

Think Less, Know More: State-Aware Reasoning Compression with Knowledge Guidance for Efficient Reasoning

Yi Sui¹, Chaozhuo Li², Dawei Song^{1,3} *

¹Beijing Institute of Technology, China

²Beijing University of Posts and Telecommunications, China

³The Open University, UK

Correspondence: dawei.song2010@gmail.com

Abstract

Large Reasoning Models (LRMs) achieve strong performance on complex tasks by leveraging long Chain-of-Thought (CoT), but often suffer from overthinking, leading to excessive reasoning steps and high inference latency. Existing CoT compression methods struggle to balance accuracy and efficiency, and lack fine-grained, step-level adaptation to redundancy and reasoning bias. Therefore, we propose State-Aware Reasoning Compression with Knowledge Guidance (STACK), a framework that performs step-wise CoT compression by explicitly modeling stage-specific redundancy sources and integrating with a retrieval-augmented guidance. STACK constructs on-line long-short contrastive samples and dynamically switches between knowledge-guided compression for uncertain or biased reasoning state and self-prompted compression for overly long but confident state, complemented by an answer-convergence-based early stopping mechanism to suppress redundant verification. We further propose a reward-difference-driven training strategy by combining Proximal Policy Optimization (PPO) and Direct Preference Optimization (DPO), enabling models to learn state-conditioned compression strategies. Experiments on three mathematical reasoning benchmarks show that STACK achieves a superior accuracy-efficiency balance, reducing average response length by 59.9% while improving accuracy by 4.8 points over existing methods.

1 Introduction

Driven by reinforcement learning (RL) with verifiable rewards, Large Reasoning Models (LRMs) (Jaech et al., 2024; Guo et al., 2025; Liu et al., 2026) have developed long Chain-of-Thought (CoT) reasoning patterns (Wei et al., 2022; Chen et al., 2025). By leveraging deep logi-

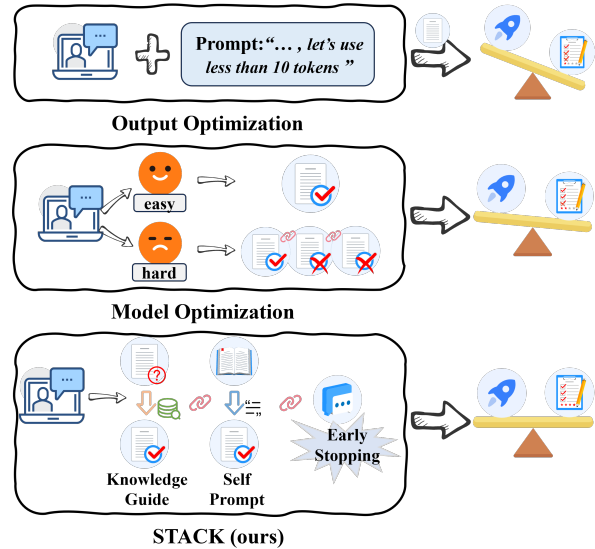


Figure 1: Comparison of CoT Compression Methods.

cal deductions, extensive exploration, and mechanisms for reflection and self-verification, this slow-thinking paradigm substantially enhances reasoning depth and flexibility (Yeo et al., 2025; Wang et al., 2026). It induces a qualitative shift from shallow pattern matching toward structured and verifiable reasoning in challenging tasks such as mathematical proofs, complex programming and logical analysis. However, long CoT reasoning also introduces an overthinking phenomenon: regardless of task complexity, the model consistently follows a deep reasoning pattern, leading to extended reasoning steps that substantially increase response latency and computational cost (Chiang and Lee, 2024; Sui et al., 2025).

To address this issue, CoT compression techniques (Liu et al., 2025b; Qu et al., 2025) aim to improve reasoning efficiency while preserving performance, primarily through output or model optimization, as shown in Figure 1. *Output optimization* typically relies on prompt design to encourage concise reasoning (Xu et al., 2025; Han et al., 2025). However, it may compromise reasoning completeness and coherence, and is inherently

*Corresponding author.

constrained by performance ceilings. In contrast, *model optimization* promotes shorter reasoning by introducing reward functions or penalty terms within RL, to train the models to adaptively decide when to engage explicit reasoning (Zhang et al., 2025a; Hou et al., 2025). This type of methods are effective in reducing reasoning length at the cost of reasoning accuracy. However, in long CoT, the redundancy arises from heterogeneous sources, leading to varied *reasoning states* at different stages of reasoning, such as *repetitive elaboration under high confidence* and *excessive verification under uncertainty*. Although existing model optimization methods adopt online learning strategies, they primarily adjust compression intensity according to task-level complexity (Yan et al., 2025a; Fang et al., 2025). The absence of a finer-grained, step-wise modeling of redundancy sources prevents the existing methods from dynamically selecting the appropriate compression strategies required for different reasoning states along the reasoning chain.

Moreover, recent studies show that long CoT may aggravate hallucinations (Lu et al., 2025): early-stage local biases are propagated and amplified through reflection and self-persuasion, leading to error accumulation that undermines reasoning reliability and accuracy. Retrieval-augmented generation (RAG) is a widely used approach to tackle the problem. However, given the extended reasoning chains of LRMs, the external knowledge introduced as contextual input tends to be gradually dominated by internal inference as reasoning unfolds (Zhao et al., 2022; Oche et al., 2025). This observation motivates a step-wise integration of CoT compression and knowledge guidance, enabling timely correction of reasoning deviations while suppressing redundancy.

To address these challenges, we propose State-Aware Reasoning Compression with Knowledge Guidance (STACK), a unified framework that dynamically regulates step-level CoT compression while integrating with a retrieval-augmented guidance. During sampling, STACK constructs online contrastive long-short CoT samples and adaptively switches between different compression strategies for different reasoning states: specifically, a knowledge-guided compression for uncertain or biased reasoning state and a self-prompted compression for overly long but confident state, to address stage-specific sources of redundancy. This would enable the model to learn when to incorporate external knowledge for correction and when

to condense redundant reasoning. In addition, we introduce an early stopping mechanism based on the convergence of answer distribution to further identify and suppress redundant verification steps.

At the training stage, STACK combines the reward modeling capability of Proximal Policy Optimization (PPO) (Schulman et al., 2017) with the lightweight contrastive learning paradigm of Direct Preference Optimization (DPO) (Rafailov et al., 2023). By driving policy updates through reward differences, the framework enhances the model’s ability to autonomously learn state-conditioned optimal compression strategies, while leveraging high-quality external evidence to guide inference under the uncertain states, thereby effectively mitigating reasoning bias and error accumulation.

Our work claims the following key innovations:

1. We propose STACK, a state-aware framework that dynamically switches targeted compression strategies to address step-level sources of redundancy.
2. We introduce a knowledge-guided contrastive decoding mechanism with an answer-distribution-based early stopping mechanism, enabling controllable suppression of redundant reasoning while correcting uncertain or biased paths.
3. Extensive experiments on three mathematical reasoning benchmarks demonstrate that STACK consistently outperforms existing CoT compression methods, achieving a superior balance between reasoning accuracy and efficiency across different model scales.

2 Related work

Chain-of-Thought Compression. Existing CoT compression strategies to mitigate overthinking primarily fall into three classes: prompt engineering (Xu et al., 2025; Yan et al., 2025b), which constrains reasoning length or format via explicit prompts; reasoning output optimization (Shen et al., 2025b; Cheng and Van Durme, 2024), which compresses reasoning steps into a latent space or adjusts strategies during inference; and model optimization (Shen et al., 2025a; Luo et al., 2025a), which enhances reasoning efficiency via online RL or offline fine-tuning. While online RL encourages concise reasoning through reward design or penalty item (Team et al., 2025; Zhang et al., 2025b), and offline fine-tuning uses variable-length CoT data (Munkhbat et al., 2025). However, these

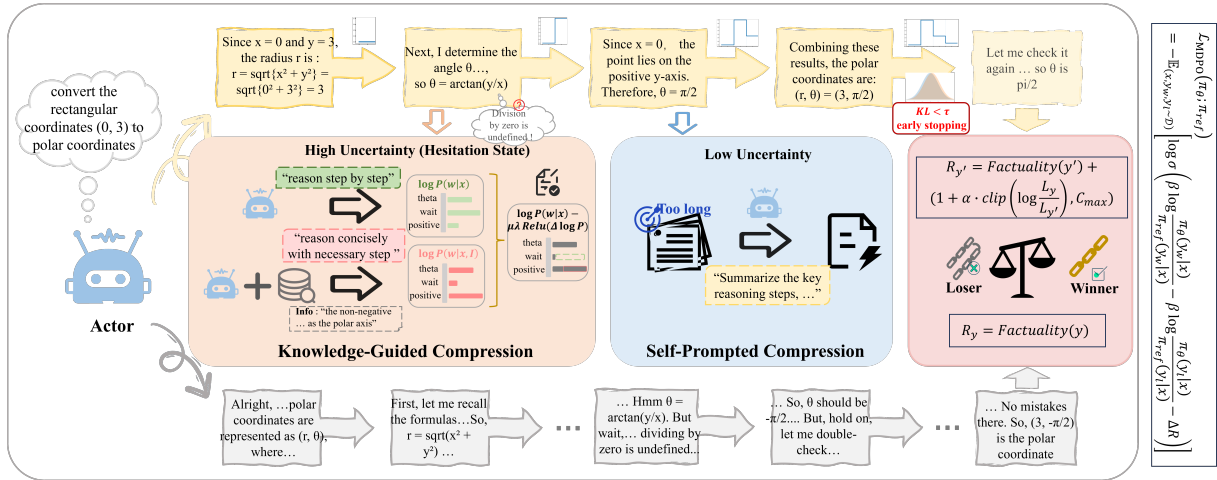


Figure 2: Framework of State-Aware Reasoning Compression with Knowledge Guidance.

methods overlook step-level redundancy causes, and frequently compromise accuracy by ignoring the influence of the model’s intrinsic biases. Our work addresses this by dynamically switching compression strategies at the step level and introducing external knowledge to correct reasoning biases.

Hallucination and Contrastive Decoding.

Hallucination remains a key challenge for LRMs (Huang et al., 2024; Li et al., 2025a; Liu et al., 2025a). RAG, which incorporates real-time external knowledge, has emerged as a mainstream approach to address hallucinations (Fan et al., 2024). To ensure models incorporate external knowledge in long-context inputs, contrastive decoding is proposed to subtly strengthen context guidance in reasoning (Li et al., 2023). Context-aware decoding (Shi et al., 2024) suppresses hallucinations by contrasting outputs distributions with and without external context, but assumes perfectly reliable knowledge. Recent work integrates both relevant and irrelevant knowledge to improve robustness in conflict scenarios (Zhao et al., 2024). By masking retrieval heads to trigger hallucinations, some approaches dynamically contrast the original and masked model outputs via conditional entropy to mitigate hallucinated responses (Gema et al., 2024). In contrast, our Knowledge-Guided Contrastive Decoding adaptively integrates external evidence with parametric knowledge to improve output accuracy and reduce redundancy.

3 State-Aware Reasoning Compression with Knowledge Guidance (STACK)

3.1 Overview

As illustrated in Figure 2, STACK builds upon the DPO paradigm to achieve adaptive chain-of-

thought compression via online long–short contrastive sampling, avoiding the training instability of PPO and the high computational cost of GRPO. For each input, the original reasoning chain is treated as the long sample, while the short sample is generated online through step-wise segmentation and fine-grained compression.

STACK enables step-wise compression by monitoring local information entropy to identify hesitation states that indicate high reasoning uncertainty. When such states are detected, a knowledge-guided compression strategy employs contrastive decoding to steer generation toward evidence-supported reasoning. If absent but the response is overly long, a self-prompted compression strategy removes redundant or repetitive steps. An answer-convergence-based early stopping mechanism further suppresses unnecessary verification once reasoning stabilizes. The resulting compressed reasoning chains are paired with their original counterparts to form contrastive samples. STACK then evaluates both chains with a reward function and integrates the reward difference into the DPO loss, enabling efficient reasoning compression while preserving accuracy. Subsequent sections detail each component.

3.2 Detection of hesitation state

Existing research addressing the overthinking issue in LRMs typically relies on fine-tuning with offline-constructed datasets or implements coarse-grained compression based on problem-level complexity (Zhang et al., 2025b), which limits the flexibility and adaptability of such approaches. In practice, redundant reasoning primarily stems from two sources: first, the presence of substantial expressions that convey little substantive information; sec-

ond, the model’s low confidence in the intermediate reasoning outcome, necessitating iterative verification to confirm correctness (Wang et al., 2025). We refer to the latter as the hesitation state.

To enable fine-grained compression, we introduce local entropy during sampling to track the model’s reasoning state at each step, allowing dynamic, targeted compression based on the source of redundancy. For the token generated at step t with length l , its generation probability distribution is denoted as $P(x)$. The local information entropy is defined as

$$H(X_t) = -\frac{1}{l} \sum_{x \in \mathcal{V}} P(x) \log P(x), \quad (1)$$

where \mathcal{V} represents the vocabulary space. During reasoning, we further maintain the mean local entropy over a sliding window of size m to model the historical confidence baseline for the current sample. If $H(X_t)$ significantly exceeds the historical mean, it indicates that the model exhibits abnormal uncertainty at this reasoning step. In this case, the model is judged to have entered a hesitation state and the framework activates a knowledge-guided compression strategy, introducing external knowledge to steer the reasoning direction and reduce ineffective repetitive reasoning. More detailed analyses are provided in the appendix C.

3.3 Step-level compression strategy

Knowledge-Guided Compression. When the model enters a hesitation state, relying solely on self-guided reasoning tends to amplify local errors or perpetuate incorrect reasoning trajectories, necessitating a knowledge-guided mechanism that simultaneously corrects reasoning direction and suppresses redundancy verification. To this end, we propose Knowledge-Guided Contrastive Decoding (KGCD). KGCD treats retrieved information as a factual anchor and dynamically regulates generation by contrasting the distributional differences between the model’s parametric knowledge and external evidence.

To enhance the discriminability of distributional contrasts, KGCD employs a prompt asymmetry strategy at the input stage: the original model P_{model} maintains its long CoT reasoning behavior, while explicit constraints for concise, evidence-driven reasoning are injected into the system instruction of the retrieval-conditioned model P_{ctx} . This induces a behavioral-level divergence between standard stepwise reasoning and retrieval-guided concise reasoning, forming the basis for

fine-grained contrastive decoding in log-probability space. Formally, KGCD modulates the generation log-probability for each token w as follows:

$$\log \tilde{P}(w) = \log P_{\text{model}}(w) - \mu \cdot \lambda_w \cdot \text{ReLU}\left(\log P_{\text{model}}(w) - \log P_{\text{ctx}}(w)\right), \quad (2)$$

where $\log P_{\text{model}}(w)$ and $\log P_{\text{ctx}}(w)$ represent the log generation probabilities for the current token from the original model and the knowledge-guided model, respectively.

Due to the prompt asymmetry strategy, when $\log P_{\text{model}}(w) \gg \log P_{\text{ctx}}(w)$, it indicates that the token is overly favored without external evidence support, manifesting as redundant reasoning or unfounded self-expansion. Conversely, when the retrieval context significantly increases the relative probability of certain tokens, it indicates that external knowledge provides a clear direction for reasoning, helping to correct reasoning bias caused by knowledge gaps. Therefore, this probability difference term serves the dual role of redundancy identification and directional guidance. The ReLU term becomes inactive when the knowledge-guided model assigns higher probability to a token, contrastive decoding is realized implicitly at the distribution level by selectively suppressing model-favored but knowledge-inconsistent tokens, thereby reallocating probability mass toward evidence-supported candidates without explicitly boosting their logits.

To dynamically adjust the intervention strength, we design a dynamic trust gate based on the relative entropy reduction: $\lambda_w = \text{clip}((H_w^{\text{model}} - H_w^{\text{ctx}})/(H_w^{\text{model}} + \epsilon), 0, 1)$. This gate quantifies the degree to which the retrieval context reduces uncertainty relative to the model itself, thereby assessing the reliability of external knowledge. When the entropy of the retrieval context is significantly lower than the model’s entropy, λ_w approaches 1, indicating that the external knowledge provides high-confidence guidance, and the system fully activates the compression mechanism. Otherwise, λ_w automatically decays to zero to avoid negative intervention from low-quality evidence. Here, $\epsilon = 1e - 06$ is a smoothing constant to prevent division by zero. Additionally, the hyperparameter μ serves as a global compression strength coefficient to modulate the suppression magnitude.

Unlike traditional retrieval-augmented generation, which incorporates external knowledge implicitly as contextual prompts, KGCD performs

continuous and controllable knowledge injection via distribution-level contrastive decoding. This enables external knowledge to precisely calibrate reasoning direction while effectively suppressing redundant generation at the token level, thereby achieving a superior balance between accuracy and reasoning efficiency.

Self-Prompted Compression. When the model is in a high-confidence reasoning state but the response exceeds the predefined maximum length of 2000 tokens (Zhao et al., 2025), STACK activates the self-prompted compression strategy. Under the constraint of lightweight prompt (e.g., “Summarize the key reasoning steps, maintaining logical completeness and removing redundant elaboration.”), LRMs are encouraged to generate reasoning steps that are both logically complete and concisely expressed based on the current reasoning context. This approach significantly reduces token usage while avoiding information loss caused by coarse-grained global compression.

3.4 Early stopping mechanism

Given that LRMs exhibit introspection and self-verification capabilities, they may continue to generate redundant confirmatory reasoning even after reaching a final answer, resulting in unnecessary computational overhead (Qiao et al., 2025). This observation motivates the incorporation of an early stopping mechanism that explicitly detects information saturation during online reasoning, enabling timely termination of redundant steps and facilitating more effective CoT compression as well as more informative training signals.

To this end, we design an early stopping mechanism based on convergence at the answer level. Effective reasoning steps are expected to contribute substantive information toward deriving the final answer. In contrast, when subsequent steps serve only as redundant verification, the model’s probability distribution over candidate answers tends to stabilize. Accordingly, we take the reasoning outputs at steps $t - 1$ and t , denoted as (a_{t-1}, a_t) , as prefix conditions, and compute the conditional probability distributions over the same answer space. The information gain introduced by the additional reasoning step is quantified by the Kullback-Leibler (KL) divergence between these two distributions, formally defined as:

$$\text{IG}_t = \text{KL}(P_t || P_{t-1}), \quad (3)$$

The probability at step t is defined as $P_t = P(A | a_t, \text{'Therefore the final answer is \boxed{'})$, where

P_{t-1} follows an analogous definition. When the information gain falls below a predefined threshold τ , it indicates that the current reasoning can no longer significantly alter the model’s answer judgment. The reasoning process is therefore regarded as converged, and early stopping is triggered.

3.5 Reward function

To jointly optimize reasoning quality and output efficiency during training, we design a reward mechanism that integrates factual accuracy with compression gain. In a complete sampling trajectory, given an input x , the intermediate results produced at each reasoning step are compressed and sequentially aggregated to form a shortened CoT y' . Together with the original long CoT y generated under the same sampling trajectory, they constitute a paired long–short contrastive sample.

The reward for the original chain y reflects only its factual accuracy, $\mathcal{R} = \text{Factuality}(y)$, serving as the baseline for subsequent compression effectiveness. The reward for the compressed chain y' , however, considers both accuracy and its compression ratio relative to y :

$$\mathcal{R}_{y'} = \text{Factuality}(y') \times \left(1 + \alpha \cdot \min \left(\log \left(\frac{L_y}{L_{y'}} \right), C_{\max} \right) \right), \quad (4)$$

where L_y and $L_{y'}$ denote the token lengths of the original and compressed reasoning chains, respectively. The logarithmic term measures compression efficiency, and the clipping function $\min(\cdot, C_{\max})$ constrains the maximum gain ($C_{\max} = 1.5$) to prevent obtaining unreasonable rewards through excessive compression. The hyperparameter α balances the accuracy constraint and the compression incentive, ensuring that improvements in efficiency do not compromise reasoning quality.

By prioritizing factuality and introducing compression benefit as a constrained multiplicative term, this design promotes concise reasoning without sacrificing accuracy. It avoids degenerate shortcut behaviors and steers the model toward learning efficient and robust chain-of-thought compression.

3.6 Loss function

To explicitly model the compression benefit while maintaining the training stability of DPO, we introduce a reward-driven dynamic margin loss into the DPO, referred to as MDPO. Given an input x , along with the preferred sample y_w and dispreferred sample y_l , the loss function is defined as:

Method	GSM8K			MATH500			AIME24			Average		
	Acc	Len	TE	Acc	Len	TE	Acc	Len	TE	Δ Acc	Δ Len	Lat (s)
<i>DeepSeek-R1-Distill-Qwen-1.5B</i>												
Original	83.2	2247	3.70	82.4	4876	1.69	29.4	16814	0.17	–	–	16.14
Prompt	85.3	2134	4.00	83.7	4313	1.94	28.3	15892	0.18	+0.77	-7.35%	15.10
ConCISE	84.1	853	9.86	83.3	2547	3.27	30.1	9315	0.32	+0.83	-51.5%	9.55
MuTIS	85.4	895	9.54	84.6	1794	4.72	31.2	8976	0.35	+2.07	-56.7%	8.83
TokenSqu.	–	–	–	83.2	2447	3.40	33.3	10672	0.31	–	–	–
STACK	87.4	684	12.78	85.3	1783	4.78	36.7	8752	0.42	+4.80	-59.9%	7.23
<i>DeepSeek-R1-Distill-Qwen-7B</i>												
Original	93.5	1478	6.32	91.2	4010	2.27	53.8	13178	0.41	–	–	13.29
Prompt	95.1	1134	8.39	91.7	3654	2.51	53.4	11462	0.47	+0.60	-15.1%	12.47
ConCISE	92.3	773	11.94	91.0	2016	4.51	48.5	7519	0.64	-2.23	-46.8%	8.60
MuTIS	94.6	614	15.41	92.3	1887	4.89	54.7	6548	0.84	+1.03	-53.9%	7.95
TokenSqu.	–	–	–	92.8	1957	4.74	56.3	9079	0.62	–	–	–
STACK	95.7	563	17.00	93.5	1733	5.39	57.4	7274	0.79	+2.70	-54.5%	6.73

Table 1: Performance comparison of reasoning compression methods on three mathematical reasoning benchmarks.

$$\mathcal{L}_{\text{MDPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \Delta R \right) \right], \quad (5)$$

where $\sigma(\cdot)$ is the sigmoid function, β controls the strength of the KL constraint relative to the reference policy π_{ref} , and $\Delta R = \mathcal{R}_w - \mathcal{R}_l$ denotes the reward gap between samples induced by the reward function. It is important to note that if the compressed reasoning chain is correct, it is designated as the preferred sample; otherwise, the uncompressed correct reasoning chain is designated as the preferred sample.

By introducing ΔR as an adaptive discrimination margin, the loss function aligns the policy advantage with the magnitude of actual reward improvement, enabling the model to learn both preference direction and strength. When the reward difference is small, the loss naturally degenerates to standard DPO, mitigating noisy or excessive updates. Meanwhile, the reference-policy log-ratio serves as a stabilizing anchor, preventing policy collapse or language degradation under aggressive compression. Overall, the loss achieves a principled trade-off between compression efficiency, reasoning quality, and training stability.

4 Experiments

4.1 Setup

We employ DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B (202, 2025) as the base models and adopt STACK as the training framework. During training, the learning rate and

batch size are set to $2e-6$ and 64, respectively. The experiments are conducted using the PyTorch framework on 4xNVIDIA A100-80GB GPUs. Furthermore, to provide the model with external knowledge, following the approach of Search-o1 (Li et al., 2025b), we utilize the Bing Web Search API and collect the top-5 retrieved results. Considering that LRMs lack the generation capability for short CoT and knowledge-guided logical reasoning, we perform a cold-start fine-tuning using annotated data to prevent mode collapse and training instability in subsequent DPO optimization. Additional training details are provided in the appendix A.

Datasets Given the task requirements and the strong mathematical capabilities of the base model, we employ three math datasets of increasing difficulty: GSM8K (Cobbe et al., 2021), MATH500 (Hunter et al., 2023), and AIME24. We further include GPQA (Rein et al.), a challenging open-domain QA benchmark involving complex and partially multi-hop reasoning. Accuracy (ACC), response length (Len) and inference latency (Lat) are used as evaluation metrics. We also adopt Token Efficiency ($TE = \frac{Acc}{Len} * 100$) to assess the method’s balance between accuracy and efficiency (Yan et al., 2025a).

Baselines We compare STACK with the following representative Chain-of-Thought compression methods: **Prompt**: Injecting external knowledge into the input and using lightweight prompts to constrain the model’s response length; **ConCISE** (Qiao et al., 2025): Constructing high-quality CoT compression data through confidence injection and early stopping, followed by fine-tuning LRMs using simple preference optimization; **MuTIS** (Zhao et al., 2025): Performing multi-

Method	Distill-Qwen-1.5B			Distill-Qwen-7B		
	Acc	Len	TE	Acc	Len	TE
original	34.8	9213	0.38	50.8	8045	0.63
prompt	32.9	8745	0.38	49.9	7612	0.66
ConCISE	35.3	7254	0.49	48.5	4932	0.98
MuTIS	35.9	6537	0.55	50.9	4378	1.16
TokenSqu.	36.3	7543	0.48	51.5	5056	1.02
STACK	38.4	6012	0.64	52.7	4573	1.15

Table 2: Performance comparison of reasoning compression methods on GPQA dataset.

round dynamic interventions on reasoning trajectories during the online sampling phase and fine-tuning LLMs with original and compressed reasoning chains; **TokenSqueeze** (Zhang et al., 2025c), denoted as TokenSqu.: Constructing high-quality long-short sample pairs via adaptive depth selection and distribution-aligned language refinement, followed by model training using DPO-L.

4.2 Performance Comparison on Reasoning Benchmarks

Table 1 reports the performance of various CoT compression methods across LRMs of different scales. Overall, STACK consistently achieves the best balance between reasoning efficiency and accuracy. Compared with the baseline, STACK reduces the average reasoning length by over 59.9% while improving average accuracy by 4.8 points, yielding the highest token efficiency and the lowest inference latency. Moreover, the distribution of reasoning lengths across datasets with varying difficulty indicates that STACK enhances adaptive reasoning behavior. On the simpler GSM8K benchmark, the 1.5B model adaptively generates shorter reasoning chains (684 tokens on average).

In contrast, prompt-based compression methods exhibit limited effectiveness, reducing inference latency by only approximately 1 second and failing to adequately support complex reasoning. Despite incorporating external knowledge, they incur a 1.1-point accuracy drop on AIME2024 using the 1.5B model. As reasoning chains grow longer, model generation becomes increasingly dominated by internal inference, thereby weakening the influence of external evidence. STACK addresses these issues via step-level fine-grained compression and explicit knowledge reinforcement at the output distribution. This advantage persists on larger models: on the 7B model, STACK improves average accuracy by 2.7 points while reducing inference latency to 6.73s (vs. 7.23s for the 1.5B model). Nevertheless, significant performance gaps remain among

compression strategies.

ConCISE and TokenSqueeze significantly improve reasoning efficiency by constructing high-quality long-short sample pairs offline and fine-tuning models via preference optimization or contrastive learning. However, their statically constructed training data introduces a growing distribution shift as model capabilities evolve, which limits further performance gains. By contrast, STACK constructs contrastive samples online, enabling its compression strategy to adapt dynamically to variations across inputs and reasoning stages, resulting in superior overall performance.

MuTIS also adopts an online intervention paradigm and demonstrates strong performance across multiple settings. However, it merely focuses on structural-level compression strategy. In contrast, STACK continuously monitors step-level reasoning states and dynamically switches between self-prompted and knowledge-guided compression, enabling it to address both stage-specific redundancy and reasoning bias. Experiments on challenging benchmarks such as MATH500 and AIME2024 show that this state-aware and knowledge-guided design substantially improves reasoning accuracy and token efficiency, underscoring the robustness of STACK in complex tasks. Concrete examples are provided in the appendix D.

Furthermore, to evaluate generalization beyond mathematical reasoning, we further conduct experiments on the GPQA dataset, a challenging open-domain QA benchmark involving complex and partially multi-hop reasoning. As shown in Table 2, STACK consistently outperforms both the original base model and existing CoT compression baselines across two model scales. On DeepSeek-R1-Distill-Qwen-1.5B, the STACK improves accuracy from 34.8 to 38.4 while reducing reasoning length from 9213 to 6012 tokens. On the 7B model, the STACK achieves the highest accuracy (52.7) with substantially shorter reasoning compared to the original model (8045→4573 tokens).

Notably, the advantage of STACK over prior compression methods is more pronounced on GPQA than on math benchmarks. While existing approaches primarily focus on structural shortening of reasoning chains, STACK’s state-aware and knowledge-guided mechanism is better suited for open-domain QA, where reasoning errors often stem from knowledge gaps rather than redundant derivations. This explains the stronger relative gains observed in QA settings.

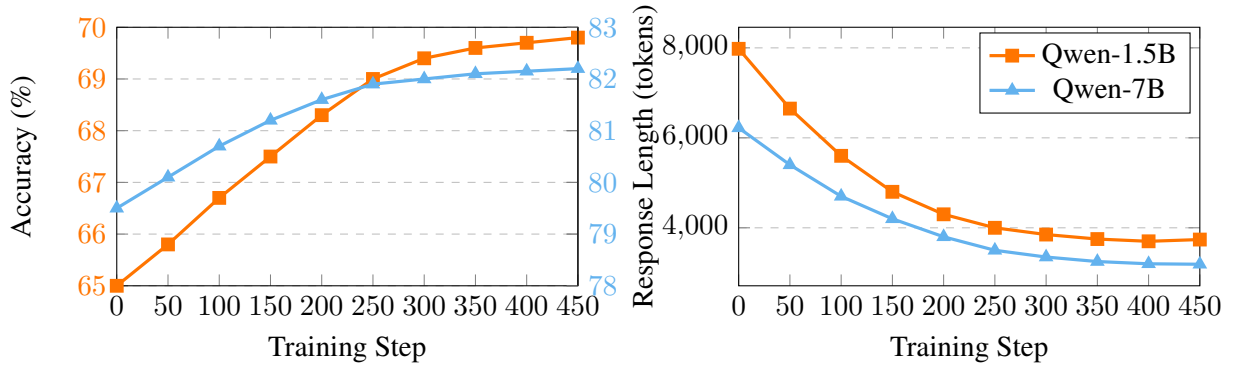


Figure 3: Average accuracy and response length of DeepSeek-R1-Distill-Qwen models across training steps.

4.3 Training Dynamics across Model Scales

As shown in Figure 3, the integration of on-line contrastive CoT construction with reward-difference-driven optimization enables an adaptive balance between reasoning accuracy and information density that aligns with model capacity. During early training, the model primarily eliminates repetitive verification and redundant reflection, resulting in a rapid reduction in response length. As training proceeds, compression becomes increasingly constrained by accuracy requirements, leading the model to preserve essential information and progressively improve factual correctness. The process eventually reaches a convergence mode characterized by stable reasoning length and accuracy.

Models of different scales exhibit pronounced differences under this framework. The 1.5B model, characterized by higher initial redundancy, achieves larger early-stage compression gains but exhibits accuracy improvements mainly in mid training and remains more sensitive to reward fluctuations in later stages due to limited capacity. In contrast, the 7B model starts with more compact reasoning and thus offers less compression headroom, yet its stronger representational capacity enables more coordinated improvements in compression and accuracy, yielding smoother training dynamics and faster convergence.

4.4 Ablation study

STACK is a principled adaptive compression framework with modular control signals, where each component addresses a distinct failure mode in long-form reasoning. The effectiveness of the framework is empirically validated through a systematic ablation study, as shown in Table 3.

First, entropy detection (ED) acts as a control signal for retrieval triggering. Without ED, retrieval is applied at every step, resulting in a degradation

Method	MATH500		AIME24	
	Acc	Len	Acc	Len
original	91.2	4010	53.8	13178
<i>Ablating entropy detection (ED)</i>				
w/o ED	92.9	1895	56.3	7580
<i>Ablating knowledge-guided compression</i>				
w/o retrieval	92.1	2680	54.7	8697
top-1 replacement	92.4	1940	55.4	7520
50% irrelevant	93.1	1810	56.8	7440
always-on	92.9	1895	56.3	7580
<i>Ablating early stopping (ES)</i>				
w/o ES	93.3	2475	57.1	9130
end signal	86.7	3374	31.7	10693
consistency	93.3	1980	57.5	8157
<i>Ablating training paradigm</i>				
SFT	92.4	3016	56.7	9761
SFT+DPO	93.2	1895	57.1	7586
STACK	93.5	1733	57.4	7274

Table 3: Ablation study on DeepSeek-R1-Distill-Qwen-7B model.

in accuracy. This indicates that ED primarily improves efficiency and stability by avoiding unnecessary retrieval calls, rather than directly improving accuracy.

Second, knowledge-guided compression aims to provide corrective external grounding during high-uncertainty (hesitation) states. When retrieval is removed, accuracy degrades significantly, especially on the more challenging AIME24 benchmark. Notably, the reasoning length remains relatively short (2680 tokens on MATH500 and 8697 on AIME24), indicating that compression still operates but loses access to corrective evidence. These results suggest that retrieval is the primary contributor to accuracy gains rather than an auxiliary component.

To further validate robustness, we conduct three

controlled settings: (1) random top-1 replacement, (2) 50% irrelevant retrieval mixed into top-k, and (3) always-on retrieval without entropy gating. Table 3 shows that STACK is robust to noisy retrieval. With random replacement, accuracy drops slightly (93.5→92.4 on MATH500; 57.4→55.4 on AIME24), while λ_w decreases to 0.11, indicating automatic suppression of unreliable knowledge. Under 50% irrelevant retrieval, λ_w increases moderately to 0.24, and performance remains above the w/o retrieval setting, suggesting partial use of valid evidence. In contrast, always-on retrieval (w/o ED) results in knowledge fusion at every step ($\lambda_w \approx 0.98$). While accuracy remains competitive (92.9 / 56.3), it is consistently lower than STACK, suggesting that unconditional retrieval may introduce unnecessary interference. This result highlights that retrieval is most beneficial when adaptively gated, rather than applied at every step.

Third, early stopping (ES) targets redundancy in the later verification phase. Removing early stopping leaves accuracy nearly unchanged (93.3 on MATH500 and 57.1 on AIME24) but significantly increases reasoning length (2475 tokens on MATH500 and 9130 on AIME24), confirming that its role is efficiency-oriented rather than correctness-driven. In addition, we replaced our stability-based early stopping strategy, which relies on answer distribution, with two alternative approaches: increasing the probability of generating an end-of-reasoning signal (Liu and Wang, 2025) and tracking answer consistency across consecutive steps (Mao et al., 2025). The results indicate that relying solely on the end-of-reasoning signal leads to premature termination of reasoning, significantly reducing both accuracy and output length, particularly on AIME24. In contrast, our early stopping mechanism demonstrates comparable reliability and stability to answer consistency while achieving greater compression of generated outputs. This highlights that our early stopping based on answer-level convergence not only ensures sufficiently thorough reasoning but also enhances the conciseness and efficiency of the outputs.

In the ablation study on training paradigms, training solely with teacher-labeled data (SFT) provides modest gains, allowing the model to begin adapting to short-chain reasoning and improving stability for subsequent training. Introducing direct preference optimization (SFT+DPO) further enhances the model’s ability to produce concise reasoning. Notably, our approach (SFT+MDPO), by incorpo-

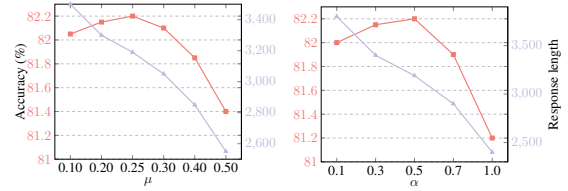


Figure 4: Sensitivity analysis of hyperparameters μ and α on average accuracy and response length.

rating reward differentials into the DPO optimization objective, achieves a more effective balance between accuracy and output length, enabling the model to generate more concise reasoning chains while maintaining stable performance.

4.5 Hyperparameter analysis

Empirically, we find that the contrastive decoding coefficient μ and the compression reward weight α are crucial for balancing reasoning accuracy and chain compactness. Specifically, μ controls the magnitude of the penalty applied when the model’s generation deviates from external evidence, while α determines the scaling factor of compression gains during training. Values that are too small make compression ineffective, while values that are too large cause premature truncation or reduced reasoning quality as shown in Figure 4. Across different model scales, $\mu = 0.25$ and $\alpha = 0.5$ consistently achieve the best balance between preserving accuracy and compressing reasoning. The detailed information of the sensitivity studies on both the sliding window size m and the early stopping threshold τ is provided in Appendix B.

5 Conclusion

In this work, we propose STACK, a State-Aware Reasoning Compression framework with Knowledge Guidance, which innovatively integrates a retrieval-augmented mechanism into the COT compression process and enables dynamic switching of online compression strategies to address diverse sources of redundancy at different steps along the reasoning chain. The introduction of early stopping mechanism based on answer-level convergence ensures thorough reasoning while further enhancing output conciseness. Moreover, by incorporating the reward differences into the DPO optimization objective, STACK achieves a superior balance between accuracy and reasoning length, attaining a state-of-the-art performance across different models and datasets.

6 Limitations

Despite the effectiveness of STACK in dynamically compressing reasoning chains and leveraging retrieval-augmented guidance, several limitations remain. First, the incorporation of online contrastive sampling and retrieval-based strategies introduces additional computational overhead, leading to increased training time and resource consumption. Second, the current retrieval mechanism does not guarantee fully accurate or relevant external knowledge, which may propagate suboptimal guidance during reasoning; joint optimization of the retrieval component and the model could further enhance performance. Finally, our framework currently relies solely on textual knowledge for external augmentation, while other modalities or external reasoning tools, such as structured databases, symbolic solvers, or multi-modal knowledge sources, represent promising directions for further extension and improvement.

7 Acknowledgements

This work was supported by Beijing Natural Science Foundation (No. L251037), the Fundamental Research Funds for the Central Universities and in part by the Natural Science Foundation of China (grant number: 62376027).

References

2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.
- Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.
- Cheng-Han Chiang and Hung-yi Lee. 2024. Over-reasoning and redundant calculation of large language models. *arXiv preprint arXiv:2401.11467*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Wenqi Fan, Yajuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.
- Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*.
- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omnimath: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.
- Aryo Pradipta Gema, Chen Jin, Ahmed Abdulaal, Tom Diethel, Philip Teare, Beatrice Alex, Pasquale Minervini, and Amrutha Saseendran. 2024. Decore: Decoding by contrasting retrieval heads to mitigate hallucinations. *arXiv preprint arXiv:2410.18860*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.
- Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning, 2025. *arXiv preprint arXiv:2504.01296*.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2024. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.
- Lightman Hunter, Kosaraju Vineet, Burda Yuri, Edwards Harrison, Baker Bowen, Lee Teddy, Leike Jan, Schulman John, Sutskever Ilya, and Cobbe Karl. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Chaozhuo Li, Pengbo Wang, Chenxu Wang, Litian Zhang, Zheng Liu, Qiwei Ye, Yuanbo Xu, Feiran

- Huang, Xi Zhang, and Philip S Yu. 2025a. Loki’s dance of illusions: A comprehensive survey of hallucination in large language models. *arXiv preprint arXiv:2507.02870*.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 12286–12312.
- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438.
- Songyang Liu, Chaozhuo Li, Jiameng Qiu, Xi Zhang, Feiran Huang, Litian Zhang, Yiming Hei, and Philip S Yu. 2025a. The scales of justitia: A comprehensive survey on safety evaluation of llms. *arXiv preprint arXiv:2506.11094*.
- Songyang Liu, Chaozhuo Li, Chenxu Wang, Jinyu Hou, Zejian Chen, Litian Zhang, Zheng Liu, Qiwei Ye, Yiming Hei, Xi Zhang, and 1 others. 2026. Clawkeeper: Comprehensive safety protection for openclaw agents through skills, plugins, and watchers. *arXiv preprint arXiv:2603.24414*.
- Xin Liu and Lu Wang. 2025. Answer convergence as a signal for early stopping in reasoning. *arXiv preprint arXiv:2506.02536*.
- Yue Liu, Jiaying Wu, Yufei He, Ruihan Gong, Jun Xia, Liang Li, Hongcheng Gao, Hongyu Chen, Baolong Bi, Jiaheng Zhang, and 1 others. 2025b. Efficient inference for large reasoning models: A survey. *arXiv preprint arXiv:2503.23077*.
- Haolang Lu, Yilian Liu, Jingxin Xu, Guoshun Nan, Yuanlong Yu, Zhican Chen, and Kun Wang. 2025. Auditing meta-cognitive hallucinations in reasoning large language models. *arXiv preprint arXiv:2505.13143*.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.
- Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, and 1 others. 2025b. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*.
- Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. 2025. Early stopping chain-of-thoughts in large language models. *arXiv preprint arXiv:2509.14004*.
- Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, and 1 others. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*.
- Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*.
- Agada Joseph Oche, Ademola Glory Folashade, Tirthankar Ghosal, and Arpan Biswas. 2025. A systematic review of key retrieval-augmented generation (rag) systems: Progress, gaps, and future directions. *arXiv preprint arXiv:2507.18910*.
- Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8021–8040.
- Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.
- 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*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. 2025a. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.
- Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025b. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*.
- Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024.

- Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- Chenxu Wang, Chaozhuo Li, Songyang Liu, Zejian Chen, Jinyu Hou, Ji Qi, Rui Li, Litian Zhang, Qiwei Ye, Zheng Liu, and 1 others. 2026. The devil behind molttbook: Anthropic safety is always vanishing in self-evolving ai societies. *arXiv preprint arXiv:2602.09877*.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, and Zhenru Zhang. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.
- Jianzhi Yan, Le Liu, Youcheng Pan, Shiwei Chen, Zike Yuan, Yang Xiang, and Buzhou Tang. 2025a. From long to lean: Performance-aware and adaptive chain-of-thought compression via multi-round refinement. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 12290–12306.
- Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. 2025b. Infythink: Breaking the length limits of long-context reasoning in large language models. *arXiv preprint arXiv:2503.06692*.
- Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.
- Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025a. Adaptthink: Reasoning models can learn when to think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3716–3730.
- Xiaoyun Zhang, Jingqing Ruan, Xing Ma, Yawen Zhu, Haodong Zhao, Hao Li, Jiansong Chen, Ke Zeng, and Xunliang Cai. 2025b. When to continue thinking: Adaptive thinking mode switching for efficient reasoning. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5808–5828.
- Yuxiang Zhang, Zhengxu Yu, Weihang Pan, Zhongming Jin, Qiang Fu, Deng Cai, Binbin Lin, and Jieping Ye. 2025c. Tokensqueeze: Performance-preserving compression for reasoning llms. *arXiv preprint arXiv:2511.13223*.
- Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*.
- Wenshuo Zhao, Haoxing Zhai, Xinyu Qiu, Zhenting Qi, Shuhe Li, and Linchao Zhu. 2025. Mutis: Enhancing reasoning efficiency through multi turn intervention sampling in reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13680–13692.
- Zheng Zhao, Emilio Monti, Jens Lehmann, and Haytham Assem. 2024. Enhancing contextual understanding in large language models through contrastive decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4225–4237.

A Training Detail

We train our models on the DeepScaleR dataset (Luo et al., 2025b), which comprises 40,000 mathematical problems sourced from AIME (1983–2023), AMC, Omni-Math (Gao et al., 2024), and STILL (Min et al., 2024). In addition, considering that LRMs are not inherently equipped to generate short Chains of Thought (Short CoT) or perform knowledge-guided logical reasoning, directly applying DPO to an unadapted model would require the new policy π_θ to simultaneously learn the two challenging objectives of knowledge utilization and logical compression. This often results in a substantial divergence from the reference policy π_{ref} , leading to KL divergence explosion and training instability. To address this, we introduce a knowledge-guided cold-start stage designed to pre-align the reference policy distribution.

Data Construction & Mixing Strategy. During the cold-start phase, we perform supervised

m	DeepSeek-R1-Distill-Qwen-7B		DeepSeek-R1-Distill-Qwen-1.5B	
	Avg Retrieval	Avg Acc	Avg Retrieval	Avg Acc
2	10.1	82.6	12.5	68.2
3	8.9	81.7	10.8	69.4
5	6.3	82.2	8.4	69.8
7	5.5	81.5	7.0	68.9
10	3.2	80.1	4.5	66.5

Table 4: Hyperparameter analysis of hesitation detection window size m across model scales.

fine-tuning (SFT) on the base model using a 5,000 sample dataset, composed of two parts mixed at an 80:20 ratio:

1. **Task-Specific Data (80%).** To enable the model to acquire compressed reasoning patterns conditioned on retrieved knowledge, we construct pairs of the form (x, y_{teacher}) , where the input $x = [Q, I]$ consists of the question and retrieval context, and the output y_{teacher} is a high-quality, knowledge-integrated, compressed Chain of Thought generated by a teacher model (GPT-4o).
2. **General Regularization Data (20%).** To prevent catastrophic forgetting caused by overfitting to specific compression patterns, we incorporate general QA and reasoning data (e.g., standard CoT), preserving the model’s fundamental language and logical generalization capabilities.

The primary objective of this phase is to construct a high-quality reference policy π_{ref} . By pre-adapting π_{ref} to the conditional distribution $P(y | Q, I)$, we constrain the exploration space in the subsequent reinforcement learning phase from unguided trial-and-error to knowledge-grounded refinement, ensuring that the KL divergence constraint acts as a benign “language competence safeguard” rather than a barrier to effective knowledge injection.

B Sensitivity Analysis of Early Stopping and Sliding Window Size

We provide a cross-scale analysis of hesitation state detection by varying the sliding window size m . Across both model sizes, the optimal setting is 5, which achieves the best balance between retrieval frequency and reasoning accuracy. The detailed experimental results are reported in Table 4.

For the 7B model, the performance remains relatively stable across different m . Although $m = 2$ yields the highest accuracy (82.6), it triggers substantially more retrieval operations (10.2 vs. 6.1 at $m = 5$), leading to higher computational cost without consistent gains. In contrast, $m = 5$ achieves the competitive accuracy (82.2) with significantly fewer retrieval calls, demonstrating a better efficiency–performance trade-off. When m is too large (e.g., 10), under-detection of hesitation states leads to accuracy degradation (80.1).

For the 1.5B model, the effect is more pronounced. Due to stronger entropy fluctuations, the retrieval frequency is consistently higher (e.g., 12.4 at $m = 2$), and the accuracy varies more sharply (69.8 at $m = 5$ vs. 66.5 at $m = 10$). This larger performance drop indicates that smaller models are more sensitive to reward instability and rely more heavily on properly triggered retrieval to stabilize reasoning.

To better understand the practical behavior of STACK, we further analyze the early stopping threshold τ . As τ increases, the reasoning length decreases monotonically, while accuracy follows an inverted-U trend.

As shown in Table 5, when τ is small (0.005–0.01), the early stopping is conservative and rarely triggered, resulting in longer reasoning chains (Len up to 3570) with slightly suboptimal accuracy. As τ increases to 0.03, the model achieves the best balance between completeness and efficiency, reaching the highest average accuracy (82.2) while reducing reasoning length to 3190. However, when τ becomes too large (≥ 0.05), early stopping becomes overly aggressive, prematurely truncating necessary reasoning steps. Although the reasoning length further decreases, accuracy drops noticeably (to 78.9 at $\tau = 0.10$). Overall, accuracy exhibits an inverted-U trend, while reasoning length decreases monotonically as τ increases.

τ	Average Acc	Average Len
0.005	81.9	3570
0.01	81.6	3420
0.02	82.1	3275
0.03	82.2	3190
0.05	81.5	2750
0.07	80.4	2660
0.10	78.9	2480

Table 5: Sensitivity analysis on the early stopping threshold τ .

In summary, both hyperparameters exhibit stable and interpretable behaviors: moderate values of τ and m provide the best balance between efficiency and reasoning completeness, while overly conservative or overly aggressive settings degrade performance.

C Analysis of Hesitation State

Due to the inherently iterative and self-reflective nature of long chain-of-thought reasoning, reflection tokens appear with high frequency across reasoning steps. Figure 5 visualizes the corresponding distributions of token-level and step-level information entropy, illustrating the variation in model uncertainty throughout the reasoning process.

Compared to single reflection token-based detection, our local-entropy-based method significantly reduces unnecessary knowledge retrievals while maintaining or slightly improving reasoning accuracy. Specifically, as shown in figure 6, across long reasoning chains, entropy-based detection triggers only 3 retrievals per chain on average, achieving 82.2% accuracy with a mean response length of 3,190 tokens. In contrast, relying on individual reflection tokens leads to 7 retrievals per chain, with slightly lower accuracy (81.5%) and longer responses averaging 3,650 tokens.

In our opinion, this seemingly counterintuitive result arises because excessive external knowledge retrieval does not necessarily improve reasoning outcomes and can even degrade DPO training performance. Frequent retrievals introduce redundant or low-relevance information, which increases context length and disperses the model’s attention across less critical tokens. During DPO optimization, the model is guided by reward differences between compressed and uncompressed chains; if the context is filled with repetitive or non-informative

knowledge, the reward signal for effective compression becomes noisy. Consequently, the policy may struggle to correctly attribute which reasoning steps are truly valuable, leading to reduced accuracy and less efficient chain compression. By contrast, local-entropy-based detection selectively introduces external knowledge only when the model exhibits genuine hesitation, ensuring that retrieved evidence is relevant and timely. This targeted approach not only preserves reasoning fidelity but also stabilizes the training signal, resulting in more accurate and compact reasoning chains.

D Case Study

The following tables present two representative cases comparing the original long CoT with the compressed CoT generated by STACK. From these examples, we observe that the excessive length of long CoT mainly stems from repeated verification under uncertain reasoning states (characterized by high local entropy) and redundant expressions that contribute little to task resolution.

By explicitly illustrating the compression process, the cases demonstrate how STACK performs step-wise state detection via local entropy, enabling the timely introduction of external knowledge to correctly steer the reasoning trajectory. This mechanism effectively mitigates reasoning bias, prevents error propagation, and suppresses ineffective reasoning. Furthermore, the answer-convergence-based early stopping mechanism truncates redundant verification once the solution stabilizes, further promoting reasoning-chain compression.

As a result, STACK achieves a favorable balance between conciseness and correctness, producing reasoning chains that are both compact and accurate.

Original output for Case 1

Question: Convert the point $(0, 3)$ in rectangular coordinates to polar coordinates. Enter your answer in the form (r, θ) , where $r > 0$ and $0 \leq \theta < 2\pi$.

Output:

Alright, so I need to convert the rectangular coordinate $(0, 3)$ to polar coordinates. Hmm, polar coordinates are represented as (r, θ) , where r is the distance from the origin to the point, and θ is the angle made with the positive x -axis. I remember that there are formulas to convert from rectangular (x, y) to polar coordinates.

First, let me recall the formulas. I think r is calculated using the Pythagorean theorem, so it should be the square root of $(x$ squared plus y squared). So, $r = \sqrt{x^2 + y^2}$.

Okay, so plugging in the values from the point $(0, 3)$. Here, x is 0 and y is 3. Let me compute r . So $r = \sqrt{0^2 + 3^2}$ which is $\sqrt{0 + 9}$ which simplifies to $\sqrt{9}$. That's 3. So, r is 3. That makes sense because the point is 3 units above the origin on the y -axis, so its distance from the origin is indeed 3.

Next, I need to find θ , the angle. The formula for θ is arctangent of y over x , so $\theta = \arctan(\frac{y}{x})$. But wait, in this case, x is 0. So, $\arctan(\frac{3}{0})$. Hmm, I remember that dividing by zero is undefined, I need to think about what angle θ gives a point on the positive y -axis.

Since x is 0 and y is 3, which is positive, the point lies on the positive y -axis. In polar coordinates, when a point is on the positive y -axis, what is the angle? I think it's $\frac{\pi}{2}$ radians because that's 90 degrees. Let me confirm. Starting from the positive x -axis, if we go counterclockwise 90 degrees, we reach the positive y -axis. So, θ should be $\frac{\pi}{2}$.

But let me make sure I'm not missing anything because sometimes when x is zero, the angle is either $\frac{\pi}{2}$ or $\frac{3\pi}{2}$ depending on the sign of y . Since y is positive here, it's definitely $\frac{\pi}{2}$. If y were negative, it would be $\frac{3\pi}{2}$, but in this case, it's positive.

Let me visualize the point $(0, 3)$ on the Cartesian plane. It's straight up along the y -axis, so the angle from the positive x -axis is a right angle, which is $\frac{\pi}{2}$ radians. So, θ is $\frac{\pi}{2}$. Therefore, putting it all together, the polar coordinates are $(r, \theta) = (3, \frac{\pi}{2})$.

Wait, just to be thorough, sometimes when converting, especially with arctangent, you have to consider the quadrant in which the point lies. In this case, since the point is $(0, 3)$, it's on the y -axis, not in any specific quadrant other than being on the axis itself. But since θ is between θ and 2π , and our point is on the positive y -axis, θ should be $\frac{\pi}{2}$.

I think that's it. So, my final answer should be $(3, \frac{\pi}{2})$. Let me just double-check my calculations. Calculating r : $\sqrt{0^2 + 3^2} = \sqrt{9} = 3$. That's correct.

Calculating θ : $\arctan(\frac{3}{0})$. As x is 0, and y is positive, θ is $\frac{\pi}{2}$. That seems right.

No mistakes there. So, yeah, $(3, \frac{\pi}{2})$ is the polar coordinate for $(0, 3)$.

****Final Answer****The polar coordinates are $(3, \frac{\pi}{2})$.

Ground Truth: $(3, \frac{\pi}{2})$

[True]

STACK's compression process for Case 1

Question: Convert the point $(0, 3)$ in rectangular coordinates to polar coordinates. Enter your answer in the form (r, θ) , where $r > 0$ and $0 \leq \theta < 2\pi$.

System Prompt:

Please reason step by step, and put your final answer within $\boxed{\}$.

Original Output: Okay, so I need to convert the rectangular coordinate $(0, 3)$ to polar coordinates. Hmm, I remember that polar coordinates are represented as (r, θ) , where r is the radius or the distance from the origin, and θ is the angle made with the positive x-axis.

Local Entropy: 0.2086

KL divergence: 5.7693

Original Output: First, let me recall the formulas for converting rectangular coordinates (x, y) to polar coordinates (r, θ) . I think r is calculated using the Pythagorean theorem, so $r = \sqrt{x^2 + y^2}$. And the angle θ can be found using the arctangent function, specifically $\theta = \arctan(\frac{y}{x})$. But wait, I also remember that sometimes $\arctan(\frac{y}{x})$ isn't enough because it doesn't account for the correct quadrant. So, I might need to adjust the angle based on the signs of x and y .

Local Entropy: 0.5049

KL divergence: 0.2825

Knowledge-Guided Compression

System Prompt:

Given the current reasoning and external knowledge:

Continue the reasoning ONLY if it adds new information.

Do NOT repeat, restate, or re-verify any known facts.

Use the external knowledge directly to reach the conclusion.

Once the answer is clear: Put your final answer within $\boxed{\}$ and STOP.

[Current Reasoning] {current reasoning}

[External Knowledge] Rectangular coordinates are represented as (x, y) , and polar coordinates are represented as (r, θ) , where r is the distance to the origin and θ is the angle with the x-axis. To convert from rectangular coordinates to polar coordinates, the formulas are as follows: $r = \sqrt{x^2 + y^2}$, $\theta = \arctan(\frac{y}{x})$.

Assistant:

Compression Output: To convert the rectangular coordinates $(0, 3)$ to polar coordinates (r, θ) , we can use the following formulas: \n1. $r = \sqrt{x^2 + y^2}$ \n2. $\theta = \arctan(\frac{y}{x})$. Given the point $(0, 3)$, we have $x = 0$ and $y = 3$.

Original Output: Alright, Let me compute r first. Plugging into the formula, $r = \sqrt{0^2 + 3^2} = \sqrt{0 + 9} = \sqrt{9} = 3$. Okay, that seems straightforward. So, r is 3.

Local Entropy: 0.3182

KL divergence: 1.3271

STACK's compression process for Case 1

Original Output: Okay, so I got r as 3. Now I need to find the angle θ . I remember that θ is the arctangent of y over x . So $\theta = \arctan(\frac{y}{x})$. But wait, I have to be careful here because x is 0 in this case. So I need to think about what \arctan of $\frac{3}{0}$ is. Hmm, 3 divided by 0 is undefined, so I need to figure out the angle that corresponds to $3 / 0$ in the arctangent function.

Local Entropy: 0.7962

KL divergence: 0.6310

Knowledge-Guided Compression

System Prompt:

Given the current reasoning and external knowledge:

Continue the reasoning ONLY if it adds new information.

Do NOT repeat, restate, or re-verify any known facts.

Use the external knowledge directly to reach the conclusion.

Once the answer is clear: Put your final answer within `\boxed{\}` and STOP.

[Current Reasoning] {current reasoning}

[External Knowledge] The preconditions are as follows: 1. Take the origin of the rectangular coordinate system as the pole; 2. Use the non-negative half of the x-axis as the polar axis.

Assistant:

Compression Output: Okay, so I got r as 3. Now I need to find the angle θ . I remember that θ is the arctangent of y over x . So $\theta = \arctan(\frac{y}{x})$. Since the angle from the positive x-axis to the point $(0, 3)$ is straight up along the y-axis, which is 90 degrees or $\frac{\pi}{2}$ radians. So θ is $\frac{\pi}{2}$.

Original Output: Okay, so putting it all together, the polar coordinates are $(3, \frac{\pi}{2})$. Let me just double-check my work to make sure I didn't make any mistakes.

Local Entropy: 0.1484

KL divergence: 0.3713

Original Output: Okay, so let me recap. The point is $(0, 3)$. To convert to polar coordinates, I use $r = \sqrt{x^2 + y^2}$. Plugging in $x = 0$ and $y = 3$, I get $r = 3$. Then, for the angle, since x is 0 and y is positive, the angle is $\frac{\pi}{2}$. So the polar coordinates are $(3, \frac{\pi}{2})$. That seems correct.

Local Entropy: 0.3764

KL divergence: 0.0208

[Early Stopping]

STACK's Output for Case 1

Question: Convert the point $(0, 3)$ in rectangular coordinates to polar coordinates. Enter your answer in the form (r, θ) , where $r > 0$ and $0 \leq \theta < 2\pi$.

External Knowledge: Rectangular coordinates are represented as (x, y) , and polar coordinates are represented as (r, θ) , where r is the distance to the origin and θ is the angle with the x-axis. To convert from rectangular coordinates to polar coordinates, the formulas are as follows: $r = \sqrt{x^2 + y^2}$, $\theta = \arctan(\frac{y}{x})$. The preconditions are as follows: 1. Take the origin of the rectangular coordinate system as the pole; 2. Use the non-negative half of the x-axis as the polar axis.

Output:

To convert the rectangular coordinates $(0, 3)$ to polar coordinates (r, θ) , we can use the following formulas: \n1. $r = \sqrt{x^2 + y^2}$ \n2. $\theta = \arctan(\frac{y}{x})$. Given the point $(0, 3)$, we have $x = 0$ and $y = 3$.

First, let's calculate r : $r = \sqrt{0^2 + 3^2} = \sqrt{0 + 9} = \sqrt{9} = 3$. So, $r = 3$.

Next, let's calculate θ . Since $x = 0$, we have a point on the y-axis. The arctangent function $\arctan(\frac{y}{x})$ is undefined when $x = 0$ (since division by zero is not allowed). However, we can determine the angle based on the position of the point.

When $x = 0$ and $y > 0$, the point is on the positive y-axis. The angle θ for this position is $\frac{\pi}{2}$ radians (90 degrees). Therefore, $\theta = \frac{\pi}{2}$.

Putting it all together, the polar coordinates are $(3, \frac{\pi}{2})$.

****Final Answer****The polar coordinates of the point $(0, 3)$ are $\boxed{\left(3, \frac{\pi}{2}\right)}$.

Ground Truth: $\left(3, \frac{\pi}{2}\right)$

[True]

Original Output for Case 2

Question: The expression $2 \cdot 3 \cdot 4 \cdot 5 + 1$ is equal to 121, since multiplication is carried out before addition. However, we can obtain values other than 121 for this expression if we are allowed to change it by inserting parentheses. For example, we can obtain 144 by writing $2 \cdot (3 \cdot 4) \cdot (5 + 1) = 144$. In total, how many values can be obtained from the expression $2 \cdot 3 \cdot 4 \cdot 5 + 1$ by inserting parentheses? (Note that rearranging terms is not allowed, only inserting parentheses).

Output:

...

Wait, so that's another way to get 144.

So, 144 is achievable in multiple ways.

But earlier, I had 36 as another value when I incorrectly grouped $3 \cdot 4$ and $5 + 1$ with 2. But that might not be a valid parenthesization because it changes the structure.

So, perhaps 36 is not a valid result.

Wait, let me check again.

If I group $2 \cdot (3 \cdot 4) + (5 + 1)$, that would be $2 \cdot 12 + 6 = 24 + 6 = 30$. But in the original expression, it's $2 \cdot 3 \cdot 4 \cdot 5 + 1$, so this grouping would require changing the order of operations, which isn't allowed.

So, that's not a valid parenthesization.

Therefore, only groupings that change the order of operations by grouping adjacent operations are allowed.

Therefore, the different groupings that result in different values are:

1. Grouping the addition with 5: 144.
2. Grouping the addition with 4 and 5: 126.
3. Grouping the addition with 3,4,5: 122.
4. Grouping the addition with 2,3,4,5: 121.

So, that's four different values.

Wait, but in the first grouping, when I grouped the addition with 5, I had 144, but when I grouped the addition with 4 and 5, I also got 144. So, that suggests that grouping the addition with 5 and grouping it with 4 and 5 both result in 144. So, it's the same value.

Similarly, grouping the addition with 3,4,5 gives 122, which is different.

Grouping the addition with 2,3,4,5 gives 121.

So, in total, we have 121, 122, 126, and 144.

Therefore, four different values.

...

Wait, but if you group $3 + 4$, it's not part of a multiplication.

So, for example, $2 \cdot (3 + 4) \cdot 5 + 1$. But in the original expression, it's $2 \cdot 3 \cdot 4 \cdot 5 + 1$. So, grouping $3 + 4$ would require changing the structure, which isn't allowed.

Therefore, that's not a valid parenthesization.

So, I think I can conclude that the different values are 121, 122, 126, and 144.

Therefore, the total number of different values is 4.

But in reality, in the original expression, it's left-associative, so $2 * 3 * 4 * 5$ is equivalent to $((2 * 3) * 4) * 5$. So, to group $3 * (4 + 5)$, you have to change the order of operations, which is allowed by inserting parentheses.

Therefore, is 55 a valid result?

Original Output for Case 2

Compute: $3 + 4 = 7$, $5 + 1 = 6$, $2 * 7 * 6 = 84$. But in the original expression, it's a linear chain, so grouping $3 + 4$ and $5 + 1$ is not allowed. Therefore, 84 is invalid.

...

Wait, when I grouped the addition with 3,4,5, I got 122. But is 122 achievable in another way? Wait, if I group 3,4,5,1, it's 121. Wait, but 122 is unique. Similarly, 55 is unique. So, yes, five different values. Therefore, I think the answer is 5.

Final Answer The number of different values is $\boxed{5}$. </think>

To determine how many different values can be obtained by inserting parentheses into the expression $2 \cdot 3 \cdot 4 \cdot 5 + 1$, we need to consider different ways of parenthesizing the multiplication and addition operations.

1. **Original Expression**: The expression is evaluated left to right without any parentheses.

$$2 \cdot 3 \cdot 4 \cdot 5 + 1 = 120 + 1 = 121$$

2. **Grouping the Addition with 5**: Insert parentheses around 4 and 5.

$$2 \cdot 3 \cdot (4 \cdot 5) + 1 = 2 \cdot 3 \cdot 20 + 1 = 120 + 1 = 121$$

This gives the same result as the original expression.

3. **Grouping the Addition with 4 and 5**: Insert parentheses around 3, 4, and 5.

$$2 \cdot (3 \cdot 4 \cdot 5) + 1 = 2 \cdot 60 + 1 = 120 + 1 = 121$$

This also gives the same result as the original expression.

4. **Grouping the Addition with 3, 4, and 5**: Insert parentheses around 3, 4, and 5.

$$2 \cdot (3 \cdot (4 \cdot 5)) + 1 = 2 \cdot 60 + 1 = 120 + 1 = 121$$

This again gives the same result as the original expression.

5. **Grouping the Addition with 4 and 5**: Insert parentheses around 4 and 5.

$$2 \cdot (3 \cdot (4 + 5)) + 1 = 2 \cdot 27 + 1 = 54 + 1 = 55$$

6. **Grouping the Addition with 3, 4, and 5**: Insert parentheses around 3, 4, and 5.

$$(2 \cdot 3) \cdot (4 \cdot 5) + 1 = 6 \cdot 20 + 1 = 120 + 1 = 121$$

7. **Grouping the Addition with 5**: Insert parentheses around 5.

$$(2 \cdot 3 \cdot 4) \cdot (5 + 1) = 24 \cdot 6 = 144$$

8. **Grouping the Addition with 3, 4, and 5**: Insert parentheses around 3, 4, and 5.

$$((2 \cdot 3) \cdot 4) \cdot (5 + 1) = 24 \cdot 6 = 144$$

After evaluating all possible groupings, the different values obtained are 55, 121, 122, 126, and 144. Therefore, the number of different values is:

$\boxed{5}$

Ground Truth: 4

[False]

STACK's Output for Case 2

Question: The expression $2 \cdot 3 \cdot 4 \cdot 5 + 1$ is equal to 121, since multiplication is carried out before addition. However, we can obtain values other than 121 for this expression if we are allowed to change it by inserting parentheses. For example, we can obtain 144 by writing $2 \cdot (3 \cdot 4) \cdot (5 + 1) = 144$. In total, how many values can be obtained from the expression $2 \cdot 3 \cdot 4 \cdot 5 + 1$ by inserting parentheses? (Note that rearranging terms is not allowed, only inserting parentheses).

External Knowledge: Step 1: Initial Calculation Calculate the initial expression without any parentheses: $2 \cdot 3 \cdot 4 \cdot 5 + 1$. Step 2: Possible Parentheses Placements Identify the possible locations where parentheses can be inserted to change the order of operations. Step 3: Evaluate Parentheses Configurations Calculate the expression for each valid placement of parentheses. Step 4: Count Unique Results Determine the total number of unique values obtained by inserting parentheses.

Output:

The expression $2 \cdot 3 \cdot 4 \cdot 5 + 1$ is equal to 121 when evaluated from left to right due to the order of operations. However, by inserting parentheses, we can obtain different values.

We need to consider all possible ways to insert parentheses in the expression $2 \cdot 3 \cdot 4 \cdot 5 + 1$ without rearranging the terms. The key is to explore different groupings of the operations.

1. **Default evaluation without parentheses:**

$$2 \cdot 3 \cdot 4 \cdot 5 + 1 = 120 + 1 = 121$$

2. **Grouping the first two terms and the rest:**

$$(2 \cdot 3) \cdot 4 \cdot 5 + 1 = 6 \cdot 4 \cdot 5 + 1 = 120 + 1 = 121$$

This does not change the result.

3. **Grouping the first three terms and the last term:**

$$(2 \cdot 3 \cdot 4) \cdot 5 + 1 = 24 \cdot 5 + 1 = 120 + 1 = 121$$

4. **Grouping the last two terms first:**

$$2 \cdot 3 \cdot (4 \cdot 5) + 1 = 2 \cdot 3 \cdot 20 + 1 = 120 + 1 = 121$$

This does not change the result.

5. **Grouping the middle terms first:**

$$2 \cdot (3 \cdot 4) \cdot (5 + 1) = 2 \cdot 12 \cdot 6 = 144$$

6. **Grouping the last term with the addition:**

$$2 \cdot (3 \cdot 4 \cdot 5 + 1) = 2 \cdot (60 + 1) = 2 \cdot 61 = 122$$

7. **Grouping the middle terms and then adding:**

$$2 \cdot (3 \cdot (4 \cdot 5 + 1)) = 2 \cdot (3 \cdot 21) = 2 \cdot 63 = 126$$

By evaluating all possible groupings, we find the distinct values: 121, 144, 126, and 122. Thus, the total number of distinct values is .

Ground Truth: 4

[True]