Mitigating Overthinking in Language Models Using Dynamic Stopping Criteria

Anonymous ACL submission

Abstract

Large language models (LLMs) often overthink, producing unnecessarily long chains-of-thought that waste computation and sometimes degrade accuracy. Several empirical mitigation approaches exist, but an operational definition for overthinking is lacking. Existing detectors rely on single signals (e.g., hesitation tokens, entropy spikes) that are brittle, while budget-based methods can over-truncate hard problems. The link between output-level signals and internal model state is underexplored, and no training-free, inference-time method exists that robustly detects overthinking without oracle labels or multi-problem calibration. We address these gaps with a training-free, inference-time framework that detects overthinking from observable output behavior and halts generation once reasoning yields diminishing returns. Across five benchmarks spanning math, multi-hop QA, code, scientific QA, and commonsense reasoning, our method reduces generated tokens by 27–89% relative to baseline. Against budget-based and cue-based baselines under matched protocols, our score-based stopping recovers more of the accuracy ceiling at comparable token savings. These results establish behavior-aware online stopping as a deployment-ready path to lower-cost LLM reasoning.

1 Introduction

Large language models (LLMs) increasingly rely on extended chain-of-thought (CoT) generation to solve complex problems. Recent reasoning-tuned models such as DeepSeek-R1 (DeepSeek-AI, 2025) routinely emit thousands of intermediate tokens per query. A growing body of work shows that much of this generation is *excessive*: models continue elaborating well past the point at which the final answer has been implicitly determined, wasting compute and sometimes degrading accuracy (Chen et al., 2025; Pu et al., 2025). This behavior, commonly called *overthinking*, typically manifests as repetition, redundancy, and hedging that adds little informational content to the trace.

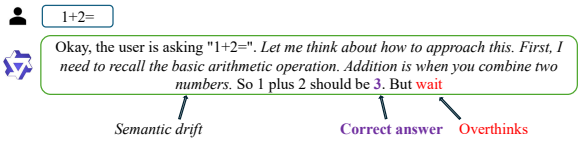

Figure 1: An illustration of overthinking: the model generates redundant or repetitive reasoning steps that do not improve the final answer.

Despite growing attention, four interrelated gaps remain: definition, detection, interpretation, and intervention. For *definition*, no operational, sentence-level criterion for overthinking exists that can be computed online from output text. Prior work treats overthinking implicitly as exceeding a fixed token budget (Han et al., 2025), or via post-hoc per-problem labels. For *detection*, existing detectors rely on a single brittle cue, such as hesitation tokens (Yang et al., 2025), entropy spikes after `</think>` (Wang et al., 2025), or run-length convergence (Mao et al., 2025), while multi-problem threshold calibration (Sun et al., 2025) is impractical for single-shot deployment. For *interpretation*, efficiency-focused work reports tokens saved without showing whether the savings stem from overthinking-related behavior (repetition, hedging, looping) or arbitrary truncation, leaving the link between output signals and internal model state underexplored. For *intervention*, top-down budgeting over-truncates hard problems when difficulty estimates are wrong, and cue-based methods fail to fire when the model overthinks silently, so a robust intervention should adapt per-instance without retraining or oracles.

We propose a training-free framework that addresses each gap in turn. For *definition*, we operationalize overthinking as low-information reasoning along three axes: semantic similarity to previ-

ous sentences (content reuse), repetition ratio (lexical reuse), and hedge density (stalling language). For *detection*, we compute a composite overthinking score t_{score} that combines all three signals continuously, rather than committing to a single cue. For *interpretation*, we pair t_{score} with a logit-lens mismatch probe and report ablations showing which signals drive savings on which tasks. For *intervention*, t_{score} supplements a small set of hard triggers (final-answer keywords, `</think>`, repetition loops); once any trigger fires we halt generation and prompt the model to emit its final answer. The framework requires no fine-tuning, no difficulty prediction, no multi-problem warm-up, and minimal access to model internals.

Across five benchmarks and four open-weight model families, our method reduces generated tokens by 27–89% relative to vanilla CoT. On MATH500 (Hendrycks et al., 2021) with DeepSeek-R1-Distill-Llama-8B, it cuts tokens by 30.5% with 1.2 points of accuracy loss (82.6% $\rightarrow$ 81.4%); on smaller distilled models (Qwen-1.5B) and instruction-tuned LLaMA-3.1-8B, it improves accuracy while shortening output by 28% and 72%, suggesting that excessive CoT can hurt smaller or less reasoning-specialized models. The method extends to HotpotQA (Yang et al., 2018), GPQA (Rein et al., 2023), and CommonsenseQA (Talmor et al., 2019) with similar trends. We also report a meaningful trade-off on HumanEval (Chen et al., 2021) (62% token savings, 3.7-point pass@1 drop), which we attribute to the higher precision requirements of code synthesis.

Our contributions are:

- An operational, sentence-level definition of overthinking.
- A training-free detection mechanism using a composite score rather than a single cue.
- Interpretability analyses (signal ablations, logit-lens diagnostics, qualitative trigger examples) linking output behavior to internal state.
- An intervention mechanism evaluated head-to-head against budget-based and cue-based baselines, with an accounting of the accuracy-efficiency trade-off.

2 Related Work

Overthinking in long reasoning. Chen et al. (2025) provide a comprehensive study of overthinking in chain-of-thought generation, showing that models frequently generate excessive tokens on problems that do not benefit from extended reasoning. Recent surveys (Wen et al., 2025) catalogue the growing body of work addressing this inefficiency across training-based and inference-time methods.

Budget-based and training-dependent methods. Pu et al. (2025) introduce THOUGHTTERMINATOR, which dynamically allocates a token budget based on estimated problem difficulty, but depends on external difficulty predictors that can misestimate the required budget. Muennighoff et al. (2025) present *s1*, combining supervised fine-tuning with budget forcing mechanisms. Han et al. (2025) propose token-budget-aware prompting that embeds budget constraints directly into the prompt. Because these methods require either difficulty calibration, training, or task-specific prompt engineering, they differ fundamentally from training-free online approaches.

Dynamic early exit during generation. Most closely related to our work are training-free methods that detect when to stop during generation. Yang et al. (2025) propose DEER, which monitors lexical cues (e.g., "Wait") and triggers trial answers based on the model's internal confidence, but depends on specific markers and logit access. Sun et al. (2025) propose REFRAIN, which combines hesitation-token triggers with a sentence-embedding similarity scorer and a UCB bandit for adaptive thresholding. Concurrent work includes EAT (Wang et al., 2025), which monitors token entropy after `</think>` as a stopping signal, and ES-CoT (Mao et al., 2025), which detects answer convergence via run-length tracking. Our method differs from these approaches in three respects: we use a *continuous* composite score (t_{score}) combining multiple output-derived signals rather than relying on a single indicator, we integrate a logit-lens mismatch monitor as a complementary internal-state signal, and we require no bandit-based warm-up or multi-sample calibration across problems.

3 Method

3.1 Operational Definition of Overthinking

We operationalize overthinking at the sentence level. Let a reasoning trace $r = (s_1, s_2, \ldots, s_t)$ be the sequence of sentences emitted so far, where sentence boundaries are determined by punctuation. We define overthinking at step t as the event

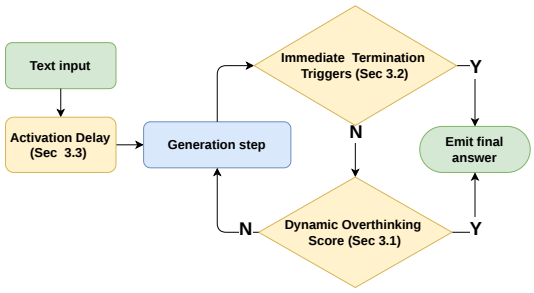

Figure 2: Dynamic stopping framework overview. Immediate triggers are checked first, then t_{score}. If any condition fires, the model emits its final answer.

in which s_t contributes negligible new information toward producing the final answer, and instead exhibits one or more of three observable behaviors: (i) *content reuse*, restating an earlier sentence's meaning; (ii) *lexical reuse*, repeating surface forms already present in r; or (iii) *stalling*, marked by a high density of hedging, hesitation, or self-correction tokens. This definition is intentionally output-level and sentence-local so it can be computed without retraining or oracle labels.

3.2 Framework Overview

Given the definition above, our framework decides at each sentence boundary whether to continue generation, halt and prompt for the final answer, or do neither. The decision combines two layers: (a) a set of *hard triggers* that fire on explicit completion cues or pathological patterns, and (b) a continuous *overthinking score* t_{score} that aggregates the three behavioral signals above. Hard triggers are checked first because they correspond to unambiguous evidence; if none fires, t_{score} is evaluated against a threshold τ.

3.3 Dynamic Overthinking Score

The score t_{score} is recomputed after each new sentence s_t from three signals, each targeting one behavior in our definition. We use a sentence-local linear composite because the signals measure complementary phenomena at different granularities (embedding-level meaning, surface-form tokens, and lexical hedging), and a linear form keeps each signal's contribution individually interpretable and ablatable. We do not claim this composite is optimal; we instead verify in Section 5.2 that all three signals contribute and that the score is robust to moderate weight perturbations.

1. **Semantic similarity (Sim).** The maximum cosine similarity between the embedding of s_t and the embeddings of $s_1, \ldots, s_{t-1}$. We use the all-MiniLM-L6-v2 sentence encoder and update embeddings incrementally. High Sim flags *content reuse*.
2. **Repetition ratio (RepRatio).** For each new sentence, we count how many earlier sentences in r contain it (or a near-paraphrase, detected via embedding similarity above 0.9), and divide by the sentence length in tokens. This proxy captures both exact and near-exact lexical reuse without explicit n-gram enumeration.
3. **Hedge density (Hedge).** The count of tokens in s_t that match a curated lexicon of hedging, hesitation, and self-correction markers (e.g., "maybe", "wait", "let me double-check"), normalized by sentence length. The lexicon is listed in Appendix C.

We combine the signals into a single score

$$t_{\text{score}} = \lambda_1 \cdot \text{Sim} + \lambda_2 \cdot \text{RepRatio} + \lambda_3 \cdot \text{Hedge}\,, \quad (1)$$

with $\lambda_1 + \lambda_2 + \lambda_3 = 1$. A stop is prepared whenever $t_{\text{score}} > \tau$; otherwise generation continues. The weights are tuned once via Optuna on a held-out subset of MATH500 (Section 4) and reused across all subsequent experiments, including out-of-distribution datasets and other model families.

Repetition-loop override. As a special case of repeated content reuse, we maintain a count of how often any sentence (or its paraphrase) reappears across r. If the count for any sentence exceeds a small constant (we use 5), we interpret this as a degenerate loop and trigger stopping immediately, bypassing t_{score}. This safeguards against pathological cycles in which individual sentences fall just below threshold but their aggregate behavior is clearly redundant.

3.4 Hard Triggers

Hard triggers fire on unambiguous evidence that the model has either signaled completion or entered a degenerate state. They override t_{score} and act as safety nets for cases the continuous score may catch only gradually. In practice, final-answer indicators and `</think>` are the most reliable signals; the mismatch trigger, repetition-loop override, and t_{score} together cover failure modes without explicit completion cue.

Final-answer indicators. We monitor for explicit completion phrases ("final an-

swer", "correct answer") and special tokens (⟨End_of_Solution⟩, </think>). We allow up to two occurrences of a completion phrase before stopping, to avoid premature halts on incidental usage, and stop immediately on </think>, which structurally marks the end of internal reasoning in models that use the tag.

Logit-lens mismatch. We additionally monitor the internal states, applying a logit lens (Nostalgebraist, 2020) to compare the top-1 prediction at intermediate transformer layers against the final-layer output. We accumulate a count of disagreements and stop when it exceeds a fixed threshold (35 in our experiments), interpreting this as drifting or unstable reasoning. Section 5.2 shows this trigger contributes modestly on MATH500 and primarily catches pathological trajectories.

3.5 Activation Delay and Intervention

The score-based trigger is disabled for the first L_{act} tokens of generation to give the model room to establish an initial reasoning trajectory; only hard triggers are active during this warm-up. We set $L_{\text{act}} = 160$, chosen by Optuna on the same held-out subset used for λ and τ. During warm-up, behavioral signals are noisy by design: short traces have few prior sentences to compare against, so Sim and RepRatio are uninformative until the trace has accumulated some content.

After warm-up, each new sentence is evaluated by the full cascade. If any hard trigger fires, generation halts immediately. Otherwise, t_{score} is computed; if it exceeds τ, generation halts. When generation halts, we append a task-specific directive to the model's context and let it emit only the final answer. For math, we use `### Final answer only. Do not explain.`; for other tasks, we adapt the wording to the expected output format (short answer, extracted span, or code completion). The directive prevents the model from re-entering a reasoning loop after the stop.

Section 5 shows that this cascade produces substantial token savings on instances where the model has effectively converged, while allowing extended reasoning on harder problems where signals remain below threshold.

4 Experiments and Setup

4.1 Datasets and Tasks

We primarily evaluate on MATH500, a benchmark of 500 challenging math word problems derived from the MATH dataset (Hendrycks et al., 2021). MATH500 is well-suited for testing overthinking because it contains problems of varying difficulty (some can be solved in a few steps, others require extended reasoning), and it was used by prior overthinking mitigation work for evaluation (Pu et al., 2025; Yang et al., 2025). We use the standard test split for MATH500.

To evaluate generalization beyond math reasoning, we additionally consider several diverse benchmarks. We include HotpotQA (Yang et al., 2018), a multi-hop question answering dataset requiring reasoning across multiple supporting documents, and HumanEval (Chen et al., 2021), which focuses on code generation and functional correctness. These tasks differ substantially in structure and highlight how dynamic stopping behaves under longer reasoning chains (HotpotQA) and precise program synthesis (HumanEval).

We further evaluate on GPQA (Graduate-Level Problem QA) (Rein et al., 2023), along with CommonsenseQA (Talmor et al., 2019). These datasets span scientific reasoning, commonsense inference, and robustness to distractors.

4.2 Models

We experiment with open-weight large language models from the DeepSeek-R1 and LLaMA families (DeepSeek-AI, 2025). In particular, we evaluate DeepSeek-R1-Distill-Llama-8B variants, which are fine-tuned for mathematical and logical reasoning and are comparable in scale to LLaMA-8B models trained with reasoning distillation. We further include smaller distilled and Qwen-based variants, namely DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-1.5B, to study how dynamic stopping behaves across different model scales and architectures. These reasoning-specialized models frequently produce overly long chains of thought on benchmarks such as MATH500, making them a suitable testbed for studying overthinking and dynamic stopping. All model inference is performed using 4-bit quantization. The same quantization setting is applied uniformly across all baselines to ensure a fair comparison.

We additionally evaluate general-purpose instruction-tuned models, including LLaMA-3.1 8B Instruct (Meta AI, 2024) and the more compact LLaMA-3.2 3B Instruct, to assess whether dynamic stopping generalizes beyond reasoning-specialized models and remains effective at

smaller scales. The results are reported in Table 1 and discussed in Section 5.

4.3 Decoding Strategies

We use a near-deterministic decoding setting ($T = 10^{-7}$, $p = 0.95$) for all main results. The framework remains effective under two other settings: greedy ($T = 0$) and stochastic ($T = 0.6$, $p = 0.95$).

4.4 Implementation and Hyperparameters

We implemented our framework for 1.5B–13B scale transformer models in the DeepSeek and LLaMA families. Hyperparameters were tuned on a held-out subset of MATH500 (20 problems per trial, 50 Optuna trials with tree-structured Parzen estimators), targeting a balance between accuracy and tokens saved. The 20-problem tuning subset is disjoint from the problems used to compute all evaluation results in Section 5. The tuned parameters were the score weights $\lambda_1, \lambda_2, \lambda_3$, the threshold τ, the mismatch count threshold, and the activation length L_{act}. The search returned $\tau \approx 0.93$, mismatch threshold 35 (the maximum tested), $L_{\text{act}} \approx 160$, and $(\lambda_1, \lambda_2, \lambda_3) = (0.50, 0.375, 0.125)$. We fix these values for *all* subsequent experiments, including out-of-distribution datasets and other model families; the per-dataset transferability is itself an empirical claim that we assess in Section 5.

4.5 Evaluation Metrics

Primary metrics. We evaluate each method on effectiveness and efficiency:

- **Accuracy:** Whether the final answer is correct. For mathematical tasks, we normalize outputs (e.g., remove `\boxed{}`) and compare against ground truth, allowing minor symbolic or numeric equivalences (via SymPy).
- **Tokens Generated:** The number of tokens produced in the chain-of-thought *before* the final answer. This serves as a proxy for inference-time computation. We report average tokens per problem and relative savings compared to vanilla CoT.

Diagnostic metrics. To better understand model behavior, we compute auxiliary metrics:

- **Mismatch Frequency:** Average number of logit-lens mismatch events (Section 3), indicating internal inconsistency.
- **Semantic Drift:** Average cosine distance between reasoning steps and the problem statement, measuring topic deviation.
- **Repetition Rate:** Fraction of tokens that repeat earlier content, capturing redundancy and looping.

These diagnostics are used for qualitative analysis rather than model selection. Final answers are extracted using a consistent regex-based pipeline across all methods, following prior work (Yang et al., 2025), ensuring fair comparison.

Following prior work on overthinking and test-time scaling, we report efficiency primarily in terms of generated tokens, which directly reflects model-side computation.

4.6 Baselines

We compare against four reproducible baselines below. Two additional training-dependent or non-public methods (s1, REFRAIN) are discussed for context in Appendix B.

Vanilla CoT. The model generates a full chain-of-thought and final answer without intervention, exhibiting the overthinking-prone behavior we aim to mitigate.

Fixed Budget. Reasoning is hard-truncated after 256 tokens. This serves as a lower-bound on accuracy for aggressive stopping rather than a competitive baseline.

ThoughtTerminator (Pu et al., 2025). We evaluate two configurations. The *budget-only* variant uses a fixed 5000-token budget with reminders every 250 tokens, omitting the difficulty predictor to avoid oracle assumptions. The *difficulty-calibrated* variant follows the original method, using pass@1 for consistency with our single-answer protocol.

DEER (Yang et al., 2025). We use the authors' released implementation. DEER monitors for cue tokens (e.g., "Wait") during generation and injects a trial-answer prompt when such cues appear. We evaluate under a single-answer protocol with the same decoding setting as our method to enable a controlled comparison.

5 Results and Analysis

5.1 Main Results

Table 1 reports accuracy and token usage across all (dataset, model) pairs. Results use near-deterministic decoding ($T = 10^{-7}$, $p = 0.95$);

token savings are computed relative to vanilla CoT under matched decoding.

MATH500. On DeepSeek-R1-Distill-Llama-8B, vanilla CoT reaches 82.6% accuracy at ∼2700 tokens per problem. Our framework retains 81.4% accuracy (a 1.2-point drop) while cutting tokens by 30.5%. Among comparable training-free baselines, our method achieves the best accuracy at comparable savings: ThoughtTerminator with difficulty calibration reaches 71.4% at 35.8% savings, a 10-point accuracy gap at slightly higher token savings; the budget-only ThoughtTerminator variant saves negligible tokens (2%) at 69.6% accuracy; and DEER's released implementation, run under our single-answer protocol with matched decoding, reaches 60.0% at 43.5% savings. We attribute DEER's lower number relative to its original report to single-answer evaluation (no answer-consistency voting) and matched 4-bit quantization; we discuss this further in Section 6.

On smaller models the picture changes: dynamic stopping *improves* accuracy. For DeepSeek-R1-Distill-Qwen-1.5B, accuracy rises from 72.0% to 77.2% while tokens drop by 28%. For LLaMA-3.1-8B Instruct (a general-purpose model that is not reasoning-tuned), accuracy improves from 57.4% to 59.4% with a 72.8% token reduction. The accuracy gains on smaller and non-reasoning models are consistent with the hypothesis that extended CoT can degrade prediction quality when the model lacks the capacity to use the additional tokens productively.

Generalization to other tasks. The framework transfers to non-mathematical reasoning, though with task-dependent trade-offs. On HotpotQA, DeepSeek-R1-Distill-Llama-8B drops from 71.9% to 68.6% (3.3 points) at 89% token savings; Qwen-1.5B improves (40.4% → 44.5%) at 59% savings. The extreme compression on HotpotQA reflects that multi-hop QA traces frequently re-read the same supporting documents, which the similarity and repetition signals catch aggressively. On HumanEval, pass@1 drops from 67.7% to 64.0% (3.7 points) at 62% savings, a meaningful regression that we attribute to code synthesis's sensitivity to truncation of partially generated programs. On GPQA, scientific QA accuracy is approximately preserved (42.9% → 40.8% on DeepSeek-R1-Distill-Llama-8B at 30.9% savings; 29.7% → 30.0% on LLaMA-3.1-8B at 62.2% savings). CommonsenseQA shows the smallest swings (0.7-point drop on DeepSeek, 0.2-point gain on LLaMA-3.2-3B) at 27–65% savings.

Across the table, the method shifts token usage downward in every (dataset, model) cell while changing accuracy by at most a few points in either direction. Where accuracy improves, the gain is largest on smaller or non-reasoning models; where it regresses, the regressions cluster on tasks that require precise multi-step synthesis (HumanEval, GPQA on the larger model).

5.2 Ablation Studies

We ablate the two architectural choices (dynamic score vs. hard triggers) and the three components of t_{score}. All ablations use MATH500 with DeepSeek-R1-Distill-Llama-8B under near-deterministic decoding. Results are summarized in Table 2.

Score vs. triggers. Disabling t_{score} and relying only on hard triggers (final-answer indicators and mismatch) cuts token savings from 30.5% to 17.5% with a small accuracy drop (81.4% → 80.4%). Without the continuous score, the framework stops mostly when the model has already emitted explicit completion cues, missing the cases where it overthinks silently.

Mismatch trigger. Removing the logit-lens mismatch trigger while keeping t_{score} leaves token savings essentially unchanged ($\approx 31\%$) but reduces accuracy by 3 points (81.4% → 78.4%). The trigger rarely fires, yet when it does it tends to catch trajectories that drift mid-reasoning. We read this as evidence that mismatch acts as an accuracy-preserving safety net rather than an efficiency lever.

Score components. We re-tune τ when individual signals are removed to isolate each signal's contribution at a comparable operating point. Removing RepRatio (Sim+Hedge only) drops token savings to 24.0% at 81.2% accuracy: the score becomes less sensitive to surface-form reuse and stops less often, retaining nearly all the accuracy of the full model but recovering less efficiency. Removing Hedge (Sim+RepRatio only) keeps savings near the full configuration at 30.0% but accuracy slips to 80.8%, suggesting that the hedge signal occasionally vetoes stops that would have hurt accuracy. Semantic similarity is the single most informative signal: its removal collapses

Dataset	Method	Model	Acc. (%)	Avg Tokens	Saved (%)
MATH500	Vanilla CoT	DeepSeek-R1-Distill-Llama-8B	82.6	2700	0.0
	Fixed Budget	DeepSeek-R1-Distill-Llama-8B	50.5	256	90.5
	ThoughtTerminator	DeepSeek-R1-Distill-Llama-8B	69.6	2500	2.0
	ThoughtTerminator (w/diff.)	DeepSeek-R1-Distill-Llama-8B	71.4	1653	35.8
	DEER	DeepSeek-R1-Distill-Llama-8B	60.0	1525	43.5
	Ours	DeepSeek-R1-Distill-Llama-8B	**81.4**	1900	**30.5**
	Vanilla CoT	Qwen-1.5B Distill	72.0	1500	0.0
	Ours	Qwen-1.5B Distill	**77.2**	1080	**28.0**
	Vanilla CoT	LLaMA-3.1-8B	57.4	4555	0.0
	Ours	LLaMA-3.1-8B	**59.4**	1241	**72.8**
HotpotQA	Vanilla CoT	DeepSeek-R1-Distill-Llama-8B	71.87	2457.95	0.0
	Ours	DeepSeek-R1-Distill-Llama-8B	**68.60**	188.00	**89.37**
	Vanilla CoT	Qwen-1.5B Distill	40.4	846.71	0.0
	Ours	Qwen-1.5B Distill	**44.5**	344.78	**59.28**
HumanEval	Vanilla CoT	DeepSeek-R1-Distill-Qwen-7B	67.68	1500	0.0
	Ours	DeepSeek-R1-Distill-Qwen-7B	**64.02**	564	**62.48**
GPQA	Vanilla CoT	DeepSeek-R1-Distill-Llama-8B	42.85	4188.34	0.0
	Ours	DeepSeek-R1-Distill-Llama-8B	**40.84**	2893.3	**30.92**
	Vanilla CoT	LLaMA-3.1-8B	29.67	4151.11	0.0
	Ours	LLaMA-3.1-8B	**30.00**	1568.74	**62.21**
CSQA	Vanilla CoT	DeepSeek-R1-Distill-Llama-8B	59.56	1406.19	0.0
	Ours	DeepSeek-R1-Distill-Llama-8B	**58.90**	1029.89	**26.76**
	Vanilla CoT	LLaMA-3.2-3B	52.0	686.67	0.0
	Ours	LLaMA-3.2-3B	**52.2**	241.77	**64.79**

Table 1: Main results (best configurations only). Token savings are relative to vanilla CoT.

paraphrase detection and pushes savings below the no-t_{score} variant. The Optuna-selected weights $(0.50, 0.375, 0.125)$ reflect this ordering: similarity carries the most weight, repetition contributes substantially to efficiency, and hedging plays a smaller but accuracy-stabilizing role.

Robustness to decoding. Under stochastic decoding ($T = 0.6$), the full framework retains 76.2% accuracy (a 4-point drop from greedy) at comparable token savings (30.0%). The drop is consistent with the higher variance of stochastic generation rather than a failure of the stopping mechanism. Figure 3 visualizes the accuracy–efficiency trade-off across all stopping configurations. The Optuna-selected configuration (red star) achieves a favorable balance, retaining high accuracy while saving substantial tokens. Notably, more aggressive configurations yield diminishing accuracy returns, confirming that the tuned threshold $\tau \approx 0.93$ is near the Pareto frontier.

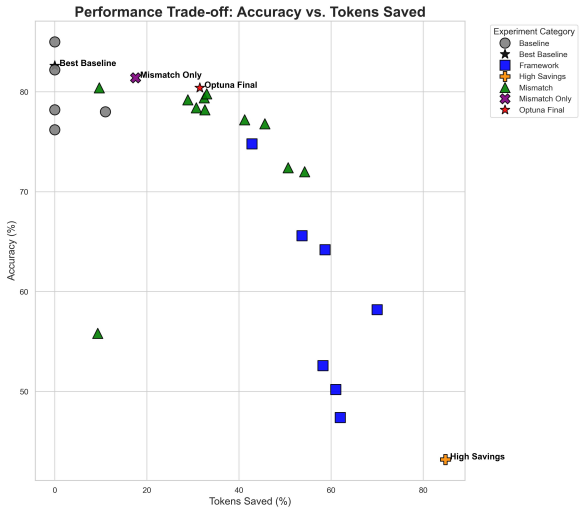

Figure 3: Accuracy vs. tokens saved on MATH500 across stopping strategies. Gray circles denote baselines; colored markers correspond to variants of our framework. The red star marks the Optuna-selected configuration.

6 Discussion

We position our results against prior work along three axes: how the method decides when to stop (top-down vs. bottom-up), how many signals it combines (single cue vs. composite), and how much model access it needs (training, prompts, logits, or output text), as Table 3 summarizes.

Top-down budgeting (ThoughtTerminator). ThoughtTerminator allocates a token budget before generation, optionally informed by a difficulty predictor. When the predictor is well-calibrated the method is competitive on tokens; when it is not, the budget either over-truncates

Setting	Decoding	Acc. (%)	Tokens Saved (%)
Vanilla CoT (no stopping)	Near-det. ($T = 10^{-7}$)	82.6	0.0
Full (Optuna-tuned)	Near-det. ($T = 10^{-7}$)	**81.4**	**30.5**
No t_{score} (triggers only)	Near-det. ($T = 10^{-7}$)	80.4	17.5
No mismatch trigger	Near-det. ($T = 10^{-7}$)	78.4	31.0
RepRatio + Hedge only (no Sim)	Near-det. ($T = 10^{-7}$)	79.2	20.0
Sim + Hedge only (no RepRatio)	Near-det. ($T = 10^{-7}$)	81.2	24.0
Sim + RepRatio only (no Hedge)	Near-det. ($T = 10^{-7}$)	80.8	30.0
Full (Optuna-tuned)	Stochastic ($T = 0.6$)	76.2	30.0

Table 2: Ablation on MATH500 with DeepSeek-R1-Distill-Llama-8B. "No t_{score}" relies only on hard triggers; "No mismatch trigger" disables the logit-lens probe. Component rows isolate the contribution of each signal in the score. Tokens Saved are relative to Vanilla CoT under the same decoding setting.

Method	Train?	Signals	Access
ThoughtTerm.	No	Difficulty	Prompt
DEER	No	Cues / confidence	Logits
s1	Yes	None	Tokens
REFRAIN	No	Triggers	Output
Ours	No	Triggers	Output/Logits

Table 3: Comparison of early-stopping methods.

hard problems or fails to constrain easy ones. Our MATH500 results illustrate the failure mode: the difficulty-calibrated configuration reaches 71.4% accuracy at 35.8% savings under a single-answer protocol, compared to 81.4% at 30.5% savings for our method. The 10-point gap suggests that even a calibrated top-down budget cannot match online behavior-aware stopping on a benchmark with heterogeneous problem difficulty.

Single-cue early exit. Under our single-answer protocol with matched 4-bit quantization, DEER (Yang et al., 2025) reaches 60.0% accuracy on MATH500 at 43.5% token savings, compared to 81.4% at 30.5% savings for our method. The 21-point gap aligns with the brittleness of single-cue stopping: when the chosen cue (e.g., "Wait") is absent from a trace, the method either fails to fire or stops at an unhelpful moment, while a composite score can still trigger on residual semantic or lexical reuse.

Takeaway. The empirical picture supports a simple claim: online, behavior-aware stopping with multiple complementary signals beats single-cue or top-down approaches on a benchmark where problem difficulty varies. The gap is largest where the cue or budget is most likely to be miscalibrated. Combining behavior-aware stopping with a difficulty-aware prior is an obvious next step that we leave to future work.

7 Conclusion

We presented a training-free framework for mitigating overthinking in LLMs that operationalizes overthinking as low-information reasoning along three measurable axes (content reuse, lexical reuse, hedging), detects it online via a continuously updated composite score, and intervenes via lightweight hard triggers and a final-answer prompt. Across five benchmarks and four model families, the framework reduces generated tokens by 27–89% relative to vanilla CoT. Accuracy improves on smaller and non-reasoning-tuned models (e.g., +5.2 points on MATH500 with Qwen-1.5B; +4.1 points on HotpotQA with the same model). On reasoning-specialized 7–8B models, accuracy regresses by 1–4 points on HumanEval, GPQA, HotpotQA, and (smallest) MATH500, where precise multi-step synthesis is more sensitive to early stopping. CommonsenseQA accuracy is essentially preserved across both model classes.

Compared to budget-based (ThoughtTerminator) and single-cue (DEER) baselines under matched protocols, the composite-score approach recovers more of the accuracy ceiling at comparable token savings. The biggest single contributor to efficiency is the semantic-similarity signal; the hedging signal contributes least but stabilizes accuracy. Combining behavior-aware stopping with difficulty-aware budgeting is a natural next step.

8 Limitations

The most pronounced limitation of our framework is task-dependent regression. On HumanEval (67.7% → 64.0%), GPQA with DeepSeek-

R1-Distill-Llama-8B (42.9% → 40.8%), and HotpotQA with DeepSeek-R1-Distill-Llama-8B (71.9% → 68.6%), the method loses two to four accuracy points relative to vanilla CoT. Tasks that demand precise multi-step synthesis, such as code generation and scientific reasoning, are more sensitive to truncation than the current signal vocabulary anticipates. We did not implement task-aware aggressiveness scheduling that would, for example, raise τ or lengthen L_{act} on code prompts; this is a natural extension and would likely recover some of the gap on HumanEval and GPQA.

A second limitation is the heuristic, English-only hedge lexicon. The lexicon was curated by inspection of DeepSeek- and LLaMA-style outputs, and other languages, model families, or domains may produce different stalling vocabularies that fall outside it. The ablation in Section 5.2 shows hedging contributes the least of the three signals to efficiency, so this dependency is small, but generalization beyond English remains untested. Replacing the lexicon with a learned hedging detector, or extending it through multilingual annotation, would relax this constraint.

A third limitation is that our evaluation is restricted to single-answer protocols. We extract one answer per problem and do not use self-consistency voting, k-best sampling, or self-verification. Methods designed for $k > 1$ samples may interact non-trivially with early stopping: aggressive truncation could reduce sample diversity, while voting could mask accuracy losses. Characterizing this interaction would be especially relevant for deployment scenarios that already use majority voting, and is a direction we leave to future work.

These limitations identify where additional work would strengthen the framework rather than constraints on its core idea. The method continues to operate fully at inference time, requires no fine-tuning or oracle, and transfers across four model families without re-tuning, with the trade-offs above clearly bounded.

9 Ethics Statement

This work focuses on improving the computational efficiency of large language model inference by reducing unnecessary reasoning steps. Our framework does not alter model weights, training data, or the content of model outputs beyond truncating redundant reasoning. We do not foresee direct negative societal impacts from this work. However, we note that aggressive early stopping could in principle suppress useful deliberation on sensitive or nuanced queries; our design mitigates this through conservative thresholds and an activation delay that allows initial reasoning to proceed. All experiments use publicly available benchmarks and open-weight models.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Tingxu Han and 1 others. 2025. Token-budget-aware llm reasoning. In *Findings of ACL*.

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. *NeurIPS*.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. 2025. Early stopping chain-of-thoughts in large language models. *arXiv preprint arXiv:2509.14004*.

Meta AI. 2024. The llama 3 model family. *arXiv preprint arXiv:2407.21783*.

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. *Preprint*, arXiv:2501.19393.

Nostalgebraist. 2020. Interpreting gpt: the logit lens. Online forum post on LessWrong. https://www.lesswrong.com/posts/jqXMYk9xEKEu6LGS6/interpreting-gpt-the-logit-lens.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. 2025. Thoughtterminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models. *Preprint*, arXiv:2504.13367.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof qa benchmark. *Preprint*, arXiv:2311.12022.

Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. 2025. Stop when enough: Adaptive early-stopping for chain-of-thought reasoning. *Preprint*, arXiv:2510.10103.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Xi Wang, James McInerney, Lequn Wang, and Nathan Kallus. 2025. Entropy after ⟨/Think⟩ for reasoning model early exiting. *arXiv preprint arXiv:2509.26522*.

Andrew Wen and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Preprint*, arXiv:1809.09600.

A Computation and Runtime

Experiments were conducted on 4 NVIDIA A6000 Ada GPUs, with the 500 MATH500 problems split into four batches processed in parallel. The vanilla CoT baseline required approximately 4 hours per batch, while our dynamic stopping framework required approximately 4 hours and 40 minutes. This modest overhead reflects controller-side computations (online embedding similarity, repetition statistics, and logit probe evaluation) in an unoptimized research prototype. We expect token count as the primary hardware-agnostic measure of efficiency, and that the runtime overhead to diminish significantly with standard optimizations such as embedding caching and batched similarity computation.

The reported wall-clock times were obtained using a research prototype that prioritizes clarity over execution speed, without optimizations such as embedding caching, batched similarity computation, or kernel fusion. Absolute runtime can vary substantially with GPU utilization, batching strategy, and system load. We expect the overhead to diminish significantly in an optimized deployment.

B Other related works

A related work, **s1** (Muennighoff et al., 2025) requires supervised fine-tuning to learn the special tokens and syntax to control the budgets. Then, the special tokens can control the reasoning length. This delivers control at the cost of retraining and reduced portability across models. Our framework is complementary: it could be applied on top of an s1-trained model to recover additional efficiency without further fine-tuning.

Another related work, REFRAIN (Sun et al., 2025), reports results on a different mix of model families and has no public implementation at the time of writing. For context only, our method generates approximately 950K tokens in total across MATH500 versus the ∼1.6M reported by REFRAIN on comparable benchmarks; we caution that the model families differ and this comparison should not be read as a head-to-head result.

C Overthinking Hedging Lexicon

As part of the dynamic overthinking score (Section 3), we include a lightweight *hedging signal* that captures tentative, self-corrective, or stalling language in the model's generated reasoning. This signal is computed using a small, manually curated lexicon of discourse markers and phrases commonly associated with hesitation, revision, or redundant explanation.

Table 4 lists the hedging words and phrases used in our experiments. The lexicon is not intended to be exhaustive or linguistically precise; rather, it serves as a coarse heuristic that complements repetition- and similarity-based signals. In practice, this component receives a relatively small weight in the overall overthinking score and primarily acts as a secondary indicator of stalled or uncertain reasoning.

D Additional Figures

This section presents supplementary visualizations that provide additional insight into the behavior of our dynamic stopping framework.

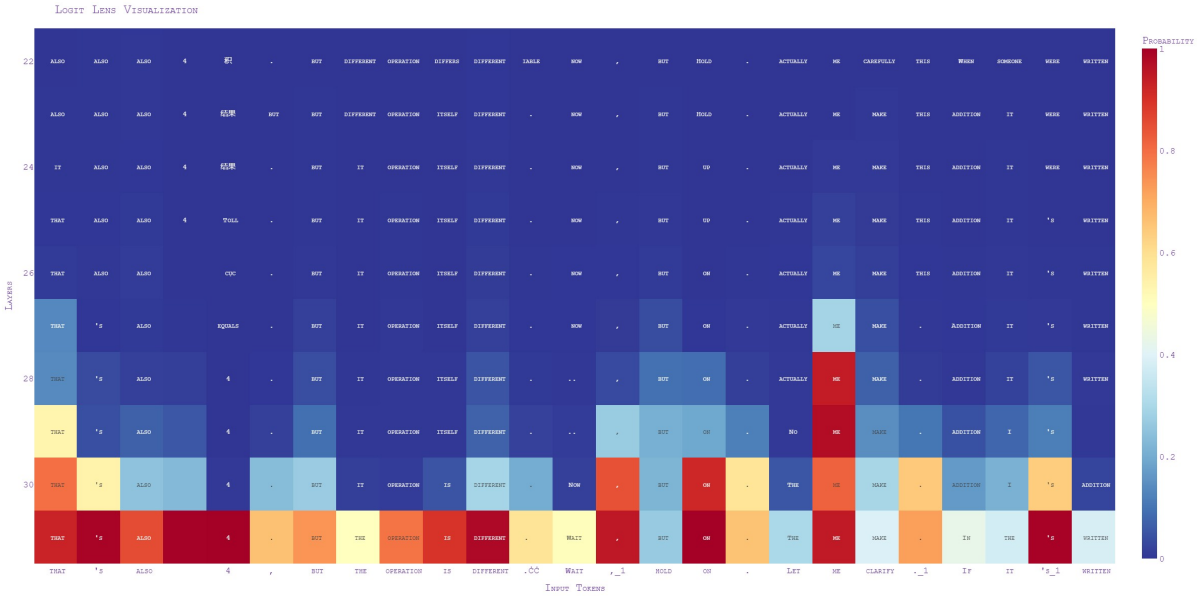

Figure 4: Logit lens heatmap during chain-of-thought generation. Each column corresponds to an output token, and each row corresponds to a transformer layer. Color intensity reflects the layer-wise logit lens probe score (see colorbar). Localized regions of disagreement across layers illustrate mismatch patterns used by our stopping signal.

Hedging and Overthinking Markers
Classic hedges: but, however, though, although, yet, still, maybe, perhaps, possibly, probably, apparently, seems, appear *Hesitation markers:* wait, hold on, actually, in fact, well, hmm, uh, erm, alternatively *Self-doubt and correction:* I think, I guess, I suppose, I believe, not sure, unsure, uncertain, let me, let's see, on second thought, reconsider, recalculate, retry, again, sorry, my mistake, correction, instead *Discourse fillers:* so, anyway, okay, right, well then, alright, as I was saying *Verbose self-explanations:* what I mean, in other words, to clarify, let me explain, let me try again

Table 4: Lexicon of hedging and overthinking-related words and phrases used to compute the Hedge component of the dynamic overthinking score.

E Intervention Mechanism Examples

Our dynamic stopping framework employs several complementary intervention mechanisms, each targeting a distinct manifestation of overthinking. Below, we illustrate three representative mechanisms with examples drawn from actual model outputs during evaluation. These do not constitute an exhaustive list; additional triggers (final-answer keyword detection, end-of-sentence token matching, and logit-lens mismatch stopping) are also active during inference and are described in Section 3.

Overthinking score trigger (t_{score}). Figure 6 shows an example where the model has already arrived at the correct answer ($14/3$) but continues to re-derive intermediate function values it has previously computed. The t_{score} mechanism detects this semantic redundancy: at the marked point, the cumulative score exceeds the threshold τ (score $= 1.063 > 0.925$), combining high semantic similarity with prior sentences, elevated repetition ratio, and hedging language ("Wait, but just to make sure"). Generation is halted and the model is prompted to emit its final answer directly. Without this trigger, the model would have continued re-checking each function evaluation indefinitely.

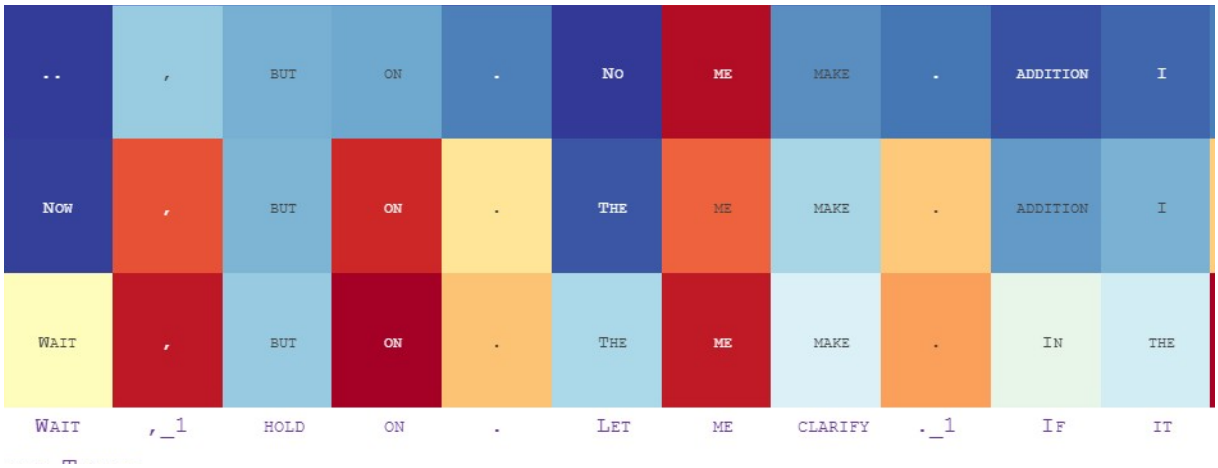

Figure 5: Zoomed-in logit lens heatmap highlighting token-level mismatch. This figure shows a focused subset of Figure 4, centered on a short span of tokens where the final output token does not appear among the top predictions of intermediate transformer layers, illustrating localized internal disagreement monitored by our mismatch-based stopping signal.

(a) Overthinking score trigger (t_{score})

```
Convert all to thirds:  2 is 6/3, 1
is 3/3, and 5/3 is already thirds.
So, 6/3 + 5/3 + 3/3 = (6 + 5 + 3)/3 =
14/3.
Yes, that's correct.  So, the final
answer is 14/3.
Wait, but just to make sure, let me
compute each function value again.
f(-2):  (3*(-2) - 2)/(-2 - 2) = (-6 -
2)/(-4) = (-8)/(-4) = 2.  Correct.
f(-1):  (3*(-1) - 2)/(-1 - 2) = (-3 -
2)/(-3) = (-5)/(-3) = 5/3.  Correct.
Early stop:  S=75, hits=3, L=2, ow=0,
score=1.063 Early stopping signal
received.  Halting generation.
Final Answer:  \boxed{14/3} </think>
```

Figure 6: Overthinking score trigger: the model re-derives previously computed values. The t_{score} exceeds the threshold and generation is halted.

End-of-think token match. Figure 7 illustrates a case where the model produces a `</think>` token, signaling the end of its internal reasoning phase. Our framework monitors for this structured delimiter and terminates generation upon detection. In this example, the model has already expressed confidence in the polar coordinate conversion and explicitly stated its final answer within the reasoning trace. The `</think>` token confirms that the model considers its reasoning complete, and the framework extracts the final boxed answer without permitting further generation that could introduce second-guessing or drift.

(b) End-of-think token match

```
I think I'm confident now that the
polar coordinates are (3, π/2).  Let
me just write that in the required
format.
**Final Answer** The polar
coordinates are \boxed{(3,
\frac{\pi}{2})}.  </think>
To convert the rectangular
coordinates \((0, 3)\) 'End of Think'
detected -- early stopping.  Final
Answer:  \boxed{3, \frac{\pi}{2}}
</think>
```

Figure 7: End-of-think token match: the model emits a `</think>` delimiter, and the framework halts generation and extracts the final answer.

Repetition loop detection. Figure 8 shows a pathological case where the model enters a repetition loop, producing near-identical "Correct Answer: Yes." lines with minor formatting variations. Our framework maintains a count of semantically equivalent sentences throughout the reasoning trace. When this count exceeds a predefined threshold, the loop is flagged as a strong overthinking signal and generation is terminated immediately, regardless of the current t_{score}. This hard trigger acts as a safety net for cases where the model becomes trapped in a degenerate cycle that the continuous score may detect only gradually.

(c) Repetition loop detection

```
Correct Answer:  Yes.  Correct
Answer :  Yes.   Correct Answer:
Yes.   Correct Answer:  Yes.
Correct Answer:  Yes.   Correct
```

```
Answer: Yes. CorrectAnswer: Yes.
CorrectAnswer: Yes. Repetition
loop detected CorrectAnswer: Yes.
CorrectAnswer: Yes Early stopping
signal received. Halting generation.
Yes.
```

Figure 8: Repetition loop detection: the model produces semantically identical outputs in a loop. The framework detects the cycle and halts generation.

Additional intervention mechanisms. Beyond the three mechanisms illustrated above, the framework includes several other active triggers during inference: (i) *final-answer keyword stop*, which monitors for phrases such as "final answer" or "correct answer" and terminates after a predefined number of occurrences (Section 3); and (ii) *logit-lens mismatch stop*, which monitors cumulative inconsistencies between intermediate-layer predictions and final-layer outputs, triggering termination when the mismatch count exceeds a threshold (35 in our experiments). These mechanisms operate in parallel and are designed to be complementary: the continuous t_{score} captures gradual onset of redundancy, while the hard triggers intercept clear structural signals of completion or degenerate behavior. Together, they provide layered coverage across diverse overthinking patterns.