Learning Adaptive Reasoning Budgets via Constraint-Rectified Training

Qinhang Wu [1] Sen Lin [2] Ming Zhang [3] Yingbin Liang [4] Ness B. Shroff [1 4]

Abstract

Foundation models increasingly rely on long chain-of-thought reasoning traces, improving task performance but imposing substantial inference-time token cost. Existing efficient-reasoning methods often shorten outputs through fixed budgets or scalar rewards that mix correctness and length, making the resulting quality-resource tradeoff difficult to control. We propose Constraint-Rectified Training (CRT), a reference-guarded post-training framework for adaptive efficient reasoning. CRT first minimizes reasoning length while preserving accuracy relative to a frozen reference policy, thereby discovering the shortest reliable reasoning regime for the model. It then treats this discovered regime as a learned token budget and refines correctness without allowing reasoning traces to expand back to unnecessary verbosity. Across multiple reasoning benchmarks and foundation-model backbones, CRT improves the accuracy-efficiency frontier over length-aware baselines. Additional redundancy analysis shows that CRT reduces repetitive reasoning structure beyond what is captured by average token count alone. These results suggest that constrained, budget-aware post-training is a practical mechanism for resource-efficient foundation model inference.

1. Introduction

Long-form chain-of-thought (CoT) reasoning has become a central form of test-time compute in foundation models. Frontier reasoning models such as OpenAI o1 (Jaech et al., 2024) and Gemini 2.5 (Comanici et al., 2025) demonstrate strong performance on mathematics, programming, and logical reasoning tasks, while open-source model families such as Qwen3 and Phi-4 provide increasingly accessible backbones for studying reasoning behavior. These advances suggest that explicit reasoning traces are a powerful mechanism for improving model capability. However, they also expose a growing deployment challenge: reasoning tokens are not free. Longer traces increase inference latency and generation cost, and they often contain redundant verification, restatement, or self-doubt that does not materially improve the final answer.

This motivates *adaptive efficient reasoning*: a model should use no more reasoning computation than necessary while preserving its problem-solving competence. Existing approaches to efficient reasoning primarily span supervised trace compression, inference-time pruning or control, and post-training reinforcement learning (RL). Among them, RL-based post-training is particularly attractive because it can directly modify the model's reasoning policy rather than relying only on prompts or decoding-time add-ons. Yet many existing methods control brevity through fixed length budgets, pruning schedules, target-length proxies, or scalar rewards that mix correctness and response length. Such designs can be difficult to tune and may produce fragile quality-resource tradeoffs: aggressive compression can erode correctness, while conservative settings may leave substantial redundant reasoning intact (Arora & Zanette, 2025; Yi et al., 2025; Su & Cardie, 2025). The key question is therefore not simply how to make CoT shorter, but how to *adapt reasoning-token expenditure under an explicit correctness boundary*.

We propose *Constraint-Rectified Training* (CRT), a reference-guarded post-training framework for adaptive efficient reasoning in foundation models. CRT treats response length as an inference resource and formulates efficient reasoning as a constrained quality-resource optimization problem. Its design has three components. First, CRT uses a frozen reference policy as a relative competence guard, avoiding hand-set absolute accuracy thresholds. Second, it treats reasoning length as the optimization objective only when the current policy remains accuracy-feasible, avoiding scalar reward-mixing coefficients. Third, it uses a two-stage objective/constraint reversal: Stage I minimizes reasoning length while preserving reference-relative accuracy, thereby discovering a short reliable reasoning regime; Stage II then treats this discovered regime as a learned token budget and

[1]Department of CSE, The Ohio State University [2]Department of Computer Science, University of Houston [3]Google [4]Department of ECE, The Ohio State University. Correspondence to: Qinhang Wu <wu.5677@osu.edu>.

International Conference on Machine Learning, AdaptFM: Resource-Adaptive Foundation Model Inference, Seoul, South Korea. PMLR 306, 2026.

improves correctness without allowing the model to drift back to unnecessarily verbose reasoning.

This framing makes CRT adaptive in a specific and practical sense: the reasoning budget is not imposed as a fixed external cap, but is learned from the model's own accuracy-efficiency boundary during post-training. When the policy remains above the reference-guarded competence floor, CRT continues compression; when the constraint is violated, it immediately rectifies toward correctness. The resulting training trajectory searches for a favorable operating point on the quality-resource frontier rather than optimizing a single scalar preference over accuracy and length.

We evaluate CRT across in-domain, out-of-domain, and cross-domain benchmarks using multiple foundation-model backbones. Empirically, CRT improves the accuracy-efficiency frontier over length-aware baselines, reducing reasoning-token cost while better preserving task accuracy. Beyond average length, we further analyze internal language redundancy with a compression-based diagnostic and an LLM-as-a-judge evaluation, showing that CRT reduces repetitive reasoning structure rather than merely truncating outputs. Together, these results suggest that reference-guarded, budget-aware post-training is a practical mechanism for resource-efficient foundation model inference.

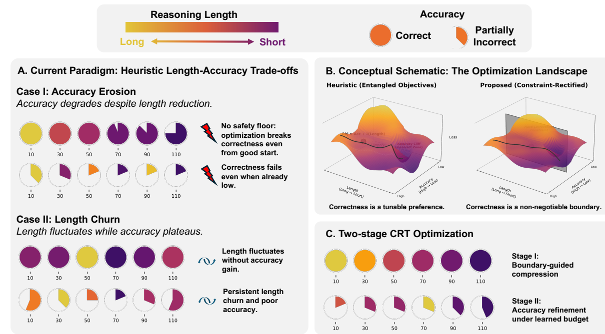

Figure 1. Motivation for boundary-controlled efficient reasoning. Color indicates response length; circle fill indicates correctness; numbers denote training steps. Panel (A) shows two failure modes of heuristic length–accuracy tradeoffs: accuracy erosion and length churn. Panel (B) contrasts scalar objectives, where correctness is a tunable preference, with constraint-rectified optimization, where correctness is a feasibility boundary. Panel (C) illustrates CRT: Stage I discovers a short reliable reasoning regime, and Stage II refines accuracy under the learned length budget.

2. Related Work

Reasoning post-training and reasoning efficiency. Chain-of-thought (CoT) prompting improves language-model reasoning by eliciting intermediate solution steps (Wei et al., 2022), with extensions such as self-consistency (Wang et al., 2023) and tree-structured search (Yao et al., 2023). More recently, RL-based post-training has become a practical recipe for large reasoning models (Cobbe et al., 2021; Lightman et al., 2024; Luo et al., 2025b; Wang et al., 2024; Zeng et al., 2025; Zhu et al., 2025), especially with scalable group-relative objectives such as GRPO (Shao et al., 2024). At the same time, reasoning traces are increasingly recognized as structured computational objects whose length, validity, and redundancy affect both model behavior and inference cost (Chen et al., 2025; Hu et al., 2025; Singla et al., 2025). Recent studies show that CoT traces can exhibit invalid or illusionary reasoning (Weatherhead et al., 2025; He et al., 2025; Jose, 2025; Liu et al., 2025b), answer-attribution failures (Wang et al., 2026), overthinking (Chen et al., 2024), underthinking (Wang et al., 2025), and capability forgetting after reasoning-centric post-training (Phan et al., 2025). These findings motivate methods that regulate not only final-answer accuracy, but also how reasoning compute is allocated.

Adaptive and resource-efficient reasoning. Efficient-reasoning methods reduce CoT cost through supervised trace shortening (Kang et al., 2025; Xia et al., 2025; Liu et al., 2024), inference-time or representation-level controls such as early exiting and latent reasoning (Dai et al., 2025; Hao et al., 2024), and RL-based length-aware post-training (Yi et al., 2025; Luo et al., 2025a; Hou et al., 2025; Arora & Zanette, 2025; Shen et al., 2025; Liu et al., 2025a; Li et al., 2025b). Closely related work also studies difficulty-aware compression, which allocates reasoning effort according to problem difficulty (Singh et al., 2025), and content-level filtering, which removes invalid or redundant reasoning steps (Cheng et al., 2025b). These approaches improve efficiency, but they often rely on externally specified budgets, pruning schedules, target-length proxies, adaptive heuristics, or scalar rewards that mix correctness and length. CRT studies a complementary formulation: efficient reasoning is treated as reference-guarded quality-resource optimization. Rather than treating correctness as a tunable reward component or imposing a fixed length budget, CRT minimizes reasoning length only while a reference-relative competence constraint is satisfied, then uses the discovered short-reasoning regime as a learned token budget for accuracy refinement.

3. Motivation: Learning Reliable Reasoning Budgets

CoT length is an inference resource, but reducing it is not equivalent to improving reasoning efficiency. A fixed token budget can be too aggressive for some training regimes and too conservative for others, while scalar length-aware rewards collapse correctness and resource cost into a single preference. In both cases, the method does not directly specify when compression has crossed the boundary where reasoning competence begins to degrade.

Figure 1 illustrates this issue. Heuristic length–accuracy tradeoffs can produce two common failure modes: *accuracy erosion*, where shorter responses are obtained by dropping reasoning needed for correctness, and *length churn*, where response length changes without meaningful quality improvement. These failures arise because correctness is treated as a negotiable reward component rather than a protected condition. Empirically, when conditioning on prompts whose average length is reduced, heuristic baselines such as *ThinkPrune* (Hou et al., 2025) and *ACPO* (Cheng et al., 2025a) more frequently decrease accuracy, whereas CRT better preserves or improves correctness under length reduction; detailed statistics are deferred to Table 6.

This motivates a boundary-controlled view of adaptive efficient reasoning. Instead of asking how to tune a length penalty, we ask how to reduce reasoning-token expenditure while remaining within a model-specific competence region. This leads to three design requirements. First, correctness should be guarded relative to a frozen reference policy rather than by a hand-set absolute threshold. Second, length should be optimized only while the current policy remains accuracy-feasible. Third, once the shortest reliable regime is reached, this regime should become a learned token budget, and subsequent training should improve correctness without drifting back to unnecessarily long reasoning. CRT implements these principles through reference-guarded feasibility, boundary-guided length minimization, and a two-stage objective/constraint reversal.

4. Constraint-Rectified Training

We now present *Constraint-Rectified Training* (CRT), a post-training framework for learning a short but reliable reasoning regime. CRT treats reasoning length as an inference resource and optimizes it under an explicit correctness boundary, rather than combining correctness and length into a single scalar reward.

4.1. Problem Setup

Let $x \sim \rho$ denote an input prompt and let $y \sim \pi_\theta(\cdot \mid x)$ denote a model response containing both the reasoning trace and final answer. We evaluate correctness with a verifier-based indicator $\mathbb{1}\{y = y^*(x)\}$, where $y^*(x)$ is the ground-truth answer. Let $\ell_{\mathrm{norm}}(y)$ denote the normalized response length; implementation details are provided in Appendix C. With expectations taken over $x \sim \rho$ and $y \sim \pi(\cdot \mid x)$, we define

$$L(\pi) = \mathbb{E}[\ell_{\mathrm{norm}}(y)], \qquad A(\pi) = \mathbb{E}[\mathbb{1}\{y = y^*(x)\}]. \tag{1}$$

4.2. Constraint-Rectified Training Framework

CRT as a two-stage boundary-control procedure. CRT consists of two stages for exploring the model's quality–resource frontier. Stage I addresses the question of *how far reasoning can be safely compressed*: it minimizes reasoning length while maintaining accuracy relative to a frozen reference policy, thereby keeping the model near the competence boundary of reliable reasoning. Stage II addresses the complementary question of *how well the model can reason within the discovered short-reasoning regime*: it freezes the Stage I policy as a learned length reference and optimizes correctness under this budget. Together, the two stages first discover a short reliable reasoning regime and then improve accuracy within it, turning efficient reasoning from a heuristic length penalty into a controlled boundary-seeking optimization process.

Stage I: reference-guarded length discovery. In Stage I, we formulate efficient reasoning as a constrained optimization problem: reducing expected reasoning length while ensuring that accuracy remains above a competence level. A natural formulation would be

$$\min_\pi \ L(\pi), \qquad \text{s.t.} \quad A(\pi) \geq \alpha. \tag{2}$$

However, it is often impractical to specify an appropriate accuracy threshold α before post-training. If the threshold is too strict, the feasible region becomes overly restrictive and the model cannot meaningfully shorten its reasoning. Conversely, if the threshold is too loose, training may collapse to short but incorrect responses. In short, the scalar threshold α is difficult to tune and does not provide a stable accuracy safeguard during training.

To address this problem, CRT uses a frozen reference policy π_{ref}, instantiated as the initial model before length-aware post-training, to define a model-specific competence floor. Since π_{ref} represents the original reasoning capability of the model, the updated policy should not perform significantly worse than this reference in terms of correctness. This leads to the reference-guarded objective

$$\min_\pi \ L(\pi), \qquad \text{s.t.} \quad A(\pi) \geq A(\pi_{\mathrm{ref}}) - \varepsilon. \tag{3}$$

Here $A(\pi_{\mathrm{ref}})$ is estimated by sampling reference responses $y_{\mathrm{ref}} \sim \pi_{\mathrm{ref}}(\cdot \mid x)$ on the same minibatch, and $\varepsilon \geq 0$ controls the allowed degradation relative to the reference policy. This formulation makes length the optimization target and correctness the protected quantity. Unlike reward-mixing methods (Arora & Zanette, 2025; Yi et al., 2025), which combine accuracy and length into a single scalar preference, the reference guard defines an explicit feasibility boundary: length reduction is pursued only while the policy remains above the reference accuracy floor.

Stage II: accuracy refinement under a learned budget.

Stage I discovers a short reasoning regime that remains feasible under the reference-guarded accuracy constraint in Equation (3). Once the boundary-walking behavior stabilizes, CRT freezes the current policy as $\pi_{\text{ref}}^{(I)}$ and uses it to define a learned length reference. CRT then optimizes the complementary objective: correctness is improved while the discovered Stage I length profile is maintained as a budget. Formally, Stage II solves

$$\max_{\pi} \ A(\pi), \qquad \text{s.t.} \quad L(\pi) \le L(\pi_{\text{ref}}^{(I)}) + \delta. \tag{4}$$

Here $L(\pi_{\text{ref}}^{(I)})$ is estimated from rollouts $y_{\text{ref}}^{(I)} \sim \pi_{\text{ref}}^{(I)}(\cdot \mid x)$, and $\delta \ge 0$ controls how much the Stage II policy may deviate from the learned length budget. To ensure a consistent length scale across stages, we freeze the normalization statistics used in ℓ_{norm} based on rollouts from $\pi_{\text{ref}}^{(I)}$ at the start of Stage II. The two stages therefore play complementary roles: Stage I identifies a short feasible reasoning regime, while Stage II uses that regime as the operating region for subsequent accuracy refinement.

Autonomous stage transition. CRT switches from Stage I to Stage II when the Stage I boundary-walking behavior stabilizes, rather than at a manually specified transition step. During Stage I, we record whether the controller activates accuracy rectification,

$$v_t = \mathbb{1}\{\hat{A}_t < \hat{A}^{\text{ref}} - \varepsilon - \tau\},$$

where $\hat{A}_t$ is the minibatch accuracy estimate for the current policy, $\hat{A}^{\text{ref}}$ is the frozen-reference accuracy estimate, and τ is a switching margin. We track the moving violation rate over a window of size K and trigger Stage II once this rate becomes stable and non-degenerate:

$$\begin{aligned} \bar{v}_t &= \frac{1}{K}\sum_{s=t-K+1}^{t} v_s, \\ \text{switch if} \quad \bar{v}_t &\in (0,1) \quad \text{and} \quad |\bar{v}_t - \bar{v}_{t-K}| \le K^{-1}. \end{aligned} \tag{5}$$

The condition $0 < \bar{v}_t < 1$ indicates that CRT has entered a boundary-walking regime in which both compression and rectification occur, while the stability condition indicates that this regime has saturated. At this point, CRT freezes the current policy as $\pi_{\text{ref}}^{(I)}$ and uses it as the learned length reference for Stage II.

Implementation, optimization trajectory, and computational cost. CRT implements the two constrained subproblems above through a lightweight reward-selection controller within the same GRPO training loop. Let $r_{\text{acc}}(y, x) = \mathbb{1}\{y = y^*(x)\}$ denote the correctness reward and let $r_{\text{len}}(y) = -\ell_{\text{norm}}(y)$ denote the length reward. At each update, the controller selects one active reward: correctness reward for rectification or negative normalized length reward for compression. In Stage I, accuracy is protected while length is optimized; in Stage II, the learned length budget is protected while accuracy is optimized.

To see how this keeps the optimization trajectory near the competence boundary, consider Stage I. CRT continuously optimizes the length-minimization objective while monitoring accuracy against the reference floor. Whenever accuracy falls below this floor, CRT immediately rectifies the update direction by following the correctness reward, pushing the model back into the feasible region. In Stage II, the same principle is applied with the roles reversed: CRT improves correctness while using the Stage I policy as a learned length budget, and activates length rectification when the policy drifts back toward overly verbose reasoning. Full pseudocode is deferred to Algorithm 1.

The reference policies are used only for forward estimates of the active constraint: π_{ref} estimates the Stage I accuracy floor, and $\pi_{\text{ref}}^{(I)}$ estimates the Stage II length budget. They do not require gradient computation, optimizer state, or backward graph retention. Although CRT references frozen policies during training, this does not double the per-step cost of standard GRPO-based pipelines because the reference policies are used only for forward computation. Stage II also continues from the Stage I boundary checkpoint within the same training run, so CRT does not require a separate post-training run.

5. Experiments

5.1. Experimental Setup

We implement our method within the Verl (Sheng et al., 2025) RL framework. All experiments are conducted using the hardware configuration detailed in Appendix A, with the full GRPO training configuration shared by all baselines reported in Appendix B. We evaluate our approach on three backbone language models: DeepSeek-R1-Distill-Qwen models at 1.5B and 7B scales (Guo et al., 2025) (hereafter *DeepSeek-1.5B* and *DeepSeek-7B*), and *Qwen3-4B* (Yang et al., 2025). Across all settings, we use a fixed batch size of 128 and generate 16 rollouts per prompt during training and testing, unless otherwise specified.

Dataset. Following SimpleRL-Zoo (Zeng et al., 2025), we perform RL on a mixture of the training splits from GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b). GSM8K consists of $7.4k$ training and $1.3k$ test examples, while MATH provides $7.5k$ training and $5k$ test examples. These datasets provide complementary reasoning regimes: GSM8K emphasizes elementary multi-step arithmetic, while MATH contains more challenging symbolic problems. For evaluation, we consider both in-domain and out-domain benchmarks. In-domain performance is assessed on GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2024), which follow the same problem

Table 1. Evaluation results across three backbone models. For DeepSeek-7B, we report averaged results over the same six math benchmarks; full dataset-wise results are reported in Tables 7 to 9.

Method	**DeepSeek-1.5B**				**Qwen3-4B**				**DeepSeek-7B**			
	$\overline{A}$	$\overline{L}$	AES_1	AES_2	$\overline{A}$	$\overline{L}$	AES_1	AES_2	$\overline{A}$	$\overline{L}$	AES_1	AES_2
Base	62.16	7739.0	0.00	0.00	65.77	2472.8	0.00	0.00	74.57	6775.0	0.00	0.00
+ *ThinkPrune-1k*	60.84	6828.0	0.01	-0.09	64.46	2006.9	0.09	-0.01	73.93	6334.8	0.02	-0.02
+ *ShorterBetter*	58.67	5731.5	-0.02	-0.30	61.22	1343.4	0.11	-0.23	72.50	5596.5	0.04	-0.10
+ *ACPO*	62.26	6752.7	0.13	0.13	63.12	1675.6	0.12	-0.08	74.84	6045.1	0.12	0.12
+ *CRT (ours)*	**63.40**	6690.7	**0.20**	**0.20**	62.99	1361.4	**0.24**	**0.03**	**75.48**	5912.3	**0.16**	**0.16**

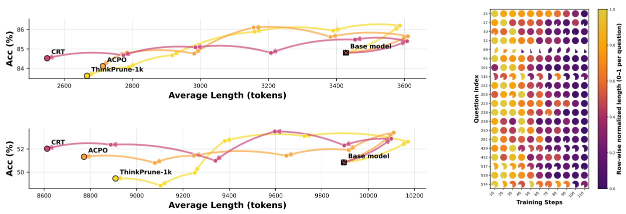

Figure 2. Quality–resource behavior over training checkpoints. Left: inference trajectories over intermediate checkpoints. The base model is DeepSeek-1.5B; the upper panel reports in-domain performance, and the lower panel reports out-domain performance. Right: checkpoint-level CRT behavior on GSM8K. Each row corresponds to a question and each column to an intermediate CRT checkpoint; colors denote row-wise normalized response length, and circle markers follow Figure 1.

distributions as the RL training data. To evaluate generalization beyond the training domain, we additionally report results on out-domain math reasoning benchmarks, including SAT Math (Zhong et al., 2024), AMC23 (math ai, 2023), AIME24 (Jia, 2024), and OLYMPIAD Bench (He et al., 2024).

Baselines. In addition to the base models, we compare our proposed method CRT against reproduced results from recent approaches for efficient reasoning, including ThinkPrune (Hou et al., 2025), O1-Pruner (Luo et al., 2025a), ShorterBetter (Yi et al., 2025), and ACPO (Cheng et al., 2025a). For ThinkPrune, we evaluate two length budgets, $1k$ and 500.

Metrics. We report **Acc**, defined as the pass@1 accuracy (in $\%$) averaged over rollouts per question, and **Len**, defined as the average number of generated tokens per response. In addition, to capture the quality–resource tradeoff, we report two composite metrics: $\textbf{AES}_1$ and $\textbf{AES}_2$. Higher AES values (Luo et al., 2025a) indicate more favorable accuracy–efficiency tradeoffs, with AES_2 imposing a stricter penalty on accuracy degradation than AES_1. The formal definition of AES is provided in Appendix C.

5.2. Main Results

We report evaluation results for DeepSeek-1.5B, Qwen3-4B, DeepSeek-7B in Table 1. Across all settings, CRT achieves the highest AES scores among the evaluated baselines, indicating a stronger ability to balance accuracy and reasoning-token efficiency. Moreover, CRT often preserves or improves accuracy, especially on the DeepSeek backbones, while in other cases trading modest accuracy loss for substantial length reduction. As shown by the training trajectories in Figure 2 (left), CRT consistently moves toward shorter operating points while largely maintaining task performance in both domains. In contrast, ThinkPrune exhibits substantial performance degradation beginning in the mid-training phase. These training-time dynamics directly account for the observed differences in final inference performance.

The right panel of Figure 2 further illustrates the budget-control behavior induced by CRT. Intermediate checkpoints expose multiple reasoning-length operating points within a single training run: many questions move from verbose traces toward shorter responses while retaining correct final answers. This supports the workshop-facing interpretation of CRT as learned reasoning-budget control rather than a fixed externally imposed length cap.

Although CRT is trained only on math reasoning data (GSM8K + MATH), the reference-guarded constraint is task-agnostic and may help reduce cross-domain degradation. To probe this, we evaluate the same DeepSeek-1.5B checkpoints on three out-of-distribution task families without any further tuning: code generation on MBPP (Austin et al., 2021), and multiple-choice QA on MMLU (Hendrycks et al., 2021a) and ARC-Challenge (Clark et al., 2018). Per-domain summaries are reported in Table 2; full per-method numbers are deferred to Tables 10 to 12. CRT is the only method that attains a non-negative AES_2 across all three domains, confirming that the gains observed on math reasoning do not induce severe cross-domain degradation under the AES_2 trade-off metric.

5.3. Analyzing Redundancy Beyond Average Length

Average response length is a useful but incomplete proxy for reasoning efficiency: two traces of similar length can still differ substantially in repetitive structure. In our qualitative inspection, inefficient traces often contain repeated verification, restated intermediate results, or self-doubt statements

Table 2. Cross-domain evaluation of DeepSeek-1.5B-trained checkpoints, reported as Acc / Len / AES_2 relative to the DeepSeek-1.5B base. MBPP uses 10 rollouts per problem evaluated by execution against hidden tests; MMLU and ARC-Challenge use 16 rollouts per question with letter-extraction grading. Full per-method results are in Tables 10 to 12.

Method	MBPP			MMLU			ARC-C		
	Acc	Len	AES_2	Acc	Len	AES_2	Acc	Len	AES_2
DeepSeek-1.5B	55.84	3134	0.00	49.61	1519	0.00	60.46	1154	0.00
+ *ThinkPrune-1k*	55.21	2767	0.01	47.21	1035	-0.16	56.42	621	-0.21
+ *ShorterBetter*	53.62	2365	-0.15	42.96	327	-0.55	54.31	149	-0.15
+ *ACPO*	54.82	2697	-0.04	47.83	1101	-0.08	57.29	647	-0.08
+ *CRT (ours)*	55.60	2755	**0.08**	48.49	1118	**0.04**	57.83	611	**0.04**

Table 3. Gzip-based non-redundancy on GSM8K. We report the compressed-size ratio $R_{\text{zip}} = |\text{zip}(y)|/|y|$ over all, correct, and wrong rollouts; higher values indicate lower byte-level repetition.

Method	$\bar{R}_{\text{zip}}$ ↑	$\bar{R}_{\text{zip}}^{\text{correct}}$ ↑	$\bar{R}_{\text{zip}}^{\text{wrong}}$ ↑
DeepSeek-1.5B	0.356	0.364	0.323
+ *ThinkPrune-1k*	0.388	0.395	0.363
+ *ACPO*	0.397	0.405	0.366
+ *CRT (ours)*	**0.423**	**0.433**	**0.383**

that do not materially contribute to solving the problem. We therefore analyze *internal language redundancy* in addition to token count.

Compression Ratio as a Redundancy Metric. Redundant text is inherently more compressible. Let $\text{zip}(\cdot)$ denote a deterministic compression algorithm (e.g., gzip), and let $|\cdot|$ denote the byte length of a rollout-level response, including chain-of-thought tokens. We define the compression ratio as $R_{\text{zip}}(y) = \frac{|\text{zip}(y)|}{|y|}$, where a higher value of R_{zip} indicates lower byte-level repetition. As shown in Table 3, CRT achieves the highest R_{zip} on GSM8K across all responses and within both correct and wrong subsets. Notably, although CRT has average length comparable to ThinkPrune-1k (Table 7, GSM8K column), its higher R_{zip} indicates that the remaining reasoning is less repetitive, showing that average length alone misses structural redundancy. This deterministic diagnostic supports the interpretation that CRT reduces redundant reasoning patterns rather than merely shortening outputs.

5.4. Ablation Study

We evaluate the effectiveness of the proposed two-stage training scheme by comparing full CRT against a single-stage variant that follows only Stage I for the same number of training steps, denoted as *CRT (1s)*. In addition, we include a primal-dual baseline, denoted as *PD*, which jointly optimizes accuracy and length within a single-stage primal-dual formulation. As shown in Table 4, *CRT (1s)* consistently underperforms the full two-stage CRT across both in-domain and out-domain evaluations in terms of AES scores. Notably, it exhibits degraded accuracy and longer length on in-domain data, indicating that aggressively minimizing length alone is insufficient to recover reasoning quality once compression saturates. In contrast, the second stage of CRT effectively restores accuracy without substantially increasing response length, leading to improved accuracy-efficiency trade-off. This effect is especially pronounced under AES_2, which heavily penalizes accuracy degradation, highlighting the importance of Stage II in preserving correctness while maintaining the learned efficiency profile discovered in Stage I.

Table 4. Evaluation of CRT variants. DeepSeek-1.5B is used as the base model. Detailed per-dataset results are reported in Table 13.

Method	$\overline{A}$	$\overline{L}$	AES_1	AES_2
In-domain				
CRT	85.35	2499.2	0.29	0.29
CRT (1s)	84.52	2549.6	0.24	0.22
PD	82.28	2461.0	0.13	-0.02
Out-domain				
CRT	52.43	8786.4	0.21	0.21
CRT (1s)	52.03	8613.0	0.20	0.20
PD	50.10	8607.4	0.06	-0.02

6. Conclusion

We presented Constraint-Rectified Training (CRT), a reference-guarded post-training framework for adaptive efficient reasoning in foundation models. CRT treats reasoning tokens as an inference resource and formulates efficient reasoning as boundary-controlled quality–resource optimization. Instead of imposing a fixed length cap or mixing correctness and brevity into a single scalar reward, CRT first minimizes reasoning length while preserving accuracy relative to a frozen reference policy, thereby discovering a short reliable reasoning regime. It then uses this regime as a learned token budget and refines correctness without drifting back toward unnecessary verbosity.

Empirically, CRT improves the accuracy–efficiency frontier over length-aware baselines across multiple benchmarks and backbones, while cross-domain evaluation suggests that the reference guard helps avoid severe degradation outside the math training domain. Beyond token count, our gzip-based and judge-based analyses indicate that CRT reduces repetitive reasoning structure rather than merely truncating outputs. These results support constrained, budget-aware post-training as a practical mechanism for resource-efficient foundation model inference.

Several limitations remain. First, our evaluation covers open-source backbones and reasoning domains, but does not fully characterize CRT on frontier-scale models or broader long-form generation tasks. Second, the method relies on reliable automatic correctness signals, which are easier to obtain for math, code, and multiple-choice QA than for open-ended tasks. Finally, CRT currently learns model-level reasoning-budget behavior through post-training, but does not yet implement explicit per-instance budget routing at inference time. Future work can extend CRT toward instance-adaptive reasoning budgets, richer constraints such as factuality, safety, and faithfulness, and integration with test-time compute allocation mechanisms.

References

Arora, D. and Zanette, A. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Chen, X., Xu, J., Liang, T., He, Z., Pang, J., Yu, D., Song, L., Liu, Q., Zhou, M., Zhang, Z., et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Chen, Y., Mao, Y., Yang, X., Ge, S., Bi, S., Liu, L., Hosseini, S., Tan, L., Nie, Y., and Nie, S. Your thoughts tell who you are: Characterize the reasoning patterns of lrms. *arXiv preprint arXiv:2509.24147*, 2025.

Cheng, X., Li, J., Zhang, Z., Tang, X., Zhao, X., KONG, X., and Zhang, Z. Incentivizing dual process thinking for efficient large language model reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=ezOfR26pGQ.

Cheng, Z., Chen, D., Fu, M., and Zhou, T. Optimizing length compression in large reasoning models. *arXiv preprint arXiv:2506.14755*, 2025b.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Dai, M., Yang, C., and Si, Q. S-grpo: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., and Tian, Y. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., Liu, J., Qi, L., Liu, Z., and Sun, M. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *ACL (1)*, pp. 3828–3850, 2024. URL https://doi.org/10.18653/v1/2024.acl-long.211.

He, C., Jiang, X., Wang, L., Yang, H., Peng, C., Yan, P., Shen, F., and Xu, X. What makes reasoning invalid: Echo reflection mitigation for large language models. *arXiv preprint arXiv:2511.06380*, 2025.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Hou, B., Zhang, Y., Ji, J., Liu, Y., Qian, K., Andreas, J., and Chang, S. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

Hu, S., Cai, X., Huang, Y., Yao, Z., Zhang, L., Zhang, P., Deng, Y., and Chen, K. How llms learn to reason: A complex network perspective. *arXiv preprint arXiv:2509.23629*, 2025.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jia, M. Aime_2024. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024. Hugging Face Datasets.

Jose, A. Reasoning models sometimes output illegible chains of thought. *arXiv preprint arXiv:2510.27338*, 2025.

Kang, Y., Sun, X., Chen, L., and Zou, W. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24312–24320, 2025.

Li, G., Chen, Y., Lin, M., and Yang, T. Drpo: Efficient reasoning via decoupled reward policy optimization. *arXiv preprint arXiv:2510.04474*, 2025a.

Li, R., Luo, Z., Zhang, Q., Li, R., Zhou, B., Payani, A., and Du, X. Aalc: Large language model efficient reasoning via adaptive accuracy-length control. *arXiv preprint arXiv:2506.20160*, 2025b.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Liu, H., Cao, L., Ren, Y., Zhou, M., Dong, H., Ma, X., Han, S., and Zhang, D. Bingo: Boosting efficient reasoning of llms via dynamic and significance-based reinforcement learning. *arXiv preprint arXiv:2506.08125*, 2025a.

Liu, J., Dai, W., Huang, Z., Miao, N., and Chen, E. Verifying large language models' reasoning paths via correlation matrix rank. *arXiv preprint arXiv:2510.24299*, 2025b.

Liu, T., Guo, Q., Hu, X., Jiayang, C., Zhang, Y., Qiu, X., and Zhang, Z. Can language models learn to skip steps? *Advances in Neural Information Processing Systems*, 37: 45359–45385, 2024.

Luo, H., Shen, L., He, H., Wang, Y., Liu, S., Li, W., Tan, N., Cao, X., and Tao, D. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. In *2nd AI for Math Workshop @ ICML 2025*, 2025a. URL https://openreview.net/forum?id=ioYybCRcyW.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J.-G., Tao, C., Geng, X., Lin, Q., Chen, S., Tang, Y., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=mMPMHWOdOy.

math ai. amc23, 2023. URL https://huggingface.co/datasets/math-ai/amc23.

Phan, H., Yang, X., Yao, K., Zhang, J., Bi, S., Tang, X., Khabsa, M., Liu, L., and Lei, D. Beyond reasoning gains: Mitigating general capabilities forgetting in large reasoning models. *arXiv preprint arXiv:2510.21978*, 2025.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shen, Y., Zhang, J., Huang, J., Shi, S., Zhang, W., Yan, J., Wang, N., Wang, K., Liu, Z., and Lian, S. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Singh, J., Chen, J. C.-Y., Prasad, A., Stengel-Eskin, E., Nambi, A., and Bansal, M. Think right: Learning to mitigate under-over thinking via adaptive, attentive compression. *arXiv preprint arXiv:2510.01581*, 2025.

Singla, P., Garg, S., Singh, A., Garg, I., and Saichandran, K. S. Thinking about thinking: Evaluating reasoning in post-trained language models. *arXiv preprint arXiv:2510.16340*, 2025.

Su, J. and Cardie, C. Thinking fast and right: Balancing accuracy and reasoning length with adaptive rewards. *arXiv preprint arXiv:2505.18298*, 2025.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. 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.

Wang, Y., Liu, Q., Xu, J., Liang, T., Chen, X., He, Z., Song, L., Yu, D., Li, J., Zhang, Z., et al. Thoughts are all over the place: On the underthinking of long reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Wang, Y., Li, C., Chen, G., Liang, J., and Wang, T. Reasoning or retrieval? a study of answer attribution on large reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DfxkLdy2Cd.

Weatherhead, S., Salim, F., and Belbasis, A. Illusions of reflection: open-ended task reveals systematic failures in large language models' reflective reasoning. *arXiv preprint arXiv:2510.18254*, 2025.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xia, H., Leong, C. T., Wang, W., Li, Y., and Li, W. Token-skip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yi, J., Wang, J., and Li, S. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=MJvwM5dBZM`.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., MA, Z., and He, J. SimpleRL-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=vSMCBUgrQj`.

Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., Chen, W., and Duan, N. Agieval: A human-centric benchmark for evaluating foundation models. In *NAACL-HLT (Findings)*, pp. 2299–2314, 2024. URL `https://doi.org/10.18653/v1/2024.findings-naacl.149`.

Zhu, J., Zheng, C., Lin, J., Du, K., Wen, Y., Yu, Y., Wang, J., and Zhang, W. Retrieval-augmented process reward model for generalizable mathematical reasoning. *arXiv preprint arXiv:2502.14361*, 2025.

A. Hardware Details

All baselines and our proposed method were trained on systems with the following hardware configuration. For reference, post-training the 1.5B backbone typically finishes within 21 hours under this setup.

- Operating system: Red Hat Enterprise Linux (RHEL) 9
- Type of CPU: Intel Xeon Platinum 8470 (Sapphire Rapids, 52 cores) running at 2.0 GHz
- Type of GPU: NVIDIA H100 ”Hopper” with 94 GB memory

All model inference and evaluation were conducted on separate systems with the following hardware configuration. As a representative example, evaluating a 1.5B checkpoint on AIME24 takes approximately 30 minutes under this setup.

- Operating system: Red Hat Enterprise Linux (RHEL) 9
- Type of CPU: AMD EPYC 7H12 (64 cores) running at 2.6 GHz
- Type of GPU: NVIDIA A100 ”Ampere” with 40 GB memory

B. Training Hyperparameters

All baselines and our proposed CRT use a shared GRPO training configuration unless explicitly noted. Table 5 lists the values used in our DeepSeek-1.5B experiments; the same configuration is reused for the Qwen3-4B and DeepSeek-7B runs. The training corpus is the SimpleRL-Zoo math reasoning mixture obtained by concatenating the GSM8K and MATH training splits (14,973 prompts total). For CRT, we use the default tolerances $\varepsilon = \delta = 0$ and switching margin $\tau = 0$ unless otherwise specified. These values are not tuned beyond the shared GRPO configuration.

Table 5. Shared training configuration for all GRPO-based methods (DeepSeek-1.5B reference run).

Hyperparameter	**Value**
Optimizer	AdamW ($\beta = (0.9, 0.999)$)
Learning rate	1×10^{-6}
Weight decay	0.01
Gradient clipping (global norm)	1.0
LR warmup steps	0
Training epochs	1 ($\sim$ 116 steps for DeepSeek-1.5B)
Batch size (prompts/step)	128
Rollouts per prompt	16
Max training prompt length	2048 tokens
Max training response length	4096 tokens
PPO clip ratio	0.2
Sampling temperature (train)	0.6
Sampling top-p (train)	1.0
CRT-specific	
Accuracy tolerance ε	0
Length tolerance δ	0
Switching margin τ	0

C. Additional Details

Length Normalization. We implement response-length normalization using a per-prompt standardization scheme:

$$\ell_{\text{norm}}(y) = \sigma\left(\frac{\text{LEN}(y) - \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[\text{LEN}(y)]}{\sqrt{\text{Var}_{y\sim\pi_\theta(\cdot|x)}[\text{LEN}(y)]}}\right). \tag{6}$$

where the expectation and variance are computed over responses sampled from the current policy. σ denotes the sigmoid function. This normalization preserves the relative ordering of response lengths while making length penalties comparable across prompts with different intrinsic difficulty.

Accuracy-Efficiency Score (AES). This metric was first introduced in O1-Pruner (Luo et al., 2025a), which jointly considers the relative change in accuracy (ΔA) and response length (ΔL), defined as

$$\Delta A = \frac{A_{\text{model}} - A_{\text{base}}}{A_{\text{base}}}, \qquad \Delta L = \frac{L_{\text{base}} - L_{\text{model}}}{L_{\text{base}}},$$

where $\Delta L > 0$ indicates a shorter response than the baseline. The AES is computed as

$$\text{AES} = \begin{cases} \alpha \Delta L + \beta |\Delta A|, & \Delta A \geq 0, \\ \alpha \Delta L - \gamma |\Delta A|, & \Delta A < 0, \end{cases}$$

where α controls the sensitivity to efficiency gain, while β and γ govern the reward and penalty magnitudes for accuracy gain or loss, respectively. We report two variants: AES_1 with $(\alpha, \beta, \gamma) = (1, 3, 5)$ (Luo et al., 2025a) and AES_2 with $(\alpha, \beta, \gamma) = (1, 3, 10)$ (Li et al., 2025a).

Full CRT Algorithm.

Algorithm 1 Constraint-Rectified Training

Input: θ_0, π_{ref}, tolerances ε, δ, margin τ, learning rate η_θ, total steps T, transition window K.
$\theta \leftarrow \theta_0$, stage $\leftarrow$ I; define $r_{\text{acc}}(y, x) = \mathbb{1}\{y = y^*(x)\}$, $r_{\text{len}}(y) = -\ell_{\text{norm}}(y)$.
for $t = 0, \ldots, T-1$ **do**
  Sample $\{x_i\}_{i=1}^B \sim \rho$, $y_i \sim \pi_\theta(\cdot \mid x_i)$.
  if stage = I **then**
    Sample $y_i^{\text{ref}} \sim \pi_{\text{ref}}(\cdot \mid x_i)$; compute $\hat{A}_t = \frac{1}{B}\sum_i r_{\text{acc}}(y_i, x_i)$, $\hat{A}^{\text{ref}} = \frac{1}{B}\sum_i r_{\text{acc}}(y_i^{\text{ref}}, x_i)$.
    $v_t \leftarrow \mathbb{1}\{\hat{A}_t < \hat{A}^{\text{ref}} - \varepsilon - \tau\}$.
    $r_i \leftarrow r_{\text{acc}}(y_i, x_i)$ if $v_t = 1$, else $r_i \leftarrow r_{\text{len}}(y_i)$.
  else
    Sample $y_i^{(I)} \sim \pi_{\text{ref}}^{(I)}(\cdot \mid x_i)$; compute $\hat{L}_t = \frac{1}{B}\sum_i \ell_{\text{norm}}(y_i)$, $\hat{L}^{(I)} = \frac{1}{B}\sum_i \ell_{\text{norm}}(y_i^{(I)})$.
    $u_t \leftarrow \mathbb{1}\{\hat{L}_t > \hat{L}^{(I)} + \delta + \tau\}$.
    $r_i \leftarrow r_{\text{len}}(y_i)$ if $u_t = 1$, else $r_i \leftarrow r_{\text{acc}}(y_i, x_i)$.
  end if
  GRPO-update θ using $\{r_i\}_{i=1}^B$.
  if stage = I and $\bar{v}_t$ is stable by Eq. (5) **then**
    Freeze $\pi_{\text{ref}}^{(I)} \leftarrow \pi_\theta$ and length-normalization statistics; stage $\leftarrow$ II.
  end if
end for
Output: π_θ.

D. Additional Results

D.1. Accuracy Stability Under Length Reduction

Table 6. Accuracy stability conditioned on successful length reduction. AD/AP/AI report the percentage of prompts whose average response length is reduced and whose accuracy decreases, is preserved, or improves, respectively.

Method	AD↓	AP↑	AI↑	Acc↑
SAT Math				
+ *ThinkPrune-1k*	37.5	45.8	16.7	81.45
+ *ACPO*	27.3	40.9	31.8	91.02
+ *CRT (ours)*	**12.5**	**56.2**	31.2	**93.16**
GSM8K				
+ *ThinkPrune-1k*	34.0	46.1	19.9	81.54
+ *ACPO*	28.7	51.1	20.2	82.78
+ *CRT (ours)*	**20.0**	**55.6**	**24.4**	**84.64**

D.2. Full DeepSeek-1.5B Details

Table 7. Full results of experiments based on DeepSeek-1.5B.

Method	GSM8K		MATH500		SAT		AMC23		AIME24		Olymp.		Avg		Metrics	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	$\overline{A}$	$\overline{L}$	AES_1	AES_2
DeepSeek-1.5B	84.43	1869	85.19	4987	89.84	1373	69.84	8634	30.42	14044	13.25	15527	62.16	7739.0	0.00	0.00
+ *ThinkPrune-500*	78.91	995	85.25	4455	81.45	609	71.88	7715	27.92	13378	13.63	14791	59.84	6990.5	-0.09	-0.28
+ *ThinkPrune-1k*	81.54	1128	85.69	4206	81.45	759	72.97	7548	28.96	13001	14.44	14326	60.84	6828.0	0.01	-0.09
+ *O1-Pruner*	84.39	1561	85.56	4452	91.21	1500	71.88	7848	30.63	12669	12.62	14547	62.72	7096.2	0.11	0.11
+ *ShorterBetter*	71.24	294	82.47	3205	84.77	200	74.84	6068	27.08	11714	11.62	12908	58.67	5731.5	-0.02	-0.30
+ *ACPO*	82.78	1252	85.44	4175	91.02	1282	71.25	7251	30.42	12367	12.62	14189	62.26	6752.7	0.13	0.13
+ *CRT (ours)*	84.64	1172	86.06	3826	93.16	1582	74.38	6934	28.54	12674	13.63	13956	63.40	6690.7	**0.20**	**0.20**

D.3. Full Qwen3-4B Details

Table 8. Full results of experiments based on Qwen3-4B.

Method	GSM8K		MATH500		SAT		AMC23		AIME24		Olymp.		Avg		Metrics	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	$\overline{A}$	$\overline{L}$	AES_1	AES_2
Qwen3-4B	92.20	310	85.62	1113	99.02	1260	69.38	1895	24.38	6382	24.00	3876	65.77	2472.8	0.00	0.00
+ *ThinkPrune-500*	91.45	281	84.53	1253	99.22	772	67.97	1806	22.71	5332	23.88	3214	64.96	2109.8	0.09	0.02
+ *ThinkPrune-1k*	90.32	287	84.78	1167	98.83	1004	66.09	1631	23.33	4879	23.38	3074	64.46	2006.9	0.09	-0.01
+ *O1-Pruner*	90.20	186	82.12	849	98.05	287	63.44	1459	19.58	3222	22.56	2364	62.66	1394.3	0.20	-0.04
+ *ShorterBetter*	90.16	133	80.28	598	96.48	1092	62.19	1064	16.04	3235	22.19	1939	61.22	1343.4	0.11	-0.23
+ *ACPO*	89.98	196	83.38	755	99.22	549	62.81	1285	21.25	4586	22.06	2683	63.12	1675.6	0.12	-0.08
+ *CRT (ours)*	90.20	222	82.94	790	98.83	318	64.06	1387	20.21	3264	21.69	2188	62.99	1361.4	**0.24**	**0.03**

D.4. Full DeepSeek-7B Details

Table 9. Evaluation results using DeepSeek-7B.

Method	GSM8K		MATH500		SAT		AMC23		AIME24		Olymp.		Avg		Metrics	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	$\overline{A}$	$\overline{L}$	AES_1	AES_2
DeepSeek-7B	93.29	1715	93.00	3895	98.24	1685	89.22	6247	52.29	12193	21.35	14915	74.57	6775.0	0.00	0.00
+ *ThinkPrune*	92.72	1410	93.03	3593	96.09	1130	90.00	5853	50.00	11669	21.74	14355	73.93	6334.8	0.02	-0.02
+ *ShorterBetter*	88.84	761	92.84	3159	92.77	494	89.06	5261	51.67	10710	19.79	13194	72.50	5596.5	0.04	-0.10
+ *ACPO*	92.62	1172	93.59	3300	98.83	1076	90.16	5439	53.54	11484	20.31	13800	74.84	6045.1	0.12	0.12
+ *CRT (ours)*	93.78	1391	92.38	3299	99.02	1508	90.47	5288	55.21	10553	22.01	13435	75.48	5912.3	**0.16**	**0.16**

D.5. Full Cross-domain Evaluation Details

This section accompanies the cross-domain summary in Table 2. All checkpoints are obtained by training DeepSeek-1.5B on the math reasoning mixture in Section 5.1 and are evaluated on each task without further tuning. We use the MBPP-sanitized test split for code generation (Austin et al., 2021), where Acc denotes the average pass rate over 10 sampled completions per problem (temperature 0.6, top-p 0.95), each executed against the dataset's test cases (one of which is also shown in the prompt as a worked example). For MMLU (Hendrycks et al., 2021a), we sample 285 test questions stratified across all 57 subjects and evaluate 16 rollouts per question, scoring each response by extracting the letter answer from the boxed expression and matching it against the ground truth. ARC-Challenge (Clark et al., 2018) uses the first 300 of 1,172 science test questions with 16 rollouts and the same extraction protocol.

Table 10. Code generation results on MBPP-sanitized.

Method	Acc	Len	AES_1	AES_2
DeepSeek-1.5B	55.84	3134	0	0
+ *ThinkPrune-1k*	55.21	2767	0.06	0.01
+ *ThinkPrune-500*	55.14	2868	0.02	-0.04
+ *O1-Pruner*	55.29	2880	0.03	-0.02
+ *ShorterBetter*	53.62	2365	0.05	-0.15
+ *ACPO*	54.82	2697	0.05	-0.04
+ *CRT (ours)*	55.60	2755	0.10	0.08

Table 11. Multiple-choice QA results on MMLU.

Method	Acc	Len	AES_1	AES_2
DeepSeek-1.5B	49.61	1519	0	0
+ *ThinkPrune-1k*	47.21	1035	0.08	-0.16
+ *ThinkPrune-500*	47.52	950	0.16	-0.05
+ *O1-Pruner*	48.46	1191	0.10	-0.01
+ *ShorterBetter*	42.96	327	0.12	-0.55
+ *ACPO*	47.83	1101	0.10	-0.08
+ *CRT (ours)*	48.49	1118	0.15	0.04

Table 12. Multiple-choice QA results on ARC-Challenge.

Method	Acc	Len	AES_1	AES_2
DeepSeek-1.5B	60.46	1154	0	0
+ *ThinkPrune-1k*	56.42	621	0.13	-0.21
+ *ThinkPrune-500*	56.27	484	0.23	-0.11
+ *O1-Pruner*	57.23	691	0.13	-0.13
+ *ShorterBetter*	54.31	149	0.36	-0.15
+ *ACPO*	57.29	647	0.18	-0.08
+ *CRT (ours)*	57.83	611	0.25	0.04

D.6. Full Ablation Results

Table 13. Full results of CRT ablation study on all testing datasets.

Method	GSM8K		MATH500		SAT		AMC23		AIME24		Olymp.		Avg		Metrics	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	$\overline{A}$	$\overline{L}$	AES_1	AES_2
DeepSeek-1.5B	84.43	1869	85.19	4987	89.84	1373	69.84	8634	30.42	14044	13.25	15527	62.16	7738.9	0	0
CRT	84.64	1172	86.06	3826	93.16	1582	74.38	6934	28.54	12674	13.63	13956	63.4	6690.7	0.20	0.20
CRT (1s)	83.23	1102	85.81	3998	93.16	1214	72.66	6982	29.17	12426	13.12	13831	62.86	6591.8	0.18	0.18
PD	79.24	975	85.31	3947	88.67	803	71.72	7337	27.5	12540	12.5	13750	60.82	6558.6	0.04	-0.06

D.7. Hyperparameter Ablation: Switching Margin τ

The CRT controller in Algorithm 1 uses a margin τ when deciding whether to activate constraint rectification. In Stage I, the controller compares the current accuracy estimate $\hat{A}_t$ against the reference-guarded threshold $\hat{A}^{\text{ref}} - \varepsilon - \tau$, where ε is the accuracy tolerance. This indicator also determines the violation sequence used by the autonomous stage-transition criterion. In Stage II, the same margin is applied to the learned length-budget constraint, activating length rectification when the current length exceeds the Stage I length reference by more than $\delta + \tau$. A negative margin makes compression more conservative by requiring the current policy to exceed the reference accuracy floor before length minimization is activated, whereas a positive margin permits a small constraint gap before rectification is triggered. We ablate $\tau \in \{-0.0625,\ 0,\ 0.125\}$ in Tables 14 and 15. The default $\tau = 0$ remains within 0.02 AES of the best setting in both regimes, suggesting that CRT is not sensitive to this margin and can use the parameter-free default without additional tuning.

Table 14. In-domain τ ablation on GSM8K and MATH500. Base model: DeepSeek-1.5B. AES values are reported relative to the default $\tau = 0$ setting.

Setting	$\overline{A}$	$\overline{L}$	AES_1	AES_2
$\tau = 0$	85.42	2968.1	0	0
$\tau = -0.0625$	85.48	2915.9	0.02	0.02
$\tau = 0.125$	84.95	2889.8	0	-0.03

Table 15. Out-of-domain τ ablation on SAT MATH, AMC23, AIME24, OLYMPIAD Bench. Base model: DeepSeek-1.5B. AES values are reported relative to the default $\tau = 0$ setting.

Setting	$\overline{A}$	$\overline{L}$	AES_1	AES_2
$\tau = 0$	52.19	9225.9	0	0
$\tau = -0.0625$	52.28	9122.9	0.02	0.02
$\tau = 0.125$	51.65	9156.3	-0.04	-0.10

E. Controlling Explanation Length with Correctness Guarantees

A distinctive advantage of CRT is its ability to provide fine-grained, correctness-preserving control over explanation detail. Unlike reward-mixing methods that typically optimize toward a single length–accuracy operating point, CRT yields a sequence of intermediate checkpoints spanning different verbosity levels while maintaining answer quality. This allows a single training run to support multiple deployment regimes: later checkpoints can serve latency- or token-budget-constrained settings, while earlier checkpoints can serve educational tutoring, audit trails, or human-in-the-loop debugging where more intermediate reasoning is desirable.

To illustrate this behavior, we visualize checkpoint-level rollout behavior on GSM8K in Figure 2 (Right), and provide a qualitative case study in Figure 3.

As shown in Figure 3, earlier checkpoints generate detailed, self-verifying reasoning traces, whereas later checkpoints converge to compact and direct solutions. Despite substantial differences in explanation length and structure, all checkpoints

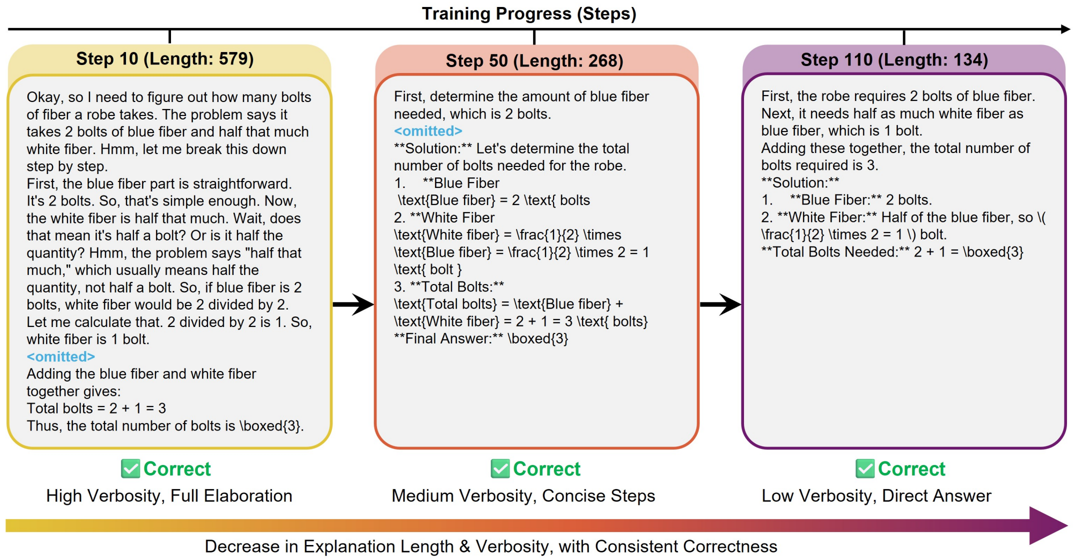

Figure 3. Controlling explanation length while preserving correctness across training checkpoints. The same GSM8K example remains correct across checkpoints, while the reasoning trace becomes progressively shorter and more direct.

produce the same correct final answer. This progression shows that explanation length can be adjusted continuously through training progress, rather than discretely tuned via heuristic length budgets, while preserving correctness.

Full rollout text for Figure 3.

Step 10 (Length: 579)

Okay, so I need to figure out how many bolts of fiber a robe takes. The problem says it takes 2 bolts of blue fiber and half that much white fiber. Hmm, let me break this down step by step.
First, the blue fiber part is straightforward. It's 2 bolts. So, that's simple enough. Now, the white fiber is half that much. Wait, does that mean it's half a bolt? Or is it half the quantity? Hmm, the problem says "half that much," which usually means half the quantity, not half a bolt. So, if blue fiber is 2 bolts, white fiber would be 2 divided by 2. Let me calculate that. 2 divided by 2 is 1. So, white fiber is 1 bolt.
Now, to find the total number of bolts, I need to add the blue fiber and white fiber together. So, that's 2 bolts plus 1 bolt. Let me do the addition: 2 + 1 equals 3. So, altogether, the robe takes 3 bolts of fiber.
Wait, just to double-check, sometimes in math problems, especially with word problems, the wording can be tricky. The problem says "half that much." So, is it 50% more than half of the blue fiber? No, that doesn't make sense because it's saying half that much, not half more. So, half of 2 is 1. That still seems correct.
Alternatively, if someone misreads it as half as much, that would be 1 bolt, but the problem says half that much, which is 1. So, yes, 1 bolt white fiber. Adding them together gives 3 total bolts.
I don't think there's any other way to interpret "half that much." If it were half of the blue fiber, that would be 1, but half that much would mean half the amount, so half of 2 is 1. Either way, it's 1. So, adding them gives 3.
I think that's solid. So, the total number of bolts is 3.
Final Answer The robe takes a total of $\boxed{3}$ bolts of fiber.
The robe takes 2 bolts of blue fiber. The white fiber is half that much, which is calculated as follows:

$$\text{White fiber} = \frac{2}{2} = 1 \text{ bolt}$$

Adding the blue fiber and white fiber together gives:

$$\text{Total bolts} = 2 + 1 = 3$$

Thus, the total number of bolts is $\boxed{3}$.

Step 50 (Length: 268)

```
First, determine the amount of blue fiber needed, which is 2 bolts.
Next, calculate the white fiber required.  It's half of the blue fiber, so 2
divided by 2 equals 1 bolt.
Finally, add the two amounts together to find the total number of bolts:  2
plus 1 equals 3 bolts.
**Solution:**
Let's determine the total number of bolts needed for the robe.
1.  **Blue Fiber:**
```

$$\texttt{Blue fiber} = 2 \texttt{ bolts}$$

```
2.  **White Fiber:**
```

$$\texttt{White fiber} = \frac{1}{2} \times \texttt{Blue fiber} = \frac{1}{2} \times 2 = 1 \texttt{ bolt}$$

```
3.  **Total Bolts:**
```

$$\texttt{Total bolts} = \texttt{Blue fiber} + \texttt{White fiber} = 2 + 1 = 3 \texttt{ bolts}$$

```
**Final Answer:**
```

$$\boxed{3}$$

Step 110 (Length: 134)

```
First, the robe requires 2 bolts of blue fiber.
Next, it needs half as much white fiber as blue fiber, which is 1 bolt.
Adding these together, the total number of bolts required is 3.
**Solution:**
1.  **Blue Fiber:** 2 bolts.
```
`2.  **White Fiber:** Half of the blue fiber, so` $\frac{1}{2} \times 2 = 1$ `bolt.`

`**Total Bolts Needed:**` $2 + 1 = \boxed{3}$

F. LLM-as-a-Judge Evaluation

To complement the deterministic redundancy diagnostic in Table 3, we run an LLM-as-a-judge analysis along nine quality and efficiency dimensions, using DeepSeek-V3.2 as the judge model. Rollouts are sampled from each method's final checkpoint on GSM8K problems with the same sampling parameters used in the main experiments. For each rollout, the judge scores reasoning coherence, solution directness, mathematical rigor, step completeness, reasoning clarity, brevity cost, crystal clear reasoning, wasted tokens, and overall efficiency. Quality dimensions (coherence, directness, rigor, completeness, clarity, no-brevity-cost) are reported as the percentage of rollouts judged at the top of the corresponding 4-level ordinal scale; efficiency dimensions (crystal clear reasoning, no wasted tokens, excellent efficiency) are reported as the percentage of rollouts at the strictest tier of each scale.

Results are reported in Table 16. CRT compresses to a median length comparable to ThinkPrune (820 vs. 807 tokens), while achieving higher programmatic accuracy and the strongest scores on all strict efficiency dimensions. It is also best or tied-best on most quality dimensions, indicating that the efficiency gain does not come from broad degradation of reasoning quality. In particular, CRT yields $3.4\times$ as many *Excellent Efficiency* ratings as ThinkPrune (15.5% vs. 4.5%) and $2.5\times$ as many *Crystal Clear Reasoning* ratings (13.5% vs. 5.5%). Together with the gzip analysis in Table 3, these results support the interpretation that CRT reduces redundant reasoning patterns rather than merely shortening outputs.

Table 16. LLM-as-a-judge evaluation on GSM8K rollouts at each method's final checkpoint, scored by DeepSeek-V3.2. Quality dimensions report % of rollouts at the top tier; efficiency dimensions report % of rollouts at the strictest tier. Best or tied-best values are in **bold**.

Metric	**Base**	**CRT**	**ThinkPrune**	**ACPO**
Median Length (tokens)	1276	820 (−36%)	**807** (−37%)	895 (−30%)
Programmatic Accuracy (%)	82.1	**82.9**	79.9	81.2
Quality dimensions (% at top tier)				
Coherence	96.0	**98.5**	97.5	98.5
Direct Path	96.0	**98.5**	95.5	97.5
Math Rigor	96.0	**97.0**	95.5	95.5
Step Completeness	97.0	**98.0**	97.5	98.0
Reasoning Clarity	94.0	**96.5**	96.5	96.5
No Brevity Cost	99.5	99.5	**100.0**	99.0
Efficiency dimensions (% at strictest tier)				
Crystal Clear Reasoning	1.0	**13.5**	5.5	5.0
No Wasted Tokens	0.0	**5.5**	0.0	1.5
Excellent Efficiency	1.5	**15.5**	4.5	7.0

G. Evaluation Prompts

For full reproducibility, we list below the system prompts used for each evaluation domain. The math prompt follows the DeepSeek-R1 chat template; the multiple-choice and code variants append a domain-specific final-answer-formatting clause.

Mathematical reasoning

```
A conversation between User and Assistant. The user asks a question, and the
Assistant solves it. The assistant first thinks about the reasoning process
in the mind and then provides the user with the answer. The reasoning process
and answer are enclosed within <think></think> and <answer></answer> tags,
respectively. Please reason step by step, and put your final answer within
boxed{}.
```

Multiple-choice QA

```
A conversation between User and Assistant. The user asks a question, and the
Assistant solves it. The assistant first thinks about the reasoning process
in the mind and then provides the user with the answer. The reasoning process
and answer are enclosed within <think></think> and <answer></answer> tags,
respectively. Please reason step by step, and put your final answer within
boxed{} which contains exactly one letter (A/B/C/D).
```

Code generation

```
Write a Python function to solve the following problem. {description} Your
code should pass this test: {test_case} Please reason step by step, and put
your final Python function inside a markdown code block.
```