Stop When Enough: Adaptive Early-Stopping for Chain-of-Thought Reasoning

Renliang Sun[1*], **Wei Cheng**[2*], **Dawei Li**[3*], **Haifeng Chen**[2], **Wei Wang**[1]
[1]UCLA [2]NEC Labs America [3]Arizona State University
sunrenliang@ucla.edu, weiwang@cs.ucla.edu
daweili5@asu.edu, {weicheng, haifeng}@nec-labs.com

Abstract

Chain-of-Thought (CoT) reasoning has driven recent gains of large language models (LLMs) on reasoning-intensive tasks by externalizing intermediate steps. However, excessive or redundant reasoning — so-called overthinking — can increase inference costs and lead LLMs toward incorrect conclusions. In this paper, we present **REFRAIN** (REFlective-Redundancy for Adaptive INference), a training-free framework that adaptively determines when to stop reasoning to mitigate overthinking. REFRAIN integrates a two-stage stop discriminator to identify reflective yet redundant reasoning and a sliding-window Upper Confidence Bound (SW-UCB) multi-armed bandit controller to dynamically adjust stopping thresholds according to problem difficulty without supervision or fine-tuning. Across four representative benchmarks and two model families, REFRAIN reduces token usage by 20-55% while maintaining or improving accuracy compared to standard CoT prompting. Extensive ablation and robustness analyses demonstrate its stability across models, scorers, and prompt variations. In summary, our findings highlight when-to-stop as a new and practical axis of test-time scaling — enabling models to reason not just more, but just enough.

1 Introduction

Large language models (LLMs) have recently achieved outstanding performance across a wide range of reasoning-intensive tasks, encompassing fields such as mathematics and commonsense reasoning (Tong et al., 2024; Guo et al., 2025; Li et al., 2025b; Yeo et al., 2025; Yu et al., 2025; Wang et al., 2026). A key driver of recent gains is that LLMs unfold a chain-of-thought (CoT) reasoning process before the final answer (Wei et al., 2022; Xu et al., 2025; Zhao et al., 2025). However, a key limitation of these models is their tendency to *overthink*—producing overly long reasoning trajectories filled with redundant or irrelevant steps (Chen et al., 2024; Cui et al., 2025). Such behavior not only increases inference overhead but can also steer the model toward incorrect conclusions (Jiang et al., 2025; Sui et al., 2025).

Recent works such as HALT-CoT (Laaouach, 2025) attempt to address this by introducing confidence or entropy-based early stopping signals, but these methods rely on manually designed heuristics for each task. Other methods like CoT-Valve (Ma et al., 2025b) and Deepconf (Fu et al., 2025) require additional computational resources to learn short-thinking behaviors or search for the best stopping threshold. While they achieve promising results, their reliance on manual heuristics and additional computational resources makes them difficult to generalize across diverse tasks and impractical in low-resource reasoning scenarios.

We argue that the core challenge lies not merely in stopping earlier, but in automatically and efficiently determining an optimal threshold that dynamically adapts the reasoning depth to each task's difficulty. This calls for a framework that ❶ automatically detects when further reasoning becomes redundant, ❷ adapts its stopping behavior online without external supervision or retraining, and ❸ maintains or even improves accuracy while reducing inference cost.

To this end, we introduce **REFRAIN** (REFlective-Redundancy for Adaptive INference), a training-free, dynamic framework that transforms 'when-to-stop' into a new axis of test-time scaling for reasoning LLMs. REFRAIN integrates two synergistic ideas:

- Reflective redundancy detection — a discriminator that identifies when reasoning transitions from reflective self-correction to redundant repetition, based on semantic similarity and trigger cues.

[*]Equal Contributions.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27250–27268
July 2-7, 2026

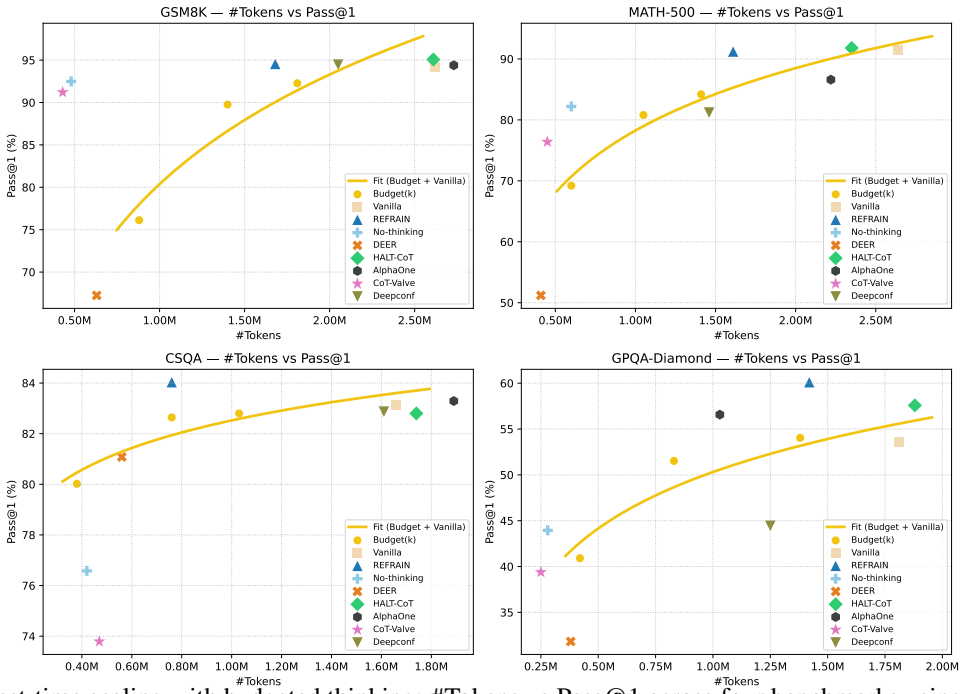

Figure 1: Test-time scaling with budgeted thinking: #Tokens vs Pass@1 across four benchmarks using Qwen3-8B. We fit a log curve using budget points and vanilla. REFRAIN lies in the upper-left of the fitted curve, indicating a better accuracy-efficiency trade-off.

• Adaptive thresholding via Sliding-Window Upper Confidence Bound (SW-UCB) — a multi-armed bandit controller that continuously balances exploration (longer reasoning when uncertain) and exploitation (early stop when confident), enabling task-specific efficiency.

Across four representative benchmarks and two model families, REFRAIN achieves 20-55% fewer tokens while maintaining or exceeding vanilla CoT accuracy, outperforming prior stopping methods and even matching fine-tuned baselines. As shown in Figure 1, REFRAIN consistently shifts the accuracy–efficiency frontier upward and leftward relative to vanilla CoT. Thus, REFRAIN scales reasoning not by thinking more, but by allocating thought where it matters.

In summary, this work makes the following key contributions:

• We identify reflective redundancy as the underlying signal behind overthinking and make it actionable through a two-stage stop discriminator.

• We propose a training-free adaptive thresholding algorithm based on SW-UCB bandits that dynamically balances exploration and exploitation during reasoning.

• We show consistent accuracy-efficiency gains across diverse reasoning tasks, establishing when-to-stop as a practical and generalizable dimension of test-time scaling.

2 Related Work

2.1 Overthinking in LLMs

Recent work characterizes overthinking in reasoning LLMs as *reasoning beyond what is needed to solve a problem*. Chen et al. (2024) show it is widespread: redundant steps add little to correctness or diversity, yet waste computation on simple problems. Sui et al. (2025) define it as verbose, redundant outputs that induce substantial overhead.

Beyond definitional studies, several works have empirically analyzed how overthinking manifests and affects reasoning performance. Chiang and Lee (2024) show that models often conduct redundant reasoning even for simple problems, sometimes leading to errors. Fan et al. (2025) show that with missing premises, models generate lengthy yet unhelpful chains instead of stopping. Gema et al. (2025) report an inverse scaling effect: longer reasoning can reduce accuracy. Cuadron et al. (2025) analyze forms that can impair accuracy, such as analysis paralysis and rogue actions. Collectively, these studies show overthinking is inefficient and can degrade reliability, motivating methods that dynamically regulate reasoning length.

2.2 Overthinking Mitigation

Existing efforts to mitigate overthinking generally fall into two categories: (1) post-training methods that teach models to shorten reasoning without

harming accuracy, and (2) inference-time strategies that adaptively decide when to stop reasoning.

Representative post-training methods include CoT-Valve (Ma et al., 2025b), which fine-tunes LLMs to control reasoning length by identifying parameter-space 'valves' that adjust the verbosity of generated thoughts; AALC (Li et al., 2025a), which uses an RL reward to balance correctness and conciseness; and SmartThinker (He et al., 2025), which combines SFT on short-form data with RL that allocates tokens to critical steps.

In contrast, inference-time approaches avoid retraining and focus on adaptive stopping — deciding when the model has reasoned enough. They seek reliable stopping signals at inference. Methods such as ESC (Li et al., 2024), s1 (Muennighoff et al., 2025), and Answer Convergence (Liu and Wang, 2025) regulate reasoning by monitoring the stability of the answer distribution. ESC dynamically stops sampling once the predicted answer converges, s1 inserts wait tokens to postpone termination until enough evidence is gathered, whereas Answer Convergence halts generation when consecutive reasoning chunks yield identical answers.

Other methods use internal confidence or entropy for early stopping. HALT-CoT (Laaouach, 2025) and Think Just Enough (Sharma and Chopra, 2025) terminate reasoning based on entropy computed from token-level distributions, while DEER (Yang et al., 2025b) and DeepConf (Fu et al., 2025) exploit confidence signals to prune redundant reasoning paths, enhancing efficiency and accuracy. RCPD (Wei et al., 2026) and NEAT (Liu et al., 2026) exploit semantic trajectory dynamics and neuron-level activations respectively to identify early-exit points. AlphaOne (Zhang et al., 2025a) and No-thinking (Ma et al., 2025a) use deterministic or skip-based mechanisms to stop when enough information is available.

Overall, existing methods either rely on supervised fine-tuning or handcrafted confidence signals. However, few provide a unified unsupervised framework that adaptively balances reasoning depth and efficiency across tasks, which is a challenge that our work aims to address.

3 Methodology

3.1 Definition

To formally describe our adaptive stopping mechanism, we first define the reasoning process of an LLM. Given a question Q, the LLM model $p_\theta(\cdot)$ produces a sequence of reasoning steps:

$$S = (s_1, \ldots, s_N) \tag{1}$$

followed by a final answer y. In practice, reasoning steps are segmented from the decoded text using blank-line delimiters. This choice follows a common practice in recent adaptive reasoning work (Yang et al., 2025b; Zhang et al., 2025a), where blank lines are treated as natural structural boundaries. Our goal is to identify the optimal stopping point-halting generation once further reasoning becomes redundant-and produce a concise, well-formed final answer.

3.2 Two-Stage Stopping Discriminator

We propose a discriminator D that triggers at step n only when (i) the step is *reflective* and (ii) its content is *semantically redundant* relative to the previous steps. The pseudocode for implementing discriminator D is shown in Algorithm 1.

Reflection detector. We first detect reflective reasoning cues. Following prior analyses of reflective or self-corrective cues in CoT reasoning (Ma et al., 2024; Yang et al., 2025c; Huang et al., 2025; Qiao et al., 2025), we define reflection operationally through a consolidated vocabulary derived from these studies. Concretely, we adopt the category structure reported in existing studies — self check, strategy shift, uncertainty expression, and retrospective revisions — and merge the trigger terms into four sets $V_{\text{check}}, V_{\text{shift}}, V_{\text{uncert}}, V_{\text{retro}}$. The full term lists used in our experiments are documented in Appendix A to facilitate replication. Let $V = V_{\text{check}} \cup V_{\text{shift}} \cup V_{\text{uncert}} \cup V_{\text{retro}}$ denote the complete reflection trigger vocabulary. We flag step s_n as reflective if any $v \in V$ occurs:

$$r_n = \mathbb{I}(\exists\, v \in V \text{ s.t. } v \subseteq s_n) \tag{2}$$

where $\mathbb{I}(\cdot)$ is the indicator function which returns 1 if the condition inside holds and 0 otherwise. To prevent premature triggers early in the trace, we additionally require the history to include a provisional answer cue c (e.g., 'answer is/should be'), defined as:

$$h_{n-1} = \mathbb{I}\left(\exists j < n \text{ s.t. } c \subseteq s_j\right) \tag{3}$$

Semantic redundancy scorer. We embed each step with a sentence encoder $f(\cdot)$ (we use `all-MiniLM-L6-v2` in practice) and compute the maximum cosine similarity to the previous steps:

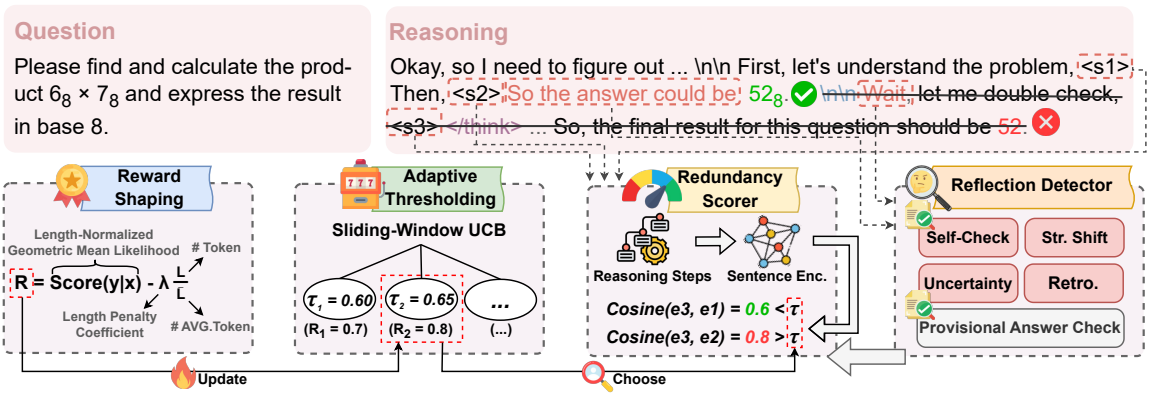

Figure 2: Overview of the proposed REFRAIN method.

Algorithm 1 Early-Stop via Reflective Redundancy

Require: Question Q; model p_θ; encoder f; trigger sets V, cue c; threshold τ.

Initialize step list $S \leftarrow [\,]$, embeddings $E \leftarrow [\,]$, flags $h \leftarrow 0$.
for $n = 1, 2, \ldots$ **do**
  Generate next step s_n (until blank-line delimiter).
  $S \leftarrow S \,\|\, s_n$; $e_n \leftarrow f(s_n)$; $E \leftarrow E \,\|\, e_n$.
  $r_n \leftarrow \mathbb{I}(\exists v \in V : v \subseteq s_n)$.
  $h \leftarrow h \vee \mathbb{I}(\exists j < n : c \subseteq s_j)$.
  $\phi_n \leftarrow \max_{1 \le j < n} \cos(e_n, e_j)$ (**if** $n > 1$, **else** 0) .
  if $h = 1$ **and** $r_n = 1$ **and** $\phi_n \ge \tau$ **then**
    break ▷ stop thinking
return stop trace S

$$e_n = f(s_n), \qquad \phi_n = \max_{1 \le j < n} \cos\big(e_n, e_j\big). \quad (4)$$

Given the threshold $\tau \in [0, 1]$, the stop rule is:

$$\begin{aligned} \textsc{Stop}(Q, s_{1:n}) \; &= \mathbb{I}(h_{n-1} = 1) \cdot \mathbb{I}(r_n = 1) \\ &\cdot \mathbb{I}(\phi_n \ge \tau). \end{aligned} \quad (5)$$

Once triggered, we halt further reasoning and proceed to the final answer generation stage.

3.3 Answer-Only Likelihood Scoring

After stopping the thinking process, we elicit an answer using a forced-closure prompt: `Final Answer: \boxed{` $\cdots$ `}`. To mitigate stylistic variance in reasoning traces, we evaluate only the answer token sequence y within the boxed region. Let x denote the answer prefix context, we use the length-normalized geometric mean likelihood:

$$\textbf{Score}(y \mid x) = \exp\left(\frac{1}{|y|} \sum_{i=1}^{|y|} \log p_\theta(y_i | x, y_{<i})\right) \quad (6)$$

3.4 Adaptive Thresholding via Sliding-Window UCB

Due to the varying complexity of reasoning across different tasks, employing a uniform fixed threshold τ is not the optimal solution, and manually defining an optimal threshold for each task is also extremely challenging. Therefore, we adaptively select τ from a discrete set $\mathcal{T}$ with a sliding-window UCB (SW-UCB) bandit (Garivier and Moulines, 2008). The pseudocode is shown in Algorithm 2. For each arm $t \in \mathcal{T}$, we maintain a window of the most recent W rewards $R^{(t)}$ with running mean $\bar{R}^{(t)}$. Let $n^{(t)}$ be the number of pulls of arm t in the window, and define the effective time

$$t_{\text{eff}} = \min\big(k,\ W \cdot |\mathcal{T}|\big), \quad (7)$$

where k is the round index. We then select

$$t_k = \arg\max_{t \in \mathcal{T}} \ \bar{R}^{(t)} + C\sqrt{\frac{2 \log t_{\text{eff}}}{n^{(t)}}}, \quad (8)$$

with each arm explored once for initialization to ensure unbiased reward estimation.

Algorithm 2 SW-UCB for Adaptive τ Selection

Require: Candidate set $\mathcal{T}$; window size W; exploration constant $C > 0$.

Initialize buffers $\{\mathcal{B}^{(t)}\}_{t \in \mathcal{T}} \leftarrow$ empty deques of capacity W.
for $k = 1, 2, \ldots$ **do**
  Cold-start: if $\exists\, t \in \mathcal{T}$ with $|\mathcal{B}^{(t)}| = 0$, set $t_k \leftarrow t$ (arbitrary such t).
  Otherwise:
  $t_{\text{eff}} \leftarrow \min\big(k,\ W \cdot |\mathcal{T}|\big)$.
  for all $t \in \mathcal{T}$ **do**
    $\bar{R}^{(t)} \leftarrow \text{mean}(\mathcal{B}^{(t)})$; $n^{(t)} \leftarrow |\mathcal{B}^{(t)}|$.
    $\text{UCB}^{(t)} \leftarrow \bar{R}^{(t)} + C\sqrt{2\log(t_{\text{eff}})/\max(1, n^{(t)})}$.
  $t_k \leftarrow \arg\max_{t \in \mathcal{T}} \text{UCB}^{(t)}$.
  Run Alg. 1 with $\tau = t_k$; obtain stopped trace and final answer y_k.
  Compute reward R_k (Eq. 9); push R_k into $\mathcal{B}^{(t_k)}$ (evict oldest if full).

3.5 Reward Shaping

To guide the adaptive selection process, we design a reward function that jointly accounts for answer

quality and reasoning efficiency. Let L be the total output token count (thinking + answer), and $\bar{L}$ its running mean. We define

$$R = \text{Score}(y \mid x) - \lambda \frac{L}{\bar{L}}, \quad (9)$$

with $\lambda > 0$. For the very first sample where $\bar{L}$ is undefined, we use a linear cold start penalty $0.0001 \cdot L$. This reward formulation encourages accurate yet concise reasoning, guiding the bandit to prefer earlier stopping (i.e., smaller τ) when answer confidence is high. We deliberately avoid process evaluators to keep REFRAIN training-free and single-forward at test time. When process fidelity is paramount, more expensive rewards can be plugged into Algorithm 2 without changing the current early-stop mechanism.

3.6 Online-Adaptive Thresholding

Figure 2 provides an overview of our method, REFRAIN (REFlective-Redundancy for Adaptive INference). REFRAIN operates entirely at test time, requiring no fine-tuning or validation data. REFRAIN does not suppress all reflective behavior. Instead, it distinguishes productive reflection from redundant repetition by halting reasoning only when a reflective step provides no new semantic insight after a provisional answer has been proposed. The sliding-window UCB controller dynamically adjusts the threshold τ based on recent reward feedback, allowing the model to balance accuracy and efficiency across varying task difficulties and to autonomously determine when to stop reasoning.

4 Experimental Settings

4.1 Models

We evaluate our approach using two representative reasoning-oriented LLMs: Qwen3-8B (Yang et al., 2025a) and gpt-oss-20B (Agarwal et al., 2025).

More details of the implementation can be found in Appendix E.

4.2 Benchmarks

We conduct experiments on four benchmarks that cover mathematical problems, commonsense reasoning problems, and STEM problems:

GSM8K. GSM8K (Cobbe et al., 2021) consists of 1,319 grade-school math problems requiring multi-step arithmetic reasoning.

MATH-500. MATH-500 (Hendrycks et al., 2021) consists of 500 challenging math problems selected from the MATH benchmark.

CommonsenseQA (CSQA). CSQA (Talmor et al., 2019) is a multiple-choice commonsense reasoning dataset with 1,221 questions. Since the official test set does not provide labels, we report results on the validation set.

GPQA-Diamond. GPQA-Diamond (Rein et al., 2024) is the most challenging subset of the GPQA benchmark, comprising 198 graduate-level interdisciplinary questions in a multiple-choice format.

4.3 Evaluation

We evaluate model performance in terms of accuracy and efficiency using two metrics:

Pass@1. For GSM8K and MATH-500, we extract the final boxed expression or numerical answer from the model's output and apply strict string matching to ground truth. For CSQA and GPQA-Diamond, we evaluate multiple-choice accuracy by comparing the predicted option (A–E) with the gold label.

#Tokens. We measure the number of tokens generated (reasoning steps + final answer). At comparable accuracy levels, lower token usage indicates more efficient reasoning and reduced overthinking.

5 Results

In this section, we first present the overall performance of REFRAIN across benchmarks and models (§5.1), followed by ablation and robustness studies (§5.2-5.6) that examine the contribution of individual components and generalization across settings.

5.1 Main Results: Accuracy-Efficiency Trade-off

We evaluated REFRAIN against baselines on four benchmarks. Our baselines are No-thinking (Ma et al., 2025a), DEER (Yang et al., 2025b), HALT-CoT (Laaouach, 2025), AlphaOne (Zhang et al., 2025a), CoT-Valve (Ma et al., 2025b), and Deepconf (Fu et al., 2025). We introduced these methods in Section 2.2, so we briefly summarized them here. For No-thinking, the reasoning trace is replaced by a fixed phrase '`Okay, I think I have finished thinking.`' For DEER, HALT-CoT, and AlphaOne, we have reproduced the methods described in the paper. For CoT-Valve, we have fine-tuned Qwen3 and gpt-oss using the short thinking process detailed in the paper. For Deepconf, we have reproduced the method and adapted it to our single-trace setting. Notably, all methods ex-

Qwen3-8B	GSM8K		MATH-500		CSQA		GPQA-Diamond	
	Pass@1	**#Tokens**	**Pass@1**	**#Tokens**	**Pass@1**	**#Tokens**	**Pass@1**	**#Tokens**
Vanilla	94.24	2.62M	91.40	2.64M	83.13	1.66M	53.54	1.81M
No-thinking	92.49	0.48M	82.20	0.60M	76.58	**0.42M**	43.94	0.28M
DEER	67.25	0.63M	51.20	**0.41M**	81.08	0.56M	31.82	0.38M
HALT-CoT	**95.07**	2.61M	**91.80**	2.35M	82.80	1.74M	57.58	1.88M
AlphaOne	94.39	2.73M	86.60	2.22M	83.29	1.89M	56.57	1.03M
CoT-Valve	91.21	**0.43M**	76.40	0.45M	73.79	0.47M	39.39	**0.25M**
Deepconf	94.47	2.05M	81.20	1.46M	82.87	1.61M	44.44	1.25M
REFRAIN	$94.54_{+0.30}$	$1.68M_{-35.9\%}$	$91.20_{-0.20}$	$1.61M_{-39.0\%}$	**84.03**$_{+0.90}$	$0.76M_{-54.2\%}$	**60.10**$_{+6.56}$	$1.42M_{-21.5\%}$

gpt-oss-20B	GSM8K		MATH-500		CSQA		GPQA-Diamond	
	Pass@1	**#Tokens**	**Pass@1**	**#Tokens**	**Pass@1**	**#Tokens**	**Pass@1**	**#Tokens**
Vanilla	91.66	0.76M	80.80	1.07M	**82.96**	0.80M	34.85	0.93M
No-thinking	90.83	**0.38M**	74.20	0.43M	76.33	**0.40M**	30.30	0.23M
DEER	93.63	0.63M	83.20	0.76M	70.52	1.14M	15.66	0.67M
HALT-CoT	92.65	0.50M	79.80	0.66M	77.15	0.60M	40.40	0.50M
AlphaOne	94.00	0.49M	**84.60**	0.84M	79.85	0.60M	**46.97**	0.79M
CoT-Valve	89.08	0.43M	62.20	**0.32M**	71.42	0.49M	15.15	**0.14M**
Deepconf	93.33	0.58M	83.00	0.60M	81.00	0.75M	25.76	0.73M
REFRAIN	**94.39**$_{+2.73}$	$0.42M_{-44.7\%}$	$84.20_{+3.40}$	$0.69M_{-35.5\%}$	$81.74_{-1.22}$	$0.45M_{-43.8\%}$	$41.92_{+7.07}$	$0.62M_{-33.3\%}$

Table 1: Overall accuracy–efficiency results across four benchmarks (Pass@1↑ / #Tokens↓). Vanilla means we use default generation settings with no early stopping. CoT-Valve requires fine-tuning, while Deepconf needs additional CoT generation. **Bold** marks the best value per column. For REFRAIN, subscripts denote its change vs. Vanilla: Pass@1 uses $+/-$ absolute points, #Tokens shows percentage reduction.

cept CoT-Valve and Deepconf require no additional fine-tuning or CoT generation.

Table 1 compares REFRAIN with the baselines across all benchmarks. Overall, REFRAIN attains vanilla-level or higher Pass@1 while consuming 20-55% fewer tokens, delivering the best accuracy-efficiency balance among training-free methods and competitive against fine-tuned baselines.

For Qwen3-8B, REFRAIN matches the vanilla accuracy on GSM8K and MATH-500 while cutting token usage by approximately 40%. It slightly surpasses vanilla on CSQA with a 55% reduction in tokens, and yields notable improvements on the more challenging GPQA-Diamond benchmark with around 20% fewer tokens.

Similar trends are observed with gpt-oss-20B. REFRAIN shows obvious improvements in Pass@1 compared to Vanilla on GSM8K, MATH-500, and GPQA-Diamond, while maintaining a comparable Pass@1 on CSQA. REFRAIN also achieves token savings of ~45%, ~35%, ~45%, and ~35% respectively across the four benchmarks. These consistent cross-model trends suggest REFRAIN's benefits are architecture-agnostic.

We also compare REFRAIN with other representative baselines. No-thinking uses the fewest tokens but lowers Pass@1, especially on the higher-difficulty benchmarks MATH-500 and GPQA-Diamond. DEER and CoT-Valve often save tokens, but Pass@1 significantly drops on some benchmarks. For example, when using Qwen3, DEER performs noticeably worse than Vanilla on GSM8K and MATH-500. When using gpt-oss, CoT-Valve performs poorly on GPQA-Diamond. HALT-CoT maintains Pass@1 but saves few tokens. AlphaOne achieves strong accuracy on several benchmarks, but its token usage often remains close to Vanilla especially on Qwen3. It also requires multiple full reasoning generations to estimate the token budget in advance. Deepconf shows competitive performance on some benchmarks such as GSM8K and MATH-500, but is inconsistent in others, such as the GPQA-Diamond benchmark when using gpt-oss.

Overall, REFRAIN systematically curbs overthinking by detecting reflective yet redundant steps and stopping early when additional reasoning is unlikely to help. This yields substantial token reductions at equal or higher Pass@1 across tasks and backbones, positioning when-to-stop as a practical axis of test-time scaling that complements longer chains and larger models. Case studies illustrating both successful and failed early stops are provided in Appendix H.

5.2 Completeness of Trigger Vocabulary and Categories

To assess the adequacy of our four predefined categories of reflective triggers, we conducted three controlled experiments on the MATH-500 dataset using Qwen3-8B. All other configurations remained consistent with the main experiment.

Leave-One-Category-Out. We ablated one trigger category at a time while keeping the others unchanged.

In-Category Vocabulary Expansion (In-cat Expansion). We augmented each category with natural synonyms and semantically similar expressions. This tests whether enlarging the lexical coverage within categories yields additional benefits.

New Category Addition (New Category). We introduced an additional trigger category with no semantic overlap with the existing categories to evaluate whether additional functional categories can further improve performance.

For the second and third experiments, we used GPT-5 to generate candidate synonyms and new category terms, which were subsequently filtered to ensure semantic validity. The complete vocabulary is listed in Table 5.

Variants	Pass@1↑	#Tokens↓
Vanilla	91.40	2.64M
REFRAIN	91.20	1.61M
Shorten the Vocabulary		
w/o self-check	90.20	1.77M
w/o strategy-shift	90.20	1.68M
w/o uncertainty	90.80	1.66M
w/o retrospective	91.40	1.63M
Expand the Vocabulary		
In-cat Expansion	91.60	1.65M
New Category	91.20	1.69M

Table 2: Ablation and expansion of trigger categories.

According to Table 2, the removal of self-check or strategy-shift triggers leads to a notable drop in Pass@1, while removing uncertainty or retrospective triggers has minor effects. Moreover, expanding the vocabulary within existing categories or adding a new category does not yield further improvements. These findings suggest that our four categories already capture the essential reflective behaviors, with self-check and strategy-shift being particularly critical, while the current vocabulary provides sufficient coverage without requiring further expansion.

5.3 Importance of UCB in Optimal Threshold Selection

Although our method adaptively selects thresholds based on sliding-window UCB and demonstrates stable performance in experiments, it remains to be verified whether the UCB strategy is essential or if simpler heuristics could achieve comparable results. To this end, we compared it against four alternative early-stopping heuristics on the MATH-500 benchmark using Qwen3-8B:

Step-based Entropy Early Stop (SE-Early Stop): At each step, we compute the average token entropy of the step and stop reasoning once the entropy drops below a low fixed threshold 0.1, effective from the 10th step to prevent insufficient thinking.

Step-based Probability Early Stop (SP-Early Stop): Each step terminates reasoning early with a probability that increases linearly with the number of steps, where $Stop\_prob(n) = min(0.5, 2 \times 10^{-3} \times n)$, effective from the 10th step.

Step-based Probability Early Stop with Trigger Words (SPTW-Early Stop): Stop thinking based on probability only when trigger words are included in the generated steps. The remaining settings remain the same as SP-Early Stop.

Randomly Select Threshold (RST): Instead of using UCB to optimize the threshold, a threshold is randomly selected for each problem from the candidate threshold set, simulating a scenario of no exploration for optimal thresholds.

Method	Pass@1↑	#Tokens↓
Vanilla	91.40	2.64M
SE-Early Stop	63.40	0.80M
SP-Early Stop	68.60	0.74M
SPTW-Early Stop	83.20	0.91M
RST	88.40	1.68M
REFRAIN	91.20	1.61M

Table 3: Comparison of REFRAIN with alternative early-stopping heuristics.

Table 3 demonstrates that fixed-threshold entropy-based stopping (SE-Early Stop) achieved very poor performance, suggesting that low entropy alone does not reliably indicate that the reasoning is complete, but may instead reflect locally confident yet globally incorrect intermediate states. Purely probability-based stopping (SP-Early Stop) severely harms accuracy despite reducing token usage, indicating that indiscriminate truncation prematurely cuts off necessary reasoning. Incorporating trigger words (SPTW-Early Stop) partially mitigates this issue but still falls short of our method,

suggesting that lexical cues alone are insufficient for robust stopping. Random threshold selection (RST) yields moderate accuracy but fails to balance efficiency and correctness. By contrast, REFRAIN consistently achieves a better trade-off between accuracy and efficiency, confirming that trigger words, semantic redundancy scorer, adaptively selecting thresholds are crucial for stable performance.

5.4 Is Answer-Only Likelihood Sufficient for Adaptive Stopping?

REFRAIN uses the average log-likelihood of the boxed final answer as the reward signal, which naturally aligns with the pre-training objective of decoder-only LLMs and avoids additional forward passes. More importantly, it reflects a fundamental constraint of our setting: REFRAIN is a fully training-free and supervision-free test-time method, and thus cannot rely on ground-truth labels when adjusting stopping thresholds. As a result, the reward signal must be computable online during inference without access to correctness supervision. This raises the question of whether answer-only likelihood — an imperfect proxy for correctness-provides sufficient alignment for adaptive early stopping, especially when compared to external reward or process reward models (RMs/PRMs) that explicitly evaluate solution quality.

To address this, we replace the likelihood-based reward with three widely used alternatives. On GSM8K and MATH-500, we evaluate two math-specific reward models AceMath-7B (RM) (Liu et al., 2024) and Qwen2.5-MATH-PRM-7B (PRM) (Zhang et al., 2025b), and on multiple-choice benchmarks CSQA and GPQA-Diamond, we use a general-purpose reward model Skywork-Reward-V2-Llama-3.1-8B (RM) (Liu et al., 2025). More details are provided in Appendix E.

Method	GSM8K		MATH-500	
	Pass@1↑	#Tokens↓	Pass@1↑	#Tokens↓
Vanilla	94.24	2.62M	91.40	2.64M
AceMath-7B	94.47	1.71M	90.60	1.70M
Qwen2.5-MATH-PRM-7B	94.24	1.66M	91.60	1.74M
REFRAIN (Likelihood)	94.54	1.68M	91.20	1.61M

Method	CSQA		GPQA-Diamond	
	Pass@1↑	#Tokens↓	Pass@1↑	#Tokens↓
Vanilla	83.13	1.66M	53.54	1.81M
Skywork-Reward-V2-8B	83.46	0.75M	55.56	1.44M
REFRAIN (Likelihood)	84.03	0.76M	60.10	1.42M

Table 4: Likelihood vs. external reward models.

As summarized in Table 4, the PRM provides a Pass@1 similar to the answer-only likelihood, while the RM slightly underperforms the likelihood on math reasoning benchmarks. Then, on multiple-choice benchmarks, replacing likelihood with a general-purpose RM results in slightly lower performance on GPQA-Diamond compared to the likelihood-based variant. Overall, these results indicate that REFRAIN does not require a perfectly aligned correctness signal. Instead, a lightweight proxy that reflects the model's internal confidence, such as answer likelihood, is sufficient to guide adaptive early stopping in practice.

It is also worth noting that both RM and PRM require an additional forward pass, increasing computational cost, whereas the likelihood-based objective incurs no extra computation. Therefore, the answer-only likelihood achieves better accuracy-efficiency trade-off under our early-stopping framework and is a practical default for CoT adaptive stopping.

5.5 Test-time Scaling with Budgeted Thinking

We treat budgeted thinking as test-time scaling that dynamically adjusts reasoning length. Concretely, for a given instance, we limit the cumulative number of thinking tokens. Once the limit is reached, the model is forced to stop the reasoning chain and generate the final answer. We sweep the budget from small to large while keeping all other decoding settings unchanged, and evaluate on Qwen3-8B across the four benchmarks.

The experimental results are shown in Figure 1 and numerical values are provided in Table 7. For the relatively simple datasets GSM8K and CSQA, optimal results can be approximated using fewer reasoning tokens. Our adaptive method rapidly detects and stops reasoning, thus achieving higher or comparable accuracy with fewer tokens. MATH-500 and GPQA-Diamond contain more problems that require longer reasoning to perform well. In contrast, our method can allocate more reasoning budget to challenging problems, thereby achieving a better accuracy-cost trade-off.

Conceptually, test-time scaling usually samples more chains or extends chain length. We show that learning when to stop within a single chain is similarly effective: it balances accuracy and tokens without extra passes, reallocating computation per instance. Uniform fixed budgets over-allocate easy instances and under-allocate hard ones. In contrast, adaptive stopping strategies continuously adjust to achieve the optimal trade-off.

5.6 Wall-Clock Latency and Runtime Overhead

We analyze the wall-clock inference latency of REFRAIN and clarifies how token reduction translates into real-time efficiency.

Latency decomposition in autoregressive inference For autoregressive LLMs, the dominant component of inference latency is the per-token forward pass during decoding, including attention and feed-forward computation. As a result, end-to-end inference time scales approximately linearly with the number of generated tokens.

This work is also consistent with the evaluation protocol adopted in previous research on adaptive reasoning and early stopping (Muennighoff et al., 2025; Laaouach, 2025; Zhang et al., 2025a), which reports only token usage as the efficiency metric, reflecting the common assumption that token reduction captures the dominant component of inference cost. Our paper follows the same convention and therefore reports #Tokens as the efficiency metric in the main experiments.

Wall-clock evaluation To quantify real-time efficiency, we measure wall-clock inference latency under the same decoding configuration as in the main experiments. We first evaluate Qwen3-8B on the CSQA benchmark, which exhibits the greatest token savings. Compared to vanilla Chain-of-Thought decoding, REFRAIN reduces average wall-clock inference time by approximately 40% with a 54% reduction in generated tokens. We then measure wall-clock latency for gpt-oss-20B on GSM8K. REFRAIN reduces token usage by 45% and wall-clock inference time by 28% relative to vanilla decoding. The consistent latency speedups across two model families and two benchmarks indicate that the wall-clock benefits of REFRAIN are not specific to a single backbone or task.

The observed gap between token reduction and latency reduction arises because early stopping breaks the single long decoding run. Each interruption forces the model to re-tokenize and re-enter generate, which prevents full KV-cache amortization and adds constant overhead. This overhead is intrinsic to segmented decoding in current LLMs and may be more pronounced for MoE models such as gpt-oss-20B, where expert routing introduces additional per-segment cost. This effect does not change the fundamental conclusion: REFRAIN reduces token usage substantially and produces a significant wall-clock speedup across both dense and MoE backbones.

Overhead of auxiliary components We further examine the runtime contribution of REFRAIN's auxiliary components. Semantic redundancy detection is implemented using a lightweight sentence encoder (all-MiniLM-L6-v2), and adaptive thresholding is based on a sliding-window UCB controller with constant-time updates. Our tests show that SBERT-based similarity scoring accounts for less than 1% of total inference time, while bandit updates are negligible. Therefore, the additional computation introduced by REFRAIN has no substantial effect on end-to-end latency.

6 Conclusion

We proposed REFRAIN, a training-free framework that mitigates overthinking by detecting reflective redundancy and adaptively tuning stop thresholds via a sliding-window UCB controller. Across four benchmarks and two model families, REFRAIN reduces token usage by 20–55% while preserving or improving accuracy, establishing when-to-stop as a practical axis of test-time scaling for efficient and reliable reasoning.

Limitations

REFRAIN relies on observable step-by-step reasoning during inference, which means the model must expose or stream intermediate reasoning traces so the system can determine when to stop. This assumption is common across most test-time methods for mitigating overthinking or adaptively allocating compute (e.g., stopping rules, reflection loops, or debate-style reasoning). For closed-source APIs that only return final answers without providing step-by-step control, REFRAIN, like other similar methods, cannot intervene during the generation process. Nevertheless, our framework remains applicable to any setting where partial reasoning signals or token-level outputs are available, including open-source and step-streaming models.

Acknowledgments

This work was partially supported by NIH U54OD036472, U54DK097771, U54HG012517, NSF 2312501, 2106859, and NEC Corporation.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Cheng-Han Chiang and Hung-yi Lee. 2024. Over-reasoning and redundant calculation of large language models. *arXiv preprint arXiv:2401.11467*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, and 1 others. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. 2025. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.

Aurélien Garivier and Eric Moulines. 2008. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, and 1 others. 2025. Inverse scaling in test-time compute. *arXiv preprint arXiv:2507.14417*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xingyang He, Xiao Ling, and Jie Liu. 2025. Smartthinker: Learning to compress and preserve reasoning by step-level length control. *arXiv preprint arXiv:2507.04348*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. 2025. Efficient reasoning for large reasoning language models via certainty-guided reflection suppression. *arXiv preprint arXiv:2508.05337*.

Yuxuan Jiang, Dawei Li, and Frank Ferraro. 2025. Drp: Distilled reasoning pruning with skill-aware step decomposition for efficient large reasoning models. *arXiv preprint arXiv:2505.13975*.

Yassir Laaouach. 2025. Halt-cot: Model-agnostic early stopping for chain-of-thought reasoning via answer entropy. In *4th Muslims in ML Workshop co-located with ICML 2025*.

Ruosen Li, Ziming Luo, Quan Zhang, Ruochen Li, Ben Zhou, Ali Payani, and Xinya Du. 2025a. Aalc: Large language model efficient reasoning via adaptive accuracy-length control. *arXiv preprint arXiv:2506.20160*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025b. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

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

Kang Liu, Yongkang Liu, Xiaocui Yang, Peidong Wang, Wen Zhang, Shi Feng, Yifei Zhang, and Daling Wang. 2026. Neat: Neuron-based early exit for large reasoning models. *arXiv preprint arXiv:2602.02010*.

Xin Liu and Lu Wang. 2025. Answer convergence as a signal for early stopping in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17907–17918.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Acemath: Advancing frontier math reasoning with post-training and reward modeling. *arXiv preprint*.

Jingyuan Ma, Damai Dai, Zihang Yuan, Weilin Luo, Bin Wang, Qun Liu, Lei Sha, Zhifang Sui, and 1 others. 2024. Large language models struggle with unreasonability in math problems. *arXiv preprint arXiv:2403.19346*.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning. *arXiv preprint arXiv:2505.04881*.

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

Aman Sharma and Paras Chopra. 2025. Think just enough: Sequence-level entropy as a confidence signal for llm reasoning. *arXiv preprint arXiv:2510.08146*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Richard S Sutton, Andrew G Barto, and 1 others. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can llms learn from previous mistakes? investigating llms' errors to boost for reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080.

Xiaoxuan Wang, Han Zhang, Haixin Wang, Yidan Shi, Ruoyan Li, Kaiqiao Han, Chenyi Tong, Haoran Deng, Renliang Sun, Alexander Taylor, and 1 others. 2026. Arlarena: A unified framework for stable agentic reinforcement learning. *arXiv preprint arXiv:2602.21534*.

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

Zihao Wei, Liang Pang, Jiahao Liu, Wenjie Shi, Jingcheng Deng, Shicheng Xu, Zenghao Duan, Fei Sun, Huawei Shen, and Xueqi Cheng. 2026. The evolution of thought: Tracking llm overthinking via reasoning dynamics analysis. *Preprint*, arXiv:2508.17627.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*.

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

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. 2025c. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv:2504.12329*.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Longxuan Yu, Delin Chen, Siheng Xiong, Qingyang Wu, Dawei Li, Zhikai Chen, Xiaoze Liu, and Liangming Pan. 2025. Causaleval: Towards better causal reasoning in language models. In *Proceedings of the 2025 Conference of the Nations of the Americas*

Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 12512–12540.

Junyu Zhang, Runpei Dong, Han Wang, Xuying Ning, Haoran Geng, Peihao Li, Xialin He, Yutong Bai, Jitendra Malik, Saurabh Gupta, and 1 others. 2025a. Alphaone: Reasoning models thinking slow and fast at test time. *arXiv preprint arXiv:2505.24863*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Chengshuai Zhao, Zhen Tan, Pingchuan Ma, Dawei Li, Bohan Jiang, Yancheng Wang, Yingzhen Yang, and Huan Liu. 2025. Is chain-of-thought reasoning of llms a mirage? a data distribution lens. *arXiv preprint arXiv:2508.01191*.

Appendix A Reflection Trigger Vocabulary

This appendix documents the reflection trigger vocabulary used by the two-stage stop discriminator (Sec. 3.2). We group phrases into four categories that correspond to self-check, strategy shifts, expressed uncertainty, and retrospective revisions. The following lists are verbatim and were used as is in all experiments unless otherwise specified.

Scope and Generalization of the Reflection Triggers The reflection trigger vocabulary used in REFRAIN is not arbitrarily curated. Instead, it is consolidated from previous empirical analyses of reflective and self-corrective behaviors in chain-of-thought reasoning, as reported in studies on CoT dynamics and reflection patterns (Ma et al., 2024; Yang et al., 2025c; Huang et al., 2025; Qiao et al., 2025). Moreover, REFRAIN does not rely on the trigger vocabulary alone to determine early stopping. The trigger words serve only as a gating signal to identify potentially reflective steps; the actual stopping decision is determined by semantic redundancy between the current step and previous reasoning.

The robustness of this design is supported by multiple experiments in the paper.

Across domains, we evaluate REFRAIN on mathematics (GSM8K, MATH-500), commonsense reasoning (CSQA), and interdisciplinary STEM problems (GPQA-Diamond), and observe consistent accuracy-efficiency trends, suggesting that the reflection-redundancy signal is not domain-specific.

Across styles, REFRAIN is primarily driven by semantic redundancy signals rather than surface-level phrasing, as evidenced by the following two points: (1) We re-ran all experiments under three different instruction templates (P0-P2). As shown in Table 6, REFRAIN remains stable across all templates. (2) Semantic redundancy detection is style-agnostic. As shown in Table 8, we replaced SBERT with SimCSE and obtained very similar performance, whereas purely surface-level metrics degraded token savings and accuracy.

Across model architectures and scales, we evaluate REFRAIN on two model families and parameter scales (Qwen3-8B and gpt-oss-20B), demonstrating that the method is architecture-agnostic and stable across model sizes.

Regarding language, our experiments focus on English reasoning tasks and English LLMs, which defines the intended scope of this work and aligns with the majority of existing CoT research. We emphasize that the framework itself does not assume English-specific properties: both the trigger extraction and the semantic redundancy scorer are modular components. We leave systematic evaluation across non-English languages to future work.

Appendix B Results of Test-time Scaling with Budgeted Thinking

The results of test-time scaling with budgeted thinking are shown in Table 7.

We prioritize the budgeted thinking curve as shown in Figure 1 to evaluate the accuracy-efficiency frontier, rather than single-point comparisons at fixed token or accuracy levels. This choice is motivated by the inherent stochasticity of adaptive stopping and the non-monotonic relationship between stopping thresholds and reasoning performance. REFRAIN and baseline methods differ substantially in how they terminate reasoning (e.g., convergence-based stopping, entropy thresholds, redundancy checks). As a result, neither accuracy nor token usage is directly controllable. For example, Pass@1 may not change monotonically with the stopping signal, so matching a target Pass@1 level cannot be achieved reliably without extensive hyperparameter sweeps. By presenting the entire frontier, we offer a more robust and comprehensive assessment of each method's capability under varying resource constraints, consistent with the evaluation paradigms in recent research (Yang et al., 2025b).

Appendix C Alternatives for the Redundancy Scorer

REFRAIN's early stopping strategy relies on a semantic redundancy scorer to measure similarity between the current step and historical steps. To test robustness, we compared the default sentence-transformer encoder (SBERT) with alternative similarity functions, including another embedding-based model (SimCSE) and two sequence overlap metrics (TF-IDF, ROUGE-L). All experiments were conducted using the Qwen3-8B model on the MATH-500 benchmark.

The results in Table 8 show that both SBERT and SimCSE achieve substantial token reduction with minimal accuracy loss, indicating that embedding-based semantic representations are effective for detecting redundancy. In contrast, TF-IDF and

Category	Trigger phrases
Self Check (V_{check})	wait; let me check; hold on; have made a mistake; let me double check; wait a moment; is that correct; let me re-read
Strategy Shift (V_{shift})	alternatively; let me try; think of it as; let me consider; what if we try; let's think from a different angle; an alternative method would be; instead of doing that
Uncertainty (V_{uncert})	not sure; looks like; that seems; hmm; perhaps; maybe i; i'm not certain; it seems; i suspect; my guess is
Retrospective (V_{retro})	earlier we saw; from before; so now we have; recall that; let me go back; as we established previously; based on our previous result; remember that we found; the value from step
New Category	**simplify this problem**; **the core of the problem is**; **this is equivalent to**; **this is equal to**; **the key insight here is**; **break this down**; **the overall plan is to**; **the plan is to**

Table 5: Reflection trigger vocabulary $V = V_{\text{check}} \cup V_{\text{shift}} \cup V_{\text{uncert}} \cup V_{\text{retro}}$. Additionally, we use underline to indicate the in-category expansions and **bold** to indicate the new category in Section 5.2.

Method	GSM8K		MATH-500		CSQA		GPQA-Diamond	
	Pass@1↑	#Tokens↓	Pass@1↑	#Tokens↓	Pass@1↑	#Tokens↓	Pass@1↑	#Tokens↓
Vanilla (P_0)	94.24	2.62M	91.40	2.64M	83.13	1.66M	53.54	1.81M
REFRAIN (P_0)	94.54	1.68M	91.20	1.61M	84.03	0.76M	60.10	1.42M
Vanilla (P_1)	94.69	2.84M	91.20	2.28M	84.77	1.83M	61.11	1.88M
REFRAIN (P_1)	94.77	1.89M	91.00	1.69M	82.56	0.79M	61.11	1.35M
Vanilla (P_2)	94.92	2.45M	92.00	2.52M	83.21	1.80M	59.60	1.91M
REFRAIN (P_2)	94.92	1.58M	92.20	1.68M	82.80	0.79M	59.09	1.42M

Table 6: REFRAIN vs. Vanilla under prompt variants P_0-P_2.

ROUGE-L yield higher token usage, suggesting that surface-level overlap metrics fail to capture paraphrased or semantically redundant reasoning steps. These findings confirm that embedding-based scorers best capture reflective redundancy in CoT reasoning.

Appendix D Prompt Robustness

While all main experiments use a single instruction template P_0, reasoning behavior can be sensitive to small prompt wording changes. To ensure that our conclusions are not artifacts of a particular wording choice, we evaluate prompt robustness by re-running the same method and configurations under two paraphrased prompt templates P_1 and P_2. We evaluate on four benchmarks using Qwen3-8B. The three prompts are:

- P_0 : {question}\nPlease answer step by step. End your response with: Final Answer: \boxed{your final answer here}. Make sure to wrap your final answer in \boxed{}.
- P_1 : You are a helpful AI Assistant, designed to provide well-reasoned and detailed responses. You FIRST think about the reasoning process step by step and then provide the user with the answer. \nQuestion: {question}\nPlease enclose your final answer in the box: Final Answer: \boxed{Your Answer}.
- P_2 : Please solve the following question. Question: {question}\n\nAfter reasoning step by step, conclude with the final answer in the format: Final Answer: \boxed{Your Answer}.

According to Table 6, REFRAIN maintains vanilla-level accuracy — sometimes modestly better — while consistently using far fewer thinking tokens across three paraphrased templates (P_0-P_2) on four benchmarks. In addition, changing prompts can cause significant Pass@1 and token drift in vanilla models, such as Pass@1 ranging from 53.54 to 61.11 on GPQA-Diamond and #Tokens ranging from 2.28M to 2.64M on MATH-500. REFRAIN effectively stabilizes model behavior under prompt perturbations, yielding stable cost with no catastrophic accuracy drops. Overall, these results indicate that REFRAIN is robust to reasonable prompt rewrites and reduces the need for prompt-specific tuning during deployment.

Appendix E Implementation Details

Qwen3-8B. Qwen3-8B (Yang et al., 2025a) is a dense decoder-only model with approximately 8

GSM8K	Pass@1↑	#Tokens↓
Vanilla	94.24	2.62M
Budget (512 tokens)	76.12	0.88M
Budget (1024 tokens)	89.76	1.40M
Budget (1536 tokens)	92.27	1.81M
REFRAIN	94.54	1.68M

MATH-500	Pass@1↑	#Tokens↓
Vanilla	91.40	2.64M
Budget (1024 tokens)	69.20	0.60M
Budget (2048 tokens)	80.80	1.05M
Budget (3072 tokens)	84.20	1.41M
REFRAIN	91.20	1.61M

CSQA	Pass@1↑	#Tokens↓
Vanilla	83.13	1.66M
Budget (256 tokens)	80.02	0.38M
Budget (512 tokens)	82.64	0.76M
Budget (768 tokens)	82.80	1.03M
REFRAIN	84.03	0.76M

GPQA-Diamond	Pass@1↑	#Tokens↓
Vanilla	53.54	1.81M
Budget (2048 tokens)	40.91	0.42M
Budget (4096 tokens)	51.52	0.83M
Budget (8192 tokens)	54.04	1.38M
REFRAIN	60.10	1.42M

Table 7: Test-time scaling with fixed thinking budgets vs. adaptive stopping.

billion parameters. For Qwen3-8B, we enable the official thinking mode and adopt the recommended decoding configuration: Temperature = 0.6, Top-p = 0.95, and Top-k = 20.

gpt-oss-20B. gpt-oss-20B (Agarwal et al., 2025) is a mixture-of-experts (MoE) model with 21 billion total parameters and 3.6 billion active parameters per forward pass. For gpt-oss-20B, we use bfloat16 precision and apply the recommended decoding configuration: Temperature = 1.0, Top-p = 1.0, and Top-k = 50.

We deploy the models on NVIDIA A100 GPUs using PyTorch and HuggingFace Transformers (Wolf et al., 2019). The stopping threshold τ is discretized over [0.60, 0.80] with a step size of 0.05. This range empirically captures the transition between conservative and aggressive stopping behavior. In practice, the exploration constant $C = 1$ is widely used in UCB-based algorithms and is considered a standard and stable choice in non-stationary bandit settings. Therefore, we adopt $C = 1$ following common practice. We set the sliding-window size to W=100 by default. We use a default reward coefficient of λ=0.2 to balance answer confidence and reasoning length. We set the maximum generation length to 16,384 tokens unless otherwise specified. The random seed is set to 42. We will release the code on https://github.com/RLSNLP/Adaptive-Reasoning.

Method	Pass@1↑	#Tokens↓
Vanilla	91.40	2.64M
Embedding-based Similarity Metrics		
SBERT (default)	91.20	1.61M
SimCSE	91.40	1.67M
Sequence Overlap Metrics		
TF-IDF	91.20	1.91M
Rouge-L	90.40	1.92M

Table 8: Comparison of alternative redundancy scorers

The introduction of PRM/RMs used in Section 5.4 is as follows:

AceMath-7B. AceMath-7B (Liu et al., 2024) is a math reward model (RM) that assigns a scalar score to the complete solution.

Qwen2.5-MATH-PRM-7B. Qwen2.5-MATH-PRM-7B (Zhang et al., 2025b) is a math process reward model (PRM) that assigns a score to each step of the solution, and we take the mean as the final score.

Skywork-Reward-V2-Llama-3.1-8B. Skywork-Reward-V2-Llama-3.1-8B (Liu et al., 2025) is a general-purpose reward model that assigns a single scalar score to a given prompt-response pair.

Appendix F Hyperparameter Sensitivity and Robustness of REFRAIN

This section analyzes the sensitivity of REFRAIN to its key hyperparameters, including the stopping threshold grid $\mathcal{T}$, sliding window size W, reward coefficient λ, and the cold-start penalty. Our goal is not to optimize these hyperparameters for peak performance, but rather to verify that REFRAIN exhibits stable behavior under reasonable variations, and degrades gracefully only under clearly unreasonable settings. All experiments are conducted on the MATH-500 benchmark using Qwen3-8B, with the same backbone and decoding configuration.

Threshold grid $\mathcal{T}$ The similarity threshold τ determines when a reasoning step is considered semantically redundant relative to previous steps. In the main experiments, we discretize τ over the range $[0.60, 0.80]$ with step size 0.05. To test ro-

bustness, we expand this range to a broader grid $[0.50, 0.90]$ while keeping all other components unchanged. Results are shown in Table 9.

We have two observations: (1) The performance remains stable under the expanded grid, indicating that REFRAIN does not rely on a narrowly tuned threshold range to function effectively. (2) SW-UCB still predominantly selects thresholds in the 0.60–0.80 region during inference. It suggests that the adaptive mechanism naturally concentrates probability mass on effective thresholds, rather than being sensitive to the grid boundaries.

Threshold grid $\mathcal{T}$	**Pass@1↑**	**#Tokens↓**
0.6-0.8 (step 0.05)	91.20	1.61M
0.5-0.9 (step 0.05)	90.60	1.69M

Table 9: Sensitivity of REFRAIN to the similarity threshold grid $\mathcal{T}$.

Window size W The window size W controls how many recent rewards are retained when estimating the value of each threshold arm in Eq. 7. In addition to the default setting $W = 100$, we tested a smaller but still reasonable value of $W = 50$, and a very small unreasonable value $W = 3$. The results are shown in Table 10.

Window size W	**Pass@1↑**	**#Tokens↓**
100 (default)	91.20	1.61M
50 (reasonable)	91.80	1.68M
3 (very small)	90.20	1.72M

Table 10: Sensitivity of REFRAIN to the window size W.

We observe that $W = 50$ yields comparable results to the default setting. This indicates that REFRAIN is not sensitive to the exact choice of window size, as long as it remains within a reasonable range. In contrast, setting $W = 3$ significantly degrades performance: reward estimates become highly unstable, leading to suboptimal threshold selection and reduced accuracy. This behavior is expected and highlights that extremely small windows break the underlying assumption of reward smoothing in non-stationary bandits.

Reward coefficient λ It controls the trade-off between answer correctness and reasoning length in Eq. 9. Besides the default value $\lambda = 0.2$, we selected another reasonable alternative $\lambda = 0.3$ and an extreme value $\lambda = 100$ that aggressively penalizes token usage. The results are shown in Table 11.

Reward coefficient λ	**Pass@1↑**	**#Tokens↓**
0.2 (default)	91.20	1.61M
0.3 (reasonable)	91.80	1.71M
100 (extreme)	89.00	1.69M

Table 11: Sensitivity of REFRAIN to the reward coefficient λ.

For λ in a reasonable range, performance remains stable. This suggests that REFRAIN does not require fine-grained tuning of the reward scale to function effectively. In contrast, an excessively large λ significantly reduces accuracy, as the multi-armed bandit tends to stop earlier, ignoring the confidence level of the answer.

Cold-start coefficient At the very beginning of inference, the running mean token length $\bar{L}$ is undefined in Eq. 9. To avoid unstable rewards during this stage, we apply a linear cold-start penalty proportional to the total token length. We vary the cold-start coefficient across three orders of magnitude, and report results in Table 12.

We find that REFRAIN is insensitive to this coefficient within a broad range. All three settings achieve comparable results, which indicates that the cold-start penalty primarily serves as a stabilizing mechanism for the initial few rounds, rather than a critical hyperparameter that affects long-term behavior.

Cold-start coefficient	**Pass@1↑**	**#Tokens↓**
0.001	91.60	1.70M
0.0001 (default)	91.20	1.61M
0.00001	91.20	1.67M

Table 12: Sensitivity of REFRAIN to the cold-start coefficient.

Summary Across all examined hyperparameters, REFRAIN demonstrates strong robustness under reasonable variations. Performance remains stable for broad ranges of $\mathcal{T}$, W, λ, and cold-start coefficients, and degrades only under clearly unreasonable settings that violate the assumptions of adaptive thresholding or reward shaping. These results support our claim that REFRAIN is insensitive to hyperparameter choices and can be deployed in practice with reasonable hyperparameter settings.

Appendix G Bandit Variants: ϵ-greedy MAB vs. SW-UCB

Our adaptive threshold method requires online selection of the current question's threshold from a set of similarity thresholds $\mathcal{T}$, thereby balancing 'token savings' with 'generating correct answers'. Because task difficulty and model state vary over time, the optimal threshold is non-stationary. Therefore, we compare SW-UCB (Garivier and Moulines, 2008) with another classic multi-armed bandit (ϵ-greedy MAB) strategy (Sutton et al., 1998). Specifically, ϵ-greedy explores any arm $t \in \mathcal{T}$ with probability ϵ or selects the arm t with the highest average reward $\bar{R}_t$ with probability 1-ϵ:

$$t = \arg\max_{t \in \mathcal{T}} \bar{R}_t, \quad \bar{R}_t = \frac{\sum R_t}{n_t}. \tag{10}$$

The reward definition follows Eq. 9.

Method	GSM8K		MATH-500	
	Pass@1↑	#Tokens↓	Pass@1↑	#Tokens↓
MAB	94.62	1.70M	91.20	1.72M
SW-UCB	94.54	1.68M	91.20	1.61M

Method	CSQA		GPQA-Diamond	
	Pass@1↑	#Tokens↓	Pass@1↑	#Tokens↓
MAB	82.47	0.79M	55.05	1.39M
SW-UCB	84.03	0.76M	60.10	1.42M

Table 13: ϵ-greedy MAB vs. sliding-window UCB for adaptive threshold selection.

Table 13 shows that both methods perform comparably on GSM8K and MATH-500, with SW-UCB generating slightly fewer tokens. On CSQA and GPQA-Diamond, where the question semantics fluctuate more, SW-UCB attains higher accuracy at similar or lower token budgets. These results indicate that SW-UCB better adapts to non-stationary reward dynamics, providing more consistent thresholding behavior. Overall, SW-UCB is a more robust choice: it conserves tokens in relatively stable settings and improves accuracy in volatile ones.

Appendix H Case Study

To understand how our UCB-based early-stopping mechanism shapes model reasoning, we examined representative examples across datasets. In these domains, early stopping acts as a selective gate on the decoding trajectory: it halts continuation once confidence peaks, preserving early coherent reasoning while discarding later, low-confidence expansions. This mechanism can prevent semantic or numeric drift—but it can also truncate necessary self-corrections. The four cases below illustrate both sides of this trade-off within a unified narrative.

When early stop succeeds, it prevents **over-elaboration**, **re-computation**, or **format degradation**—errors that typically appear after the model has already reached a correct intermediate conclusion. When it fails, it interrupts **self-repair**: the model has not yet completed its reasoning, but the confidence temporarily spikes on an incorrect intermediate token, causing premature termination.

Across all examples, early stopping behaves like a *confidence filter*—it favors precision at the cost of completeness. In commonsense and physical reasoning (CSQA, GPQA), it prunes away irrelevant elaboration or unstable numerical recomputation. In multi-step quantitative reasoning (MATH, GSM8K-type), it can freeze an intermediate value before the chain converges to the correct expression. Overall, early stopping helps when the first coherent burst already contains the solution essence, but harms when correctness emerges only after longer deliberation.

In the first case presented in Table 14, early stopping prevented *semantic drift*: the baseline continued elaborating until it replaced "office" with the more frequent but irrelevant "classroom." In the second, it prevented *numeric hallucination*: the baseline re-integrated equations and inflated the distance to 9 Gpc, whereas early stop froze the stable 6 Gpc block. In the third, it avoided *format degradation*: stopping right after the correct $\boxed{52_8}$ preserved syntax that the baseline later corrupted to plain "52." In the fourth, however, the same mechanism caused a *truncation error*: the model had not yet applied the geometric constraint that halves the angle, so early stop froze an intermediate 62° instead of the correct 28°.

Together these examples reveal that early stopping serves as a precision-biased filter: it trims the low-probability tail of reasoning—removing noise and overthinking—but can also cut off genuine late corrections. In practice, combining early stop with a minimum reasoning length or explicit "Final Answer" check mitigates this risk while retaining its benefits.

Dataset	Question	Gold	Baseline	Ours
CSQA	**Where do adults use glue sticks?** (A) classroom (B) desk drawer (C) at school (D) office (E) kitchen drawer	D	. . . reasons that glue sticks are for sticking papers, then shifts to "classroom" context; overthinks into child-use scenario. **Final Answer: A (classroom)**	. . . identifies adults use them in professional settings, stops at "office" without drift. **Final Answer: D (office)**
GPQA	**Observations of a quasar across the electromagnetic spectrum are being carried out.** Focusing on the near infrared and optical spectrum, there is a peak at wavelength of about 790 nm, and at shorter wavelengths $<$ 790 nm the flux drops significantly. If we live in a flat ΛCDM universe ($H_0 = 70$ km s^{-1} Mpc^{-1}, $\Omega_m = 0.3$, $\Omega_\Lambda = 0.7$), what is the most likely comoving distance (for $a = 1$) of this quasar from the Earth? (A) 8 Gpc (B) 7 Gpc (C) 6 Gpc (D) 9 Gpc	A	. . . computes ∼6 Gpc correctly at first, then re-derives equations with slightly altered constants, inflating to **9 Gpc (D)**. Numeric hallucination from over-expansion.	. . . halts right after consistent calculation, preserving **6 Gpc (A)** before re-computation noise.
MATH	**Find the product** $6_8 \times 7_8$ **and express the result in base 8.**	52_8	. . . correctly gets 42_{10}, but continues verbose explanation, drops subscript and outputs **52**. Formatting error after over-elaboration.	. . . stops immediately after producing boxed 52_8, avoiding format decay.
MATH	**In a triangle with** $AB = BC$ **and** $BC \parallel$ **a line through** A**, find** $\angle x$**.**	28°	. . . initially gets 60°, continues reasoning, self-corrects to 28° after applying geometric constraint.	. . . confidence peaks early, stops at intermediate 62°, missing late correction.

Table 14: Representative examples showing how early stopping prevents or causes specific reasoning errors. Green = correct; Red = incorrect. CSQA and GPQA questions and options are taken verbatim from the original datasets.

Dataset	Question	Gold	Baseline	Ours
CSQA	**Where do adults use glue sticks?** (A) classroom (B) desk drawer (C) at school (D) office (E) kitchen drawer	D	<think> ... "Glue sticks are often used in schools." ... "teachers (adults) use glue sticks in a classroom." ... "Final Answer: A (classroom)" ... </think>	<think> ... "Adults would more typically use glue sticks in an office for attaching papers." ... "At school is too general." ... "Final Answer: D (office)" ... </think>
GPQA	**Observations of a quasar across the electromagnetic spectrum are being carried out.** Focusing on the near infrared and optical spectrum, there is a peak at wavelength of about 790 nm, and at shorter wavelengths $<$ 790 nm the flux drops significantly. If we live in a flat ΛCDM universe ($H_0 = 70$ km s^{-1} Mpc^{-1}, $\Omega_m = 0.3$, $\Omega_\Lambda = 0.7$), what is the most likely comoving distance (for $a = 1$) of this quasar from the Earth? (A) 8 Gpc (B) 7 Gpc (C) 6 Gpc (D) 9 Gpc	A	<think> ... "$\lambda_{\text{obs}}/\lambda_{\text{rest}} \approx 790/400 \Rightarrow z \approx 1$." ... "comoving distance ≈ 6 Gpc." ... then "maybe $z = 2 \rightarrow 9$ Gpc." ... "Final Answer: D (9 Gpc)" ... </think>	<think> ... "At $z \approx 1$, $d_c \approx 6$ Gpc under ΛCDM." ... "consistent with given parameters." ... "Final Answer: A (6 Gpc)" ... </think>
MATH	**Find the product $6_8 \times 7_8$ and express the result in base 8.**	52_8	<think> ... "$6 \times 7 = 42$." ... "Convert 42 to base $8 \rightarrow 52_8$." ... then continues ... "drops subscript $\rightarrow$ Final Answer: 52." ... </think>	<think> ... "$6 \times 7 = 42 \rightarrow 52_8$." ... "Stops right after boxed 52_8." ... </think>
MATH	**In a triangle with $AB = BC$ and $BC \parallel$ a line through A, find $\angle x$.**	28°	<think> ... "Parallel lines $\rightarrow$ alternate interior angles equal." ... "$x = 28^\circ$." ... </think>	<think> ... "maybe supplementary, $180 - 118 = 62^\circ$." ... "Final Answer: 62°." ... </think>

Table 15: Excerpted raw model responses showing only critical reasoning segments (dark green = correct, red = incorrect). Non-essential text is omitted as "..." for brevity.