Efficient Reasoning Through Suppression of Self-Affirmation Reflections in Large Reasoning Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

While recent advances in large reasoning models have demonstrated remarkable performance, efficient reasoning remains critical due to the rapid growth of output length. Existing optimization approaches highlights a tendency toward "overthinking", yet lack fine-grained analysis. In this work, we focus on **Self-Affirmation Reflections**: redundant reflective steps that affirm prior content and often occurs after the already correct reasoning steps. Observations of both original and optimized reasoning models reveal pervasive self-affirmation reflections. Notably, these reflections sometimes lead to longer outputs in optimized models than their original counterparts. Through detailed analysis, we uncover an intriguing pattern: compared to other reflections, the leading words (i.e., the first word of sentences) in self-affirmation reflections exhibit a distinct probability bias. Motivated by this insight, we can locate self-affirmation reflections and conduct a train-free experiment demonstrating that suppressing self-affirmation reflections reduces output length without degrading accuracy across multiple models (R1-Distill-Models, QwQ-32B, and Qwen3-32B). Furthermore, we also improve current train-based method by explicitly suppressing such reflections. In our experiments, we achieve length compression of 18.7% in train-free settings and 50.2% in train-based settings for R1-Distill-Qwen-1.5B. Moreover, our improvements are simple yet practical and can be directly applied to existing inference frameworks, such as vLLM. We believe that our findings will provide community insights for achieving more precise length compression and step-level efficient reasoning.

1 Introduction

Recent advancements [23, 43, 2, 17] in large language models have delivered remarkable performance across various tasks [10, 4], yet they still struggle with complex reasoning tasks demanding advanced mathematical [15, 22] and formal logical deduction [35] abilities. Works [18, 17, 12, 38, 36] like Deepseek-R1 [12] have driven progress in reasoning models via reinforcement learning. However, current large reasoning models are troubled by high token usage and computational costs due to generating lengthy reasoning chains. Researchers have been delving into different technical strategies to enhance reasoning efficiency, like refining latent representations [14, 33, 7] and combining small models [27, 11]. In order to directly optimize the model itself, several approaches have emerged as promising methods, using supervised fine-tuning (SFT) [41, 13, 25], direct preference optimization (DPO) [32], and reinforcement learning (RL) [3, 24, 3, 1, 44].

Although these methods have yielded impressive results, such as producing more semantically efficient or direct solutions, we observed that redundant reflection persists in the optimized models [3, 24]. Rather than attributing this phenomenon to the more general "overthink" issue (defined in

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

work [34] as the generation of excessively detailed or unnecessary lengthy reasoning steps by large reasoning models during inference, which undermines reasoning efficiency), we conducted a more detailed examination of these examples (for more details, refer to ***Section 3***). As illustrated in Figure 1, we have identified a common phenomenon: affirming prior content typically occurs following correct steps, which we refer to as Self-Affirmation Reflection. In some cases, this superfluous reflection even leads to longer answers for certain problems in the optimized models.

Since this phenomenon is also widespread in the optimized models [3, 24], such reflection essentially originates from the base models. Therefore, we investigate scenarios where original reasoning models frequently execute self-affirmation reflection. Empirical analysis reveals a notable phenomenon: During repeated rollout processes at reflection points, models do not consistently engage in reflection behavior, as illustrated in the yellow box of Figure 1. In fact, after generating correct intermediate steps, models often exhibit ambiguity in transitioning between problem-solving steps and reflection initiation. Since prior works [31, 26] have established that certain keywords, termed leading words in this paper, significantly influence a model's decision to trigger the reflection mechanism. To characterize the self-affirmation reflection type, we conduct statistical analysis comparing its leading word distribution to those of other reflection categories. As shown in ***Section 4***, self-affirmation reflection exhibits statistically significant divergence in leading word distribution patterns.

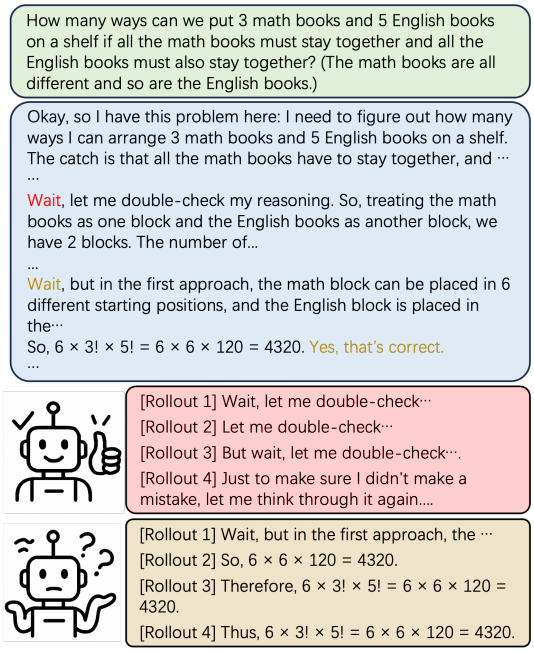

Figure 1: **Self-Affirmation Reflection** phenomenon. We investigate the characteristics when the reflection occurs by performing multiple rollouts at reflection points (marked by the colored "Wait"). As illustrated in the figure below, the model sometimes consistently performs a reflection (marked in pink box), while at other times it takes different actions (marked in yellow box). Since the latter reflections generally serve to reaffirm previous content, we refer to this behavior as the self-affirmation reflection.

Building on this insight, we identify conditions under which self-affirmation reflections occur during model reasoning and present a straightforward yet effective method that explicitly suppresses rethinking confidence at these crucial confusion points. Notably, our approach entails no extra cost and is extremely simple to implement. It can be seamlessly integrated into current inference frameworks such as vLLM [19]. Under a train-free experimental setup, while maintaining nearly the same performance level, we have achieved length reductions of 18.7%, 14.3%, 11.1%, 9.1%, and 8.4% for R1-Distill-Qwen-1.5B/7B/32B, QwQ-32B, and Qwen3-32B respectively, and in some cases, even better performance is obtained. Furthermore, we apply our method to a representative train-based approach [3], which generates responses of varying lengths by model self-sampling. Correct responses are prioritized, with shorter lengths receiving higher rewards. Our method complements this strategy by explicitly suppressing self-affirming reflections within positive samples and achieve significantly shorter outputs with competitive performance. Extensive experiments in ***Section 5*** across different settings validate the feasibility and significance of the findings in this paper.

Our work offers three key benefits: (1) We provides the first focused analysis of the Self-Affirmation Reflection phenomenon and **provides actionable insights for improving reasoning models**. (2) **We present a simple yet practical intervention strategy**: suppressing crucial leading tokens. This directly lessens Self-Affirmation Reflection, achieving efficient output compression without modifying the model architecture or training process. (3) Extensive experiments across diverse models and datasets in **both train-free and train-based settings** demonstrate that mitigating Self-Affirmation Reflection can effectively reduce output length while preserving or even enhancing performance.

2 Related Work

Recent surveys [29, 39, 34] have summarized progress in efficient reasoning. In this section, we highlight representative methods to provide a concise overview of key developments in this area.

The first category of methods [13, 42, 20] aim to obtain some information from the input to estimate the cost in advance. Some methods [13, 42, 20] choose to estimate the reasoning cost. For example, TokenBudget [13] instructs models to estimate the token count required to solve a problem upfront, compelling them to generate concise responses within a predefined budget. Other methods [27, 5, 9, 8] estimate the model size. Approaches like RouterLLM [27] and Sot [5] pre-train classifiers to route problems to the most suitable LLM based on task complexity. In contrast, router-free methods such as Self-ReF [9] and Seek-Router [8] predict the necessity of invoking additional LLMs by analyzing internal uncertainty scores. However, how to accurately determine the optimal cost is still a problem.

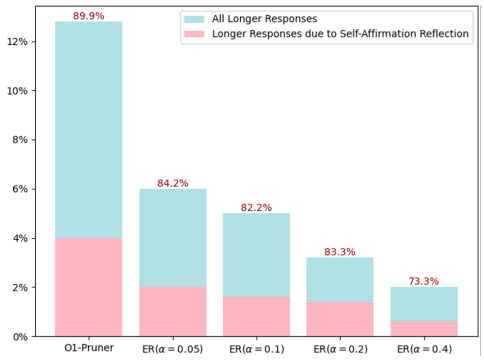

Figure 2: The ratio of questions with longer average response lengths in MATH500 [22] when testing released checkpoints from EfficientReasoning [3] and O1-Pruner [24](marked as blue bar). Through manual inspection, we additionally show the proportion of responses that are longer due to output self-affirmation reflections (marked as pink bar). Moreover, the accuracy on MATH500 is labeled at the top of each bar.

Recent advances in the second category of methods have focused on refining the output of optimization models. As demonstrated in [28], simply adding intermediate "thinking" tokens, or even nonsensical fillers like ".....", can lead to satisfactory performance. Subsequent works, such as those in [14, 33, 7], have explored treating the last layer hidden states of LLMs as "continuous thinking" to replace traditional discrete markers. These approaches optimize latent representations through training. However, such methods may render parts of the thought chain invisible, potentially introducing security risks. In addition to optimizing the output sequence, there are also methods to choose to switch the model during output, such as RSD [21] integrating multiple LLMs to achieve dynamic reasoning. These methods demand precise determination of when to switch models and which model to switch to.

The third category of methods centers on fine-tuning LLM itself to directly address redundancy in the output of the original model. Generally, these methods [24, 3, 3, 1, 44] first obtain responses of varying lengths via model self-sampling. Then, they design a length-based reward conditioned on correctness, with shorter correct answers receiving higher rewards. However, balancing accuracy and output length remains a significant challenge. Although these methods reduce average response length of the entire datasets, they do not guarantee shorter outputs across all instances. Specifically, the figure 2 highlights that at the instance level, some outputs (depicted in blue) remain longer than those from the original model. Our train-based experiments in Section 5.2 is closely related to these methods but differs in focus. Rather than focusing on reward function design, our objective is to directly optimize sampling outcomes, thereby refining the quality of the constructed data pairs and achieving more succinct responses.

3 Observation

Our observation begins with experiments evaluating two representative open-source works [24, 3]. We tested R1-Distill-Qwen-1.5B [12] and QwQ-Preview [37] on the MATH500 dataset [22] alongside their compressed counterparts (EfficientReasoning [3] and O1-Pruner [24]). For each question, three responses were sampled, with average response length recorded.

As shown in Figure 2, results exhibit a counterintuitive trend: certain ratio of instance-level questions elicit longer responses, yet most of these extended responses remain correct (e.g., 91.6% in QwQ-Preview settings). This pattern persists across varying compression levels. For instance, the parameter

α in EfficientReasoning[3] balances brevity and accuracy. Larger α values shorten outputs at the cost of reduced accuracy. As shown in Figure 2, even at mild compression ($\alpha = 0.05$), 6% of questions generated longer responses. Aggressive compression ($\alpha = 0.4$) still failed to resolve this behavior, though the trade-off led to a significant drop in accuracy.

To gain deeper insights into the optimized models, we conducted a manual examination and analyzed their qualitative behaviors, as shown in Figure 3. Two key characteristics consistently emerge: (1) the model tends to produce more concise steps during the problem-solving phase. For instance, the brown text in Figure 3 illustrates how the optimized model succinctly summarizes the problem background. (2) certain questions have been solved using more direct approaches. This arises because, during the self-sampling process, the model generates multiple potential problem-solving methods. By favoring shorter solution ideas, the model discards superfluous strategies.

However, we also observe that the model frequently engages in prolonged reflections even after executing correct reasoning steps or arriving at the correct answer, resulting in excessively lengthy outputs. Figure 3 illustrates this issue: compared to the baseline R1-Distill-Qwen-1.5B model [12], the optimized EfficientReasoning model ($\alpha = 0.05$) [3] achieves the correct solution more rapidly but produces lengthier responses due to increased self-reflections. These reflections often show a tendency to affirm the previous content. Quantitative results in Figure 2 further confirm that this pattern is prevalent across current efficient reasoning models [3, 24]. Overall, these findings suggest that naively optimizing for solution-level simplicity is not enough to effectively compress the model's outputs. This indicates that existing frameworks may still have opportunities to enhance output simplicity by suppressing step-level self-affirmation reflections.

Notably, while this phenomenon was widely summarized as an overthink problem in previous works [29, 39, 34], its persistence in optimized reasoning models underscores the need for targeted solutions. Therefore, we formally define this behavior as self-affirmation reflection, as introduced in Section 1, and focus on it in this paper. To address this issue fundamentally, we analyze how to locate and suppress self-affirmation reflection in the original model, as detailed in the subsequent Section 4.

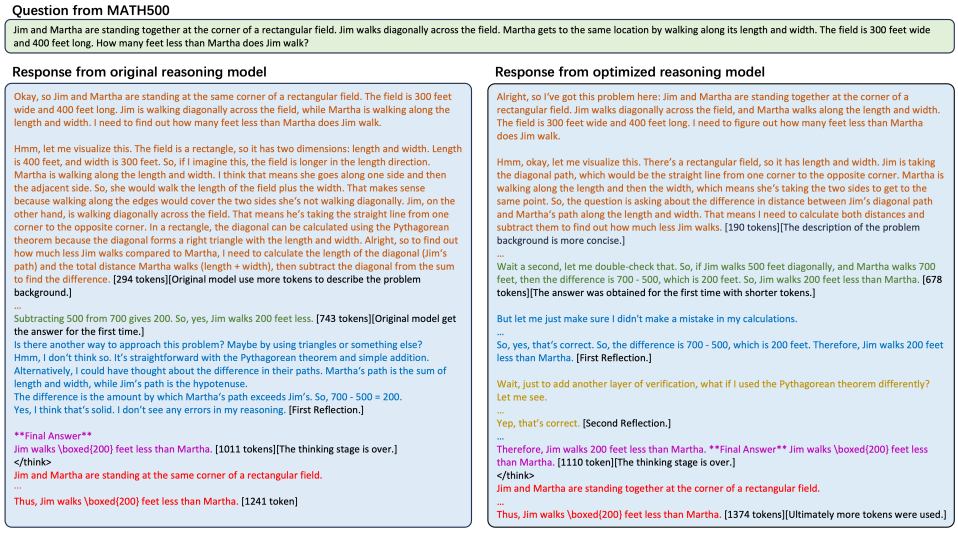

Figure 3: Qualitative analysis of responses from original and optimized R1-Distill-Qwen-1.5B. The colored text represents the original response contents of the two models, and the black text indicates our description of the progress of solving the problem. Provide the number of tokens that have been used when necessary.

4 Analysis

4.1 Locate Self-Affirmation Reflections

To investigate the self-affirmation reflection, we first randomly sampled 500 training instances from the MATH dataset. For each problem, we generated a single solution using R1-Distill-1.5B [12]. The generated reasoning steps were split using the "$\backslash n \backslash n$" delimiter.

Then, we utilized the Qwen2.5-72B-Instruct [43] to classify each step as either a reasoning reflection or non-reflective reasoning. To improve the accuracy, we further combine the previous method [6] to assist in judgment. For steps identified as reasoning reflections, we invoked the Qwen2.5-72B-Instruct again to determine whether the reflection affirmed the preceding reasoning, i.e., whether it constituted a self-affirmation reflection. The specific prompts used for classification are detailed in Appendix A.3. To validate this automated classification, we manually annotated self-affirmation reflections in all responses of 20 problems to form a test set. Evaluation revealed the Qwen2.5-72B-Instruct achieved an accuracy of 80.6%. This is primarily because while most self-affirmation reflections contain explicit signal words such as "that's correct" in Figure 3, some self-affirmation reflections merely repeat previous conclusions to indicate affirmation of prior content, making these more challenging to detect.

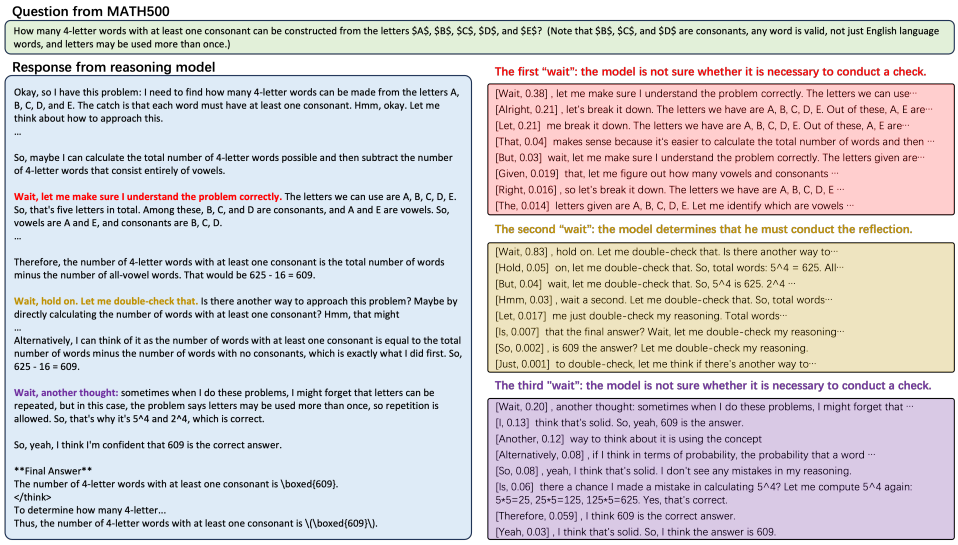

Figure 4: A qualitative analysis result of self-affirmation reflections and other reflections. At the beginning of each reflective sentence, we save the top 8 high-probability words and continue rollout based on these words. We also marked the first word and the corresponding confidence level. For more details, please refer to Section 4.1.

Based on the results of classification, our analysis proceeded from two complementary perspectives to identify patterns in self-affirmation reflection occurrence. **First**, prior researches [31, 26] has established that language patterns are critical in determining whether models activate specific reflection mechanisms during decision-making. Therefore, we sampled 20 self-affirmation reflections and other reflection types, then performed multiple rollout iterations from their contextual positions. We hope to explore whether models can elicit distinct behavioral responses by producing diverse leading words. As shown in Figure 4, we found that self-affirmation reflections exhibit significantly greater diversity in behavior-guiding continuation patterns compared to other reflection types.

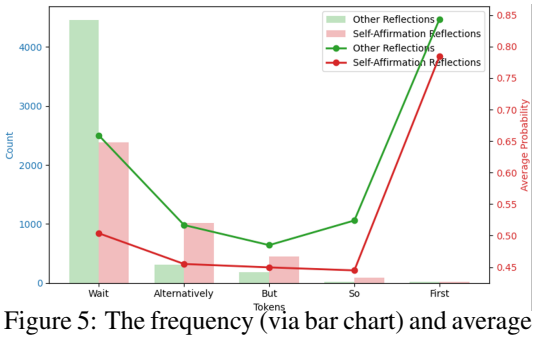

Figure 5: The frequency (via bar chart) and average confidence scores (via line plot) of the first words in all reflective sentences. The top five words are presented in our results. We found that when the model engages in self-affirmation reflections, it exhibits a certain level of uncertainty compared to other reflections.

Specifically, during the initial reflection (Self-Affirmation Reflection), the model attempts to verify its understanding of the problem context. Examining the top 8 predicted word continuations, we observe the model's indecision between further reflection and progressing toward a solution, as evidenced by suggestions such as "break it down". In the second reflection (Other Reflection), the trajectories of the model show a high degree of consistency. Here, the top 8 predicted words uniformly advocate for continued reflection, such as "double-check". The final reflection (Self-Affirmation Reflection) occurs after the model has

already provided the correct answer. Similar to the first reflection, we observe the model confronting uncertainty, contemplating whether to terminate the task or initiate another round of reflection.

Second, we conducted an additional analysis of the quantity and probability of the leading words across all reflections. It is worth noting that, for simplicity, we only selected the first word as leading words, but a typical leading word is sometimes not necessarily a single word. For example, when sentences start with "But", we found that the next word is "wait" in 38% of self-affirmation reflections. In this case, "But wait" is more suitable to be used as a unique leading word. During statistical processing, we excluded repetitive tail-end reflections from model outputs. Because self-affirmation reflections sometimes exhibited recurring patterns where incremental reinforcement of leading-word probabilities led to closed-loop generation. These loops could skew statistical accuracy. While this phenomenon warrants further study, it falls outside the scope of this work. For an illustrative example of such looping behavior and comprehensive results including looped sentences, refer to Appendix A.4.

The final result is shown in Figure 5. We present the top 5 high-frequency words in self-affirmation reflection. To our surprise, the confidence level of leading words in self-affirmation reflection appears to be lower than that in other Reflections. This finding echoes the results from the Figure 4, indicating that when the model is uncertain about its actions, the output leading words also lack confidence.

4.2 Suppress Self-Affirmation Reflections

Up to now, our analysis highlights a critical insight: leading tokens in self-affirmation reflections exhibit significantly lower generation probabilities compared to other reflection types. Our objective is to suppress self-affirmation reflections in order to achieve shorter outputs while maintaining performance. Consequently, we propose a straightforward intervention: when the model assigns a low probability to generating leading words, we set their probability to zero to suppress self-affirmation reflections. Given the statistical significance of the "Wait" token, we first focus on it in this paper.

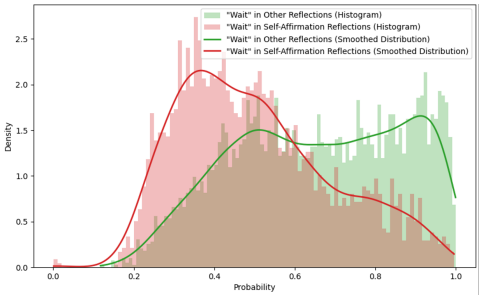

Figure 6: The distribution of "Wait" in the two types of reflections.

To validate our method, we analyze the probability density of the "Wait" token across two reflection types in Figure 6. Though their distributions exhibit partial overlap, they remain distinguishable through thresholding. We acknowledge that suppressing low-probability "Wait" instances may inadvertently filter out other reflections. However, empirical results show that even when such instances are suppressed, other high-probability tokens can still facilitate emergence of necessary reflections via compensatory mechanisms, as exemplified by the second reflection in Figure 4. We also recognize the limitation of exclusively targeting the "Wait" token, which relates to balancing interventions across multiple leading words. In practical results, we found that since "wait" frequently follows "But", suppressing this token also partially mitigates self-affirmation reflections. Consequently, we intervene on both "Wait" and "wait" (collectively referred to as the "wait" tokens hereafter). **More discussions of other interfered tokens appear in Appendix A.5.**

Our results also demonstrate that suppressing low-probability "wait" tokens maintains performance integrity and produces shorter outputs. In train-free settings (e.g., R1-Distill-Qwen-1.5B in Table 1), output lengths shorten while performance improves when thresholds decrease below 0.5. For training-based scenarios, ablation studies identify optimal performance at a threshold of 0.3. These findings curiously align with the probability distributions in Figure 6, suggesting that thresholds ≤ 0.3 effectively suppress self-affirmation reflections while having minimal impact on other reflection types.

5 Experiments

We first assess the impact of removing Self-Affirmation Reflections in Section 5.1. Subsequently, in Section 5.2, we integrate a representative train-based approach to verify whether explicitly suppressing self-affirmation reflections can lead to improved results.

5.1 Train-free Experiments

5.1.1 Settings

We hypothesize that removing low-probability self-affirmation reflection will not degrade model performance. To test this hypothesis, we conducted experiments across several prominent reasoning models: R1-Distill-Qwen-1.5B/7B/32B [12], QwQ-32B [37] and Qwen3-32B [38]. We also additionally compared Underthink [40], which intervened with logits within a certain window length. For more details and our discussion on this work, please refer to Appendix A.2. Similar to the previous approaches [6, 3, 24, 32], our experiments are conducted on four datasets: (1) MATH500 [22], comprising 500 problems across seven mathematical domains (algebra, geometry, number theory, etc.) that challenge both humans and LLMs; (2) AIME24, featuring 2024 American Invitational Mathematics Examination problems to test advanced mathematical competition problem-solving skills; (3) AMC23 [15], with problems from the 2023 American Mathematics Competitions covering secondary school mathematics; and (4) GSM8K [10], containing 8.5K linguistically diverse elementary school math problems to evaluate simple arithmetic reasoning. **Regarding the results of the out-of-domain dataset, please refer to the Appendix A.6.**

For AIME24 and AMC23, we sampled eight responses per question. For MATH500 and GSM8K, we collected one response per problem. The maximum allowed output length was 32k tokens. In order to evaluate Qwen3-32B [38], we utilized the latest vLLM implementation [19] in zero-shot inference settings. We report the average accuracy (Acc) and average token count (LEN) per response across all datasets. All experiments were conducted on NVIDIA L20 GPUs.

5.1.2 Results

As shown in Table 1, we evaluated the impact of different thresholds on performance. Our results indicate that an appropriate threshold can effectively reduce output length without compromising performance, and in some cases, even enhances performance with shorter outputs, which is consistent with our previous analysis in Figure 6. However, as the threshold increases, high-probability "wait" tokens are also filtered out, leading to a noticeable decline in performance due to the removal of some necessary reflections. Despite its simple implementation, our method achieves a significant reduction in output length with almost no performance loss. Specifically, it reduces average length by 18.7%, 14.3%, 11.1%, 9.1% and 8.4% for R1-Distill-Qwen-1.5B (Threshold-0.9), R1-Distill-Qwen-7B (Threshold-0.7), R1-Distill-Qwen-32B (Threshold-0.7), QwQ-32B (Threshold 0.7) and Qwen3-32B (Threshold 0.9), respectively. Moreover, we were surprised to find that the compression ratio adapts dynamically across different datasets. For instance, AIME24 demonstrates a lower compression ratio than MATH500. This aligns with the intuition that AIME24, being more challenging, inherently possesses less compressible structure compared to MATH500.

5.2 Train-based Experiments

5.2.1 Settings

In this section, we incorporate our approach into a representative training-based method. We utilize EfficientReasoning [3], which assigns positive rewards to correct responses and zero rewards to incorrect ones, with longer correct responses receiving smaller rewards. A penalty intensity coefficient α is included to fine-tune the compressive intensity. Owing to resource constraints inherent in training reinforcement learning (RL) models, we restrict our evaluation to the Deepseek-R1-Distill-Qwen-1.5B checkpoint [12]. All other experimental configurations are inherited from the EfficientReasoning [3], with rollout budgets specifically set at 10 for AIME24, 3 for MATH500, and 1 for GSM8K. Our intervention is confined to the rollout phase during training. For testing, we rigorously adhere to the original implementation details: environments are constructed using the provided Dockerfile, and inference is conducted with the officially released checkpoints when evaluating EfficientReasoning [3].[1] While the performance on specific datasets showed minor

[1] Owing to variations in experimental settings, R1-Distill-Qwen-1.5B exhibits slight performance differences between train-free and train-based experiments. This discrepancy aligns with findings reported in prior studies [16]. We present our results as measured truthfully. Notably, when evaluating the average performance across the three datasets, the outcomes from both experimental environments demonstrate remarkable consistency.

discrepancies, the average performance across the three datasets aligned closely with the results reported in the EfficientReasoning [3].

During the rollout phase, we filter out all "wait" tokens that have confidence scores below 0.3. Reinforcement learning (RL) requires generating samples of varying lengths to strengthen shorter responses. However, indiscriminate intervention across all samples may compromise the learning of negative samples. Negative samples do not need to be shortened in length by suppressing self-affirmation reflections. Therefore, our objective is to suppress self-affirmation reflections in positive samples only. Yet, determining in advance which responses are correct and should be intervened is challenging. Therefore, we adopt an approximate method and intervene with a 25% probability each time.

Table 1: The influence of different thresholds on the original model. The degradation and improvement of performance are marked with Red and Green.

Models	AIME24		AMC23		GSM8K		MATH500		Average	Average
	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓
R1-Distill-1.5B	31.7	16415.7	69.7	10370.1	75.7	676.7	82.7	5488.0	64.9	8237.6
Underthink	27.5	16866.6	71.3	9907.1	75.8	639.5	82.0	5641.9	64.1$_{-0.8\%}$	8263.8$_{+0.3\%}$
Threshold 0.1	32.5	15468.9	71.9	9412.5	75.3	734.8	84.6	5268.1	66.1$_{+1.2\%}$	7721.1$_{-6.3\%}$
Threshold 0.3	27.5	16143.3	72.5	9101.5	77.6	667.4	83.0	4893.7	65.2$_{+0.3\%}$	7701.5$_{-6.5\%}$
Threshold 0.5	27.9	14153.8	65.6	8370.6	76.0	587.0	82.0	4924.9	62.9$_{-2.0\%}$	7009.1$_{-14.9\%}$
Threshold 0.7	26.7	14989.6	69.7	7679.0	75.3	635.6	80.6	4549.0	63.1$_{-1.8\%}$	6963.3$_{-15.5\%}$
Threshold 0.9	28.3	14121.2	71.6	7765.5	75.2	698.9	81.6	4191.3	64.2$_{-0.7\%}$	6694.2$_{-18.7\%}$
R1-Distill-7B	50.8	13781.8	89.7	6258.3	92.1	1533.2	92.6	4096.8	81.3	6417.5
Underthink	53.4	12964.6	89.7	6219.8	92.8	1405.3	93.2	4101.4	82.3$_{+1.0\%}$	6172.8$_{-3.8\%}$
Threshold 0.1	52.1	14111.2	90.3	6205.8	91.1	1504.4	91.4	4319.6	81.2$_{-0.1\%}$	6535.3$_{+1.8\%}$
Threshold 0.3	57.1	12852.2	89.1	6471.4	91.8	1373.6	93.2	3760.5	82.8$_{+1.5\%}$	6114.4$_{-4.7\%}$
Threshold 0.5	50.8	12652.2	88.4	6009.2	91.7	1397.4	92.4	3747.2	80.8$_{-0.5\%}$	5951.5$_{-7.3\%}$
Threshold 0.7	53.3	11548.3	89.7	5426.9	92.7	1291.3	92.4	3720.1	82.0$_{+0.7\%}$	5496.6$_{-14.3\%}$
Threshold 0.9	47.1	11477.3	87.8	5145.2	92.2	1261.5	93.6	3365.2	80.2$_{-1.1\%}$	5312.3$_{-17.2\%}$
R1-Distill-32B	70.4	11363.4	95.0	5679.6	93.6	644.4	93.8	3640.9	88.2	5332.1
Underthink	70.9	10744.3	95.3	5849.0	94.0	633.0	94.8	3498.7	88.7$_{+0.5\%}$	5181.3$_{-2.8\%}$
Threshold 0.1	70.4	10955.9	96.6	5783.8	93.6	663.4	93.6	3653.4	88.6$_{+0.4\%}$	5264.2$_{-1.3\%}$
Threshold 0.3	72.1	11170.5	95.3	5929.7	94.0	629.4	94.4	3456.5	89.0$_{+0.7\%}$	5296.5$_{-0.6\%}$
Threshold 0.5	68.8	10350.5	94.7	5788.8	94.1	628.4	93.6	3239.2	87.8$_{-0.4\%}$	5001.7$_{-6.2\%}$
Threshold 0.7	70.0	9747.5	96.9	5232.1	93.5	625.5	92.6	3339.2	88.2$_{+0.0\%}$	4736.1$_{-11.1\%}$
Threshold 0.9	66.3	10193.4	94.4	4774.4	94.2	593.4	93.8	3163.9	87.2$_{-1.0\%}$	4681.3$_{-12.2\%}$
QwQ-32B	78.3	13709.0	98.8	7591.2	96.5	1646.4	95.4	4267.6	92.2	6803.5
Underthink	77.9	13579.9	97.5	7798.2	96.4	1540.5	96.0	4319.5	92.0$_{-0.2\%}$	6809.5$_{+0.1\%}$
Threshold 0.1	79.6	13239.4	98.8	7683.3	96.2	1617.6	95.2	4405.7	92.4$_{+0.2\%}$	6736.5$_{-1.0\%}$
Threshold 0.3	77.5	13296.3	99.1	7392.8	96.4	1575.3	95.8	4241.1	92.2$_{+0.0\%}$	6626.4$_{-2.6\%}$
Threshold 0.5	77.1	13241.8	95.6	7490.8	96.5	1546.4	95.6	4225.9	91.2$_{-1.0\%}$	6626.2$_{-2.6\%}$
Threshold 0.7	77.1	12547.0	98.4	6693.0	96.5	1491.3	96.2	4018.4	92.1$_{-0.1\%}$	6187.4$_{-9.1\%}$
Threshold 0.9	76.7	12613.3	97.5	6652.5	96.4	1456.9	94.4	3883.9	91.2$_{-1.0\%}$	6151.6$_{-9.6\%}$
Qwen3-32B	80.4	13369.1	96.9	7092.2	96.3	1704.2	96.4	4570.0	92.5	6683.9
Underthink	79.6	12579.5	97.2	7082.3	96.1	1604.9	96.0	4667.7	92.2$_{-0.3\%}$	6483.6$_{-3.0\%}$
Threshold 0.1	80.4	13054.4	97.8	7396.5	95.5	1723.5	95.6	4665.9	92.3$_{-0.2\%}$	6710.1$_{+0.4\%}$
Threshold 0.3	82.1	12601.8	95.6	7131.1	96.4	1627.6	95.2	4653.4	92.3$_{-0.2\%}$	6503.5$_{-2.7\%}$
Threshold 0.5	81.7	12571.7	97.2	6898.0	96.2	1607.7	95.0	4535.1	92.5$_{+0.0\%}$	6403.1$_{-4.2\%}$
Threshold 0.7	81.2	12751.7	98.4	6531.2	96.1	1601.2	95.8	4316.8	92.9$_{+0.4\%}$	6300.2$_{-5.7\%}$
Threshold 0.9	78.8	12152.2	97.2	6510.8	96.1	1561.5	95.6	4277.5	91.9$_{-0.6\%}$	6125.5$_{-8.4\%}$

5.2.2 Results

The main results as shown in Table 2. As anticipated, our method achieved substantially shorter average lengths while maintaining comparable performance. It should be noted that text length and accuracy are inherently conflicting objectives. The results from EfficientReasoning [3] demonstrate that the stronger the emphasis on shorter text, the more severe the drop in accuracy. The model pursues shorter positive responses at the same intensity, which is approximately equivalent to using a stronger compression intensity. This explains the slight dip in accuracy. However, as shown in Table 2, the accuracy decrease is minimal and still surpasses the baseline (R1-Distill-Qwen-1.5B). Moreover, at the same text length, our method delivers superior performance.

5.2.3 Ablation Study

In this section, we compared the different thresholds for filtering "wait" tokens and the impact of the probability of intervention responses. All ablation experiments were conducted based on $\alpha = 0.05$.

Table 2: The results of EfficientReasoning [3] combines our method. ER is the abbreviation of EfficientReasoning [3]. * denote the results from EfficientReasoning [3] paper. The degradation and improvement of performance are marked with Red and Green.

Models	AIME24		MATH500		GSM8K		Average Acc↑	Average LEN↓
	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓		
R1-Distill-1.5B	28.7	15651.0	85.1	5274.0	75.9	709.0	63.2	7211.3
SFT*	24.3	13805.5	77.8	3701.2	77.6	508.2	$59.9_{-3.3\%}$	$6004.9_{-16.7\%}$
DPO*	28.7	15145.8	83.3	4478.6	76.3	831	$62.8_{-0.4\%}$	$6818.4_{-5.4\%}$
ER($\alpha = 0.05$)	30.3	9452.9	84.2	2648.3	85.2	776.9	$66.6_{+3.4\%}$	$4292.7_{-40.5\%}$
ER($\alpha = 0.1$)	30.7	12071.2	82.2	2652.0	82.1	628.0	$65.0_{+1.8\%}$	$5117.1_{-29.0\%}$
ER($\alpha = 0.2$)	29.0	10043.7	83.3	2378.6	80.3	297.3	$64.2_{+1.0\%}$	$4239.9_{-41.2\%}$
ER($\alpha = 0.4$)	26.3	8767.6	73.3	1869.8	68.1	138.6	$55.9_{-7.3\%}$	$3592.0_{-50.2\%}$
Ours($\alpha = 0.05$)	27.7	7712.6	83.6	2366.2	85.7	760.1	$65.7_{+2.5\%}$	$3613.0_{-49.9\%}$
Ours($\alpha = 0.1$)	30.0	8312.5	81.1	2103.8	79.8	368.3	$63.6_{+0.4\%}$	$3594.8_{-50.2\%}$
Ours($\alpha = 0.2$)	26.3	7807.4	80.1	1798.0	80.9	341.9	$62.4_{-0.8\%}$	$3315.8_{-54.0\%}$
Ours($\alpha = 0.4$)	29.1	7268.0	75.9	1614.4	78.2	184.8	$61.1_{-2.1\%}$	$3022.4_{-58.0\%}$

Table 3: Impact of different thresholds for filtering "wait" tokens. The baseline represent EfficientReasoning [3]($\alpha = 0.05$).

Models	AIME24		MATH500		GSM8K		Average Acc↑	Average LEN↓
	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓		
R1-Distill-1.5B	28.7	15651.0	85.1	5274.0	75.9	709.0	63.2	7211.3
Baseline	30.3	9452.9	84.2	2648.3	85.2	776.9	66.6	4292.7
Threshold-0.1	29.7	12249.9	80.8	3031.8	81.3	538.5	63.9	5273.4
Threshold-0.3	27.3	7918.9	82.5	2269.1	85.8	703.4	65.2	3630.5
Threshold-0.5	29.0	9457.8	80.5	2241.6	82.2	478.8	63.9	4059.4
Threshold-0.7	30.3	8695.8	80.7	2358.2	83.4	632.0	64.8	3895.3
Threshold-0.9	30.3	9609.1	83.5	2883.2	82.5	676.9	65.5	4389.8

Impact of different threshold: In this experiment, the probability of intervention responses is fixed on 50%. As shown in Table 3, when the threshold is below 0.9, the model's performance initially increases but then decreases. This indicates that excessive filtering of self-affirmation reflections can significantly degrade performance. However, when the threshold is set to 0.9, performance improves due to increased length. Overall, there is no linear relationship between the threshold and length. We attribute this to the uncontrollable attribute of the RL training process.

Table 4: Impact of different the probability of intervention responses. The baseline represent EfficientReasoning [3]($\alpha = 0.05$).

Models	AIME24		MATH500		GSM8K		Average Acc↑	Average LEN↓
	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓		
R1-Distill-1.5B	28.7	15651.0	85.1	5274.0	75.9	709.0	63.2	7211.3
Baseline	30.3	9452.9	84.2	2648.3	85.2	776.9	66.6	4292.7
Probability-25%	27.7	7712.6	83.6	2366.2	85.7	760.1	65.7	3613.0
Probability-50%	27.3	7918.9	82.5	2269.1	85.8	703.4	65.2	3630.5
Probability-75%	29.3	12299.4	82.4	3114.6	82.8	610.8	64.8	5341.6
Probability-100%	24.0	8136.7	81.5	2144.7	81.6	497.6	62.3	3593.1

Impact of different probability: In this experiment, the thresholds for filtering "wait" tokens fixed on 0.3. As shown in Table 4, as the probability increases, performance exhibits a gradual decline. Overly aggressive intervention in the rollout process may generate shorter negative samples, thereby undermining the model's adversarial learning capabilities.

6 Conclusion

Our paper delves into the distinctive features of self-affirmation reflections in the distribution of leading words and their correlation with confidence. Leveraging these insights, we introduce a strategy to suppress critical leading words. Experimental results demonstrate that this method effectively reduces output length in both training-free and training-based scenarios while sustaining or even elevating model performance. While we acknowledge certain limitations (see Appendix A.1), we aim for this study to offer a fresh viewpoint on achieving more precise length compression at step-level efficient reasoning and to improve the efficiency of large reasoning models.

References

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

[2] AI@Meta. Llama 3 model card. 2024.

[3] Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025.

[4] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[5] Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*, 2025.

[6] Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2504.07986*, 2025.

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

[8] Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui, Vladimir Braverman, and Xia Hu. Confident or seek stronger: Exploring uncertainty-based on-device llm routing from benchmarking to generalization. *arXiv preprint arXiv:2502.04428*, 2025.

[9] Yu-Neng Chuang, Helen Zhou, Prathusha Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, and Xia Hu. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 3, 2025.

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

[11] Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. Energy considerations of large language model inference and efficiency optimizations. *arXiv preprint arXiv:2504.17674*, 2025.

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

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

[14] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

[15] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[16] Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *arXiv preprint arXiv:2504.07086*, 2025.

[17] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

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

[19] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[20] Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*, 2025.

[21] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

[22] 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.

[23] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[24] Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

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

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

[27] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024.

[28] Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

[29] Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, et al. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*, 2025.

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

[31] Darsh J Shah, Peter Rushton, Somanshu Singla, Mohit Parmar, Kurt Smith, Yash Vanjani, Ashish Vaswani, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, et al. Rethinking reflection in pre-training. *arXiv preprint arXiv:2504.04022*, 2025.

[32] Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.

[33] Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

[34] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

[35] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

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

[37] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025.

[38] Qwen3 Team. Qwen3, 2025.

[39] Rui Wang, Hongru Wang, Boyang Xue, Jianhui Pang, Shudong Liu, Yi Chen, Jiahao Qiu, Derek Fai Wong, Heng Ji, and Kam-Fai Wong. Harnessing the reasoning economy: A survey of efficient reasoning for large language models. *arXiv preprint arXiv:2503.24377*, 2025.

[40] Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025.

[41] Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

[42] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

[43] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

A Technical Appendices and Supplementary Material

A.1 Limitations

Two core challenges persist. First, methods for systematically identifying candidate tokens for intervention remain underdeveloped. Second, developing principles to balance dependencies among multiple intervened words represents an open research question. Consequently, our intervention framework in this work focuses solely on the most salient specific tokens. Future research will explore strategies to generalize token intervention approaches, addressing both identification scalability and multi-word coordination.

We further recognize that the foundational mechanisms governing the model's reasoning process, as well as the emergence of redundant reflections during generation, are not yet fully understood. These knowledge gaps persist within the large reasoning model community. As the field matures, we anticipate opportunities to integrate insights from complementary work and conduct more in-depth analyses of these phenomena in subsequent studies.

A.2 Comparison with concurrent works

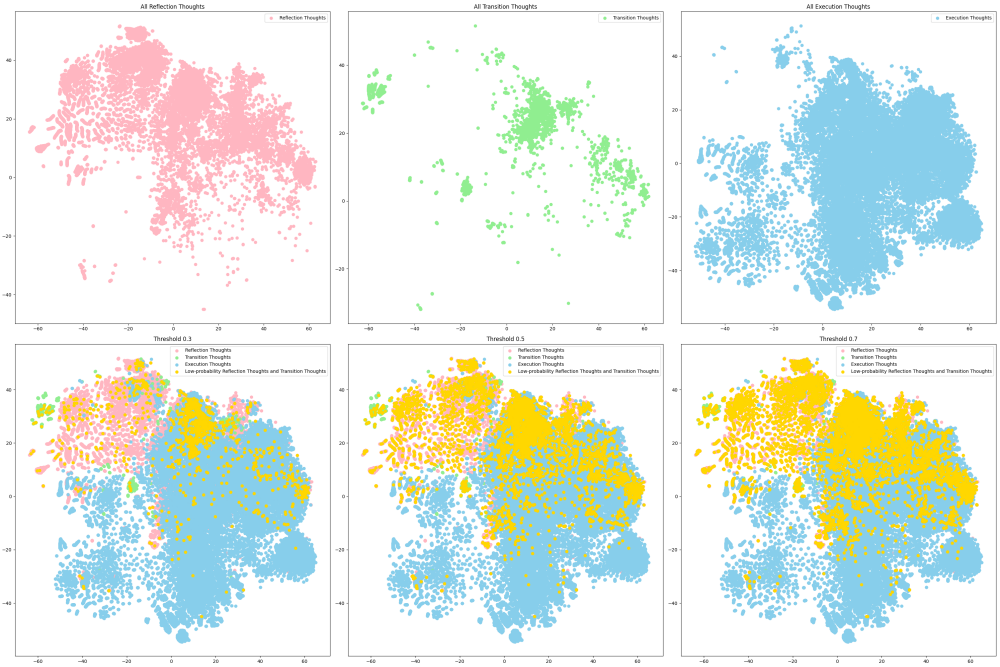

Figure 7: Results of t-SNE visualization of different reasoning thoughts in the SEAL [6]. We additionally visualized "reflection thoughts" and "transition thoughts" with confidence levels below the threshold.

We analyze two concurrent excellent works:

SEAL [6]: SEAL focuses on modifying activation values to convert reflective steps (termed "reflection thoughts" and "transition thoughts" in their paper) into executive steps (termed "execution thoughts"). For more details, please refer to the SEAL [6]. In contrast, our work identifies which reflective steps can be preserved and which should be removed. We extend SEAL's t-SNE visualization by highlighting reflective steps with low confidence (see Figure A.2), successfully distinguishing ambiguous reflective steps overlapping with executive steps that typically indicate model confusion.

It is important to note that SEAL only attempts a maximum output length of 10,000 tokens. This limitation arises because SEAL requires intervention on activation values during the "think" phase (between "<think>" and "</think>"), which presents significant implementation challenges, such as integration with vLLM. In contrast, our method achieves a maximum output length of 32,000 tokens, consistent with DeepSeek's testing configuration. This advancement is attributed to our approach's ability to directly integrate with vLLM.

Table 5: Comparison of Underthink [40] and our method. α and β represent the intensity and length of the intervention respectively. For more details, please refer to Underthink [40].

Models	AIME24		AMC23		GSM8K		MATH500		Average	Average
	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓
R1-Distill-1.5B	31.7	16415.7	69.7	10370.1	75.7	676.7	82.7	5488.0	64.9	8237.6
Threshold 0.1	32.5	15468.9	71.9	9412.5	75.3	734.8	84.6	5268.1	66.1	7721.1
Threshold 0.3	27.5	16143.3	72.5	9101.5	77.6	667.4	83.0	4893.7	65.2	7701.5
Threshold 0.5	27.9	14153.8	65.6	8370.6	76.0	587.0	82.0	4924.9	62.9	7009.1
Threshold 0.7	26.7	14989.6	69.7	7679.0	75.3	635.6	80.6	4549.0	63.1	6963.3
Threshold 0.9	28.3	14121.2	71.6	7765.5	75.2	698.9	81.6	4191.3	64.2	6694.2
$\alpha = 1, \beta = 200$	28.8	16689.0	68.1	10315.2	76.9	729.9	84.0	5375.0	64.4	8277.3
$\alpha = 1, \beta = 600$	25.4	17709.3	72.5	9776.8	75.4	736.6	86.6	5117.2	65.0	8335.0
$\alpha = 1, \beta = 1000$	31.3	16736.7	70.3	10313.4	75.8	601.9	84.8	5404.9	65.5	8264.2
$\alpha = 1, \beta = +\infty$	29.6	15053.0	71.9	8445.1	75.1	615.6	83.8	4804.9	65.1	7229.6
$\alpha = 3, \beta = 200$	27.9	16278.8	70.6	9100.6	76.1	748.7	84.2	5433.1	64.7	7890.3
$\alpha = 3, \beta = 600$	27.5	16866.7	71.3	9907.1	75.8	639.5	82.0	5641.9	64.1	8263.8
$\alpha = 3, \beta = 1000$	27.9	16497.6	68.4	10085.1	74.3	671.0	81.8	5533.4	63.1	8196.8
$\alpha = 3, \beta = +\infty$	28.8	13490.3	68.4	7167.3	75.2	602.5	83.4	4575.3	63.9	6458.8
$\alpha = 10, \beta = 200$	28.8	17063.6	67.8	9989.1	75.7	692.8	83.8	5612.0	64.0	8339.4
$\alpha = 10, \beta = 600$	30.4	15733.6	66.9	9917.3	75.0	635.9	82.4	5633.3	63.7	7980.0
$\alpha = 10, \beta = 1000$	32.9	16708.9	70.6	9649.2	75.5	566.9	83.2	5214.8	65.6	8035.0
$\alpha = 10, \beta = +\infty$	29.2	13938.6	71.9	7276.2	75.9	648.0	82.0	4405.8	64.7	6567.2

Underthink [40]: This paper aims to address the lack of path exploration capability in models by adjusting the logits of certain reflective leading words within a fixed window. This approach encourages models to explore current reasoning paths more thoroughly during the exploration process. Our focus differs. We aim to suppress self-affirmation reflections in the whole output to achieve more concise reasoning. Whether it is during the exploration process (self-affirmation reflection after already correct steps) or after the completed exploration (self-affirmation reflection after outputting correct answers). Notably, while this work provides a preliminary analysis of underthinking, our work offers a more in-depth and specific analysis of self-affirmation reflections, presenting a fresh perspective on efficient reasoning area.

We also compared our method with Underthink [40], and the results are presented in Table 5 and Table 1. To ensure a fair comparison, we only intervened on the same tokens (all "wait" tokens) as described in the main text. In this context, α and β denote the intensity and window length of the intervention, respectively. Underthink [40] recommends parameters of $\alpha = 3$ and $\beta = 600$, which are also the default parameters in Table 1. As shown in Table 5, the phenomenon identified in this paper also contributes to the improvement of Underthink [40]. For instance, when the intervention window length β in Underthink [40] is set to $+\infty$, the performance remains nearly unchanged and sometimes even improves, while the outputs become shorter. However, compared with Underthink [40], we consider that intervening based on probability is more straightforward and better aligned with the analysis in our paper. Therefore, we ultimately chose not to directly intervene in the logits.

A.3 Template

There are the templates we use. During the second judgment phase, the current reflection and the sequence of steps spanning from current reflection step to the subsequent reflection step are processed simultaneously.

The first judgment template

"""

Current step: {str1}

Please help me determine the function of the current step.

Is the current step a reflective behavior?

Output the answer directly to <answer></answer>, for example, <answer>Yes</answer> or <answer>No</answer>.

"""

The second judgment template

"""

The previous steps: {str1}

The initial step of reflection: {str2}

The subsequent steps of reflection: {str3}

Please help me judge the role of the reflection steps. Is the result of the reflection affirms the previous content? Output your answer directly to <answer></answer>, for example, <answer>Yes</answer> or <answer>No</answer>.

"""

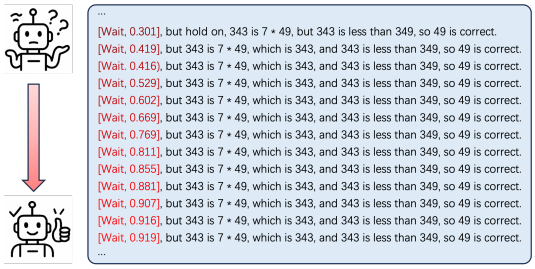

Figure 8: An example where output gets trapped in a loop.

A.4 Influence of the tail repeated reflections

Figure 8 illustrates a case where the model becomes trapped in a loop. The correct answer is 49, yet the model repeatedly performs self-affirmation reflection and enhances the probability of the leading word. As demonstrated in Figures 9 and 10, the repeated steps filtered to the end of the output significantly impact the statistical results. This presents an intriguing phenomenon worth in-depth investigation. Moreover, the intervention proposed in this paper can partially address this issue. Notably, despite this issue, Self-affirmation reflections still retains significantly confidence bias in leading tokens compared to other reflections. This proves the generalization of our discovery.

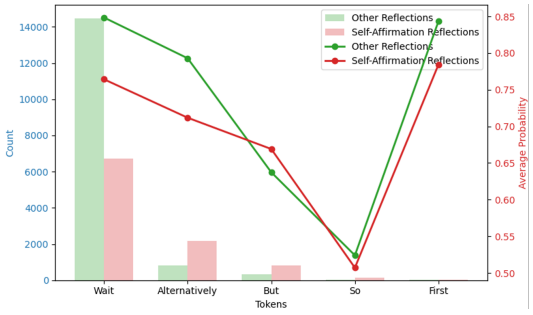

Figure 9: The frequency (via bar chart) and average confidence scores (via line plot) of the first words in all reflective sentences before filtering out the repetitive reflections at the tail.

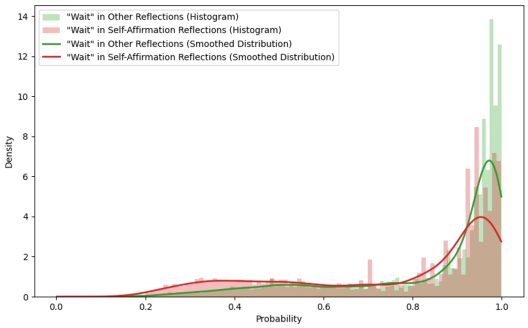

Figure 10: The distribution of "Wait" in the two types of reflections before filtering out the repetitive reflections at the tail.

A.5 Discussion on interfered tokens

In this section, we evaluate the impact of intervening on different tokens for R1-Distill-Qwen-1.5B [12]. Results in Table 6 demonstrate consistent patterns across settings. When threshold values are small, all results align with our hypothesis in Section 5.1: removing the Self-Affirmation Reflection does not significantly degrade performance but generates shorter outputs. This behavior is observed in interventions targeting "wait" tokens (Threshold=0.9), "wait" + "alternatively" tokens (Threshold=0.5), and "wait" + "alternatively" + "but" tokens (Threshold=0.5). Increasing threshold values shorten outputs at the cost of reduced performance.

For fixed thresholds, output length decreases progressively as the number of intervened tokens increases. However, addressing dependencies between multiple tokens remains challenging due to combinatorial complexity. Notably, "wait" tokens exhibits statistically significant effects in our analysis. Given these factors, we defer exploration of multi-token interactions to future work and focus primarily on "wait" tokens in this paper.

Table 6: Comparison of different interfered tokens. Baseline represent the results of R1-Distill-Qwen-1.5B [12].

Models	AIME24		AMC23		GSM8K		MATH500		Average	Average
	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓
Baseline	31.7	16415.7	69.7	10370.1	75.7	676.7	82.7	5488.0	64.9	8237.6
"wait" tokens										
Threshold 0.1	32.5	15468.9	71.9	9412.5	75.3	734.8	84.6	5268.1	66.1	7721.1
Threshold 0.3	27.5	16143.3	72.5	9101.5	77.6	667.4	83.0	4893.7	65.2	7701.5
Threshold 0.5	27.9	14153.8	65.6	8370.6	76.0	587.0	82.0	4924.9	62.9	7009.1
Threshold 0.7	26.7	14989.6	69.7	7679.0	75.3	635.6	80.6	4549.0	63.1	6963.3
Threshold 0.9	28.3	14121.2	71.6	7765.5	75.2	698.9	81.6	4191.3	64.2	6694.2
"wait" + "alternatively" tokens										
Threshold 0.1	25.8	16756.1	66.9	10206.1	74.6	679.0	83.4	5458.2	62.7	8274.8
Threshold 0.3	30.0	16393.9	69.4	8903.5	75.4	703.0	83.8	4944.0	64.6	7736.1
Threshold 0.5	29.2	14704.1	72.8	7606.9	76.0	567.2	84.0	4443.0	65.5	6830.3
Threshold 0.7	29.2	13302.3	69.7	6983.7	75.8	595.9	81.8	4012.1	64.1	6223.5
Threshold 0.9	26.7	12735.7	70.0	6708.7	76.0	595.7	82.4	4086.4	63.8	6031.6
"wait" + "alternatively" + "but" tokens										
Threshold 0.1	28.3	16280.2	71.9	9770.4	77.1	652.8	81.8	5427.4	64.8	8032.7
Threshold 0.3	30.4	14283.2	69.4	7973.4	75.1	619.4	82.8	4546.2	64.4	6855.5
Threshold 0.5	26.7	12117.4	72.8	6287.5	76.1	528.8	82.8	3672.2	64.6	5651.5
Threshold 0.7	27.1	9525.1	68.1	5459.7	76.1	518.9	80.4	3357.7	62.9	4715.4
Threshold 0.9	24.6	9432.5	68.4	5259.5	76.0	509.4	80.6	3221.1	62.4	4605.6

Table 7: The influence of different thresholds on the original model in out-of-domain dataset.

	R1-Distill-1.5B		R1-Distill-7B		R1-Distill-32B		QwQ-32B		Qwen3-32B	
	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓	Acc↑	LEN↓
Baseline	33.8	10328.9	50.9	9264.1	60.0	6723.2	62.7	9103.7	67.7	8086.5
Threshold 0.1	35.6	9288.1	48.3	8927.4	60.9	6761.1	64.9	8934.5	68.0	8270.6
Threshold 0.3	36.5	9411.5	50.7	8462.5	60.6	6539.8	64.9	8610.2	68.7	7973.7
Threshold 0.5	33.7	9083.3	47.1	8055.1	60.5	6123.7	63.1	8355.0	68.8	7826.6
Threshold 0.7	34.9	8896.1	46.5	8637.6	60.6	6245.8	63.6	7987.2	67.4	7771.4
Threshold 0.9	32.6	10850.9	51.0	9498.3	58.9	6953.1	63.4	7597.0	68.0	7571.6

A.6 Results of out-of-domain dataset

We additionally evaluate our model on out-of-domain test data using the GPQA-Diamond benchmark [30]. GPQA-Diamond serves as a rigorous evaluation dataset designed to assess models' capacity for deep reasoning and domain expertise. This dataset represents the highest-quality resource within the GPQA series, comprising 198 graduate-level or competition-level multiple-choice questions. The questions primarily focus on core STEM disciplines including biology, physics, and chemistry, presenting complex problems that require sophisticated reasoning abilities. We sampled four responses for each question and restricted the output length to 32k tokens. We report the average accuracy (Acc) and average token count (LEN) per response. As shown in Table 7, our method generates significantly shorter outputs on out-of-domain dataset while maintaining competitive performance.

A.7 Implementation in vLLM [19]

```python
from transformers import AutoTokenizer, AutoModelForCausalLM
from vllm import LLM, SamplingParams
import torch
from functools import partial
import os
os.environ["VLLM_USE_V1"] = "0" # we use latest vLLM version for
                                  testing Qwen3. In order to use
                                  logits_processors, we need to set
                                  this to 0

def prob_adjustment(token_ids, logits, adjust_ids, values, threshold):
    assert len(logits.shape) == 1
    probs = torch.softmax(logits, dim=-1)
```

```
    logits[adjust_ids.to(logits.device)] = torch.where(probs[
                                        adjust_ids.to(logits.device)] <
                                         threshold, values, logits[
                                        adjust_ids.to(logits.device)])
    return logits

# load the tokenizer and the model
deepseek_1_5b_path="your path"
tokenizer = AutoTokenizer.from_pretrained(deepseek_1_5b_path)
llm = LLM(model=deepseek_1_5b_path, tensor_parallel_size=4)

# prepare the model input
Question = "Carlos went to a sports store to buy running shoes.
                                    Running shoes were on sale, with
                                    prices reduced by $20\%$ on every
                                    pair of shoes. Carlos also knew
                                    that he had to pay a $7.5\%$ sales
                                    tax on the discounted price. He had
                                     $$43$ dollars. What is the
                                    original (before discount) price of
                                     the most expensive shoes he could
                                    afford to buy?"
Template = '<|begin_of_sentence|><|User|>Please reason step by step,
                                    and put your final answer within \\
                                    boxed{{}}. Question: {input}<|
                                    Assistant|><think>\n'  # from
                                    EfficientReasoning
Input_text = Template.format(input=Question)

# prepare the sampling params, just for a example without any
                                    randomness. During the formal test,
                                     we strictly adhered to the
                                    official configuration of Deepseek.
ori_sampling_params = SamplingParams(
    max_tokens=32768,
    n=1,
)
our_sampling_params = SamplingParams(
    max_tokens=32768,
    n=1,
    logits_processors=[
        partial(
            prob_adjustment,
            adjust_ids=torch.tensor([11489,  3783, 14190, 13824]),
            values=float('-inf'),
            threshold=0.3
        )
    ]
)

# test
ori_output = llm.generate(Input_text, sampling_params=
                                    ori_sampling_params)
our_output = llm.generate(Input_text, sampling_params=
                                    our_sampling_params)
ori_length = len(tokenizer.encode(ori_output[0].outputs[0].text,
                                    add_special_tokens=False))
our_length = len(tokenizer.encode(our_output[0].outputs[0].text,
                                    add_special_tokens=False))
print("original length:", ori_length)
print("our length:", our_length)
```

We provide the simplest implementation method in vLLM [19]. However, for a perfect running speed, we suggest achieving it by directly creating a new CustomSampler class. For example,

```
llm.llm_engine.model_executor.driver_worker.model_runner.model.sampler
                                    = CustomSampler(...). # when
                                   tensor_parallel_size = 1
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of our work in Section A.1.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We fully disclose all the information needed to reproduce the main experimental results of the paper in Section 5.1.1 and Section 5.2.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the necessary code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We describe the implementation in detail to make it reproductiive.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report other appropriate information about the statistical significance of the experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided our employed open-source models and the hardware environments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conduct research in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper has no social impact of either positive or negative.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The data and models are open-source that have a no risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We are the creators or original owners of assets used in the paper, properly credited, and are the license and terms of use explicitly mentioned and properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We release no new assets in this paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We didn't employ any human crowd-sourcing projects in this work.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The contents of our paper have no risk of leaking out or disclosing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We describe the usage of LLMs in Section A.3.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.