ASTRA: MITIGATING OVERTHINKING VIA TRACKING ATTENTION SINK

Anonymous authors
Paper under double-blind review

ABSTRACT

The overthinking problem has recently attracted significant attention from researchers. While the introduction of chain of thought substantially enhances the model's ability to solve complex tasks, the overthinking problem emerges when solving relatively simple questions. In these cases, overthinking can be observed as repeated verifications and divergent thoughts, which do not translate into improved problem-solving, yet generate considerable computational overhead. In this paper, we propose ASTra (Attention Sink Tracker), an inference-time method to mitigate overthinking problems in LLM. By monitoring attention weight distribution, we discovered an optimal truncation point to prompt models to terminate the chain of thought. Extensive experiments show that our method reduces 15% to 70% of chain of thought tokens, while maintaining final output quality [1].

1 INTRODUCTION

Large language models (LLMs) have recently demonstrated remarkable ability to solve complex problems, where chain-of-thought is an important factor. Chain of thought (CoT; Wei et al., 2023) is a highly effective prompting technique, specifically designed to enhance the reasoning capabilities of large language models. It encourages the model to generate a series of intermediate reasoning steps before arriving at a final answer, closely mirroring how a human would logically break down a complex problem. By integrating chain of thought with reinforcement learning (RL) (Comanici et al., 2025; Guo et al., 2025), models, such as DeepSeek (Guo et al., 2025), attain strong long-context reasoning ability, especially on tasks such as mathematical reasoning, long-context understanding, multistep problems and coding.

The overthinking problem is a by-product of the introduction of chain-of-thought reasoning (Shi et al., 2023). Although long CoT reasoning sequences improve performance, they also introduce significant computational overhead due to redundant and verbose outputs (Sui et al., 2025). This phenomenon is particularly evident in relatively simple cases, where models without CoT can solve problems using only 10% of the tokens that CoT-enabled models require (Chen et al., 2025).

In this paper, we introduce ASTra, attention sink tracker, a new overthinking mitigation method based on attention weights during inference. By identifying the "attention sink" (Xiao et al., 2024) — a phenomenon where specific tokens disproportionately absorb attention weights to maintain softmax stability or convey semantic meaning — we separately calculate their cumulative value during the prefill and decode stages. Because the total attention probability mass is bounded, we treat the unconsumed weight as the **remaining attention resources for current reasoning**. By monitoring this metric, we can **dynamically determine the proper truncation points across different settings, therefore mitigate overthinking phenomena**. We tested our method on Qwen3 family (Yang et al., 2025a), Deepseek-R1 (Guo et al., 2025) and GLM4 (GLM et al., 2024), with four datasets while maintaining output quality compared with baseline. Our contribution can be summarized as follows:

- Analyze the overthinking problem from the perspective of attention sink;
- Build a theoretical model for an inference-time overthinking mitigation strategy;
- Perform extensive experiments. Results show our method reduces output tokens by 15% to 70%, while maintaining near-lossless (< 2% difference in accuracy or F1 score) output

[1] The full codebase will be released publicly upon acceptance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

quality in most cases. In addition, our method achieves a greater token reduction for test examples with longer baseline outputs.

2 BACKGROUND

2.1 LLM OVERTHINKING

With the test-time scaling approach "longer is better" (Jin et al., 2024; Snell et al., 2024), the model accuracy is boosted on complex tasks, while some studies reveal a critical inefficiency issue — overthinking (Chen et al., 2025; Sui et al., 2025). This phenomenon can be defined as the generation of additional thought tokens that do not contribute to performance gains (Chen et al., 2025; Fang et al., 2025; Yang et al., 2025a; Zhang et al., 2025b), which is more evident when solving simple problems with evident answers. This phenomenon not only wastes computational resources (Aggarwal & Welleck, 2025; Arora & Zanette, 2025) , but also triggers performance degradation (Liu et al., 2025) in some cases, where a model may be misled by its own excessive thinking, abandoning the correct answer which has been proposed during reasoning.

In this paper, we define two typical patterns of overthinking: excessive exploration and over-verification, based on the definitions of other scholars (Chen et al., 2025; Hassid et al., 2026; Pu et al., 2025; Zhang et al., 2025b).

- **Excessive exploring**: Model tends to solve simple question in different ways, which is unnecessary. For instance, model may try to explore a new path to solve the problem halfway through reaching the final answer, or repeatedly check the meaning of part of the query for a new interpretation after generating the answer.
- **Over-verification**: Model tends to recursively verify its answer several times, regardless of whether it is correct or incorrect. For instance, model may repeatedly recalculating mathematical operations or redundantly validating general problem-solving strategies.

Methods proposed to solve overthinking can be divided into three types, based on the stage and method employed to mitigate overthinking. **Training-stage intervention**: the overthinking problem is mainly affected by reinforcement learning strategies. Therefore, adjustments to RL training with length-based penalties (Aggarwal & Welleck, 2025; Arora & Zanette, 2025; Wang et al., 2026) or compressed CoT (Deng et al., 2024) can be a solution to this problem. However, this requires an additional training process. **Inference-stage offline criteria**: these methods set fixed stopping criteria before inference. Common strategies include reasonable length approximation (Pu et al., 2025; Xu et al., 2025) and prompt engineering (Han et al., 2025). These methods introduce less overhead. However, their generalization abilities are usually limited across diverse settings, for instance, mathematical reasoning tasks with multiple intermediate steps. In addition, fixed truncation criteria lacks stability and may result in “over-truncation”, decreasing output quality. **Online inference monitoring**: these methods monitor the inference process and terminate reasoning when detecting overthinking patterns (Huang et al., 2025; Bin et al., 2026; Mao et al., 2025). Indicators employed include reasoning transition points (Yang et al., 2025b), completion points (Wei et al., 2025), output entropy (Guan et al., 2026). These methods introduce runtime overhead, yet have better generalization ability and stability. However, current methods(Bin et al., 2026; Guan et al., 2026; Huang et al., 2025; Mao et al., 2025; Yang et al., 2025b) employed during inference time usually focus on token-level indicators, including specific tokens (Yang et al., 2025b), output perplexity, output entropy (Guan et al., 2026), while **no existing methods analyze this phenomenon from the angle of attention**. These methods operate at a higher level of abstraction compared with attention mechanism, and are easily misled by noises during sampling (Bin et al., 2026), while attention features offers us basic-level signals.

2.2 ATTENTION SINK PATTERNS

During inference time, the distribution of attention weights is not totally random, instead it demonstrates certain patterns in two-dimensional space (Jiang et al., 2024). Common patterns include A-shape pattern (Xiao et al., 2024; Han et al., 2024), block-sparse pattern (Jiang et al., 2024), and vertical sink pattern (Kovaleva et al., 2019; Bondarenko et al., 2023). Attention sink pattern emerges when later tokens pay excessive amount of attention to certain tokens. This phenomenon can be interpreted in two ways: the model “dumps” residual attention weights onto meaningless tokens due

to softmax normalization (Xiao et al., 2024; Gu et al., 2025), or the token serves as a fundamental information hub worth attending to (Xiao et al., 2024; Wu et al., 2024). Research has found that a considerable amount of attention has been directed to the first token of the sequence, even though it has no significant semantic meaning, which is a common feature of a variety of models, including decoder-only LLM (Xiao et al., 2024), encoder-only models, (Kovaleva et al., 2019) and vision-language models (Darcet et al., 2024). We also observed a similar phenomenon in Table 1, featuring a wide range of current LLMs.

Models	GSM8K	Triviaqa	MMLU	MATH500
QWEN3-4B	72.72%	70.74%	73.26%	73.46%
QWEN3-8B	66.61%	67.06%	69.07%	66.68%
QWEN3-14B	62.92%	65.49%	65.47%	66.79%
QWEN3-32B	70.31%	66.90%	69.59%	69.17%
GLM4-9B	66.23%	64.51%	65.22%	65.58%
DEEPSEEK-7B	48.39%	49.31%	46.98%	51.02%

Table 1: Average attention sink proportion on meaningless tokens at the beginning of the sequence of different models. Note that DeepSeek-7B refers to DeepSeek-Distill-Qwen2.5-7B. Results show that attention weight sink **is a common pattern** in various models.

During inference, attention sink anchors the model back to the initial token to ensure coherency (Liu et al., 2026; Zhang et al., 2026; 2025a; Li et al., 2026), yet also limit the scope of future reasoning (Yan et al., 2025; Yoo et al., 2026). Some studies also claim that they indicate "reasoning saturation" (Li et al., 2026), where future reasoning potential becomes constrained. We hypothesize that redundant reasoning paths during overthinking are directly linked to attention sink distributions. Motivated by these observations, we propose attention sink accumulation as an internal signal for a lightweight, online stopping heuristic to mitigate overthinking.

3 MOTIVATION

3.1 WHY WE CAN IDENTIFY ATTENTION SINK VIA STATISTICAL METHOD

As defined by Xiao et al. (2024), attention sinks refer to specific tokens that attract a disproportionately large share of attention scores, stabilizing the attention mechanism. Numerically, attention-sink (Xiao et al., 2024) tokens have some prominent features that distinguish themselves from other tokens:

Significantly high mean value, compared with other tokens. During inference, the mean attention weight of attention sink tokens is usually 10 times higher than that of other tokens.

Relatively stable attention value throughout the CoT range. Normally, the attention weight received by other tokens will decrease as the output sequence grows. However, for attention sink tokens, their gained attention weights remain relatively stable in chain of thought range (See Figure 1).

These features lay foundation for our proposed method. Since the attention sink possesses extremely high value, a statistic method will be able to distinguish it from normal attention distribution; since the attention sink is stable, an online method can be proposed, with only nearby tokens as judges rather than the whole CoT.

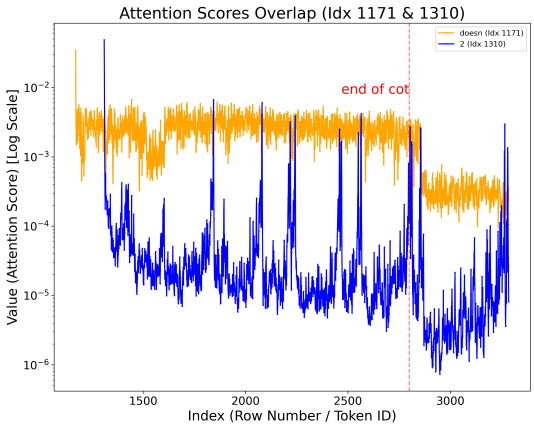

Figure 1: Attention weight comparison between attention sink token and non-attention sink token. (Qwen3-8B, Layer35). The sequence begins from a token's self attention. Note that token 1171 and token 1310 have similar mean value. However, the attention weight distribution of token 1171 is relatively stable in the range of chain of thought, while non-attention sink tokens usually depicts violent change in attention weight as the sequence grows.

3.2 A Bidirectional Relationship Between Attention Sink and Overthinking

We approach this relationship from two angles, an empirical observation and a theoretical argument, and then combine them into a single bidirectional claim:

Empirically, in chain-of-thought, attention sinks (especially those with extreme high values) often appear on tokens with a strong indication of the final answer, or serving as a fundamental intermediate result. When reasoning on mathematical and logical tasks, models tend to pay significant attention to specific numbers or crucial logical transition words (Figure 2). This phenomenon indicates either that the model has generated its final answer, or that it has generated crucial intermediate results which may contain enough information to directly produce the final answer, rather than requiring redundant chain-of-thought reasoning.

Theoretically, the emergence of attention sinks geometrically constrains the information carried by attention outputs. Let $h_t = \sum_{i=1}^{t} \alpha_{t,i} v_i$ be the output of a self-attention layer at step t, and let S be the sink set with total weight $\mu_t := \sum_{s \in S} \alpha_{t,s}$ and centroid $v_S := \mu_t^{-1} \sum_{s \in S} \alpha_{t,s} v_s$. Standard normalization bounds value-vector distances by an architecture-dependent constant M (Lemma 1, Appendix B.2. When sinks dominate, i.e., $\mu_t \geq 1 - \epsilon$,

$$\|h_t - v_S\|_2 \;\leq\; \epsilon M \text{(Theorem 1, Appendix B.3)} \tag{1}$$

Eq. (1) is a per-layer bound on representations, and three observations connect it to overthinking. *First*, when sinks emerge in the last few attention layers — those closest to the model's output representation — the pinning effect directly biases generation toward repeating the content encoded in v_S, naturally inducing over-verification; this is also the regime tracked by Algorithm 1, which aggregates the last M layers. *Second*, Eq. (1) substantially constrains the diversity of representations driving generation, lowering the marginal reasoning value of further decoding. *Third*, by the stability property described in Section 3.1, sinks persist across the CoT range and continue to influence subsequent tokens, which is also reflected in the attention-weight stability of sink tokens in Figure 1.

Taken together, the two angles describe a bidirectional relationship. The observation and the bound above are two halves of one loop. **(D1) Sink saturation suppresses reasoning progress**: once $\mu_t \geq 1 - \epsilon$, Eq. (1) pins h_t to v_S, so the representation driving generation stops changing and the marginal value of further decoding vanishes. **(D2) Redundant reasoning accumulates sink mass**: by the stability property of Section 3.1, sink weights persist over the whole CoT range, so every additional redundant token adds to, and rarely releases, μ_t. The two directions reinforce each other rather than acting one way: the longer the model over-verifies, the less attention capacity remains for genuine reasoning progress, which in turn makes further tokens even more redundant. We do *not* claim that sink saturation is necessary for overthinking; we claim that it is sufficient to eliminate marginal reasoning gain, and that μ_t is therefore a reliable online indicator of the overthinking phase.

To test our claim, we conduct an attention-deprivation experiment (Figure 3). After several hundred CoT tokens, we force the model to allocate at least a fraction k of attention to the identified sinks. On both Qwen3-8B and Qwen3-14B, modest increases in k yield lower accuracy and longer outputs — the pattern predicted by Eq. (1) through reduced representational diversity, indicating that elevated sink mass directly degrades reasoning quality and lengthens the chain-of-thought. The same loop is visible without any intervention: during unperturbed decoding, the expected marginal gain $\mathbb{E}[\Delta R_t]$, measured against the remaining attention capacity $1 - \mu_t$, collapses to a near-zero plateau exactly when sinks saturate, while the model keeps emitting tokens (Figure 4); we analyse this measurement in detail in Section 3.3.

3.3 An Optimal-Stopping View of the Truncation Threshold

The feedback loop of Section 3.2 implies that decoding does not degrade smoothly, but splits into two regimes. While attention capacity remains, every step still injects new information and the expected gain grows with $1 - \mu_t$; we call this the **productive phase**. Once the loop drives μ_t toward saturation, Eq. (1) freezes the representation and additional tokens buy almost nothing; we call this the **overthinking phase**. Mitigating overthinking therefore reduces to detecting the transition between the two phases and stopping there, which we formalize as an optimal-stopping problem:

$$\max_T J = \mathbb{E}\big[R(S_T) - \lambda \cdot T\big], \tag{2}$$

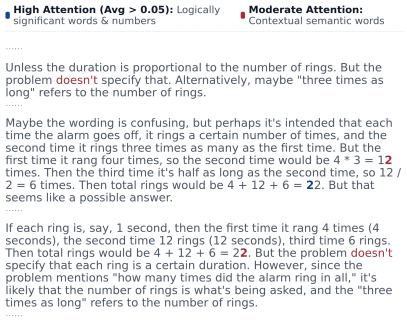

Figure 2: Example of attention sink tokens and overthinking paragraph, GSM8K Q86, Qwen3-8B. Note that blue represents extremely high mean value (average attention>0.05). The model pays high attention on logically significant words ("**doesn't**") and exact result number ("**2**").

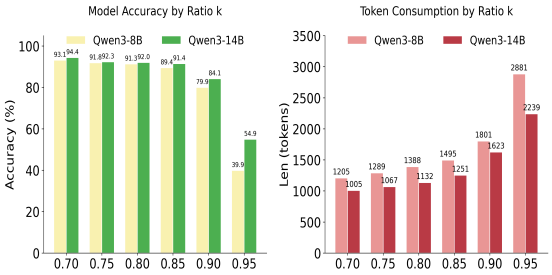

Figure 3: Result of attention-deprivation experiment on Qwen3-8B and Qwen3-14B. All layers and heads included. We disabled FlashAttention-2 and used standard attention. At each decoding step $t > t_0$ (we set $t_0 = 500$) , we apply a proportional renormalization. Suppose $\mu_t = \sum \alpha_{t,sink}$, then $\alpha_{t,sink} = \alpha_{sink} * \frac{k}{\mu_t}, \alpha_{t,normal} = \alpha_{t,normal} * \frac{1-k}{1-\mu_t}$. The test was carried out on GSM8K. Ratio k starts from 0.70, given the fact that attention sink to first token in both models are around 0.65 (See Table 1).

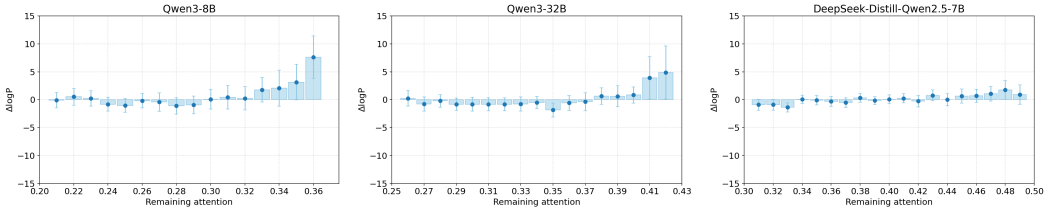

Figure 4: Marginal confidence gain versus the remaining attention capacity on three models, stride=20 tokens. The x-axis indicates the remaining attention capacity $1 - \mu_t$, i.e., the attention mass not absorbed by sinks. The y-axis represents the expected marginal gain $\mathbb{E}[\Delta R_t]$. At each point, we calculate the log-probability of the correct answer $R_t = \log P(\text{answer} \mid S_t)$ and the difference $\Delta R_t = R_t - R_{t-\text{stride}}$. Positive correlations before overthinking phase (Qwen3-8B: $r = 0.8294$ for $x > 0.30$, $r = 0.6629$ for all; Qwen3-32B: $r = 0.7936$ for $x > 0.37$, $r = 0.6415$ for all; DeepSeek: 0.5587 for $x > 0.44$, $r = 0.8576$ for all) are consistent with the first-order approximation $\mathbb{E}[\Delta R_t] \approx \kappa(1 - \mu_t)$. The graph also indicates the existence of overthinking phases, where $\Delta logP$ oscillates near zero.

where $R(S_T)$ is the terminal reasoning reward and $\lambda > 0$ penalizes each additional generated token. The relevant quantity for stopping is the expected marginal improvement $\mathbb{E}[\Delta R_t]$, which we measure as $\Delta R_t = R_t - R_{t-\text{stride}}$ with $R_t = \log P(\text{answer} \mid S_t)$.Here the stride is the sampling stride determining the calculation frequency of R_t.

Figure 4 confirms this two-regime picture quantitatively. In the **productive phases**, $\mathbb{E}[\Delta R_t]$ grows approximately **linearly** with the remaining attention capacity $1 - \mu_t$ (Pearson $r = 0.83$), consistent with both Theorem 2 (Appendix B.6) and the first-order approximation $\mathbb{E}[\Delta R_t] \approx \kappa(1 - \mu_t)$, where κ is a task- and model-dependent coefficient. In the **overthinking phases**, $\mathbb{E}[\Delta R_t]$ collapses to a **near-zero plateau**, indicating that further decoding yields negligible reasoning gain—i.e., the model has entered overthinking regime. Also, combining this with the upper bound from Theorem 2 (Appendix B.6) stopping condition $\mathbb{E}[\Delta R_t] < \lambda$ gives $\mu_t > 1 - \lambda/\kappa_{max}$. In practice, since the worst-case κ_{max} is loose, we fold the latent ratio into a single tunable threshold,

$$\tau := 1 - \frac{\lambda}{\kappa}, \tag{3}$$

which yields the truncation rule used by our algorithm: **stop generation once** $\mu_t > \tau$. The framework converts the choice of τ from a free parameter into a calibration of the cost-utility ratio λ/κ, which

can be estimated with method mentioned in Appendix E and allows a single threshold to be transferred across tasks within the same model family.

4 METHOD

We present **ASTra**, an online attention sink tracking and accumulation method to mitigate the overthinking problem. As the attention sink phenomenon intensifies, the model's reasoning ability degrades as remaining attention resources diminish. ASTra records current attention sink values to estimate remaining attention resources, cutting off the reasoning as the remaining resources fall below a threshold. Concretely, our method can be divided into three parts:

- **Attention sink identification (4.1)**: Identifies sink patterns by computing the mean attention value and decay ratio.
- **Sink value monitoring (4.2)**: Dynamically accumulates current sink values while reasoning.
- **Truncation triggering (4.3)**: Decides whether to stop reasoning via inserting a prompt.

Detailed analysis regarding the time and memory overhead of our method can be seen in Appendix C, proving that the additional cost of our method is acceptable.

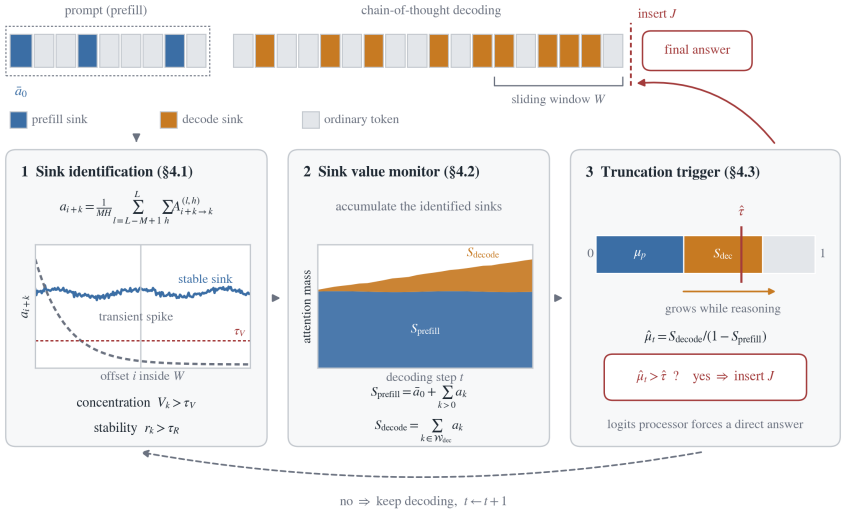

Figure 5: An overview of ASTra pipeline.

4.1 ATTENTION SINK IDENTIFICATION

Section 3.1 identifies two numerical features that distinguish sink tokens: a significantly elevated mean attention value, and stability of that attention across the CoT range. We turn these into an inference-time identification rule based on two indicators computed over a sliding window W of subsequent tokens.

To capture sink behavior across multiple model layers rather than a single layer, we aggregate the attention received by token k from token $i+k$ over the final M attention layers and all H heads: $a_{i+k} = \frac{1}{MH} \sum_{l=L-M+1}^{L} \sum_{h=1}^{H} A_{i+k \to k}^{(l,h)}$.

Attention Concentration. We measure the average attention mass that token k absorbs from the following W tokens: $V_k = \frac{1}{W} \sum_{i=1}^{W} a_{i+k}$. Sink tokens absorb a disproportionate share of attention, so V_k exceeding a threshold τ_V provides the first identification signal.

Decay Ratio. V_k alone does not separate true sinks from transient attention spikes, since a non-sink token may briefly absorb high attention before its influence decays. To capture stability, we compare

the second half of the window to the first half: $r_k = \frac{\sum_{i=W/2}^{W} a_{i+k}}{\sum_{i=1}^{W/2} a_{i+k}}$. Stable sinks maintain r_k near unity throughout the CoT range, while transient tokens exhibit rapid decay ($r_k \to 0$).

Identification rule. A token k is identified as a sink if and only if $V_k > \tau_V$ and $r_k > \tau_R$. The full online procedure is given in Algorithm 1 (Appendix A).

4.2 SINK VALUE MONITOR

In this part, we accumulate sink attention values. Given identified attention sink tokens $W_{prefill}$, W_{decode}:

$$S_{prefill} = \bar{a}_0 + \sum_{k>0} a_k, a_k \in W_{prefill}; \quad S_{decode} = \sum a_k, a_k \in W_{decode} \tag{4}$$

Here, $\bar{a}_0$ denotes the cumulative moving average of the attention weight allocated to the first-token sink of the whole sequence (See Table 1). up to the current decoding step t. Despite the stability of attention sinks, a minor change in the attention of the first-token sink can be numerically enormous due to its extremely high value. Therefore, our formulation is designed to reduce the oscillation caused by this change and better estimate remaining attention resources. The complete dynamic monitoring procedure is detailed in Algorithm 2 in Appendix A.

4.3 EMPIRICAL TRUNCATION TRIGGERING

In our conceptual model, the truncation condition is triggered when the attention sink mass exceeds a predefined threshold ($\mu_t > \tau$). However, directly applying the raw absolute attention score as μ_t is suboptimal in practice, because the total attention budget is naturally partitioned between the initial prompt (prefill) and the generated reasoning steps (decode) (See Table 1). Let $\mu_p = S_{prefill}$ denote the baseline attention absorbed by the prefill tokens. The remaining attention capacity available for active reasoning during the decoding phase is strictly bounded by $1 - S_{prefill}$. To accurately capture the extent to which the sink dominates the *current* generation process, we define our empirical estimator of the sink weight, $\hat{\mu}_t$, as the relative proportion of available attention resources ($1-S_{prefill}$) consumed by the decode sink tokens (S_{decode}), i.e., $\hat{\mu}_t = \mu_d/(1-\mu_p) = S_{decode}/(1-S_{prefill})$.

Proposition 1 (Appendix B) translates Theorem 1 into the form of: $||h_t - v_s||_2 \leq (1-\hat{\mu}_t)(1-\mu_p)M$. Combining it with Theorem 2 yields: $[E[\Delta R_t]] \leq \kappa_{max}(1-\mu_p)(1-\hat{\mu}_t)$.

Therefore, the stopping condition $E[\Delta R_t] < \lambda$ becomes:

$$\hat{\mu_t} > \hat{\tau}, \hat{\tau} = 1 - \frac{\lambda}{\kappa(1-\mu_p)} \tag{5}$$

Where τ is a coefficient mentioned in Section 3.3, depending on model size, type and task complexity. By inserting a prompt J inside the CoT via logits processor [2], we prompt the model to conclude the reasoning phase and directly output the answer. The detailed logic for determining truncation is formalized in Algorithm 3 in Appendix A.

5 EXPERIMENTS

5.1 SETUP

To validate the effectiveness of our method, we carry out experiments on six models—including the Qwen3 family (4B, 8B, 14B, and 32B) (Yang et al., 2025a), GLM4-9B (GLM et al., 2024), and Deepseek-R1-Distill (i.e., DeepSeek-R1-Distill-Qwen2.5-7B) (Guo et al., 2025) —across four diverse datasets (GSM8K, MATH500, TriviaQA, and GPQA) (See G.2). This setup covers a variety of models with different parameter scales as well as datasets with varying tasks and difficulties, providing a comprehensive evaluation of our method's generalizability in both short- and long-context reasoning. Hyperparameters and hardware settings are detailed in Table 8.

[2] A `LogitsProcessor` in HuggingFace `transformers` (Wolf et al., 2020) is a callback that intercepts and modifies the next-token logits at each decoding step. We use it to force-feed the tokens of the prompt in J into the generation stream without re-invoking the prefill.

5.2 Results and Discussion

Table 2 presents the main experimental results. We comprehensively evaluate our approach against four settings: (1) **Baseline**: the standard unconstrained CoT prompting; (2) **No-CoT**: direct answering without reasoning steps; (3) **Dynasor** (Fu et al., 2025): a recent open-source batch-level scaling and entropy-based method, selected as our primary advanced baseline; (4) **ROM** (Wang et al., 2026): a method combining streaming detection and fine-tuning. Both Dynasor and ROM are tested based on hyperparameters provided in their paper.

Main Results: Compared to the standard CoT baseline, our method achieves a notable 15% to 70% reduction in tokens while maintaining, or even slightly improving reasoning accuracy across different settings. In particular, our method demonstrates more pronounced token-saving efficiency on larger and more advanced models (e.g., Qwen3-32B). This aligns with our hypothesis that stronger models exhibit greater reasoning capabilities, but are correspondingly subjected to greater overthinking risks. Taking both accuracy and token reduction into account, our approach yields the optimal balance.

Comparison with other signals: To rigorously isolate the effectiveness of our dynamic sink-monitoring mechanism from other early-stopping signals, we conduct an ablation study including entropy signal, which terminates generation when it falls below a threshold, and fixed-length truncation signal, which the model is prompted to stop reasoning and output the final answer as soon as its CoT length reaches the average token length of our method in Table 2. Our dynamically triggered truncation significantly outperforms this rigid-length baseline, demonstrating that our method identifies meaningful boundaries rather than merely taking advantage of shorter generated lengths. The full results are reported in Appendix D.

Long Generation Examples Reduction Analysis: To further evaluate our method's performance on long-generation examples with significant overthinking, we sampled the top 20% of examples where the baseline generates the most tokens. This analysis is important: reductions on long, heavily-overthought generations save substantially more tokens than reductions on short, mildly-overthought ones. Results are presented in table 3, which shows that our method is more efficient in token reduction in significantly overthinking cases, with 60% to 80% token savings in most situations. Note that in some cases, ASTra also achieves a higher accuracy, which is consistent with previous studies showing that overthinking can mislead models (Liu et al., 2025).

5.3 Ablation Study

In this section, we investigate the optimal settings for the window size W and the thresholds τ_V and τ_R on the Qwen3-8B and Qwen3-14B models. We conduct these experiments on a held-out set of 60 validation examples, sampled evenly from three datasets (GSM8K, MATH500, and TriviaQA) and excluded from the main evaluation, each containing more than 10 sink tokens. We identify attention sink tokens using offline algorithms based on features defined in Xiao et al. (2024), where we employ comprehensive indicators including mean value across layers, proportions above several thresholds and variance to ensure accuracy. We then compare them with the predictions made by our algorithm. The results show that the chosen parameters detailed in Table 8 are legit (see Figures 9–11).

Built upon these validated parameters, we further investigate the optimal truncation ratio τ_{truncate} for Qwen3-8B and Qwen3-14B. Similar to the previous setup, we evaluate this on an independent set evenly sampled from the three datasets. The results, depicted in Figure 12, demonstrate that our chosen τ_{truncate} is a robust choice. For expert-level tasks including GPQA, we propose a new method to estimate τ_{truncate} in Appendix E. A full sensitivity analysis of τ_{truncate}, including a threshold sweep on MATH500 and its transfer across model families and across tasks of comparable difficulty, is reported in Appendix F. Prompt difference ablation and failure case analysis can be seen in J and L.

6 Conclusion

In this paper, we study overthinking problem, where we find current token-level mitigation methods lack underlying insights into the essence of overthinking. To this end, we introduced ASTra, an inference-time method to mitigate overthinking by capturing underlying attention distributions. Our method has demonstrated advantage over current open-source methods, with near-lossless performance on several datasets across various models and 15% $\sim$ 70% token decrease.

Table 2: Experimental Results across different models and datasets. All experiments are evaluated using Chain of Thought (CoT) prompting with greedy decoding. For GSM8K, MATH500, and TriviaQA, we limit the maximum tokens to 8192, while for GPQA, we set the limit to 32768. See Appendix H for error analysis.

Model	Method	GSM8K		MATH500		TriviaQA		GPQA	
		Acc	Tokens	Acc	Tokens	F1 Score	Tokens	Acc	Tokens
QWEN3-4B	Baseline	92.75	1085.19	91.90	2151.87	46.09	855.95	50.86	5061.12
	No-CoT	11.45	-	8.03	-	10.94	-	26.09	-
	Dynasor	93.59	1176.29	83.77	1742.33	39.25	629.44	41.58	1781.43
	ROM	88.08	1119.15	79.18	1795.59	38.44	392.58	36.85	2139.83
	ASTra	93.46	851.08	92.44	1139.67	44.35	100.91	51.72	4051.22
QWEN3-8B	Baseline	93.85	1240.07	93.21	2405.86	50.69	756.91	55.84	6693.40
	No-CoT	23.28	-	22.01	-	23.50	-	28.77	-
	Dynasor	94.08	1029.97	90.19	2042.78	37.09	592.23	50.49	4331.06
	ROM	90.94	970.18	78.94	1528.65	39.03	606.89	46.52	3813.26
	ASTra	94.77	539.24	92.76	1319.95	50.23	201.35	54.90	4900.53
QWEN3-14B	Baseline	94.33	1094.11	94.41	2200.76	52.55	569.32	58.82	3635.75
	No-CoT	19.09	-	14.11	-	12.33	-	32.61	-
	Dynasor	95.04	1032.23	90.15	2154.65	35.24	460.01	44.57	3191.09
	ROM	95.37	789.76	81.46	1987.09	53.65	514.89	39.60	3528.50
	ASTra	95.16	963.56	94.42	1863.25	54.11	488.23	58.50	2765.24
QWEN3-32B	Baseline	94.69	1095.06	94.32	2346.03	52.85	1095.23	56.83	3028.21
	No-CoT	41.08	-	17.11	-	38.02	-	35.71	-
	Dynasor	93.77	879.21	91.10	2148.29	41.32	923.16	51.30	2569.74
	ROM	96.04	539.36	86.82	1666.52	43.61	499.23	42.50	1145.62
	ASTra	94.54	469.31	92.91	570.81	52.11	611.83	56.89	1377.98
GLM4-9B	Baseline	94.65	299.05	67.16	993.32	39.33	289.44	31.82	678.93
	No-CoT	15.11	-	10.21	-	16.89	-	26.80	-
	Dynasor	81.28	225.09	59.41	527.12	24.35	160.78	27.22	340.62
	ROM	88.08	259.84	57.63	919.13	33.00	359.04	37.62	472.72
	ASTra	94.25	289.41	65.43	825.77	39.10	247.57	31.90	642.11
DEEPSEEK-R1-DISTILL	Baseline	88.97	321.21	90.62	1736.09	19.67	582.62	50.83	7176.59
	No-CoT	8.76	-	11.49	-	17.55	-	29.61	-
	Dynasor	83.72	303.06	77.34	1718.48	12.67	509.49	41.61	920.08
	ROM	75.50	387.74	76.82	1898.99	13.59	587.80	35.74	4578.53
	ASTra	87.37	312.58	88.45	855.91	19.02	113.39	49.77	2418.48

Table 3: Long-generation example results across different models and datasets.

Model	**Method**	**GSM8K**		**MATH500**		**TriviaQA**		**GPQA**	
		Acc	**Tokens**	**Acc**	**Tokens**	**F1 score**	**Tokens**	**Acc**	**Tokens**
QWEN3-14B	Baseline	77.98	2445.39	50.43	5844.93	28.79	1148.76	42.62	7711.80
	ASTra	**82.11**	**1839.41**	**52.22**	**3332.86**	**32.03**	**1135.87**	**39.34**	**5156.38**
QWEN3-32B	Baseline	79.09	2203.93	70.08	5495.26	18.39	2365.56	32.35	6943.85
	ASTra	**80.99**	**610.86**	**69.84**	**1610.68**	**21.28**	**855.23**	**36.65**	**1497.76**
DEEPSEEK-R1-DISTILL	Baseline	84.85	560.86	44.00	6435.31	11.50	1334.01	31.37	16087.94
	ASTra	**83.81**	**522.67**	**45.00**	**1340.22**	**10.70**	**152.75**	**35.29**	**4277.08**

7 LIMITATIONS

We have proved that the hyperparameter settings in Table 8 are suitable for our experiments, which include an extensive range of models and tasks. However, the optimal parameter settings in other situations may vary due to model capacity and task difficulty, and we cannot guaranty optimal hyperparameters under all conditions. In addition, for models with significantly weaker capabilities compared with task difficulty, our method may contribute little to overall performance. An example is GLM4-9B (See Table 2), which more depicts an insufficient thinking pattern rather than overthinking compared with other models.

In addition, we haven't expanded our test to datasets including long agent tasks and extremely long contexts, which features 128k and more tokens. To ensure near-lossless accuracy, methods superior to truncation may be needed, which we list as future work to be done.

Some emerging LLM structures may not contain attention mechanisms, or specially trained to avoid significant attention sinks, where our method does not apply.

8 AI USE STATEMENT

Generative AI tools are used to create and modify scientific images, present tables according to data generated by experiments, and monitor experiments based on ordered settings. We also use generative AI to increase overall readability and proofread mathematical proofs and modeling.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. 2025. URL https://arxiv.org/abs/2503.04697.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. 2025. URL https://arxiv.org/abs/2502.04463.

Yi Bin, Tianyi Jiang, Yujuan Ding, Kainian Zhu, Fei Ma, Jingkuan Song, Yang Yang, and Heng Tao Shen. Explore briefly, then decide: Mitigating llm overthinking via cumulative entropy regulation. 2026. URL https://arxiv.org/abs/2510.02249.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. 2023. URL https://arxiv.org/abs/2306.12929.

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. Do not think that much for 2+3=? on the overthinking of o1-like llms. 2025. URL https://arxiv.org/abs/2412.21187.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. 2021.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. 2025. URL https://arxiv.org/abs/2507.06261.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. 2023. URL https://arxiv.org/abs/2307.08691.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. 2024. URL https://arxiv.org/abs/2309.16588.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. 2024. URL https://arxiv.org/abs/2405.14838.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think. 2025. URL https://arxiv.org/abs/2505.13379.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling llm reasoning with certaindex. 2025. URL https://arxiv.org/abs/2412.20993.

Team GLM, :, Aohan Zeng, Bin Xu, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. 2024. URL https://arxiv.org/abs/2406.12793.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. 2025. URL https://arxiv.org/abs/2410.10781.

Weixin Guan, Liang Li, Jiapeng Liu, Bing Li, Peng Fu, Chengyang Fang, Xiaoshuai Hao, Can Ma, and Weiping Wang. Mitigating overthinking in large reasoning language models via reasoning path deviation monitoring. 2026. URL https://arxiv.org/abs/2603.14251.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. 2024. URL https://arxiv.org/abs/2308.16137.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. 2025. URL https://arxiv.org/abs/2412.18547.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. Don't overthink it. preferring shorter thinking chains for improved llm reasoning. 2026. URL https://arxiv.org/abs/2505.17813.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. 2020. URL https://arxiv.org/abs/2010.14701.

Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. Mitigating overthinking in large reasoning models via manifold steering. 2025. URL https://arxiv.org/abs/2505.22411.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. 2024. URL https://arxiv.org/abs/2407.02490.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. pp. 1830–1842, August 2024. doi: 10.18653/v1/2024.findings-acl.108. URL https://aclanthology.org/2024.findings-acl.108/.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. 2020. URL https://arxiv.org/abs/2001.08361.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of bert. 2019. URL https://arxiv.org/abs/1908.08593.

Gengyang Li, Wang Cai, Yifeng Gao, and Yunfang Wu. Syncthink: A training-free strategy to align inference termination with reasoning saturation. 2026. URL https://arxiv.org/abs/2601.03649.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. 2025. URL https://arxiv.org/abs/2410.21333.

Xu Liu, Guikun Chen, and Wenguan Wang. Sinktrack: Attention sink based context anchoring for large language models. 2026. URL https://arxiv.org/abs/2604.10027.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. Early stopping chain-of-thoughts in large language models. 2025. URL https://arxiv.org/abs/2509.14004.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. Thoughtterminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models. 2025. URL https://arxiv.org/abs/2504.13367.

David Rein, Betty Li Hou, et al. GPQA: A graduate-level google-proof q&a benchmark. *arXiv:2311.12022*, 2023.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. 2023. URL https://arxiv.org/abs/2302.00093.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. 2024. URL https://arxiv.org/abs/2408.03314.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. 2023. URL https://arxiv.org/abs/2206.04615.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. 2025. URL https://arxiv.org/abs/2503.16419.

Xinyan Wang, Xiaogeng Liu, and Chaowei Xiao. Rom: Real-time overthinking mitigation via streaming detection and intervention. 2026. URL https://arxiv.org/abs/2603.22016.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. 2022. URL https://arxiv.org/abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. 2023. URL https://arxiv.org/abs/2201.11903.

Zihao Wei, Liang Pang, Jiahao Liu, Jingcheng Deng, Shicheng Xu, Zenghao Duan, Jingang Wang, Fei Sun, Xunliang Cai, Huawei Shen, and Xueqi Cheng. Stop spinning wheels: Mitigating llm overthinking via mining patterns for early reasoning exit. 08 2025. doi: 10.48550/arXiv.2508.17627.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. 2020. URL https://arxiv.org/abs/1910.03771.

Yanan Wu, Jie Liu, Xingyuan Bu, Jiaheng Liu, Zhanhui Zhou, Yuanxing Zhang, Chenchen Zhang, Zhiqi Bai, Haibin Chen, Tiezheng Ge, Wanli Ouyang, Wenbo Su, and Bo Zheng. Conceptmath: A bilingual concept-wise benchmark for measuring mathematical reasoning of large language models. 2024. URL https://arxiv.org/abs/2402.14660.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. 2024. URL https://arxiv.org/abs/2309.17453.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. 2025. URL https://arxiv.org/abs/2502.18600.

Shaotian Yan, Chen Shen, Wenxiao Wang, Liang Xie, Junjie Liu, and Jieping Ye. Don't take things out of context: Attention intervention for enhancing chain-of-thought reasoning in large language models. 2025. URL https://arxiv.org/abs/2503.11154.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. 2025a. URL https://arxiv.org/abs/2505.09388.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models. 2025b. URL https://arxiv.org/abs/2504.15895.

Suho Yoo, Youngjoon Jang, and Joon Son Chung. On the nature of attention sink that shapes decoding strategy in mllms. 2026. URL https://arxiv.org/abs/2603.14337.

Stephen Zhang, Mustafa Khan, and Vardan Papyan. Attention sinks: A 'catch, tag, release' mechanism for embeddings. 2025a. URL https://arxiv.org/abs/2502.00919.

Xinliang Frederick Zhang, Anhad Mohananey, Alexandra Chronopoulou, Pinelopi Papalampidi, Somit Gupta, Tsendsuren Munkhdalai, Lu Wang, and Shyam Upadhyay. Do llms really need 10+ thoughts for "find the time 1000 days later"? towards structural understanding of llm overthinking. 2025b. URL https://arxiv.org/abs/2510.07880.

Zihou Zhang, Zheyong Xie, Li Zhong, Haifeng Liu, Yao Hu, and Shaosheng Cao. One token is enough: Improving diffusion language models with a sink token. 2026. URL https://arxiv.org/abs/2601.19657.

A ALGORITHMS

Algorithm 1 Attention Sink Identification

Require: Target token index k; Window size W; Total layers L, total heads H; Raw attention tensor A; Identification thresholds τ_V (Concentration), τ_r (Stability), M (Sampled layer)
Ensure: Boolean flag is_sink

for $i = 1$ **to** W **do**
  $a_{i+k} \leftarrow \frac{1}{MH} \sum_{l=L-M+1}^{L} \sum_{h=1}^{H} A_{i+k \to k}^{(l,h)}$
end for
$V_k \leftarrow \frac{1}{W} \sum_{i=1}^{W} a_{i+k}$ ▷ Check if $\mu_t \to 1$
$S_{\text{first}} \leftarrow \sum_{i=1}^{W/2} a_{i+k}$
$S_{\text{second}} \leftarrow \sum_{i=W/2}^{W} a_{i+k}$
$r_k \leftarrow \frac{S_{\text{second}}}{S_{\text{first}}}$
if $V_k > \tau_V$ **and** $r_k > \tau_r$ **then**
  return True
else
  return False ▷ Transient attention spike, no collapse
end if

Algorithm 2 Sink Value Monitor, Decoding Step t

Require: Attention weights $\mathbf{a}_t$; Sink sets $W_{prefill}, W_{decode}$; History $\{a_0^{(1\ldots t)}\}$
Ensure: $S_{prefill}, S_{decode}$

Smooth the first-token attention to reduce oscillation
$\bar{a}_0 \leftarrow \frac{1}{t} \sum_{i=1}^{t} a_0^{(i)}$
$S_{prefill} \leftarrow \bar{a}_0$
for each $k \in W_{prefill}$ **do**
  if $k \neq 0$ **then**
    $S_{prefill} \leftarrow S_{prefill} + a_k$
  end if
end for
$S_{decode} \leftarrow 0$
for each $k \in W_{decode}$ **do**
  $S_{decode} \leftarrow S_{decode} + a_k$
end for
return $S_{prefill}, S_{decode}$

Algorithm 3 Truncate Triggering

Require: Prefill sink value $S_{prefill}$; Decode sink value S_{decode}; Truncation coefficient $\tau_{truncate}$
Ensure: Boolean flag trigger_truncation

Estimate the remaining attention resources capacity
$R_{est} \leftarrow (1 - S_{prefill}) \times \tau_{truncate}$
# Check if the cumulative decode sink value exceeds the threshold
if $R_{est} < S_{decode}$ **then**
  trigger_truncation $\leftarrow$ **True**
Intervene via LogitProcessor to stop Chain-of-Thought
  Insert system prompt to force direct answer output
else
  trigger_truncation $\leftarrow$ **False**
  Continue reasoning generation
end if
return trigger_truncation

B Theoretical Justification of Output Degeneration via Attention Sinks

B.1 Setting and Assumptions

Consider a single self-attention layer of a pre-norm transformer with hidden dimension d and value dimension d_v. Let $u_i \in \mathbb{R}^d$ denote the residual-stream input at position i. Before the attention computation, u_i is passed through a normalization layer to produce $\tilde{x}_i$, and the value vector is

$$v_i \;=\; W_V\, \tilde{x}_i, \qquad W_V \in \mathbb{R}^{d_v \times d}. \tag{6}$$

We make two assumptions, both standard for the LLMs evaluated in this paper.

(A1) *Norm-bounded normalization.* There exists a constant $R_x > 0$ such that $\|\tilde{x}_i\|_2 \le R_x$ for all i. Specifically:

- RMSNorm with learnable scale $\gamma \in \mathbb{R}^d$: $R_x = \|\gamma\|_\infty \sqrt{d}$.
- LayerNorm with affine parameters (γ, β): $R_x = \|\gamma\|_\infty \sqrt{d} + \|\beta\|_2$.

(A2) *Bounded value projection.* $\|W_V\|_{\mathrm{op}} \le B_V$ for some $B_V > 0$.

B.2 Lemma 1 (Bounded Pairwise Distance of Value Vectors)

Lemma 1. *Under assumptions, define*

$$M \;:=\; 2\, B_V\, R_x. \tag{7}$$

Then for all token positions i, j in the same attention layer,

$$\|v_i - v_j\|_2 \;\le\; M. \tag{8}$$

Proof. By linearity of W_V,

$$\|v_i - v_j\|_2 \;=\; \|W_V(\tilde{x}_i - \tilde{x}_j)\|_2 \;\le\; \|W_V\|_{\mathrm{op}}\, \|\tilde{x}_i - \tilde{x}_j\|_2. \tag{9}$$

By the triangle inequality and (A1),

$$\|\tilde{x}_i - \tilde{x}_j\|_2 \;\le\; \|\tilde{x}_i\|_2 + \|\tilde{x}_j\|_2 \;\le\; 2R_x. \tag{10}$$

Combining with (A2), $\|v_i - v_j\|_2 \le B_V \cdot 2R_x = M$. □

B.3 Theorem 1 (ϵM-Neighborhood Bound)

Theorem 1. *Suppose Lemma 1 holds, so that $\|v_i - v_j\|_2 \le M$ for all i, j. Let S be a non-empty set of sink tokens with total attention mass $\mu_t := \sum_{s \in S} \alpha_{t,s}$, and define the weighted sink centroid*

$$v_S \;:=\; \frac{1}{\mu_t} \sum_{s \in S} \alpha_{t,s}\, v_s. \tag{11}$$

If $\mu_t \ge 1 - \epsilon$ for some $\epsilon \in [0, 1]$, then

$$\|h_t - v_S\|_2 \;\le\; \epsilon M. \tag{12}$$

Proof. We first observe that v_S is a convex combination of the sink value vectors $\{v_s : s \in S\}$, so by Lemma 1 the bound $\|v_i - v_S\|_2 \le M$ holds for every token i (whether or not $i \in S$). Indeed, for any i,

$$\|v_i - v_S\|_2 \;=\; \left\| \sum_{s \in S} \frac{\alpha_{t,s}}{\mu_t} (v_i - v_s) \right\|_2 \;\le\; \sum_{s \in S} \frac{\alpha_{t,s}}{\mu_t} \|v_i - v_s\|_2 \;\le\; M. \tag{13}$$

Next, decompose the attention output relative to v_S:

$$h_t - v_S \;=\; \sum_{i=1}^{t} \alpha_{t,i} v_i - v_S \;=\; \sum_{i=1}^{t} \alpha_{t,i} (v_i - v_S), \tag{14}$$

where the second equality uses $\sum_i \alpha_{t,i} = 1$. Note that the sink contribution telescopes:

$$\sum_{s\in S} \alpha_{t,s}(v_s - v_S) \;=\; \sum_{s\in S} \alpha_{t,s} v_s - \mu_t v_S \;=\; 0 \tag{15}$$

by the definition of v_S. Hence

$$h_t - v_S \;=\; \sum_{i\notin S} \alpha_{t,i}(v_i - v_S). \tag{16}$$

The triangle inequality, together with $\|v_i - v_S\|_2 \le M$ established above, yields

$$\|h_t - v_S\|_2 \;\le\; \sum_{i\notin S} \alpha_{t,i}\, \|v_i - v_S\|_2 \;\le\; M \sum_{i\notin S} \alpha_{t,i} \;=\; M(1-\mu_t) \;\le\; \epsilon M. \tag{17}$$

□

B.4 Proposition 1 (Decode-Conditional ϵM Bound)

In practice, ASTra distinguishes between sinks formed during the prefill phase and those that emerge during decoding, since prefill sinks are perceived as a known offset. We make this distinction explicit by partitioning the sink set as $S = S_p \cup S_d$, with prefill mass and decode mass

$$\mu_p \;:=\; \sum_{s\in S_p} \alpha_{t,s}, \qquad \mu_d \;:=\; \sum_{s\in S_d} \alpha_{t,s}, \qquad \mu_t \;=\; \mu_p + \mu_d. \tag{18}$$

The *decode-conditional sink ratio* used by Algorithm 3 is

$$\hat{\mu}_t \;:=\; \frac{\mu_d}{1-\mu_p} \;=\; \frac{S_{\text{decode}}}{1 - S_{\text{prefill}}}, \tag{19}$$

i.e., the fraction of the post-prefill attention budget consumed by decode-side sinks. The two formulations are linked by the algebraic identity

$$1 - \mu_t \;=\; 1 - \mu_p - \mu_d \;=\; (1-\mu_p)\,(1-\hat{\mu}_t), \tag{20}$$

which we use to translate every result derived in absolute mass form to the decode-conditional form actually employed at inference.

Proposition 1 (Decode-Conditional Bound). *Under the conditions of Theorem 1, suppose $\hat{\mu}_t \ge 1 - \hat{\epsilon}$ for some $\hat{\epsilon} \in [0,1]$. Then*

$$\|h_t - v_S\|_2 \;\le\; \hat{\epsilon}\,(1-\mu_p)\,M. \tag{21}$$

Proof. By Eq. (20), $1 - \mu_t = (1-\mu_p)(1-\hat{\mu}_t) \le \hat{\epsilon}(1-\mu_p)$, so $\mu_t \ge 1 - \hat{\epsilon}(1-\mu_p)$. Applying Theorem 1 with $\epsilon = \hat{\epsilon}(1-\mu_p)$ gives $\|h_t - v_S\|_2 \le \hat{\epsilon}(1-\mu_p)\,M$. □

From Proposition 1 to Eq. (5). Setting $\hat{\epsilon} = 1 - \hat{\tau}$ in Eq. (21) yields the stopping condition

$$\hat{\mu}_t > \hat{\tau} \iff S_{\text{decode}} \;>\; (1 - S_{\text{prefill}})\,\hat{\tau}, \tag{22}$$

which is exactly Eq. (5) in Section 4.3. The empirical estimator $\hat{\mu}_t$ is therefore not an unjustified substitute for μ_t but a decode-conditional *instance* of Theorem 1 obtained through Eq. (20).

B.5 Lemma 2 (Lipschitz Continuity of Log-Softmax)

Lemma 2. *For any logit vectors $z, z' \in \mathbb{R}^{|V|}$ and any token index $y \in \{1,\dots,|V|\}$,*

$$\big|\log \mathrm{softmax}(z)[y] - \log \mathrm{softmax}(z')[y]\big| \;\le\; 2\,\|z - z'\|_\infty \;\le\; 2\,\|z - z'\|_2. \tag{23}$$

Proof. Write $\log \mathrm{softmax}(z)[y] = z[y] - \mathrm{LSE}(z)$, where $\mathrm{LSE}(z) := \log \sum_k e^{z[k]}$. Then

$$\log \mathrm{softmax}(z)[y] - \log \mathrm{softmax}(z')[y] \;=\; \big(z[y] - z'[y]\big) - \big(\mathrm{LSE}(z) - \mathrm{LSE}(z')\big). \tag{24}$$

The first term is bounded by $\|z - z'\|_\infty$. For the second, set $\delta := \|z - z'\|_\infty$. Then $z[k] \le z'[k] + \delta$ for all k, so

$$\sum_k e^{z[k]} \;\le\; e^{\delta} \sum_k e^{z'[k]} \;\Longrightarrow\; \mathrm{LSE}(z) \le \mathrm{LSE}(z') + \delta. \tag{25}$$

By symmetry $|\mathrm{LSE}(z) - \mathrm{LSE}(z')| \le \delta$. Combining,

$$\big|\log \mathrm{softmax}(z)[y] - \log \mathrm{softmax}(z')[y]\big| \;\le\; 2\delta \;\le\; 2\|z - z'\|_2, \tag{26}$$

where the last step uses $\|\cdot\|_\infty \le \|\cdot\|_2$. □

B.6 THEOREM 2 (BOUNDED LOG-PROBABILITY DRIFT UNDER SINK DOMINANCE)

Setup. Let $\hat{h}_t$ denote the attention output of the final attention layer at decoding step t, so that the next-token logits are

$$z_t \;=\; W_U\,\hat{h}_t, \qquad W_U \in \mathbb{R}^{|V|\times d_v}, \tag{27}$$

where W_U absorbs any final layer-normalization factors.[3] We assume that the conditions of Theorem 1 apply to this final attention layer, with sink set S, sink centroid v_S, and sink mass $\mu_t \geq 1-\epsilon$.

Theorem 2 (Bounded Log-Probability Drift). *Define*

$$\kappa_{\max} \;:=\; 2\,\|W_U\|_{\mathrm{op}} \cdot M \;=\; 4\,\|W_U\|_{\mathrm{op}}\,B_V\,R_x. \tag{28}$$

Let $\hat{y} \in \{1,\ldots,|V|\}$ be any target token, and define $\ell_t(\hat{y}) := \log \mathrm{softmax}(z_t)[\hat{y}]$, $\ell_S(\hat{y}) := \log \mathrm{softmax}(W_U v_S)[\hat{y}]$. Then under the conditions of Theorem 1,

$$\big|\ell_t(\hat{y}) - \ell_S(\hat{y})\big| \;\leq\; \kappa_{\max}\cdot(1-\mu_t). \tag{29}$$

Proof. By Theorem 1 with $\epsilon = 1-\mu_t$, $\|\hat{h}_t - v_S\|_2 \leq (1-\mu_t)M$. Hence

$$\|z_t - W_U v_S\|_2 \;=\; \|W_U(\hat{h}_t - v_S)\|_2 \;\leq\; \|W_U\|_{\mathrm{op}}\cdot\|\hat{h}_t - v_S\|_2 \;\leq\; \|W_U\|_{\mathrm{op}}\cdot(1-\mu_t)\,M. \tag{30}$$

Applying Lemma 2 with $z = z_t$ and $z' = W_U v_S$:

$$\big|\ell_t(\hat{y}) - \ell_S(\hat{y})\big| \;\leq\; 2\,\|z_t - W_U v_S\|_2 \;\leq\; 2\,\|W_U\|_{\mathrm{op}} M\,(1-\mu_t) \;=\; \kappa_{\max}(1-\mu_t). \tag{31}$$

□

Decode-conditional form. Substituting Eq. (20) directly into Eq. (29) gives

$$\big|\ell_t(\hat{y}) - \ell_S(\hat{y})\big| \;\leq\; \kappa_{\max}\,(1-\mu_p)\,(1-\hat{\mu}_t), \tag{32}$$

the form used in Section 4.3 to derive the implemented stopping rule $\hat{\mu}_t > \hat{\tau}$.

B.7 COROLLARY 1 (BOUND ON THE MARGINAL REASONING GAIN)

Corollary 1. *Let $R_t = \log P(\hat{y} \mid S_t)$ for a single-token answer $\hat{y}$, computed from a final-layer attention output $\hat{h}_t$ with sink mass $\mu_t \geq 1-\epsilon_t$. Suppose at two decoding steps t and $t-\mathrm{stride}$ the same sink centroid v_S persists (stability property, Section 3.1). Then*

$$\big|\Delta R_t\big| \;:=\; \big|R_t - R_{t-\mathrm{stride}}\big| \;\leq\; \kappa_{\max}\cdot\big[(1-\mu_t) + (1-\mu_{t-\mathrm{stride}})\big]. \tag{33}$$

In particular, if both μ_t and $\mu_{t-\mathrm{stride}}$ exceed $1-\epsilon$, then $|\Delta R_t| \leq 2\kappa_{\max}\epsilon$.

Proof. By the triangle inequality and two applications of Theorem 2,

$$|R_t - R_{t-\mathrm{stride}}| \;\leq\; |R_t - \ell_S(\hat{y})| + |\ell_S(\hat{y}) - R_{t-\mathrm{stride}}| \tag{34}$$

$$\leq\; \kappa_{\max}(1-\mu_t) + \kappa_{\max}(1-\mu_{t-\mathrm{stride}}). \tag{35}$$

□

Decode-conditional form. Applying Eq. (20) to both terms on the right-hand side of Eq. (33),

$$|\Delta R_t| \;\leq\; \kappa_{\max}(1-\mu_p)\,\big[(1-\hat{\mu}_t) + (1-\hat{\mu}_{t-\mathrm{stride}})\big], \tag{36}$$

where we have used the fact that μ_p is the prefill sink mass and is unchanged across decoding steps.

Multi-token answers. If the answer is a sequence $\hat{y} = (\hat{y}_1,\ldots,\hat{y}_n)$ with $R_t = \sum_{i=1}^{n}\log P(\hat{y}_i \mid S_t, \hat{y}_{<i})$, applying Theorem 2 at each answer-prediction position and summing yields

$$|\Delta R_t| \;\leq\; n\,\kappa_{\max}\,\big[(1-\mu_t) + (1-\mu_{t-\mathrm{stride}})\big], \tag{37}$$

which preserves the linear-in-$(1-\mu_t)$ structure used by ASTra's stopping rule.

[3] If a final RMSNorm/LayerNorm is applied to $\hat{h}_t$ before W_U, by (A1) the operation is bounded and can be absorbed into the operator norm. The resulting effective constant satisfies $\|W_U\|_{\mathrm{op}}^{\mathrm{eff}} \leq \|W_U\|_{\mathrm{op}}\cdot\|\gamma\|_\infty\sqrt{d_v}/\min_i r_i$ in the RMSNorm case.

B.8 Discussion: From Bound to Stopping Rule

Equation (33) provides the rigorous justification for the optimal-stopping framework of Section 3.3. Specifically, the marginal gain $\mathbb{E}[\Delta R_t]$ from continued decoding satisfies

$$\big|\mathbb{E}[\Delta R_t]\big| \;\le\; \mathbb{E}\big[|\Delta R_t|\big] \;\le\; 2\kappa_{\max}\,(1-\mu_t), \tag{38}$$

where the last step assumes a stationary sink configuration. Substituting into the stopping condition $|\mathbb{E}[\Delta R_t]| < \lambda$ gives $\mu_t > 1 - \lambda/(2\kappa_{\max})$, recovering Eq. (3) with the explicit identification $\kappa = 2\kappa_{\max}$ as the worst-case Lipschitz constant.

Translation to the implemented stopping rule. Combining Eq. (38) with the partition identity (20) gives the decode-conditional envelope

$$\big|\mathbb{E}[\Delta R_t]\big| \;\le\; 2\kappa_{\max}\,(1-\mu_p)\,(1-\hat{\mu}_t). \tag{39}$$

The stopping condition $|\mathbb{E}[\Delta R_t]| < \lambda$ therefore reduces to

$$\hat{\mu}_t \;>\; \hat{\tau}, \qquad \hat{\tau} \;:=\; 1 - \frac{\lambda}{2\kappa_{\max}\,(1-\mu_p)} \;=\; 1 - \frac{\lambda}{\kappa_{\text{eff}}}, \tag{40}$$

where $\kappa_{\text{eff}} := 2\kappa_{\max}(1-\mu_p)$ is the *effective* Lipschitz constant absorbing the model-specific prefill offset. Since μ_p is approximately stable for a given model (Table 1, with cross-model variation between 0.46 and 0.73), a single $\hat{\tau}$ per model suffices, and the implemented rule $(1 - S_{\text{prefill}})\,\tau < S_{\text{decode}}$ in Eq. (5) is exactly this conditional condition with $\tau \equiv \hat{\tau}$.

Why τ_{truncate} varies across models. The decomposition $\hat{\tau} = 1 - \lambda/[2\kappa_{\max}(1-\mu_p)]$ identifies two independent sources of cross-model variation in Table 5: (i) the worst-case Lipschitz constant $\kappa_{\max} = 4\,\|W_U\|_{\text{op}} B_V R_x$, which scales with model capacity and underpins the scaling-law-based extrapolation in Appendix E; and (ii) the prefill sink fraction μ_p, which is directly observable from Table 2. Larger models tend to have both larger $\kappa_{\max}$ and slightly different μ_p, jointly explaining the $5\times$ spread of τ_{truncate} between Qwen3-4B and Qwen3-32B without invoking a free hyperparameter.

C Overhead analysis

With settings demonstrated in Table 8, given a prompt, our method causes the following overheads:

Memory overhead: Our method requires storing the attention values of the next W = 50 tokens for every token in the sequence, until it has been identified as attention sink/non-attention sink. Therefore, the number of attention values (BF16) we have to additionally store is 1275.

Time overhead: Since we apply FlashAttention-2 (Dao, 2023) settings in our method, we maintain high inference throughput during reasoning. However, minor cost is introduced to calculate attention weights with query vector and key vector in our sink value monitor in order to adapt to Flashattention-2 settings, which remains comparable with other overthinking mitigation methods.

Table 4: Overhead comparison between overthinking-mitigation methods, measured on MATH500 with 100 test cases on a single H100 GPU. *Rel. throughput* is the token-generation speed normalised by the FlashAttention-2 baseline ($\times 1.00$); for Qwen3-8B it is averaged over the datasets of Figure 7, and for Qwen3-32B it is computed from the tok/s column of this table. "–" marks configurations we did not measure. ASTra is about 20% slower than the baseline in tok/s because of the additional sink-awareness computation, and its memory overhead is negligible (< 0.1 GB); the 50–70% token reduction nevertheless makes it the fastest method in seconds per query.

Method	**Sec./query**	**tok/s**	**Rel. throughput**	**Peak mem. (GB)**
Qwen3-8B				
FlashAttention-2 (baseline)	84.4	44.4	$\times 1.00$	17.68
Dynasor	79.8	33.6	$\times 0.76$	17.69
ROM	143.0	19.1	$\times 0.43$	17.69
ASTra (ours)	**59.7**	34.0	$\times 0.82$	17.73
Qwen3-32B				
FlashAttention-2 (baseline)	131.6	27.8	$\times 1.00$	67.77
Dynasor	138.3	21.7	$\times 0.78$	67.82
ROM	201.2	12.8	$\times 0.46$	67.80
ASTra (ours)	**40.0**	22.5	$\times 0.81$	67.89

Table 4 shows that the sink-awareness mechanism costs roughly 20% of raw token-generation speed, while the additional memory footprint stays below 0.1 GB on both model sizes. Since ASTra removes 50–70% of the generated tokens, the per-query latency still drops substantially — from 84.4 s to 59.7 s on Qwen3-8B and from 131.6 s to 40.0 s on Qwen3-32B — so a lower tok/s does not translate into a slower system.

Wall-clock time analysis is shown in the following graph:

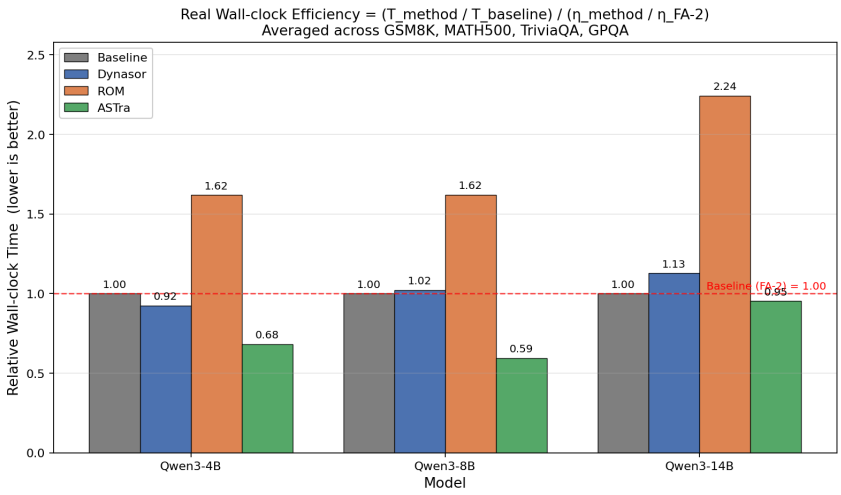

Figure 6: Result of wall-clock analysis for all methods.

Since all online mitigation methods require additional computation that cannot be fused with optimized kernel, the wall-clock time reduction will not be as fundamental as token reduction. From the graph we can see that our method (ASTra) consistently outperforms baseline (FlashAttention-2), while saving 20% to 60% tokens on average. ROM method is slower than baseline, since its additional attention head cannot be integrated in FlashAttention-2 kernel. Despite the wall-clock time reduction is not as significant as token reduction, fewer token still indicates lower computational cost.

D COMPARISON WITH OTHER SIGNALS

Comparison with other signal is made with the Qwen3-8B and Qwen3-14B models, on GPQA datasets. The truncation method remains universal, as proposed in J, leaving signal type the only difference.

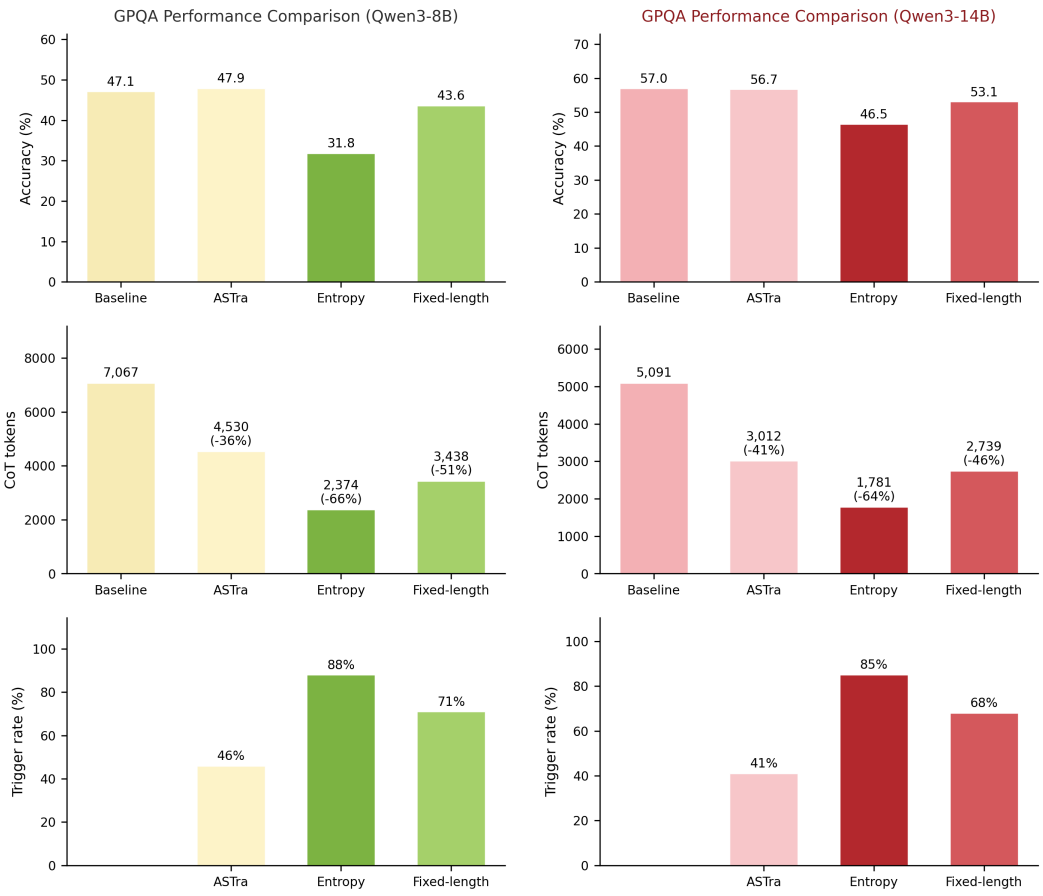

Figure 7: Result of signal effective comparison

From the results we can conclude that Astra outperforms other methods, especially simple entropy signal, which shows catastrophic decrease in accuracy. Compared with other methods, Astra triggers much less intervention, therefore achieving a much higher accuracy as well as moderate and proper token reduction rate. This demonstrates that attention sink signal identifies overthinking and distinguishes them from necessary reasoning more precisely than fixed-length truncation and entropy signal when testing on hard datasets.

E EMPIRICAL METHOD TO ESTIMATE $\tau_{truncate}$

While testing on easy and moderate-difficulty tasks (GSM8K and MATH500), our empirical setting for $\tau_{truncate}$ has been proved by both experiment results 2 and ablation study results 12. However, when applying our method to novel tasks with greater difficulty, we have to propose a general method to estimate a proper $\tau_{truncate}$. $\tau_{truncate}$ is linked to two factors: user's cost sensitivity (λ) and model capability (κ). Assume that we would like to achieve near-lossless results in all situations, the ability of models will be the key factor for $\tau_{truncate}$. To estimate the model capacity in a model family, we refer to **the Scaling Law** (Henighan et al., 2020; Kaplan et al., 2020): a model's performance predictably improves as a power-law function of its training computation, dataset size, and parameter count. This phenomenon will be fundamental while testing on complex tasks such as GPQA, where

high-level reasoning and information retrieval require significantly larger parameters. While scaling law predicts power-law performance gain, studies have also found that there is a boundary where the marginal benefits of purely scaling pre-training parameters eventually diminish, forming the upper asymptote of an S-shaped performance curve on specific downstream tasks (Srivastava et al., 2023; Wei et al., 2022). Similarly, a lower bound also exists where the model capacity falls below the critical scale N_C (Kaplan et al., 2020). Based on these discoveries, we would like to express the ability of models with different parameters as an S-shaped function while testing on hard tasks with scaling law expressions (Kaplan et al., 2020):

$$\tau_{truncate}(N) = \frac{\tau_{\max}}{1 + (N/N_c)^{\alpha}} \tag{41}$$

where $\tau_{\max}$ is the upper asymptote attained on small models, N_C is the half-saturation scale at which $\tau = \tau_{\max}/2$, and $\alpha > 0$ controls the sharpness of the transition. Eq. (41) recovers a power law $\tau \approx \tau_{\max}(N_C/N)^{\alpha}$ in the regime $N \gg N_C$ relevant to large models on hard tasks, while properly bounding τ from above as $N \to 0$. With this modeling, as soon as we find valid $\tau_{truncate}$ values for several models in a family via grid search, we can fit Eq. (41) and extrapolate to models with different parameter counts, which turns empirical results into transferable estimates for other model families with similar ability. For instance, fitting Eq. (41) on the four Qwen3 checkpoints calibrated on GPQA yields $\hat{\tau}_{\max} = 0.489$, $\hat{N}_C = 17.36\,\text{B}$, $\hat{\alpha} = 2.25$, with full-data $R^2 = 0.9997$ (Figure 8). To prove its effectiveness, we employed grid search for best parameters in advance. Our criterion is first to ensure near-lossless accuracy, then to measure token reduction proportions. After that, we conduct leave-one-out (LOO) predictions: for the four Qwen3 models of different sizes, we assume the optimal parameter for one is unknown, and predict it by re-fitting Eq. (41) on the other three. For the two non-Qwen3 models (DeepSeek-R1-Distill-Qwen2.5-7B and GLM4-9B), predictions are obtained from the full Qwen3 fit as a cross-family transfer test. Results can be seen in the following table.

The mean absolute percentage error is 6.6% across the four Qwen3 LOO predictions and 7.6% averaged over all six models, with the largest relative deviations occurring on Qwen3-32B (absolute error only 0.018, in the small-τ regime where accuracy is insensitive to τ shifts of this magnitude) and on GLM4-9B (-13%); for the latter, GLM4 is independently identified in our limitations section as exhibiting underthinking rather than overthinking, suggesting an effective κ different from the Qwen3 family. The fit thus yields predictions accurate enough to serve as a starting point on new models, with at most a small grid-search refinement around $\hat{\tau}$.

Table 5: Best settings for $\tau_{truncate}$ on all models while testing on GPQA in main results Table 2 versus predicted value. Qwen3 entries use leave-one-out (LOO); DeepSeek and GLM4 entries are cross-family predictions from the full Qwen3 fit.

Models	**Best settings**	**Predicted Result**
Qwen3-32B	0.13	0.102
Qwen3-14B	0.35	0.320
Qwen3-8B	0.45	0.413
Qwen3-4B	0.50	0.485
Deepseek-R1-Distill-Qwen2.5-7B	0.46	0.433
GLM4-9B	0.46	0.398

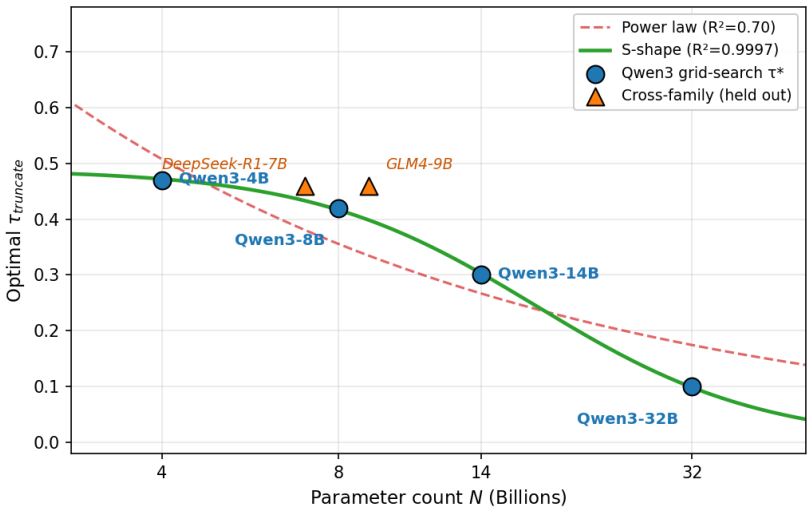

Figure 8: Optimal $\tau_{truncate}$ as a function of model parameter count N on GPQA. Blue circles: grid-searched τ^* on Qwen3 checkpoints. Orange triangles: cross-family models (DeepSeek-R1-Distill-Qwen2.5-7B, GLM4-9B), held out from the fit. Solid green: S-shaped fit from Eq. (41) ($R^2 = 0.9997$). Dashed red: a pure power-law alternative $\tau = (N_c/N)^\alpha$ shown for reference; it matches interior points roughly but diverges from the data at the boundary scales (4B and 32B), motivating the saturating form in Eq. (41).

F Parameter Sensitivity

ASTRA has two groups of hyperparameters: the sink-identification parameters (τ_V, τ_R, the window size W and the number of sampled layers M), and the truncation threshold $\tau_{truncate}$. The former are universal, i.e. shared by all models and datasets (Table 8), and their sensitivity is reported in Figures 9–11: identification quality is stable over a wide range and saturates beyond the values we adopt. This section therefore focuses on $\tau_{truncate}$, which fixes the operating point of the method.

F.1 Sensitivity on easy and moderate tasks

When overthinking is present in most queries, as on GSM8K and MATH500, accuracy is largely insensitive to $\tau_{truncate}$, while the token reduction is not. Table 6 sweeps the threshold on MATH500 with Qwen3-8B.

Table 6: Sensitivity to the truncation threshold $\tau_{truncate}$ on MATH500 with Qwen3-8B. Accuracy stays within ± 1.5 points of the baseline across the whole sweep, whereas the token reduction decreases monotonically as the threshold grows.

Config	Acc.	CoT tokens	Token reduction
Baseline	0.825	3536	–
$\tau_{truncate} = 0.05$	0.785	1013	71.3%
$\tau_{truncate} = 0.10$ (ours)	**0.840**	1294	63.4%
$\tau_{truncate} = 0.15$	0.820	1325	62.5%
$\tau_{truncate} = 0.20$	0.825	1372	61.2%
$\tau_{truncate} = 0.30$	0.835	2219	37.3%
$\tau_{truncate} = 0.40$	0.840	3012	14.8%

When $\tau_{truncate} > 0.05$, accuracy varies within a narrow band (0.820–0.840 against a baseline of 0.825), so no setting in this range is harmful; the token reduction, by contrast, falls from 71.3% to 14.8%. In this regime $\tau_{truncate}$ therefore trades saved tokens against a safety margin rather than

against correctness, and we adopt the elbow of the curve. The same trend appears independently in the threshold ablation of Figure 12, where accuracy on Qwen3-8B and Qwen3-14B stops improving once $\tau_{truncate}$ passes 0.1 while the average token count keeps rising, so Table 6 and Figure 12 locate the same operating point from two different sweeps.

F.2 Transfer across models

On harder tasks the optimal threshold reflects the gap between model capability and task difficulty rather than the model family. On GPQA the grid-searched optima of DeepSeek-R1-Distill-Qwen2.5-7B, Qwen3-8B and GLM4-9B lie close to each other (Table 5), and a single capability-based fit predicts held-out cross-family models to within 0.07 in absolute threshold value (Appendix E), which is why we do not tune the threshold per model.

F.3 Transfer across tasks

The threshold also transfers across datasets of comparable difficulty. Reusing the GPQA threshold without any re-calibration, we evaluate Qwen3-8B on Omni-MATH, an olympiad-level mathematics benchmark, over 540 test cases. As Table 7 shows, the transferred threshold cuts 43% of the chain of thought at a 0.5-point accuracy cost, and the reduction adapts to difficulty rather than being uniform.

Table 7: Cross-task transfer of $\tau_{truncate}$: the threshold calibrated on GPQA applied unchanged to Omni-MATH (Qwen3-8B, 540 cases, single H100). ΔAcc. is in percentage points.

Difficulty band	Baseline	ASTra	ΔAcc.(%)	Baseline CoT	ASTra CoT	Reduction
1–3 (easy)	0.801	0.767	-3.3	4903	4186	-15%
4–6 (medium)	0.444	0.422	-2.2	12024	6722	-44%
7–9 (hard)	0.178	0.211	$+3.3$	14017	6300	-55%
Overall	0.473	0.468	-0.5	10050	5696	-43%

Longer chains are pruned more aggressively than short ones, and the hardest band is not the one that suffers most, indicating that the monitored signal adapts to the instance instead of firing at a fixed depth.

G Experiment settings

G.1 Hyperparameters

Table 8: Hyperparameters and Experimental Setup

Parameter / Setting	Value/Setting
Sampled layers (M)	6
Window Size (W)	50
Threshold (τ_V)	10^{-3}
Threshold (τ_R)	0.5
Threshold ($\tau_{truncate}$)	0.1, See table 5 for GPQA
Hardware	NVIDIA H100-80GB GPU
Prompting Strategy	Chain of Thought (CoT)

G.2 DATASETS

Table 9: Statistics of the four evaluation datasets used in our experiments. The four datasets span over an order of magnitude in both input length ($\sim$21 to $\sim$372 tokens) and baseline output length (757 to 6,694 tokens), and cover four distinct task formats (free-form numerical, free-form mathematical expression, open-ended factual, multiple-choice scientific). Input/output lengths are measured with the Qwen3 tokenizer; output length is the mean generated tokens by the Qwen3-8B baseline (matching Table 2).

Dataset	Case number	Input len	Domain	Answer format
GSM8K (Cobbe et al., 2021)	1,319	65	Grade-school math word problems	Free-form numerical
MATH500 (Lightman et al., 2023)	500	71	Competition math (algebra, geometry, calculus, number theory)	Free-form expression / number
TriviaQA (Joshi et al., 2017)	17,944	21	Open-domain factual / trivia QA	Open-ended short answer
GPQA (Rein et al., 2023)	448	372	Graduate-level physics, chemistry, biology	Multiple choice (4-way)

H MAIN TABLE ERROR ANALYSIS

In this section we provide a statistical error analysis of our main results to rigorously justify the effectiveness of ASTra. Since all experiments use greedy decoding, run-to-run variance is by construction zero; the relevant statistical question is therefore the *sample-level* uncertainty over each finite test set. Accordingly, we compare Baseline and ASTra using paired tests that exploit the fact that both methods are evaluated on identical inputs: McNemar's exact binomial test for binary correctness (Accuracy), and Wilcoxon signed-rank test for continuous metrics (F1 score and token counts). The analysis is organized into three parts: (i) quality preservation (Table 10), (ii) token distribution shift (Table 11), and (iii) statistical significance and effect size of token reduction (Table 12).

H.1 QUALITY PRESERVATION

Table 10 reports paired-test p-values for the quality metric on each (model, dataset) cell. Across all 24 cells, **none reaches the $p < 0.05$ threshold**, indicating that the accuracy/F1 differences between Baseline and ASTra are statistically indistinguishable from sampling noise. Even on TriviaQA — where ASTra applies its most aggressive truncation (cf. Table 11) — the F1 differences are not significant for any of the six models ($p \geq 0.136$). This empirically supports our central claim that ASTra is *near-lossless*: the substantial token savings reported in our main results are not achieved at the cost of measurably degraded output quality.

Table 10: McNemar p-values (and Wilcoxon p-values for F1 score on TriviaQA) comparing Baseline versus ASTra. No cell reaches statistical significance at $p < 0.05$, supporting our claim that ASTra is near-lossless in output quality.

Model	GSM8K	MATH-500	TriviaQA	GPQA
QWEN3-4B	0.210	1.000	0.569	0.719
QWEN3-8B	0.625	1.000	0.423	0.549
QWEN3-14B	0.249	1.000	0.180	0.481
QWEN3-32B	0.888	0.500	0.136	0.896
GLM4-9B	0.500	0.830	1.000	1.000
DEEPSEEK-R1-DISTILL	0.096	0.129	0.167	0.215

H.2 TOKEN DISTRIBUTION SHIFT

Table 11 reports the median and interquartile range (IQR) of token counts before and after applying ASTra. Reporting median and IQR — rather than mean and standard deviation — is more informative

here because token-count distributions are typically right-skewed: a long tail of overthinking-heavy examples inflates the mean and obscures the typical behavior. Three patterns emerge.

First, on **larger Qwen3 models on harder datasets**, ASTra achieves dramatic median reductions: Qwen3-32B / MATH-500 drops from 2,201 to 529 (-76%); Qwen3-32B / GPQA drops from 3,037 to 1,223 (-60%); DeepSeek-R1-Distill / GPQA drops from 4,614 to 1,563 (-66%). These are precisely the (model, dataset) regimes where the baseline exhibits the most overthinking, confirming that ASTra disproportionately benefits the cases that need it most.

Second, on **TriviaQA — a short-answer factual task — ASTra collapses the token distribution onto the answer token itself**. For Qwen3-4B and Qwen3-8B, the median drops from 473–480 to 89–90 tokens with extremely tight IQRs ([110] and [106]), indicating that ASTra correctly recognizes that factual QA requires no extended reasoning and truncates almost immediately.

Third, on **GLM4-9B**, the median changes are minimal across all datasets (e.g., GSM8K $216 \rightarrow 203$, GPQA $557 \rightarrow 533$). This is consistent with our limitations discussion (Section 7): GLM4-9B exhibits an under-thinking pattern rather than overthinking, so there is little overthinking budget for ASTra to reclaim.

Table 11: Token count distribution: Baseline and ASTra. Each cell shows median token count with interquartile range [IQR] in brackets. For GSM8K, MATH-500, and TriviaQA, maximum tokens is set to 8192; for GPQA, 32768.

Model	Method	GSM8K Median [IQR]	MATH-500 Median [IQR]	TriviaQA Median [IQR]	GPQA Median [IQR]
QWEN3-4B	Baseline	768 [670]	2012 [1237]	473 [565]	5961 [5058]
	ASTra	643 [448]	968 [594]	90 [110]	3585 [4373]
QWEN3-8B	Baseline	1028 [893]	2324 [1369]	480 [528]	6132 [6380]
	ASTra	499 [208]	966 [872]	89 [106]	4618 [3798]
QWEN3-14B	Baseline	802 [705]	2117 [1419]	407 [364]	3962 [4450]
	ASTra	780 [631]	1739 [863]	358 [371]	2795 [2614]
QWEN3-32B	Baseline	887 [578]	2201 [1341]	629 [354]	3037 [2326]
	ASTra	414 [218]	529 [320]	546 [274]	1223 [906]
GLM4-9B	Baseline	216 [97]	682 [491]	210 [89]	557 [340]
	ASTra	203 [96]	634 [465]	199 [81]	533 [329]
DEEPSEEK-R1-DISTILL	Baseline	299 [196]	1530 [1080]	320 [226]	4614 [7379]
	ASTra	274 [157]	608 [626]	95 [18]	1563 [1630]

H.3 SIGNIFICANCE AND EFFECT SIZE OF TOKEN REDUCTION

While Table 11 demonstrates a visible shift in the typical token count, statistical inference is required to confirm that the reduction is not merely a few outliers being trimmed. Table 12 reports two complementary quantities: (i) the paired Wilcoxon signed-rank p-value, which tests whether the *distribution* of paired differences is centered at zero; and (ii) Cohen's d on paired differences, which measures the standardized *magnitude* of the reduction. Following the conventional interpretation ($|d| \geq 0.2$ small, ≥ 0.5 medium, ≥ 0.8 large), we observe:

Token reduction is statistically significant in 20 of 24 cells. The four exceptions are GLM4-9B on all four datasets ($p \in [0.081, 0.823]$, $|d| \leq 0.10$) and Qwen3-14B on TriviaQA ($p = 0.109$, $d = -0.13$). The GLM4-9B results are consistent with the under-thinking pattern noted in Section 7; the Qwen3-14B / TriviaQA cell reflects a model that already produces reasonably short answers on TriviaQA, leaving little room for ASTra to act.

Effect sizes scale with model size and task difficulty. On Qwen3-32B / MATH-500, $d = -2.36$ indicates an extremely large reduction relative to the within-pair variability — the largest effect in the table. More generally, for the four Qwen3 models, the median $|d|$ across datasets generally increases with parameter count (approximately 0.78, 0.84, 0.42, 1.19 for 4B/8B/14B/32B), supporting our hypothesis that larger models have more overthinking budget for ASTra to reclaim. Datasets that

elicit longer chains-of-thought (MATH-500, GPQA) consistently exhibit medium-to-large effects ($|d| \geq 0.58$ for all evaluated non-GLM4 models on these datasets).

Together, Tables 10–12 establish that ASTra reduces token consumption by a statistically significant and practically meaningful margin while leaving output quality statistically unchanged — the formal substantiation of the "near-lossless token reduction" claim made in our main results.

Table 12: Token count statistical comparison: paired Wilcoxon signed-rank test and Cohen's d effect size. Each dataset shows Wilcoxon p-value and Cohen's d on paired differences between Baseline and ASTra. ASTra significantly reduces tokens on 20 of 24 cells, with medium-to-large effects ($|d| \geq 0.30$) on most cells.

Model	GSM8K		MATH-500		TriviaQA		GPQA	
	p	d	p	d	p	d	p	d
QWEN3-4B	< 0.001	-0.42	< 0.001	-1.26	< 0.001	-0.73	< 0.001	-0.69
QWEN3-8B	< 0.001	-0.86	< 0.001	-1.15	< 0.001	-0.78	< 0.001	-0.58
QWEN3-14B	< 0.001	-0.32	< 0.001	-0.60	0.109	-0.13	< 0.001	-0.61
QWEN3-32B	< 0.001	-0.95	< 0.001	-2.36	< 0.001	-0.18	< 0.001	-1.25
GLM4-9B	0.144	-0.09	0.081	-0.10	0.823	-0.06	0.795	-0.03
DEEPSEEK-R1-DISTILL	< 0.001	-0.13	< 0.001	-1.09	< 0.001	-0.52	< 0.001	-0.84

I ABLATION STUDY GRAPHS

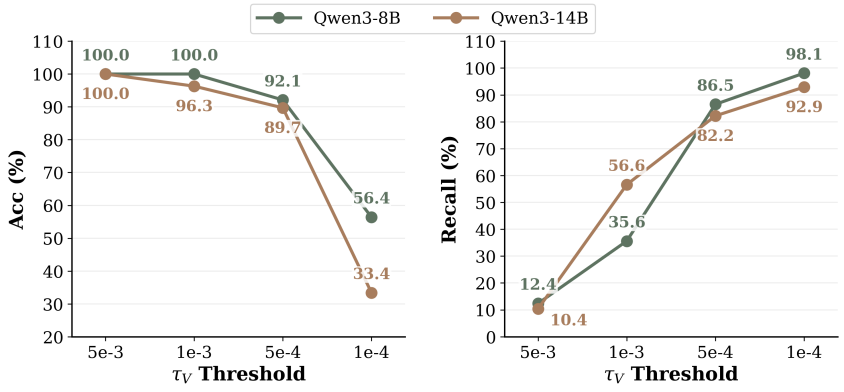

Figure 9: Result of different τ_V settings on Qwen3-8B and Qwen3-14B. Note that accuracy denotes the proportion of identified sinks are true, while recall rate denotes the proportion of identified sink compared with all sink. Despite the accuracy/recall are not extremely high, the high-value sinks are always detected in all problems, playing a major role in the effectiveness of our algorithm.

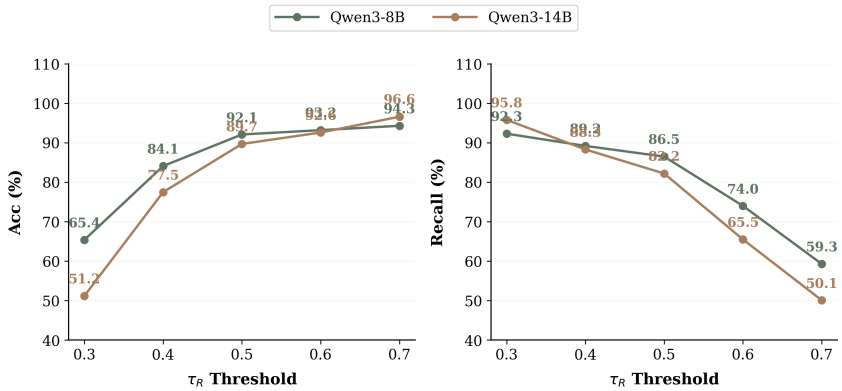

Figure 10: Result of different τ_R settings on Qwen3-8B and Qwen3-14B.

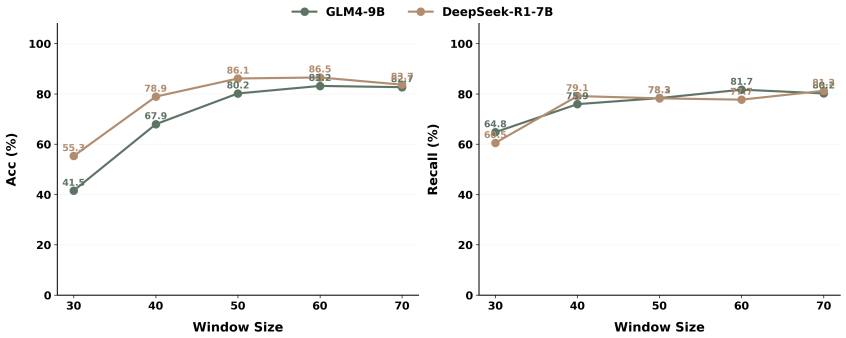

Figure 11: Result of different W settings on GLM4-9B and Deepseek-R1-Distill-Qwen2.5-7B. Note that after window size 50, the rising trend of accuracy and recall rate is not prominent. To balance overhead and performance, we employ window size 50.

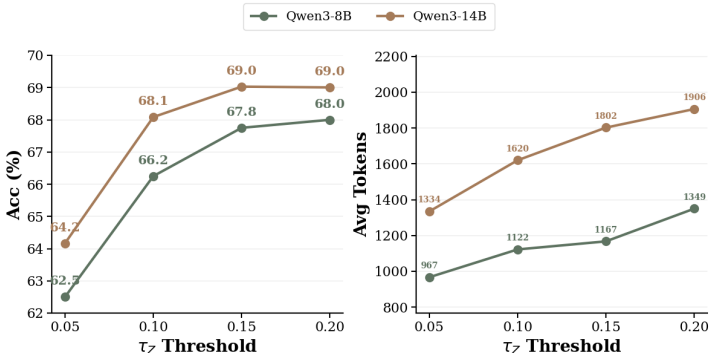

Figure 12: Result of different $\tau_{truncate}$ settings on Qwen3-8B and Qwen3-14B. Results show that after $\tau_{truncate} = 0.1$, there is no significant increase in accuracy, yet there is a steady increase in average tokens.

J Inserted Prompt Analysis

For all models and datasets, once ASTra decides to terminate the chain of thought, we insert a prompt via logitProcessor: **I have finished the necessary reasoning. Directly generate the answer.** To analyze Astra's sensitivity to injected prompt, we conduct a study to analyze the effect of different

prompts. Results have shown that Astra is insensitive to different expressions, given they have the same semantic meaning, urging the model to stop reasoning.

Table 13: Prompt-injection ablation on GPQA with Qwen3-8B. ASTRA is not sensitive to the wording of the stop prompt: accuracy, CoT token cost, and trigger rate remain nearly identical across variants.

Variant	Stop prompt	Acc.	Tokens	Trig.
default	"I have finished the necessary reasoning. Directly generate the answer."	0.544	4701	45%
timeup	"Time is up. Based on the reasoning above, I now state the final answer."	0.539	4703	45%
confident	"I am confident in the result obtained above. The final answer is:"	0.541	4785	45%
concise	"Enough reasoning. Final answer:"	0.543	4775	45%

K DIFFICULTY-LEVEL ANALYSIS

While studying overthinking naturally pairs relatively strong models with easy-to-moderate tasks, a reasonable concern is whether ASTra attains its token reduction by cutting off *productive* reasoning on hard problems. To examine this, we conduct a difficulty-level ablation analysis: we retrieve the per-example experiment data of the four Qwen3 models (4B/8B/14B/32B) in Table 2 on GPQA, and aggregate accuracy by difficulty level. Results are reported in Table 14.

Table 14: Accuracy by difficulty level on GPQA, aggregated over the four Qwen3 models.

Level	n	Baseline	ASTra	ΔAcc(%)
0 (hardest)	275	0.538	0.512	-2.06
1	231	0.641	0.658	$+1.70$
2 (easiest)	28	0.536	0.500	-3.60

Across all difficulty levels of GPQA, the accuracy difference between Baseline and ASTra remains marginal. For an even harder, olympiad-level evaluation, the Omni-MATH transfer experiment (Table 7, Appendix F) shows that, reusing the GPQA truncation threshold on Qwen3-8B over 540 test cases, ASTra reduces CoT length by 43% with only a 0.5-point overall accuracy loss, while accuracy on the hardest band (7–9) even improves by 3.3 points. This further indicates that the truncation threshold is transferable between datasets in different domains without too much difference in difficulty.

L FAILURE CASE ANALYSIS

To better analyze the effect Astra poses on each testcase, we aggregate the testlog of Qwen3-8/14/32B on GPQA as an example. Separating them into 4 types:

- Baseline correct, Astra correct
- Baseline correct, Astra wrong
- Baseline wrong, Astra correct
- Baseline wrong, Astra wrong

Results can be seen in the following table:

Table 15: Average CoT length by failure type: baseline vs. ASTra.

Type	Share	Baseline CoT	ASTra CoT
1	49.4%	3758	2958
2	7.1%	7646	3537
3	6.9%	6397	3447
4	36.6%	5803	3840

From the table, we can conclude that failures and recoveries are nearly symmetric, with only 0.2% drop in accuracy at a 32.4% token reduction. To look deep into the failure reasons, we define 4 of them:

- overtruncation: Astra truncates reasoning at an early stage, while extending the reasoning by 25% percent from the truncation point still produces the wrong answer.
- marginal miss: Astra truncates reasoning at early stage, while extending the reasoning by 25% percent from the truncation point reaches the real reasoning saturation point and generates the correct answer. (Excluding type 3)
- ong-reasoning needed: Astra truncates reasoning after 80%-percent position of baseline output, but still produces wrong answer.
- Unexpected failures, including missing answers, dead loops, etc.

Table 16: Breakdown of failure cases by category. The **Mean truncation position** is the result compared with baseline.

Category	Share of failures	Mean truncation position
1. Over-truncation (still wrong at $+25\%$)	55.4%	43%
2. Narrow miss (correct at $+25\%$)	15.4%	30%
3. Long reasoning genuinely required	28.7%	87%
4. Unexpected failure	0.5%	65%

Of all these failures, category 1 can be interpreted as the minor ineffectiveness of Astra detecting overthinking, which accounts for 3.6% in all cases; category 2 and 3 can be linked with parameter robustness problem, which accounts for 3.5% in all cases; category 4 is rare enough to be overlooked.

Summary. By analyzing the failure cases, we can conclude that compared with 6.9% successful cases saved from misleading overthinking that generates incorrect answer, an 3.6% inner ineffectiveness in signal and 3.5% miss from parameter robustness as well as the unavoidable error from online identification algorithm is acceptable, compared with token-level signals.