# THINK RIGHT: LEARNING TO MITIGATE UNDER-OVER THINKING VIA ADAPTIVE, ATTENTIVE COMPRESSION

## Anonymous authors

000

001

002003004

010 011

012

013

014

016

017

018

019

021

024

025

026

027

028

029

031

033

034

037

040

041

042

043

044

046

047

048

049

051

052

Paper under double-blind review

#### **ABSTRACT**

Recent thinking models are capable of solving complex reasoning tasks by scaling test-time compute across various domains, but this scaling must be allocated in line with task difficulty. On one hand, short reasoning (underthinking) leads to errors on harder problems that require extended reasoning steps; but, excessively long reasoning (overthinking) can be token-inefficient by generating unnecessary steps even after reaching a correct intermediate solution. We refer to this as **under-adaptivity**, where the model fails to modulate its response length appropriately given problems of varying difficulty. To address under-adaptivity and strike a balance between under- and overthinking, we propose TRAAC (Think Right with Adaptive, Attentive Compression), an online post-training RL method that leverages the model's self-attention over a long reasoning trajectory to identify important steps and prune redundant ones. TRAAC also estimates difficulty and incorporates into training rewards, thereby learning to allocate reasoning budget commensurate with example difficulty. Our approach improves accuracy, reduces reasoning steps, and enables adaptive thinking compared to base models and other RL baselines. Across a variety of tasks (AIME, AMC, GPQA-D, BBEH), TRAAC (Qwen3-4B) achieves an average absolute accuracy gain of 8.4% with a relative reduction in reasoning length of 36.8% compared to the base model, and a 7.9% accuracy gain paired with a 29.4% length drop compared to the best RL baseline. TRAAC also shows strong generalization: although the models are trained on math datasets, they show accuracy and efficiency gains on out-of-distribution non-math datasets like GPQA-D, BBEH, and OptimalThinkingBench. Our analysis further verifies that TRAAC provides fine-grained adjustments to thinking budget based on difficulty and that a combination of task-difficulty calibration and attention-based compression yields gains across diverse tasks.

#### 1 Introduction

Recent advancements in thinking models have enabled language models to solve complex reasoning tasks (DeepSeek-AI et al., 2025; OpenAI et al., 2024; Team, 2025). These models extend the chain-of-thought (CoT; Wei et al., 2023) paradigm with online reinforcement learning (RL; Shao et al., 2024), allowing them to refine intermediate solutions as well as sequentially scaling the number of tokens (i.e., compute) to arrive at the final answer. While such approaches show strong promise for harder problems in domains like mathematics, programming, and logical puzzles (Xie et al., 2025; Chen et al., 2025), their accuracy and utility remain capped by a failure to regulate their reasoning length. On one hand, *underthinking* arises when models terminate too early on harder problems, yielding an incorrect final answer. On the other hand, *overthinking* occurs when models think excessively for simpler tasks, inflating test-time computation (Marjanović et al., 2025; Wu et al., 2025; Cuadron et al., 2025), and reducing efficiency. This highlights the need for adaptive thinking (Saha et al., 2025; Chen et al., 2024; Snell et al., 2024; Aggarwal & Welleck, 2025), where models dynamically allocate thinking based on difficulty.

We refer to the phenomenon of models misallocating thinking budget – illustrated in Fig. 1 – as *under-adaptivity*. Addressing under-adaptivity is crucial for improving both performance and efficiency of long-thinking models, as dynamic reasoning effort allocation can enable better reason-

<sup>&</sup>lt;sup>1</sup>Codebase available in the supplementary, and will be released publicly upon acceptance.

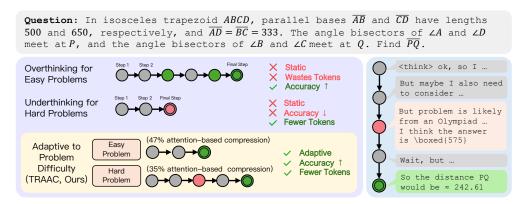


Figure 1: Overthinking on easy problems wastes tokens despite being able to maintain decent accuracy. On the other hand, underthinking on hard problems saves token budgets but fails to maintain accuracy. TRAAC addresses this trade-off by adapting to problem difficulty (estimated during training), via attention-based compression and, enabling intelligent resource allocation while improving both accuracy and efficiency.

ing exploration in harder problems, while avoiding wasteful computation on problems requiring minimal reasoning. Prior work has generally addressed the upper end of under-adaptivity, i.e., improving thinking efficiency. These works employ supervised fine-tuning on compressed CoT (Xia et al., 2025), using user control signals such as early stopping during inference (Muennighoff et al., 2025), or RL methods with length penalties (Arora & Zanette, 2025; Hou et al., 2025). Other more adaptive work has employed budget-aware reward shaping with a binary choice between thinking or not thinking (Zhang et al., 2025b). While such work can reduce token usage, its performance is typically bounded by the accuracy of the underlying model being trained, and often trades performance for efficiency. Our work aims to beat this trade-off and improve both efficiency and accuracy by providing finer-grained feedback through difficulty-adaptive compression, where the degree of compression is dynamically adapted to task difficulty to address under-adaptivity.

To address these gaps, we introduce TRAAC (Think Right with Adaptive, Attentive Compression), a GRPO-based (Shao et al., 2024) post-training method that incorporates an **online, difficulty-adaptive, attention-based compression module** to adaptively prune the reasoning trajectory (an entire chain in Fig. 1) based on estimated task difficulty. Our method teaches the model to compress the context that it should pay attention to, such that it contains only relevant material without getting distracted or skewed in wrong directions (Weston & Sukhbaatar, 2023). Specifically, we compute the attention score averaged across layers and heads of the model for each reasoning step (illustrated as nodes in Fig. 1 (right)) from the 

 Think> token and compress reasoning steps that are least attended to, based on the assumption that these are the least important tokens contributing to the final answer. During online training, the level of attention-compression is determined by task difficulty, as estimated by the pass rate during GRPO rollout, making the model more adaptive. For harder problems, TRAAC maintains a low compression rate, allowing the model to extend its reasoning trajectory, which increases the likelihood of reaching the correct final answer. For easier problems, it applies a higher compression rate to aggressively compress once the correct final answer is reached.

We evaluate TRAAC on two strong off-the-shelf reasoning models, Qwen3-4B (Team, 2025) and Deepseek-Qwen-7B (DeepSeek-AI et al., 2025), across multiple benchmarks: AMC (AMC, 2023), AIME (AIME, 2024), GPQA-Diamond (Rein et al., 2023), BBEH (Big Bench Extra Hard; Kazemi et al., 2025), and OptimalThinkingBench (Aggarwal et al., 2025). Our experiments demonstrate that TRAAC consistently adapts to problem difficulty, yielding improvements in efficiency on simple tasks and stronger accuracy on complex tasks. Averaged across AMC, AIME, GPQA-D, and BBEH, TRAAC (Qwen3-4B) achieves an average absolute improvement of 8.4% in accuracy while a relative reduction in reasoning length by 36.8% compared to the base model. When compared to the next-best performing baseline, AdaptThink (Zhang et al., 2025b), we achieve an average accuracy improvement of 7.9% and 29.4% efficiency gain. We test TRAAC on OptimalThinkingBench (Aggarwal et al., 2025), and find TRAAC improves by 7.36 points on Qwen3-4B and 12.55 points on Deepseek-Qwen-7B over the base model according to Aggarwal et al. (2025)'s F1 metric – designed to measure both performance and efficiency. Moreover, TRAAC is trained on a math-specific dataset;

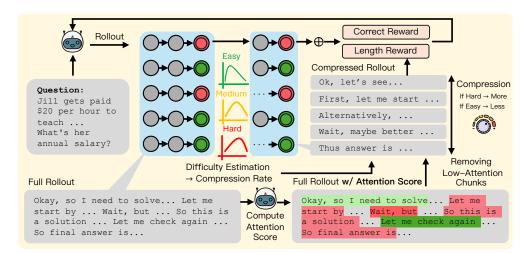


Figure 2: Overview of TRAAC. Given a problem, the model first generates N rollouts, and the pass rate of these rollouts is used to estimate the problem's difficulty (easy, medium, or hard). Next, the generated reasoning is fed back into the model, which is asked to compute the attention score of each reasoning token from 
 buring this attention-based compression step, we remove steps with lower scores. The degree of removal is determined by the estimated difficulty: easier problems undergo more aggressive compression. Finally, we compute the correctness and length rewards using the compressed reasoning trajectory, and these rewards are used to update the policy.

evaluation on various benchmarks such as GPQA-D, BBEH, OverthinkingBench, and UnderthinkingBench shows generalizability performance. Among these OOD tasks, TRAAC shows an average improvement of 3% on Qwen3-4B, with a maximum improvement of 6.8% on UnderthinkBench, along with an average 40% reduction in response length across OOD tasks. Our analysis and ablations demonstrate that through difficulty level calibration, TRAAC learns to dynamically adjust its compression ratio – with lower compression on difficult tasks and higher compression on easier ones, which translates into performance gains across diverse difficulty tasks.

# 2 TRAAC: THINK RIGHT WITH ADAPTIVE ATTENTIVE COMPRESSION

In this section, we introduce our proposed TRAAC method in detail (also shown in Fig. 2). It is designed to mitigate under-adaptivity, which leads to resource misallocation during test-time. The main challenge lies in the efficient identification of low-importance tokens and making the attention-based compression adaptive to the task's difficulty. To this end, TRAAC employs an attention-based compression module that calibrates its degree of compression based on estimated task difficulty and prunes unnecessary reasoning steps while preserving essential information.

## 2.1 PROBLEM FORMULATION IN TRAAC

TRAAC is based on Group Reward Policy Optimization (GRPO; Shao et al., 2024), which is an online reinforcement learning (RL) framework that extends Proximal Policy Optimization (Schulman et al., 2017) by eliminating the critic and instead estimating the baseline from a group of sampled responses. Let  $\pi_{\theta}$  denote the policy model and q the input query. Given q, the model generates an output  $y = \cot(r, a)$  where cat is the concatenate function, r is the complete reasoning trajectory, and a is the final answer, separated by the delimiter </think>. An attention-based compression module  $\mathcal{C}$  (described below) produces a compressed reasoning trajectory:  $r_{\text{comp}} = \mathcal{C}(r)$ . At each training step, the model generates N rollouts,  $\{y^i\}_{i=1}^N$ , where each rollout  $y^i = \cot(r^i, a^i)$  (see "rollout" arrow in Fig. 2). The advantage of each rollout is estimated using the standard GRPO objective (details in Appendix A.4). The task difficulty d is estimated from these rollouts as the proportion of correct answers among the N samples (Zhang & Zuo, 2025; Huang et al., 2025). We show this in Fig. 2 by classifying a problem to easy, medium or hard based on d. Task difficulty d is then used to (i) modulate the compression ratio applied to the reasoning trajectory r, and (ii)

assign rewards to each rollout. The answer is regenerated based on the compressed trajectory and the advantage is estimated using both the original rollouts and their compressed counterparts.

#### 2.2 Adaptive, Attentive Compression Module

The goal of the compression module is to identify and remove redundant reasoning steps by evaluating attention scores assigned to each token.

**Attention-Based Compression.** To calculate the attention score assigned to each token, we pass the reasoning trajectory r (full rollout in Fig. 2) through the initial policy model. As compared to other compression-based methods (Cheng et al., 2025; Lu et al., 2025), TRAAC does not rely on external models for annotating reasoning steps. To segment the reasoning trajectory r into reasoning steps, we split it at special control tokens such as "wait", "alternative", "Let me think again", etc (complete list Appendix A.3.2). For the current thinking models, </think> marks the end of a reasoning trajectory, followed by the final answer. Choi et al. (2025) show that </think> attends to key reasoning steps that contain crucial information for deriving the final answer, therefore, for each token  $t_j$  in the reasoning steps, its importance score is defined as the aggregated attention from the delimiter </think> across all layers and heads:

$$s_j = \frac{1}{LH} \sum_{\ell=1}^L \sum_{h=1}^H \alpha^{(\ell,h)}_{ \to t_j},$$

where L is the number of layers, H is the number of heads per layer, and  $\alpha^{(\ell,h)}_{</\text{think}>\to t_j}$  is the attention weight from </think> to token  $t_j$  in head h of layer  $\ell$ . Before computing the attention score of each token, consistent with prior work (Muennighoff et al., 2025; Choi et al., 2025), we also append an auxiliary prompt at the end of the reasoning trajectory (see Appendix A.3.1). This encourages the model to distill the reasoning process into its most salient steps, thereby enabling the delimiter token </think> to attend to the most informative parts of the reasoning trajectory (highlighted in green). As shown in Fig. 2 (bottom-right), the model assigns low attention scores to reasoning steps that do not contribute to the final correct answer (highlighted in red), effectively pruning unnecessary cyclic self-corrections and verification loops. Finally, the importance score of a reasoning step  $C_k$ , consisting of tokens  $\{t_j\}_{j\in C_k}$ , is then computed as the mean of its token-level scores:  $s_{C_k} = \frac{1}{|C_k|} \sum_{j\in C_k} s_j$ . Steps with lower importance scores are pruned, yielding the compressed reasoning trajectory  $r_{\text{comp}}^i$ .

**Difficulty-Level Calibration.** To address **under-adaptivity**, the pruning strategy is further adapted to task difficulty, i.e., for easier tasks, a larger proportion of reasoning steps are removed, encouraging the model to condense its reasoning more aggressively (see "compression" on the right of Fig. 2). The difficulty of a task is estimated, based on the pass rate of each problem, during rollout. From the estimated difficulty level, each problem set is categorized among three difficulty levels: easy, medium, and hard, with a higher pass rate indicating easier problems and vice versa. Each category is assigned a compression rate to determine the degree of redundant steps to prune from the reasoning trajectory, with a higher compression for easier problems and a lower compression for hard problems. In addition, to keep these constraints adaptive to the amount of redundancy in the steps, we calculate the *uniformity* of the attention score distribution. When the distribution of  $\{s_j\}$  is close to uniform, indicating that no step or token within a step stands out as significantly more important, the compression rate is reduced to avoid removing potentially useful reasoning steps. More details on calculating the uniformity score can be found in Appendix A.3.3. The difficulty estimate d is further incorporated with the reward calculation described below.

## 2.3 REWARDS

Following standard GRPO practice of having a verifiable reward system (Shao et al., 2024), our setup comprises three different reward signals to guide the model to generate correct adaptive length responses based on the difficulty of the task:

• Correctness Reward (CR): A high-weight reward is assigned to outputs that produce the correct final answer. A high score over other rewards is used to ensure that correctness remains the primary optimization objective, regardless of the reasoning trajectory length.

- **Format Reward:** A structure reward to ensure the presence of special delimiter tokens such as <think> and </think>, ensuring that trajectory r and final answer a are easily distinguishable.
- Length Reward (LR): To regulate the verbosity of the reasoning process, we define a length-based reward that penalizes unnecessarily long reasoning traces while adapting to task difficulty. Based on our initial experiment, simply favoring shorter rollouts led to a drastic decrease in response length along with model accuracy; therefore we introduce a sigmoid-based smoothing mechanism that provides a soft bonus ( $\beta$ ) for rollouts beyond the median length. This prevents sharp drops in reward for slightly longer reasoning and helps stabilize training. During each training step, rollouts are partitioned into bins according to their calculated difficulty. As mentioned above, we use the pass rate of the rollouts to categorize them into three difficulty bins: easy, medium and hard. For each bin, we maintain a different distribution  $\mathcal{L}_d = \{\ell_1, \ell_2, \ldots, \ell_m\}$  for each difficulty category, where each  $\ell_i$  denotes the reasoning length of a rollout within that difficulty category d. Let  $\ell$  be the length of the current rollout. The normalized length score is computed as:  $L_{\text{norm}} = (L_{\text{max}} \ell)/\text{max}(L_{\text{max}} L_{\text{min}}, \epsilon)$ , where  $\epsilon > 0$  prevents division by zero and  $L_{\text{min}} = \min(\mathcal{L}), L_{\text{max}} = \max(\mathcal{L})$ . To avoid a sharp cutoff around the median, we add a smooth bonus term:

$$\beta = 1/\Big(1 + \exp\Big(\frac{\ell - \operatorname{median}(\mathcal{L})}{0.1 \times \operatorname{median}(\mathcal{L})}\Big)\Big),$$

where median  $(\mathcal{L})$  = median of the set. The final length reward becomes  $r_{\text{length}} = \max(L_{\text{norm}}, \beta)$ . Note that length reward is only provided to a rollout if it reaches a final correct answer. Moreover, to ensure stability when calculating  $L_{min}$ ,  $L_{max}$ , and medium  $(\mathcal{L})$ , we maintain a sliding window over the last 10 steps for each difficulty bin, thereby avoiding drastic fluctuations during training.

The final reward for each rollout during GRPO training is the combination of correctness, format, and length rewards (c.f. range of each reward in Appendix A.5.2).

## 3 EXPERIMENTAL SETUP

**Models.** We adopt two reasoning models, DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI et al., 2025) (Deepseek-Qwen-7B) and Qwen3-4B (Team, 2025) as our base models.

**Datasets.** We train the model using DAPO-Math-17k (Yu et al., 2025), a math dataset that has verifiable answer. For evaluation, we use a diverse set of benchmarks, including AIME (AIME, 2024), AMC (AMC, 2023), GPQA-D (Rein et al., 2023), OverthinkingBench/ UnderthinkingBench (Aggarwal et al., 2025), and Big Bench Extra Hard (BBEH) (Kazemi et al., 2025). Among the evaluation datasets, only AIME and AMC are math-specific, while the remaining benchmarks represent out-of-distribution settings. Further dataset details and their sizes are provided in Appendix A.1.

**Evaluation.** For each evaluation run, we set temperature to 1.0, and the maximum response length is set to 10k. For each dataset, the mean accuracy and mean response length across 5 runs are reported. For the overthink split, we also report the  $AUC_{OAA}$  (Aggarwal et al., 2025), directly used from their work. To aggregate performance across all thresholds, we compute the Area Under the Curve (AUC). More details about these metrics can be found in Appendix A.5.3. Intuitively, a higher  $AUC_{OAA}$  indicates that the model sustains stronger accuracy while minimizing unnecessary reasoning across thresholds. Following evaluation from Aggarwal et al. (2025) for computing the OptimalThinkingBench score, we combined the  $AUC_{OAA}$  from OverthinkingBench and accuracy from UnderthinkingBench into a single F1 score.

**Training.** During the GRPO rollout, we keep a high temperature of 1.0 and sample 8 rollouts at each step. Due to computational constraints, we set the maximum response length to 10k (see Appendix A.5.3 for other hyperparameter details). For difficulty calibration, we bin problems into easy, medium, and hard categories, assigning the categories decreasing compression scores.

Baselines. We compare TRAAC with 5 strong baselines: (1) Base model: off-the-shelf reasoning model, (2) TokenSkip: An SFT based baseline as described by Xia et al. (2025) that fine-tunes the model over compressed CoT training data. (3) L1-Max: An RL framework proposed by Aggarwal & Welleck (2025) that optimizes for accuracy while adhering to user-specific length constraints. We used the constraint "Think for a maximum of 10000 tokens." during its training. (4) LC-R1: A compression-based RL framework by Cheng et al. (2025) that uses an externally trained model to remove invalid portions of the thinking process. (5) AdaptThink: Different from the

Table 1: Performance comparison of TRAAC with various baselines. Each model is evaulate across various benchmarks, and Acc: accuracy(%) and Len: average Response Length(k) are reported. TRAAC on average shows the highest performance gain.

| Method      | AIME  |       | AMC   |         | GPQA-D    |        | BBEH  |       | Average |       |
|-------------|-------|-------|-------|---------|-----------|--------|-------|-------|---------|-------|
|             | Acc.↑ | Len.↓ | Acc.↑ | Len.↓   | Acc.↑     | Len.↓  | Acc.↑ | Len.↓ | Acc.↑   | Len.↓ |
|             |       |       |       | Qwo     | en3-4B    |        |       |       |         |       |
| Base Model  | 27.64 | 9.2   | 68.19 | 7.0     | 45.18     | 7.6    | 18.28 | 6.7   | 39.8    | 7.6   |
| TokenSkip   | 5.84  | 9.6   | 27.71 | 8.7     | 32.32     | 7.8    | 11.91 | 7.2   | 19.4    | 8.3   |
| L1-Max      | 30.11 | 7.1   | 63.61 | 5.8     | 43.23     | 5.8    | 14.91 | 5.0   | 38.0    | 5.9   |
| LC-R1       | 13.48 | 2.6   | 56.38 | 1.7     | 26.67     | 1.5    | 12.35 | 1.9   | 27.2    | 1.9   |
| Adapt Think | 36.63 | 8.4   | 72.77 | 5.8     | 44.04     | 6.7    | 7.87  | 6.2   | 40.3    | 6.8   |
| TRAAC       | 45.45 | 6.7   | 79.52 | 4.2     | 47.21     | 4.2    | 20.59 | 4.3   | 48.2    | 4.8   |
|             |       |       | Deep  | Seek-R1 | Distill-Q | wen-7B |       |       |         |       |
| Base Model  | 33.71 | 8.2   | 74.22 | 5.7     | 43.55     | 7.1    | 10.61 | 5.9   | 40.5    | 6.7   |
| TokenSkip   | 24.94 | 8.5   | 52.05 | 6.8     | 34.24     | 7.0    | 6.30  | 6.4   | 29.4    | 7.2   |
| L1-Max      | 31.01 | 3.1   | 75.90 | 2.2     | 23.54     | 1.9    | 13.43 | 2.1   | 36.0    | 2.3   |
| LC-R1       | 6.07  | 4.0   | 37.35 | 3.5     | 28.78     | 2.5    | 9.09  | 1.7   | 20.3    | 2.9   |
| Adapt Think | 38.88 | 7.1   | 75.66 | 4.1     | 19.29     | 4.8    | 6.17  | 5.2   | 35.0    | 5.3   |
| TRAAC       | 38.60 | 7.3   | 77.83 | 4.5     | 47.31     | 6.2    | 11.55 | 5.2   | 43.8    | 5.8   |

above baselines, AdaptThink is an adaptive RL framework described by Zhang et al. (2025b), that enables reasoning models to choose between "thinking" and "no-thinking" modes and poses it as a constraint optimization problem that encourages the model to choose no-thinking while maintaining performance. Prompts used for all baselines in Appendix A.5.5.

### 4 RESULT AND DISCUSSION

#### 4.1 MAIN RESULTS

**TRAAC improve both performance and efficiency.** Tables 1 show the performance of TRAAC compared to other baselines on AIME, AMC, GPQA-D, BBEH (Big Bench Extra Hard) benchmarks. TRAAC (Owen3-4B) achieves an average accuracy improvement of 8.4% while reducing reasoning length by 36.8% compared to the base model. Similarly, TRAAC (Deepseek-Qwen-7B) improves accuracy by 3.3% with a 13.4% reduction in length. When compared to the SFT baseline Token-Skip (Xia et al., 2025), TRAAC outperforms in terms of performance and efficiency for both models, Qwen3-4B and Deepseek-Qwen-7B. Similarly, L1-Max (Aggarwal & Welleck, 2025), an RL-based method that penalizes long responses, also solely focuses on efficiency gains, at a slight cost of overall performance. Additionally, the compression-based RL framework LC-R1 (Cheng et al., 2025) improves the efficiency of the model at the cost of a 12.6% drop for Qwen3-4B and 20.2% drop for Deepseek-Qwen-7B, when compared with base models, respectively. On average for Qwen3-4B, TRAAC outperforms L1-Max by 10.2% on Qwen3-4B and by 7.9% on Deepseek-Qwen-7B. Similarly, TRAAC also outperforms LC-R1 by 21% on Qwen3-4B and 23% on Deepseek-Qwen-7B. Moreover, given the same token budget, of approximately 7k, TRAAC (Qwen3-4B) on AIME outperforms L1-Max by 15%. These results highlight that, unlike methods that prioritize only efficiency, TRAAC simultaneously delivers both higher accuracy and shorter reasoning traces.

TRAAC generalizes across domains. Recall that for training TRAAC we used data from DAPO-Math-17k (Yu et al., 2025), which is a math reasoning dataset. In addition to math datasets, we also evaluate TRAAC on several out-of-domain (OOD) tasks, including GPQA-D, BBEH, OverthinkingBench, and UnderthinkingBench (Table 2). Among these OOD tasks, TRAAC shows an average improvement of 3% on Qwen3-4B and 2.8% on Deepseek-Qwen-7B compared to the base model, with improvement as high as 6.8% on UnderthinkingBench, which covers 100 diverse reasoning tasks from Reasoning Gym (Stojanovski et al., 2025). In addition, TRAAC reduces reasoning tokens by 40% on Qwen3-4B and 20% on Deepseek-Qwen-7B, demonstrating substantially higher efficiency while also boosting accuracy across benchmarks. This indicates that TRAAC learns a generalizable compression strategy that transfers from math to other reasoning domains.

Table 2: Performance of TRAAC and various baselines on OptimalThinkingBench (OTB). For UnderthinkingBench we report the Acc: Accuracy(%), and Len: Average Response length(k). For OverthinkingBench, in addition to Acc. and Len. we also report the  $AUC_{OAA}$ .

| Method      | Ov                          | erthinkir | ngBench              | Underth | ОТВ   |       |  |  |  |
|-------------|-----------------------------|-----------|----------------------|---------|-------|-------|--|--|--|
|             | Acc.↑                       | Len.↓     | AUC <sub>OAA</sub> ↑ | Acc.↑   | Len.↓ | F1↑   |  |  |  |
|             |                             |           | Qwen3-4B             |         |       |       |  |  |  |
| Base Model  | 90.02                       | 1.2       | 80.06                | 34.33   | 7.1   | 48.05 |  |  |  |
| TokenSkip   | 78.15                       | 3.5       | 57.88                | 14.80   | 7.9   | 23.57 |  |  |  |
| L1-Max      | 87.22                       | 0.9       | 1.11                 | 21.27   | 6.3   | 2.10  |  |  |  |
| LC-R1       | 78.62                       | 0.3       | 64.20                | 14.95   | 1.3   | 24.25 |  |  |  |
| Adapt Think | 68.83                       | 8.2       | 63.44                | 18.80   | 6.0   | 29.01 |  |  |  |
| TRAAC       | 89.79                       | 0.6       | 85.06                | 41.09   | 4.7   | 55.41 |  |  |  |
|             | DeepSeek-R1-Distill-Qwen-7B |           |                      |         |       |       |  |  |  |
| Base Model  | 78.45                       | 0.9       | 72.38                | 12.69   | 6.2   | 21.60 |  |  |  |
| TokenSkip   | 57.03                       | 3.9       | 40.77                | 8.55    | 7.2   | 14.13 |  |  |  |
| L1-Max      | 73.18                       | 1.0       | 66.01                | 20.07   | 2.0   | 30.78 |  |  |  |
| LC-R1       | 76.08                       | 0.9       | 69.81                | 7.16    | 2.5   | 12.99 |  |  |  |
| Adapt Think | 73.41                       | 0.4       | 70.72                | 13.13   | 4.6   | 22.14 |  |  |  |
| TRAAC       | 81.81                       | 1.0       | 72.89                | 22.30   | 5.9   | 34.15 |  |  |  |

**TRAAC learns to adaptively allocate token budget.** Among the baselines in Tables 1 and 2, we also compare TRAAC against an adaptive RL method, AdaptThink (Zhang et al., 2025b), which teaches the model to use distinct "thinking" vs. "non-thinking" modes for hard and easy problems, respectively. On Qwen3-4B, TRAAC outperforms AdaptThink by 7.9% while also reducing tokens by 29.4%, highlighting that a flexible adaptive strategy is more effective in handling diverse problem difficulties. Table 2 further tests on the OverthinkingBench/UnderthinkingBench (Aggarwal et al., 2025). OverthinkingBench is designed to measure excessive use of thinking tokens on simple queries. On the other hand, UnderthinkingBench evaluates how necessary "thinking" is based on problem difficulty. Taken together, TRAAC improves overall F1 performance by 7.36% on Qwen3-4B, and 12.55% on Deepseek-Qwen-7B over base model, indicating that TRAAC enables the model to avoid both overthinking on simple problems and underthinking on complex ones(Aggarwal et al., 2025). Against AdaptThink, TRAAC achieves a 26% gain on Qwen3-4B and a 12% gain on Deepseek-Qwen-7B, underscoring its ability to adaptively allocate reasoning effort and adjust token budgets based on problem difficulty. On OverthinkingBench, we measure overthinking using the AUCOAA metric, which rewards models that solve very easy problems correctly while using minimal tokens (ideally 0). Compared to the base model, TRAAC (Qwen3-4B) improves AUCOAA by 5% and Deepseek-Qwen-7B by 0.5%. Relative to AdaptThink, TRAAC gains 21.6% for Qwen3-4B and 6.9% for Deepseek-Qwen-7B.

## 4.2 Ablations and Analysis

To understand the importance of each component of the training setup we conducted an ablation study, removing each component of our method. Table 3 and Table 4 show the performance of these ablations compared with the base model. Specifically, we start with the base model and the ablations:

(i) Base Model + CR: The base model trained with GRPO using only the correctness reward, (ii) Base model + CR + LR: The base model trained with GRPO using both correctness and length rewards, but without difficulty-level calibration, (iii) Base model + CR + LR + Compression: The base model trained with GRPO using correctness and length rewards, along with the compression module, with no difficulty-level calibration. Our findings are as follows.

Combining difficulty-adaptiveness and attention-based compression is crucial for accuracy and efficiency. Table 3 shows that on Qwen3-4B, removing the difficulty-based calibration (Base Model + CR + LR + compression) reduces the average performance across AIME, AMC, GPQA-D, and BBEH by 3.4%, while also making the model less efficient by 23.8%. Additionally, removing the attention-based compression (Base Model + LR + CR) leads to a further drop in performance by 0.3%. Similarly, on OptimalThinkingBench (Table 4), we observe a comparable degradation: the

Table 3: Ablation Results of TRAAC on Qwen3-4B and Deepseek-Qwen-7B tested across 4 datasets: AIME, AMC, GPQA-D, and BBEH. Each component addition adds to the previous method.

| Method        | AIME     |       | AMC   |       | GPQA-D |       | ВВЕН  |       | Average |       |
|---------------|----------|-------|-------|-------|--------|-------|-------|-------|---------|-------|
|               | Acc.↑    | Len.↓ | Acc.↑ | Len.↓ | Acc.↑  | Len.↓ | Acc.↑ | Len.↓ | Acc.↑   | Len.↓ |
|               | Qwen3-4B |       |       |       |        |       |       |       |         |       |
| Base Model    | 27.64    | 9.2   | 68.19 | 7.0   | 45.18  | 7.6   | 18.28 | 6.7   | 39.8    | 7.6   |
| + CR          | 44.36    | 7.9   | 77.35 | 5.5   | 46.29  | 5.7   | 18.13 | 5.2   | 46.5    | 6.1   |
| + LR          | 37.84    | 4.5   | 77.35 | 2.4   | 44.06  | 2.3   | 18.57 | 2.1   | 44.5    | 2.8   |
| + Compression | 38.37    | 8.1   | 75.90 | 5.5   | 46.40  | 6.2   | 18.41 | 5.4   | 44.8    | 6.3   |
| TRAAC         | 45.45    | 6.7   | 79.52 | 4.2   | 47.21  | 4.2   | 20.59 | 4.3   | 48.2    | 4.8   |

Table 4: Ablation Results of TRAAC (Qwen3-4B and Deepseek-Qwen-7B) on OptimalThinking-Bench (OTB). Each component addition adds to the previous method.

| Method        | OverthinkingBench |       |                      | Underth | ОТВ   |      |  |
|---------------|-------------------|-------|----------------------|---------|-------|------|--|
|               | Acc.↑             | Len.↓ | AUC <sub>OAA</sub> ↑ | Acc.↑   | Len.↓ | F1↑  |  |
| Qwen3-4B      |                   |       |                      |         |       |      |  |
| Base Model    | 90.02             | 1.2   | 80.06                | 34.33   | 7.1   | 48.1 |  |
| + CR          | 90.02             | 0.9   | 78.86                | 37.06   | 5.7   | 50.4 |  |
| + LR          | 90.94             | 0.4   | 75.86                | 29.62   | 2.3   | 42.6 |  |
| + Compression | 90.12             | 0.9   | 80.41                | 36.51   | 6.0   | 50.2 |  |
| TRAAC         | 89.79             | 0.6   | 85.06                | 41.09   | 4.7   | 55.4 |  |

F1 score decreases by 5.2% when task-difficulty level calibration is removed and drops further by 7.6% when the attention-based compression module is also removed. These results highlight that a combination of task-difficulty calibration and attention-based compression is crucial for achieving both high performance and efficiency gains across tasks.

**TRAAC adapts to task difficulty.** To further understand the level of adaptivity of TRAAC compared to other methods, we plot the relative compression ratio and absolute accuracy gains (w.r.t. the base model) in Fig. 3 as a function of task difficulty. Here, we rank tasks in order of increasing difficulty. We conduct these experiments on SuperGPQA (Team et al., 2025) – a benchmark to evaluate model knowledge and reasoning capabilities, which is stratified into easy, medium, and hard splits, and BBH (Big Bench Hard) (Suzgun et al., 2022) – an easier version of BBEH. To get oracle difficulty ratings, we rank the datasets by the performance of frontier models on them (Kazemi et al., 2025; Team et al., 2025), with harder datasets being those with lower performance. From Fig. 3(a), we see that as the difficulty of the dataset increases from left to right, the compression rate steadily drops for TRAAC, underscoring its ability to compress more for easier tasks and less for difficult tasks. However, without task-difficulty level calibration, the compression rate remains roughly uniform across the tasks. Fig. 3(b) highlights the performance difference, and shows that even with more compression, TRAAC always maintains higher accuracy than Qwen3-4B + CR + LR + compression, reiterating the effectiveness of adapting to problem difficulty in TRAAC. Moreover, most of the accuracy gains stems from harder problems, indicating the average accuracy gains seen in Table 1 come from difficulty-adaptive thinking. Deepseek-Qwen-7B results are shown in Appendix A.2 and follow a similar trend as Qwen3-4B.

TRAAC scales to larger response length, maintaining its improvement. During TRAAC training, we set a maximum token budget of 10k. To test the scalability of our method, we increase the max training and test-time response length to 15k. Table 5 shows the accuracy and average response length for AIME, AMC, and GPQA-D datasets, for the Qwen3-4B and TRAAC with increased token budget. Similar to the

Table 5: TRAAC with 15k training and test-time response length. For each dataset, Accuracy (%) and Response Length (in  $\times 1000$  tokens) are reported.

|       | AIME               | AMC                | GPQA-D             |
|-------|--------------------|--------------------|--------------------|
|       | 47.74 / 12.3       |                    | 49.64 / 8.6        |
| TRAAC | <b>51.93</b> / 9.7 | <b>81.68</b> / 6.6 | <b>51.27</b> / 6.2 |

prior results, we see an average accuracy improvement of 3.5% and 23.4% efficiency gains. This underscores that scaling TRAAC still shows consistent gains for both accuracy and efficiency.

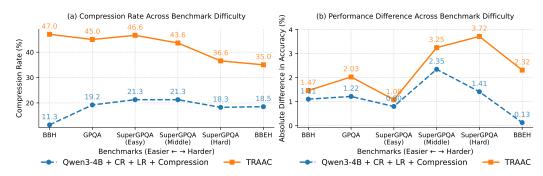


Figure 3: (a) Relative change in compression rate of TRAAC and Qwen3-4B + Compression compared to Qwen3-4B across varying problem difficulty. (b) Absolute accuracy drop of TRAAC and Qwen3-4B + Compression compared to Qwen3-4B across varying problem difficulty.

## 5 RELATED WORK

In the past years, reasoning performance of language models has vastly improved via the introduction of chain-of-thoughts (Wei et al., 2023), parallel scaling through self-consistency (Wang et al., 2023), and best-of-N sampling (Lightman et al., 2023). More recently, several works have found sequential scaling - i.e., increasing the number of reasoning tokens - to be the most effective approach (Muennighoff et al., 2025), especially when combined with online reinforcement learning or distillation from such models (Aggarwal & Welleck, 2025; Shao et al., 2024; DeepSeek-AI et al., 2025). Consequently, the area of efficient reasoning – maintaining high performance from sequential scaling with minimal token usage - has become a central research focus (Chen et al., 2024; Marjanović et al., 2025; Wu et al., 2025). To this end, prior works compress or prune chain-of-thoughts via early exiting (Zhang et al., 2025a; Fu et al., 2025), train models under pre-specified budgets (Aggarwal & Welleck, 2025), learn thoughts latently without generating them (Hao et al., 2025), use supervised finetuning to avoid overthinking (Xia et al., 2025; Cheng et al., 2025; Lu et al., 2025), or add length-based penalties for conciseness (Arora & Zanette, 2025; Hou et al., 2025). However, this line of work does not explicitly account for varying problem difficulty, instead relying on the model to learn to allocate budget implicitly; in contrast, TRAAC introduces difficulty-based supervision for budget allocation. Moreover, prior approaches typically address only overthinking - reducing output length at the cost of performance drops – whereas we tackle both over- and underthinking.

Improving *both* reasoning performance and efficiency requires a more *adaptive* approach through explicit training. Prior work such as Zhang et al. (2025b) frames adaptivity as a binary decision of *whether* to think, whereas we argue that for harder problems it must involve deciding *how much* to think – and empirically outperform this baseline in Appendix 4.1. A similar insight appears in planning, where Saha et al. (2025) show that mixing "system 1" and "system 2" reasoning within the same instance outperforms a binary choice between them. Shen et al. (2025) pursue difficulty-adaptive training via repeated sampling and offline preference optimization to prefer shorter responses. In contrast, TRAAC provides attention-based supervision in the compression module through online RL (DeepSeek-AI et al., 2025). Unlike concurrent work by Choi et al. (2025), who prune redundant tokens post hoc, our method adapts compression during training itself – yielding difficulty-aware reasoning and improved test-time efficiency without generating unnecessary tokens.

## 6 Conclusion

We introduced TRAAC, a post-training RL method that operates online and uses a difficulty-adaptive, attention-based compression module. Through its adaptive attentive compression, TRAAC is able to prune its reasoning steps adaptively based on the task difficulty. TRAAC addresses the issue of under-adaptivity, which helps improve both performance and efficiency, as thinking longer on harder problems helps in better exploration, and thinking shorter on easier problems avoids wasting of test-time compute. Moreover, our method also shows strong generalizability, with evaluation done on various OOD tasks. Through our analysis and ablation, we further verify that our adaptive method can provide fine-grained adjustments to the thinking budget based on the difficulty of the problem, and a combination of task-difficulty calibration and attention-based compression helped achieve both accuracy and efficiency gains.

# **ETHICS STATEMENT**

TRAAC is a reinforcement learning method that rewards models based on the correctness of the final answer. Therefore, the trained LLMs may still generate hallucinations, since their intermediate reasoning steps are neither guided nor evaluated – only the final result is checked. This means outputs from TRAAC can pose risks of misinformation or hallucination. Future work is needed to more thoroughly evaluate and mitigate these issues.

## REPRODUCIBILITY STATEMENT

We are making our code available in the supplementary materials to help reproduce our findings. We also provide detailed descriptions, hyperparameters, and prompts about the implementation of TRAAC in Appendix A.5.

## REFERENCES

- Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.
- Pranjal Aggarwal, Seungone Kim, Jack Lanchantin, Sean Welleck, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. Optimalthinkingbench: Evaluating over and underthinking in llms, 2025. URL https://arxiv.org/abs/2508.13141.
- AIME. American invitational mathematics examination, 2024. URL https://artofproblemsolving.com/wiki/index.php/American\_Invitational\_Mathematics\_Examination.
- AMC. American mathematics competitions, 2023. URL https://maa.org/student-programs/amc/.
- Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025. URL https://arxiv.org/abs/2502.04463.
- Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Qiying Yu, Xuefeng Li, Jiaze Chen, Hao Zhou, and Mingxuan Wang. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles, 2025. URL https://arxiv.org/abs/2505.19914.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.
- Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. Optimizing length compression in large reasoning models, 2025. URL https://arxiv.org/abs/2506.14755.
- Daewon Choi, Jimin Lee, Jihoon Tack, Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Bhavana Ganesh, Jinwoo Shin, Aram Galstyan, and Sravan Babu Bodapati. Think clearly: Improving reasoning via redundant token pruning, 2025. URL https://arxiv.org/abs/2507.08806.
- Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, et al. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.
- DeepSeek-AI et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.
- Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv* preprint arXiv:2508.15260, 2025.

- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language model to reason in a continuous latent space, 2025. URL https://openreview.net/forum?id=tG4SgayTtk.
- Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.
- Shijue Huang, Hongru Wang, Wanjun Zhong, Zhaochen Su, Jiazhan Feng, Bowen Cao, and Yi R. Fung. Adactrl: Towards adaptive and controllable reasoning via difficulty-aware budgeting, 2025. URL https://arxiv.org/abs/2505.18822.
- Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. Big-bench extra hard, 2025. URL https://arxiv.org/abs/2502.19187.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.
- Ximing Lu, Seungju Han, David Acuna, Hyunwoo Kim, Jaehun Jung, Shrimai Prabhumoye, Niklas Muennighoff, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, et al. Retro-search: Exploring untaken paths for deeper and efficient reasoning. arXiv preprint arXiv:2504.04383, 2025.
- Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stańczak, and Siva Reddy. Deepseek-r1 thoughtology: Let's think about llm reasoning, 2025. URL https://arxiv.org/abs/2504.07128.
- 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, 2025. URL https://arxiv.org/abs/2501.19393.
- OpenAI et al. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.
- 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 qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.
- Swarnadeep Saha, Archiki Prasad, Justin Chih-Yao Chen, Peter Hase, Elias Stengel-Eskin, and Mohit Bansal. System-1.x: Learning to balance fast and slow planning with language models, 2025. URL https://arxiv.org/abs/2407.14414.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.
- Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models, 2025. URL https://arxiv.org/abs/2503.04472.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:* 2409.19256, 2024.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling Ilm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

- Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2505.24760.
- 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 bigbench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.
- P Team et al. Supergpqa: Scaling Ilm evaluation across 285 graduate disciplines, 2025. URL https://arxiv.org/abs/2502.14739.
- Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.
- Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too), 2023. URL https://arxiv.org/abs/2311.11829.
- Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025.
- Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms, 2025. URL https://arxiv.org/abs/2502.12067.
- Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning, 2025. URL https://arxiv.org/abs/2502.14768.
- Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.
- Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv* preprint *arXiv*:2504.05419, 2025a.
- Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think, 2025b. URL https://arxiv.org/abs/2505.13417.
- Jixiao Zhang and Chunsheng Zuo. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models, 2025. URL https://arxiv.org/abs/2504.09696.

## A APPENDIX

#### A.1 DATASET DETAILS

We evaluated the model on various benchmarks:

- AMC: All questions come from AMC12 2022, AMC12 2023, and have been extracted from the AOPS wiki page. Total Count: 83
- AIME: All questions come from AIME 22, AIME 23, and AIME 24, and have been extracted directly from the AOPS wiki page. Total Count: 90
- GPQA-D: It is a multiple-choice dataset covering physics, biology, and chemistry. Total Count:
- BBEH: A benchmark designed to push the boundaries of LLM reasoning evaluation. BBEH
  replaces each task in BBH with a novel task that probes a similar reasoning capability but exhibits
  significantly increased difficulty. Total Count: 460
- OptimalThinkingBench: A unified benchmark that jointly evaluates overthinking and underthinking in LLMs and also encourages the development of optimally-thinking models that balance performance and efficiency. Two sub benchmarks: OverthinkingBench, featuring simple queries in 72 domains, and UnderthinkingBench, containing 11 challenging reasoning tasks. UnderthinkingBench count: 550, OverthinkingBench count: 607.
- BBH: a suite of 23 challenging BIG-Bench tasks. Total Count: 2115
- SuperGPQA: A comprehensive benchmark designed to evaluate the knowledge and reasoning abilities of Large Language Models (LLMs) across 285 graduate-level disciplines. Each problem is also categorized as easy, medium and hard. 540 problems for each difficulty category, so the total count is 1620.

To calculate the accuracy, we adopt Math-Verify <sup>2</sup>. For UnderthinkingBench accuracy calculation, we used the evaluation scripts from Reasoning-Gym (Stojanovski et al., 2025)

## A.2 DEEPSEEK ABLATION AND ANALYSIS

Table 6 and Table 7 present the ablation results for (i) Base Model + CR: The base model trained with GRPO using only the correctness reward, (ii) Base model + CR + LR: The base model trained with GRPO using both correctness and length rewards, but without difficulty-level calibration.

Table 6: Ablation Results of TRAAC on Qwen3-4B and Deepseek-Qwen-7B tested across 4 datasets: AIME, AMC, GPQA-D, and BBEH. Each component addition adds to the previous method.

| Method     | AII                         | ME    | AN    | <b>ЛС</b> | GPQ   | A-D   | BB    | EH    | Ave   | rage  |
|------------|-----------------------------|-------|-------|-----------|-------|-------|-------|-------|-------|-------|
|            | Acc.↑                       | Len.↓ | Acc.↑ | Len.↓     | Acc.↑ | Len.↓ | Acc.↑ | Len.↓ | Acc.↑ | Len.↓ |
|            | DeepSeek-R1-Distill-Qwen-7B |       |       |           |       |       |       |       |       |       |
| Base Model | 33.71                       | 8.2   | 74.22 | 5.7       | 43.55 | 7.1   | 10.61 | 5.9   | 40.5  | 6.7   |
| + CR       | 35.81                       | 7.6   | 78.55 | 4.9       | 45.99 | 6.1   | 11.74 | 5.1   | 43.0  | 5.9   |
| + LR       | 32.73                       | 6.0   | 79.04 | 3.3       | 45.99 | 3.5   | 11.51 | 2.7   | 42.3  | 3.9   |
| TRAAC      | 38.60                       | 7.3   | 77.83 | 4.5       | 47.31 | 6.2   | 11.55 | 5.2   | 43.8  | 5.8   |

## A.3 COMPRESSION MODULE

## A.3.1 PROMPT

For every reasoning trajectory, auxiliary prompt was appended at the end of the trajectory. The prompt is: "Time is up. I should stop thinking and now write a summary containing all key steps required to solve the problem."

<sup>&</sup>lt;sup>2</sup>Huggingface Math-Verify

703

704 705 706

715

716 717

718

723 724

725 726

727

728

729 730 731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753 754

755

Table 7: Ablation Results of TRAAC (Qwen3-4B and Deepseek-Qwen-7B) on OptimalThinking-Bench (OTB). Each component addition adds to the previous method.

| Method                      | OverthinkingBench |       |                      | Underth | ОТВ   |      |  |  |
|-----------------------------|-------------------|-------|----------------------|---------|-------|------|--|--|
|                             | Acc.↑             | Len.↓ | $AUC_{OAA} \uparrow$ | Acc.↑   | Len.↓ | F1↑  |  |  |
| DeepSeek-R1-Distill-Qwen-7B |                   |       |                      |         |       |      |  |  |
| Base Model                  | 78.45             | 0.9   | 72.38                | 12.69   | 6.2   | 21.6 |  |  |
| + CR                        | 79.51             | 0.8   | 73.36                | 17.05   | 5.7   | 27.7 |  |  |
| + LR                        | 78.06             | 0.4   | 72.61                | 14.69   | 3.0   | 24.4 |  |  |
| TRAAC                       | 81.81             | 1.0   | 72.89                | 22.30   | 5.9   | 34.1 |  |  |

### A.3.2 SPECIAL TOKENS TO SPLIT TRAJECTORY TO CHUNKS

Below is the list that is used to split each reasoning trajectory into multiple reasoning steps.

```
split_tokens = [
                                                       "Wait", "Alternatively", "Another angle", "Another approach", "But wait", "Hold on", "Hmm", "Maybe", "Looking back", "Okay", "Let me", "First", "Then", "Alright", "Compute", "Correct", "Good", "Got it", "I don't see any errors", "I think", "Let me double-check", "Let's see", "Now " "Branch of the control 
                                                           "Now", "Remember", "Seems solid", "Similarly", "So", "Starting", "That's correct", "That seems right", "Therefore", "Thus"
]
```

## A.3.3 Uniformity Score

Algorithm 1 presents the pseudocode for calculating the uniformity score, based on which the final compression rate is calculated.

```
Algorithm 1: Calculating Eviction Percentage Based on Attention Uniformity
```

```
Input: Step importance scores \{s_1, s_2, \dots, s_n\}, target reduction \tau (default: 0.25)
Output: Eviction percentage e \in [0, 1]
Function CALCULATEUNIFORMITYSCORE(\{s_1, \ldots, s_n\}):
    if n \leq 1 then
        return 1.0;
                                                      // Only one step \Rightarrow perfectly uniform
    Clamp all s_i \geq 0;
    T \leftarrow \sum_{i} s_i;
    if T \leq 0 then
        return 1.0;
    p_i \leftarrow s_i/T;
                                                  // Normalize to probability distribution
    H \leftarrow -\sum_{i} p_{i} \cdot \log(p_{i} + \epsilon);
                                                                             // Entropy, \epsilon = 10^{-12}
    H_{\text{max}} \leftarrow \log(n);
    if H_{\text{max}} = 0 then
        return 1.0;
    return H/H_{\rm max};
                                                                    // Uniformity score in [0,1]
Function DETERMINEEVICTIONPERCENTAGE(u, \tau):
    if u > 0.8 then
        return 0.0;
                                                          // High uniformity: keep all steps
    e \leftarrow \tau \cdot (1-u);
                                                         // Scale eviction by non-uniformity
    return min(e, 0.8);
                                                                          // Cap eviction at 80\%
u \leftarrow \text{CALCULATEUNIFORMITYSCORE}(\{s_1, \dots, s_n\});
e \leftarrow \text{DETERMINEEVICTIONPERCENTAGE}(u, \tau);
```

## A.4 GRPO DETAILS

For each question q, a group of responses  $\{y^1, y^2, \dots, y^N\}$  is sampled from the old policy  $\pi_{\text{old}}$ , and the policy model  $\pi_{\theta}$  is optimized by maximizing the following GRPO objective.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|y^i|} \sum_{t=1}^{|y^i|} \min \left[ \frac{\pi_{\theta}(y^i(t)|y^i_{< t})}{\pi_{\text{old}}(y^i(t)|y^i_{< t})} \hat{A}_{i,t}, \operatorname{clip}\left(\frac{\pi_{\theta}(y^i(t)|y^i_{< t})}{\pi_{\text{old}}(y^i(t)|y^i_{< t})}, 1 - \varepsilon, 1 + \varepsilon\right) \hat{A}_{i,t} \right],$$

where  $\varepsilon$  is the clipping range hyperparameter, and  $\hat{A}_{i,t}$  represents the advantage, computed based on the relative verifiable outcome based rewards of outputs within each group.

#### A.5 EXPERIMENTAL DETAILS

We adopt verl (Sheng et al., 2024) as the training framework.

#### A.5.1 HYPERPARAMETERS

Table 8: Hyperparameters used for training, evaluation, and difficulty calibration.

| Category               | Hyperparameter      | Value |
|------------------------|---------------------|-------|
|                        | Number of rollouts  | 8     |
|                        | Temperature         | 1.0   |
|                        | top_p               | 1.0   |
|                        | top_k               | -1.0  |
| Training               | Max response length | 10k   |
|                        | clip_ratio_low      | 0.20  |
|                        | clip_ratio_high     | 0.28  |
|                        | kl_loss_coef        | 0.001 |
|                        | Learning rate (LR)  | 1e-6  |
|                        | Number of rollouts  | 8     |
|                        | Temperature         | 1.0   |
| Evaluation             | top_p               | 1.0   |
| Evaluation             | top_k               | -1.0  |
|                        | Max response length | 10k   |
|                        | N                   | 5     |
|                        | Hard                | 0.20  |
| Difficulty Calibration | Medium              | 0.40  |
|                        | Easy                | 0.60  |

## A.5.2 TRAINING REWARD

To ensure a high weight on correctness relative to other components, we assign a **correctness reward** of +4 if the final answer is correct and 0 otherwise. The **format reward** ranges from 0 to 1: a score of 0.5 is given for the presence of the <think> and </think> tokens, and an additional 0.5 is awarded if every reasoning trajectory is properly enclosed within these tokens in the correct order. The **length reward** ranges from 0 to 2. The overall reward is computed as the sum of these components:

Total Reward = Correctness Reward + Format Reward + Length Reward.

### A.5.3 EVALUATION METRICS

For each of the dataset we compute the accuracy and the average response length. Specifically for OverthinkingBench we also compute the AUC<sub>OAA</sub>. This metric is based on Overthinking-Adjusted

Accuracy (OAA), which measures model correctness under a limit on reasoning tokens. For a threshold *t*, it is defined as

$$\mathrm{OAA}_t = \frac{1}{n} \sum_{i=1}^n \left( \mathrm{Correctness}_i \cdot \mathbb{I}(\mathrm{ThinkTokens}_i < t) \right),$$

where  $Correctness_i \in \{0,1\}$  indicates whether the *i*-th response is correct, and  $\mathbb{I}(\cdot)$  is the indicator function that enforces the thinking length constraint.

$$\mathrm{AUC_{OAA}} = \int_0^{t_{\mathrm{max}}} \frac{\mathrm{OAA}_t}{t_{\mathrm{max}}} \, dt \approx \frac{1}{t_{\mathrm{max}}} \sum_{t=0}^{t_{\mathrm{max}}} \mathrm{OAA}_t,$$

where  $t_{\rm max}$  is the maximum number of allowed thinking tokens. Furthermore, following the method from (Aggarwal et al., 2025), to compute the OptimalThinkingBench metric: F1 score we combine the AUC<sub>OAA</sub> from OverthinkingBench and Accuracy (Acc<sub>ut</sub>) from UnderthinkingBench into a single F1 score:

$$F1 = 2 \cdot \frac{\text{AUC}_{\text{OAA}} \cdot \text{Acc}_{\text{ut}}}{\text{AUC}_{\text{OAA}} + \text{Acc}_{\text{ut}}}$$
(1)

### A.5.4 TRAINING PROMPT

 For each questions in the training set, instruction was provided: "Let's think step by step and output the final answer within \\boxed{}''

### A.5.5 BASELINE PROMPTS

Below we define the instruction that was provided to each baseline model:

- Base Model: ''Let's think step by step and output the final answer within \\boxed{}''
- L1-Max: ''Let's think step by step and output the final answer within  $\begin{tabular}{l} \begin{tabular}{l} \begin{tabula$
- LC-R1: '' Please reason step by step, and put your final answer within \boxed{}''
- AdaptThink: No prompt, just the question
- TokenSkip: "<|im\_start|>system You are a helpful assistant.<|im\_end|> <|im\_start|>user Please reason step by step, and put your final answer within \boxed{}. question<|eot\_id|>0.5<|eot\_id|><|im\_end|> <|im\_start|>assistant"

## A.6 COMPUTE USED

All training was done on 4\*A100 (80GB).