language model evaluation harness](#).

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. 2025. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiad-bench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. 2025. [Can large language models detect errors in long chain-of-thought reasoning?](#)

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Maor Ivgi, Ori Yoran, Jonathan Berant, and Mor Geva. 2024. From loops to oops: Fallback behaviors of language models under uncertainty. *arXiv preprint arXiv:2407.06071*.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

KimiTeam, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

586	Chunyi Li, Jianbo Zhang, Zicheng Zhang, Haoning Wu,	Matthew Renze and Erhan Guven. 2024. The benefits of	641
587	Yuan Tian, Wei Sun, Guo Lu, Xiaohong Liu, Xiongkuo	a concise chain of thought on problem-solving in large	642
588	Min, Weisi Lin, et al. 2024. R-bench: Are your large	language models. In <i>2024 2nd International Conference</i>	643
589	multimodal model robust to real-world corruptions?	on <i>Foundation and Large Language Models (FLLM)</i> ,	644
590	<i>arXiv preprint arXiv:2410.05474</i> .	pages 476–483. IEEE.	645
591	Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang,	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu,	646
592	Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubrama-	Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan	647
593	nian, and Radha Poovendran. 2025. Small models	Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath:	648
594	struggle to learn from strong reasoners. <i>arXiv preprint</i>	Pushing the limits of mathematical reasoning in open	649
595	<i>arXiv:2502.12143</i> .	language models. <i>arXiv preprint arXiv:2402.03300</i> .	650
596	Hunter Lightman, Vineet Kosaraju, Yuri Burda, Har-	Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-	651
597	rison Edwards, Bowen Baker, Teddy Lee, Jan Leike,	Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell,	652
598	John Schulman, Ilya Sutskever, and Karl Cobbe. 2023.	Subhro Das, David Cox, and Chuang Gan. 2025. Satori:	653
599	Let’s verify step by step. In <i>The Twelfth International</i>	Reinforcement learning with chain-of-action-thought	654
600	<i>Conference on Learning Representations</i> .	enhances llm reasoning via autoregressive search. <i>arXiv</i>	655
601	Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang,	<i>preprint arXiv:2502.02508</i> .	656
602	Chao Du, and Min Lin. 2025. There may not be aha	Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin	657
603	moment in rl-zero-like training — a pilot study. https://oatllm.notion.site/oat-zero . Notion Blog.	Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin,	658
604		and Chuan Wu. 2024. Hybridflow: A flexible and effi-	659
605	Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi,	cient rlhf framework. <i>arXiv preprint arXiv:2409.19256</i> .	660
606	William Tang, Manan Roongta, Colin Cai, Jeffrey Luo,	Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu	661
607	Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion	Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew	662
608	Stoica. 2025. Deepscaler: Surpassing o1-preview with	Wen, Shaochen Zhong, Hanjie Chen, et al. 2025. Stop	663
609	a 1.5b model by scaling rl. https://pretty-radio-b	overthinking: A survey on efficient reasoning for large	664
610	75.notion.site/DeepScaleR-Surpassing-O1-Pre	language models. <i>arXiv preprint arXiv:2503.16419</i> .	665
611	view-with-a-1-5B-Model-by-Scaling-RL-19681		
612	902c1468005bed8ca303013a4e2 . Notion Blog.	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten	666
613	Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs,	Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al.	667
614	Sewon Min, and Matei Zaharia. 2025. Reasoning mod-	2022. Chain-of-thought prompting elicits reasoning in	668
615	els can be effective without thinking.	large language models. <i>Advances in neural information</i>	669
616	math-ai. 2025. Minervamath dataset. https://hu	<i>processing systems</i> , 35:24824–24837.	670
617	ggingface.co/datasets/math-ai/minervamath .	Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka,	671
618	Accessed: 2025-05-19.	and Yisen Wang. 2025. When more is less: Under-	672
619	Niklas Muennighoff, Zitong Yang, Weijia Shi, Xi-	standing chain-of-thought length in llms. <i>arXiv preprint</i>	673
620	ang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke	<i>arXiv:2502.07266</i> .	674
621	Zettlemoyer, Percy Liang, Emmanuel Candès, and Tat-	Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao.	675
622	sunori Hashimoto. 2025. s1: Simple test-time scaling .	2025. Softcot: Soft chain-of-thought for efficient rea-	676
623	Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea	soning with llms. <i>arXiv preprint arXiv:2502.12134</i> .	677
624	Saracino, Giorgio Buttazzo, Nicolamaria Manes, and	Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie	678
625	Fabrizio Giacomelli. 2024. Concise thoughts: Impact of	Xia, and Pengfei Liu. 2025. Limo: Less is more for	679
626	output length on llm reasoning and cost. <i>arXiv preprint</i>	reasoning. <i>arXiv preprint arXiv:2502.03387</i> .	680
627	<i>arXiv:2407.19825</i> .	Edward Yeo, Yuxuan Tong, Morry Niu, Graham	681
628	Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin	Neubig, and Xiang Yue. 2025. Demystifying long	682
629	Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed	chain-of-thought reasoning in llms. <i>arXiv preprint</i>	683
630	Kadous, and Ion Stoica. 2024. Routellm: Learning	<i>arXiv:2502.03373</i> .	684
631	to route llms from preference data. In <i>The Thirteenth</i>	Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan,	685
632	<i>International Conference on Learning Representations</i> .	Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu,	686
633	Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan	Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole	687
634	Yuan, Hao Peng, and Alane Suhr. 2025. Tinyzero.	Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mo-	688
635	https://github.com/Jiayi-Pan/TinyZero . Accessed:	fan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze	689
636	2025-01-24.	Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan	690
637	David Rein, Betty Li Hou, Asa Cooper Stickland, Jack-	Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing	691
638	son Petty, Richard Yuanzhe Pang, Julien Dirani, Julian	Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao,	692
639	Michael, and Samuel R. Bowman. 2023. Gpqa: A	Yonghui Wu, and Mingxuan Wang. 2025. Dapo: An	693
640	graduate-level google-proof q&a benchmark .	open-source llm reinforcement learning system at scale .	694

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. [Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?](#)

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. [Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild.](#)

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2025. [Genprm: Scaling test-time compute of process reward models via generative reasoning.](#)

Appendix

A Experiment Details

A.1 Dataset filtering details

- Remove multiple-choice questions (MCQs). To focus on the model’s ability to generate answers rather than merely select them, thereby providing a more rigorous assessment of its reasoning capabilities, all MCQs were excluded.
- Remove questions with overly long (≥ 55 tokens) answers. We observed that some answers in the original dataset had non-standard formatting or contained excessive descriptive language. Such answers are challenging to evaluate accurately using a rule-based reward system.
- Remove questions with multiple answers or involving multiple variables. the presence of multiple valid answers complicates the extraction and comparison process during evaluation, potentially leading to mismatches that can negatively impact training.

Following these filtering steps, our final training dataset consisted of over 60,000 samples. From this, 0.5% was allocated as a dedicated validation set to monitor model performance throughout the training process.

A.2 Evaluation Details

We assessed our method and baseline models on the following five math reasoning benchmarks: Math-500 ([Lightman et al., 2023](#)), AIME 2024 ([AMC. American Mathematics Competitions, 2025a](#)), AMC23 ([AMC. American Mathematics Competitions, 2025b](#)), Olympiadbench ([He et al., 2024](#)), Minerva ([math-ai, 2025](#))

The dataset versions used were aligned with those available in the LIMO repository. We use greedy decoding for all evaluations, which introduces no randomness in the outputs. Consequently, the same answer is obtained regardless of the random seed, ensuring that all reported data correspond to results from a single sampling pass.

Our mathematical reasoning evaluation also leveraged LIMO’s evaluation framework, whose methodology is primarily derived from Qwen2.5-Math. This framework employs a rule-based assessment to determine answer correctness, without relying on model-based judgments.

For MCQ tasks, we predominantly utilized the Im-eval ([Gao et al., 2024](#)) framework, as LIMO’s evaluation framework offers limited support for these types of evaluations.

A.3 Experiment Model

Our experiments are conducted on DeepSeek-R1-Distill-Qwen-1.5B ([Guo et al., 2025](#)), DeepScaleR-1.5B-Preview ([Luo et al., 2025](#)), and DeepSeek-R1-Distill-Qwen-7B ([Guo et al., 2025](#)). Given the original reasoning model’s already strong mathematical problem-solving capabilities, coupled with our research emphasis on efficiency, we also included it as a key baseline for performance comparison.

A.4 Analysis of Responses Length

Figure 7 shows the generation length histogram of MuTIS and the original DeepSeek-R1-Distill-Qwen-1.5B model on Math500 dataset. It demonstrates that MuTIS evidently mitigates the overthinking problems (shown by the overall distribution) and the repetition issues (shown by the red part of the rightmost bar).

A.5 Detailed Results on Large-Parameter Models

Figure 8 shows the comparison of accuracy under token limits between the original DeepSeek-R1-Distill-Qwen-7B and our MuTIS.

A.6 further discussion on Reflection Word

The advent of sophisticated reasoning models, exemplified by OpenAI o1 ([Jaech et al., 2024](#)) and DeepSeek-R1 ([Guo et al., 2025](#)), has catalyzed a research emphasis on long Chain-of-Thought

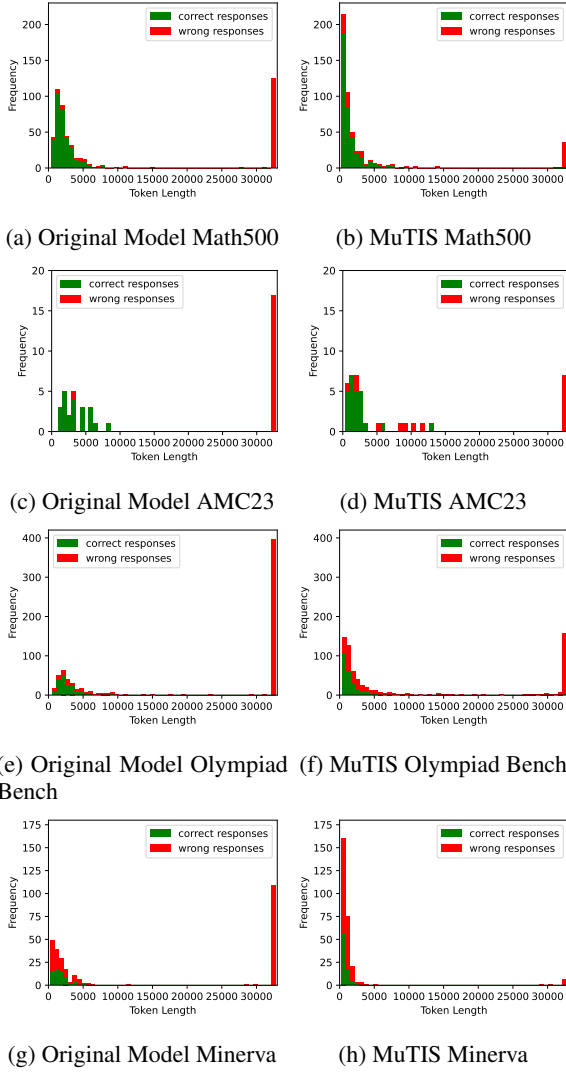


Figure 7: Generation Length

(CoT) methodologies as a primary target for optimizing model training. Nevertheless, contemporary studies indicate a prevalent "OverThinking" phenomenon within these models, characterized by excessive or non-productive cognitive steps.

Table 1 illustrates that original reasoning models often introduce significant redundancy. In contrast, our optimization (MuTIS) not only further improves accuracy but also concurrently reduces token consumption. This demonstrates that the Chain-of-Thought (CoT) in such reasoning models contains many unnecessary steps. Indeed, analysis of MuTIS's post-training reasoning CoT, reveals a significant reduction in "reflection words"—terms frequently occurring in standard distilled models.

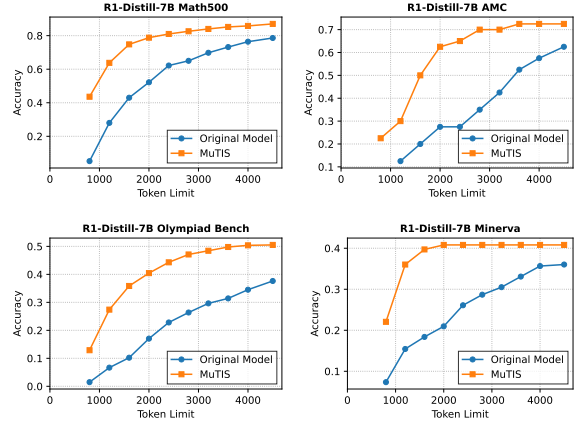


Figure 8: Accuracy vs Token Limits on 7B models. The original model is DeepSeek-R1-Distill-Qwen-7B and the MuTIS is trained on it.

B Prompt Design

B.1 Chat Template Design

We employed a system prompt inspired by DeepSeek-R1 Zero. For our two model versions, MuTIS and MuTIS-Ask, distinct chat templates were developed. Within the system role specified in these templates, we outlined the specific interaction workflow to guide the LLM.

MuTIS-Ask

role: 'system',content: The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The answer is enclosed within <answer> </answer> tags. i.e., <answer> answer here </answer>. During the assistant's reasoning process, if he realizes that his reasoning may be problematic or wrong, he can ask other agents for help. The query is inclosed within <ask> </ask> Tags. i.e., <ask> put confused point here </ask>. It will return the advice from other agent within <communicate> </communicate>. The assistant can ask other agents for help multiple times. If the assistant understand the question and find no further other agents' advice needed, the assistant can directly provide the answer inside <answer> </answer>.

MuTIS

role: 'system',content: The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The answer is enclosed within <answer> </answer> tags. i.e., <answer> answer here </answer>. If the assistant understand the question, he can directly provide the answer inside <answer> </answer>.

B.2 Intervention Prompt Design

MuTIS-Ask

Warning! My previous action is invalid. If I want to ask other agents for help, I should put the query between <ask> and </ask>. If I want to give the final answer, I should put the answer between <answer> and </answer>. Let me try again:

MuTIS

Warning! My previous action is invalid. If I want to give the final answer, I should put the answer between <answer> and </answer>. Let me try again:

B.3 Analysis of Prompt Sensitivity and Generalization

To ensure that our experimental design was not overly sensitive to prompt hyperparameter selection, we analyzed the experimental results and training processes associated with different variants of the 'Intervention Prompt.' The specific prompts used are as follows:

- Alert: The action you just performed was not valid. Please attempt it again.
- Notice: Your last move was unsuccessful. Kindly try once more.
- Error: The preceding operation failed. Please redo the action.
- Caution: That last input was not accepted. Please have another go.
- Unsuccessful Action: Your prior step could not be processed. Please try again.

C Additional Cases

C.1 MuTIS Inference Case

Figure 9 shows the inference outputs of MuTIS and the original model on the same question.

C.2 MuTIS Trajectory in Training Phase

Figure 10 shows an example of MuTIS training trajectory. This example demonstrates how MuTIS's design substantially enriches the LLM's reasoning chain, resulting in higher-quality trajectories for training.

D Additional Related Work

Reinforcement Learning for LLM Reasoning.

As advancements in Large Language Model (LLM) research encounter the challenges of escalating data scales, performance bottlenecks are becoming increasingly apparent. This issue is particularly acute for Supervised Fine-tuning (SFT), which critically depends on high-quality, novel datasets. Consequently, there is a growing research focus on Reinforcement Learning, especially techniques utilizing rule-based rewards (Zeng et al., 2025; Pan et al., 2025; Face, 2025; Liu et al., 2025), which have shown considerable potential in enhancing model performance. "Learning from reward" is thus emerging as a new training paradigm.

RL algorithms such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) leverage rewards during the training phase to optimize model parameters. Concurrently, sampling methodologies like Best of N employ rewards at the sampling stage to improve response accuracy. Furthermore, Zhao et al. (2025) rewards reasoning trajectories using a Process Reward Model (PRM), while Shen et al. (2025) rewards them using an Outcome Reward Model (ORM). Both advocate for the use of high-quality reasoning trajectories to augment model reasoning capabilities. However, these approaches are fundamentally model-based reward designs, and the potential of **rule-based** rewards within such frameworks remains largely unexplored.

Question: If $f(x) = \frac{3x-2}{x-2}$, what is the value of $f(-2) + f(-1) + f(0)$? Express your answer as a common fraction..

MuTIS(Ours): token consumption 650

Q: If $f(x) = \frac{3x-2}{x-2}$, what is the value of $f(-2) + f(-1) + f(0)$? Express your answer as a common fraction.

A: i think, To find the value of $f(-2) + f(-1) + f(0)$ for the function $f(x) = \frac{3x-2}{x-2}$, I will evaluate each term separately and then sum them up.
First, I will calculate $f(-2)$:

$$f(-2) = \frac{3(-2)-2}{-2-2} = \frac{-6-2}{-4} = \frac{-8}{-4} = 2$$

Next, I will calculate $f(-1)$:

$$f(-1) = \frac{3(-1)-2}{-1-2} = \frac{-3-2}{-3} = \frac{-5}{-3} = \frac{5}{3}$$

Then, I will calculate $f(0)$:

$$f(0) = \frac{3(0)-2}{0-2} = \frac{-2}{-2} = \frac{-2}{-2} = 1$$

Finally, I will sum the results:

$$f(-2) + f(-1) + f(0) = 2 + \frac{5}{3} + 1 = \frac{6}{3} + \frac{5}{3} + \frac{3}{3} = \frac{14}{3}$$

i think,

To find the value of $f(-2) + f(-1) + f(0)$ for the function $f(x) = \frac{3x-2}{x-2}$, we will evaluate each term separately and then sum them up.

1. Calculate $f(-2)$:

$$f(-2) = \frac{3(-2)-2}{-2-2} = \frac{-6-2}{-4} = \frac{-8}{-4} = 2$$

2. Calculate $f(-1)$:

$$f(-1) = \frac{3(-1)-2}{-1-2} = \frac{-3-2}{-3} = \frac{-5}{-3} = \frac{5}{3}$$

3. Calculate $f(0)$:

$$f(0) = \frac{3(0)-2}{0-2} = \frac{-2}{-2} = \frac{-2}{-2} = 1$$

4. Sum the results:

$$f(-2) + f(-1) + f(0) = 2 + \frac{5}{3} + 1 = \frac{6}{3} + \frac{5}{3} + \frac{3}{3} = \frac{14}{3}$$

$$\boxed{\frac{14}{3}}$$

Distill Model(Origin): token consumption 2753

Okay, so I have this function $f(x)$ which is equal to $(3x-2)$ divided by $(x-2)$. I need to find the value of $f(-2) + f(-1) + f(0)$. Hmm, let me think about how to approach this step by step. First, I should probably figure out what each of these function values is individually. That is, I need to compute $f(-2)$, $f(-1)$, and $f(0)$ separately and then add them together. Sounds straightforward, but I want to make sure I do each step correctly. Starting with $f(-2)$. The function $f(x)$ is $(3x-2)/(x-2)$. So, substituting x with -2 , I get: $f(-2) = (3*(-2)-2)/(-2-2)$. Let me compute the numerator and the denominator separately. The numerator is $3*(-2)$ which is -6 , and then subtract 2 , so that's $-6-2 = -8$. The denominator is $-2-2$, which is -4 . So, $f(-2)$ is -8 divided by -4 . Hmm, dividing two negatives gives a positive, so that's 2 . Okay, so $f(-2)$ is 2 . Next, let's compute $f(-1)$. Again, substituting x with -1 in the function: $f(-1) = (3*(-1)-2)/(-1-2)$. Calculating the numerator: $3*(-1)$ is -3 , minus 2 is -5 . The denominator is $-1-2$, which is -3 . So, $f(-1)$ is -5 divided by -3 . Again, dividing two negatives gives a positive, so that's $5/3$. So, $f(-1)$ is $5/3$. Now, moving on to $f(0)$. Substituting x with 0 : $f(0) = (3*0-2)/(0-2)$. Calculating the numerator: $3*0$ is 0 , minus 2 is -2 . The denominator is $0-2$, which is -2 . So, $f(0)$ is -2 divided by -2 , which is 1 . So, $f(0)$ is 1 . Alright, so now I have all three function values: $f(-2)$ is 2 , $f(-1)$ is $5/3$, and $f(0)$ is 1 . Now, I need to add them together. So, adding them up: $2 + 5/3 + 1$. Let me convert all these to fractions with a common denominator to make it easier. 2 is the same as $6/3$, and 1 is the same as $3/3$. So, adding them together: $6/3 + 5/3 + 3/3 = (6+5+3)/3 = 14/3$. **Wait**, hold on, that seems a bit high. Let me double-check my calculations. First, $f(-2)$: $3*(-2)$ is -6 , minus 2 is -8 . Denominator: $-2-2$ is -4 . So, $-8/-4$ is 2 . That seems correct. $f(-1)$: $3*(-1)$ is -3 , minus 2 is -5 . Denominator: $-1-2$ is -3 . So, $-5/-3$ is $5/3$. That also seems correct. $f(0)$: $3*0$ is 0 , minus 2 is -2 . Denominator: $0-2$ is -2 . So, $-2/-2$ is 1 . That's correct too. Adding them: $2 + 5/3 + 1$. Let me convert 2 and 1 to thirds: 2 is $6/3$, 1 is $3/3$. So, $6/3 + 5/3 + 3/3$ is indeed $14/3$. Hmm, $14/3$ is equal to 4 and $2/3$, which is approximately $4.666...$. **Wait**, but let me think again. Is there a possibility that I made a mistake in substituting the values? Let me check each substitution again. For $f(-2)$: $(3*(-2)-2)/(-2-2) = (-6-2)/(-4) = (-8)/(-4) = 2$. Correct. For $f(-1)$: $(3*(-1)-2)/(-1-2) = (-3-2)/(-3) = (-5)/(-3) = 5/3$. Correct. For $f(0)$: $(3*0-2)/(0-2) = (-2)/(-2) = 1$. Correct. So, adding them: $2 + 5/3 + 1$. Let me add 2 and 1 first: $2+1$ is 3 . Then, $3 + 5/3$. 3 is $9/3$, so $9/3 + 5/3$ is $14/3$. So, $14/3$ is the correct sum.

Wait, but $14/3$ is an improper fraction, and the question says to express the answer as a common fraction. So, $14/3$ is already a common fraction, so that should be the answer. But just to make sure, let me think if there's another way to approach this problem, maybe simplifying the function first before plugging in the values. Sometimes, simplifying can make the calculations easier or reveal any potential issues. Looking at $f(x) = (3x-2)/(x-2)$. Hmm, can I factor anything here? Let me see. The numerator is $3x-2$, which doesn't factor nicely with the denominator $x-2$. So, I don't think there's a way to simplify this function further. So, I think my initial approach is correct. Alternatively, maybe I can perform polynomial division or something, but since it's a linear over linear function, it's just a straight division.

Wait, another thought: maybe I can write $f(x)$ as $3 + \text{something}$. Let me try that. Let me rewrite $f(x)$ as $(3x-2)/(x-2)$. Let me perform the division: Divide $3x-2$ by $x-2$. So, how many times does $x-2$ go into $3x-2$? Well, x goes into $3x$ three times. So, $3*(x-2)$ is $3x-6$. Subtract that from $3x-2$: $(3x-2)-(3x-6) = 0x+4$. So, the division gives 3 with a remainder of 4 . Therefore, $f(x)$ can be written as $3 + 4/(x-2)$. So, $f(x) = 3 + 4/(x-2)$. Hmm, that might be helpful for checking the values. Let me compute $f(-2)$ using this expression: $3 + 4/(-2-2) = 3 + 4/(-4) = 3 - 1 = 2$. Correct, same as before. $f(-1)$: $3 + 4/(-1-2) = 3 + 4/(-3) = 3 - 4/3 = 5/3$. Correct. $f(0)$: $3 + 4/(0-2) = 3 + 4/(-2) = 3 - 2 = 1$. Correct. So, that's another way to confirm the values. So, adding them together: $2 + 5/3 + 1 = 14/3$. **Wait** but just to make sure, let me compute each step again: $f(-2)$: $3*(-2) = -6$, $-6-2 = -8$. Denominator: $-2-2 = -4$. So, $-8/-4 = 2$. Correct. $f(-1)$: $3*(-1) = -3$, $-3-2 = -5$. Denominator: $-1-2 = -3$. So, $-5/-3 = 5/3$. Correct. $f(0)$: $3*0 = 0$, $0-2 = -2$. Denominator: $0-2 = -2$. So, $-2/-2 = 1$. Correct. Adding them: $2 + 5/3 + 1$. Let me convert 2 and 1 to thirds: 2 is $6/3$, 1 is $3/3$. So, $6/3 + 5/3 + 3/3 = (6+5+3)/3 = 14/3$. So, $14/3$ is the sum. **Wait**, $14/3$ is an improper fraction, but the question says to express the answer as a common fraction. $14/3$ is already a common fraction, so that's fine. **Alternatively**, if I wanted to write it as a mixed number, it would be 4 and $2/3$, but since the question specifies a common fraction, $14/3$ is the appropriate answer. Just to make sure I didn't make any arithmetic errors, let me add $2 + 5/3 + 1$ again: $2+1$ is 3 , and $3 + 5/3$ is $3 + 1$ and $2/3$, which is 4 and $2/3$, which is $14/3$. Yep, that's correct. So, I think I've double-checked all my steps, and everything seems consistent. Therefore, the value of $f(-2) + f(-1) + f(0)$ is $14/3$.

Final Answer The value of $f(-2) + f(-1) + f(0)$ is $\boxed{\frac{14}{3}}$.

To find the value of $f(-2) + f(-1) + f(0)$ for the function $f(x) = \frac{3x-2}{x-2}$, we need to evaluate the function at each of these points and then sum the results. First, we compute $f(-2)$: $f(-2) = \frac{3(-2)-2}{-2-2} = \frac{-6-2}{-4} = \frac{-8}{-4} = 2$. Next, we compute $f(-1)$: $f(-1) = \frac{3(-1)-2}{-1-2} = \frac{-3-2}{-3} = \frac{-5}{-3} = \frac{5}{3}$. Then, we compute $f(0)$: $f(0) = \frac{3(0)-2}{0-2} = \frac{-2}{-2} = \frac{-2}{-2} = 1$. Adding these values together: $f(-2) + f(-1) + f(0) = 2 + \frac{5}{3} + 1$. Converting 2 and 1 to fractions with a common denominator:

$$2 = \frac{6}{3} \quad \text{and} \quad 1 = \frac{3}{3} \quad \text{Adding the fractions: } \frac{6}{3} + \frac{5}{3} + \frac{3}{3} = \frac{6+5+3}{3} = \frac{14}{3} \quad \text{Thus, the value of } f(-2) + f(-1) + f(0) \text{ is } \boxed{\frac{14}{3}}$$

Figure 9: Inference case. The left side is the original DeepSeek-R1-Distill-Qwen-1.5B model, and the right side is our MuTIS model.

Question: If $f(x) = \frac{3x-2}{x-2}$, what is the value of $f(-2) + f(-1) + f(0)$? Express your answer as a common fraction.

<think> Okay, so I need to figure out how many segments joining the vertices of this convex polyhedron lie entirely inside the polyhedron, not along an edge or a face. Let me first jot down all the given information to organize my thoughts. The polyhedron has: - 12 squares, - 8 regular hexagons, - 6 regular octagons as its faces. Each vertex is where one square, one hexagon, and one octagon meet. That means the configuration at each vertex is a 3-face meeting: 1 square, 1 hexagon, and 1 octagon. I remember that in polyhedrons, the number of edges can be calculated using the formula:

$$3F = 2E$$

where F is the total number of faces and E is the number of edges. Let me calculate the total number of faces first. The polyhedron has 12 squares, 8 hexagons, and 6 octagons, so:

$$F = 12 + 8 + 6 = 26$$

So, plugging into the formula:

$$3 \times 26 = 2E$$

$$78 = 2E$$

$$E = 39$$

Alright, so there are 39 edges in total.

Next, I think about the number of vertices. I remember another formula related to the vertices. Since each face is a polygon and each vertex is where three faces meet, maybe I can use a similar approach as with edges to find vertices.

For polyhedrons, there's Euler's formula:

$$V - E + F = 2$$

We know $F = 26$ and $E = 39$, so plugging into Euler's formula:

$$V - 39 + 26 = 2$$

$$V - 13 = 2$$

$$V = 15$$

So, there are 15 vertices in total.

Now, the question is about the number of segments joining the vertices that lie in the interior of the polyhedron rather than along an edge or a face. These are the space diagonals.

...

Wait, now I'm confused. Maybe the total number of face diagonals is 216, but we have only 105 segments, so actually, the face diagonals that we have calculated are the ones that cross through the interior, so those are in the space, so the actual space diagonals would be fewer.

Wait, no, perhaps the face diagonals are the ones that are not space diagonals.

My previous action is invalid If I want to ask other agents for help, I should put the query between <ask> and </ask>. If I want to give the final answer, I should put the answer between <answer> and </answer>. Let me try again:

Total number of segments: 105.

Edges: 39.

Face diagonals: 216.

Wait, this is not possible because 216 is larger than 105.

Therefore, my mistake is in calculating the face diagonals. Because the face diagonals can't exceed the number of vertex segments, since each face diagonal is a unique segment between two vertices.

Wait, but in reality, each face diagonal is only one segment, not multiple.

Wait, but if each face has $n(n-3)/2$ diagonals, and for the polyhedron, the number of such segments is:

For a square face: 2 diagonals.

But in reality, all the face diagonals belong to some face. So, in this polyhedron, each face diagonal is a vertex segment.

...

Total number of vertex pairs: $C = 15 \times 14 / 2 = 105$.

Number of edges: $E = 39$.

Number of face diagonals: for each face, diagonals are unique to that face. So, for each face, if it's a square, it has 2 face diagonals; hexagon, 9; octagon, 20.

So, face diagonals count is:

Squares: $12 \text{ faces} \times 2 \text{ diagonals} = 24$

Hexagons: $8 \times 9 = 72$

Octagons: $6 \times 20 = 120$

My previous action is invalid If I want to ask other agents for help, I should put the query between <ask> and </ask>. If I want to give the final answer, I should put the answer between <answer> and </answer>. Let me try again:

<ask> A convex polyhedron has for its faces 12 squares, 8 regular hexagons, and 6 regular octagons. At each vertex of the polyhedron one square, one hexagon, and one octagon meet. How many segments joining vertices of the polyhedron lie in the interior of the polyhedron rather than along an edge or a face? </ask>

<communicate> Another Agent's Response To find the number of segments joining the vertices of the polyhedron that lie in the interior, we can follow these steps:

1. **Calculate the total number of vertices (V)**: Using Euler's formula $V - E + F = 2$, we first need to find the number of edges (E) and faces (F).
2. **Calculate the total number of faces (F)**:

$$F = 12 \text{ (squares)} + 8 \text{ (hexagons)} + 6 \text{ (octagons)} = 26$$

3. **Calculate the number of edges (E)**: Each edge is shared between two faces. The total number of edges can be calculated as:

$$E = \frac{1}{2} (12 \times 4 + 8 \times 6 + 6 \times 8) = \frac{1}{2} (48 + 48 + 48) = \frac{144}{2} = 72$$

4. **Use Euler's formula to find the number of vertices (V)**:

$$V - 72 + 26 = 2 \implies V - 46 = 2 \implies V = 48$$

5. **Calculate the total number of segments joining the vertices**: The total number of segments (C) joining the vertices is given by:

$$C = \binom{V}{2} = \frac{48 \times 47}{2} = 1128$$

6. **Subtract the number of edges (E)**:

$$\text{Remaining segments} = 1128 - 72 = 1056$$

7. **Calculate the number of face diagonals**: - For squares: $\frac{4(4-3)}{2} = 2$ diagonals per square, total $12 \times 2 = 24$ - For hexagons: $\frac{6(6-3)}{2} = 9$ diagonals per hexagon, total $8 \times 9 = 72$ - For octagons: $\frac{8(8-3)}{2} = 20$ diagonals per octagon, total $6 \times 20 = 120$

Total face diagonals:

$$24 + 72 + 120 = 216$$

8. **Calculate the number of space diagonals**: Now, subtract the face diagonals from the remaining segments:

$$\text{Space diagonals} = 1056 - 216 = 840$$

Thus, the number of segments joining the vertices of the polyhedron that lie in the interior is **840**.

<answer> To find the number of segments joining the vertices of the polyhedron that lie in the interior, we can follow these steps:

1. **Calculate the total number of vertices (V)**: Using Euler's formula $V - E + F = 2$, we first need to find the number of edges (E) and faces (F).
2. **Calculate the total number of faces (F)**:

$$F = 12 \text{ (squares)} + 8 \text{ (hexagons)} + 6 \text{ (octagons)} = 26$$

3. **Calculate the number of edges (E)**: Each edge is shared between two faces. The total number of edges can be calculated as:

$$E = \frac{1}{2} (12 \times 4 + 8 \times 6 + 6 \times 8) = \frac{144}{2} = 72$$

4. **Use Euler's formula to find the number of vertices (V)**:

$$V - 72 + 26 = 2 \implies V - 46 = 2 \implies V = 48$$

5. **Calculate the total number of segments joining the vertices**: The total number of segments (C) joining the vertices is given by:

$$C = \binom{V}{2} = \frac{48 \times 47}{2} = 1128$$

6. **Subtract the number of edges (E)**:

$$\text{Remaining segments} = 1128 - 72 = 1056$$

7. **Calculate the number of face diagonals**: - For squares: $\frac{4(4-3)}{2} = 2$ diagonals per square, total $12 \times 2 = 24$ - For hexagons: $\frac{6(6-3)}{2} = 9$ diagonals per hexagon, total $8 \times 9 = 72$ - For octagons: $\frac{8(8-3)}{2} = 20$ diagonals per octagon, total $6 \times 20 = 120$

Total face diagonals:

$$24 + 72 + 120 = 216$$

8. **Calculate the number of space diagonals**: Now, subtract the face diagonals from the remaining segments:

$$\text{Space diagonals} = 1056 - 216 = 840$$

Thus, the number of segments joining the vertices of the polyhedron that lie in the interior is **840**.

Figure 10: The figure presents an example of a model rollout within the multi-turn intervention process. In this illustration, **red** segments indicate the Intervention Prompt (IP) introduced when the model's response surpasses the predefined response limit; **blue** tags represent the model's autonomous actions during the rollout.