TAPER: TEACHER-GUIDED, ADAPTIVE-LENGTH POLICY OPTIMIZATION FOR EFFICIENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Explicit multi-step reasoning improves the capability and reliability of large language models, but at considerable computational cost: traces are often dominated by redundant verification and unproductive backtracking. Because useful computation is entangled with this redundancy, indiscriminate shortening sacrifices both; what can be safely removed depends on how reliably the model solves the problem. We present **TAPER (Teacher-guided, Adaptive-length Policy Optimization for Efficient Reasoning)**, an on-policy framework that treats how much a model reasons and how it does so as a single learning problem. At the trajectory level, an accuracy-adaptive relative-length objective gauges the model's current competence on each problem from its own online samples, adjusting compression strength accordingly. At the token level, region-aware asymmetric teacher supervision lets a stronger teacher guide the student's own generation, with distinct objectives for the reasoning and answer regions; the two objectives are optimized jointly on the same trajectories in each update. Experiments on Qwen3.5-4B and Qwen3.5-9B across five benchmarks spanning mathematical, knowledge-intensive, and logical reasoning, against six baselines, show that TAPER improves accuracy over GRPO by 1.2–7.9 percentage points while using 18.4%–66.4% fewer tokens. Behavioral analysis further shows that TAPER reshapes the model's reasoning behavior in a difficulty-dependent manner: it reallocates the reasoning budget toward productive advancement and away from unnecessary redundancy, with the proportion of the two adjusted according to the model's competence on each problem.

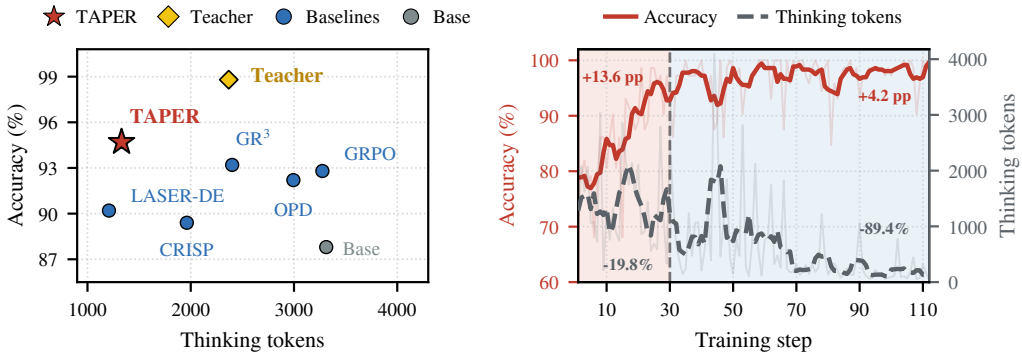

Figure 1: Accuracy–efficiency trade-off and training dynamics of TAPER on a 9B model. Left: Accuracy (avg@4) vs. average thinking tokens on MATH500 across TAPER, the base model, teacher, and baselines. Right: Rollout accuracy and thinking tokens during training. Annotations show changes between the 1–5, 28–32, and 108–112 step windows.

1 INTRODUCTION

Explicit multi-step reasoning has substantially improved the ability of large language models (LLMs) to solve complex problems (Wei et al., 2022), but at considerable computational cost (Qu et al., 2025; Wang et al., 2025), with returns diminishing as traces grow longer (Snell et al., 2025).

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

Additional computation can mix deeper reasoning with repeated verification and detours, making necessary and redundant computation difficult to separate. Evidence of this mismatch includes findings that o1-like models spend more rounds on easier problems, and that in correct traces most tokens appear after the answer is first given (Chen et al., 2024). Efficient reasoning therefore hinges on two linked questions: **which parts of a reasoning trace can be safely compressed, and how much computation each problem needs.**

The composition of a reasoning trace determines how much of it can be compressed. Tokens serve distinct roles—advancing the solution, verifying intermediate steps, and backtracking to correct errors (Yeo et al., 2025)—and differ markedly in compressibility. For problems the model already solves reliably, much of the trace is repetition and over-verification that can be pruned with little accuracy loss; pruning studies show that removing verification-like content even yields gains, whereas pruning core reasoning steps degrades performance (Zhao et al., 2025). Conversely, for problems the model has yet to master, long traces support exploratory backtracking and error correction, which emerge only given sufficient incentive (Yeo et al., 2025); compressing them costs accuracy. These observations imply that effective compression must be selective: remove what is redundant while preserving the computation each problem still requires.

Existing work approaches these two questions through reinforcement learning and distillation, but neither line alone fully addresses them. RL methods compress traces with length rewards, some further estimating difficulty online from group pass rates (Yuan et al., 2026; Xiang et al., 2025), while distillation methods impose supervision via self-distillation (Sang et al., 2026; Dong et al., 2026) or on-policy distillation (Agarwal et al., 2024). Length and outcome rewards are trajectory-level scalar signals: they demand "shorter" without indicating which steps can be removed—under pressure, the model deletes the easiest steps, which often include critical reasoning (Yuan et al., 2026; Sang et al., 2026). Process-supervision work shows that step-level feedback can improve intermediate reasoning quality, motivating similarly fine-grained guidance for compression (Lightman et al., 2024). Budget allocation and token-level guidance are also coupled: how much a trace can be compressed depends on how reliably the model solves the problem, and token-level supervision alone does not yield the desired accuracy–length trade-off. What is needed is a unified learning process that jointly decides the budget for each problem and how to use it.

To this end, we propose **TAPER (Teacher-guided, Adaptive-length Policy Optimization for Efficient Reasoning)**. Our central view is that budget allocation and generation guidance are not two pluggable modules, but two granularities of the same problem: trajectory-level signals decide how much computation each problem receives, while token-level signals decide how the budget is used internally. They must be optimized jointly because the compressible space depends on generation quality, which is in turn shaped by the budget constraint. At the trajectory level, an accuracy-adaptive group-relative length objective lets the group solve rate directly determine the compression strength—stronger when the model solves reliably, weaker when outcomes are uncertain—without any predefined target length. At the token level, region-aware asymmetric teacher supervision guides the student's own generations with different KL objectives for the reasoning and answer regions. Because both objectives act on the same batch of on-policy trajectories, compression strength and generation structure adapt to each other: the model is not taught to reason better within a fixed budget, but is allowed to use less computation as it learns to reason better.

We evaluate TAPER on Qwen3.5-4B and Qwen3.5-9B across five benchmarks covering mathematical, knowledge-intensive, and logical reasoning. Compared with standard GRPO, **TAPER reduces reasoning tokens by 18.4%–66.4% on average while improving accuracy by 1.2–7.9 percentage points**, compressing more on easier evaluation groups and less on harder ones. Behavioral analysis further shows that, relative to Base, TAPER reduces the token shares of verification and redundancy while increasing substantive advancement and revision.

Our contributions are as follows:

- We propose TAPER, a framework that jointly learns how much to reason at the trajectory level and how to reason at the token level within a single update.
- We propose an accuracy-adaptive group-relative length objective, where the group solve rate endogenously determines the compression strength, with no predefined target length.

- We propose region-aware asymmetric teacher supervision, applying separate token-level constraints to the reasoning and answer regions of student-generated trajectories.
- Across all 10 settings (two model scales $\times$ five benchmarks), TAPER is more accurate and uses fewer reasoning tokens than GRPO in every setting, and achieves the highest accuracy among student methods in all 10.

2 Related Work

Reasoning-length control and adaptive compression. Reinforcement-learning methods reduce reasoning cost with length-based rewards or penalties, along an axis that has gradually shifted: when the penalty applies, what it is relative to, and what it adapts to. Short-RL applies penalties only after a correct answer (Yuan et al., 2026); GR^3 makes the reward group-relative, favoring shorter traces within each prompt's rollouts without a fixed target length (Li et al., 2026). Further work adapts compression to training progress, model performance, or difficulty: LASER-DE adjusts its length reward by training progress and instance difficulty (Liu et al., 2026); others switch the penalty or reasoning mode by model performance or query difficulty (Su & Cardie, 2025; Zhang et al., 2025; Pardinas et al., 2026); and some estimate difficulty online from group solve rates to scale per-token penalties (Xiang et al., 2025; Feng et al., 2026).

Across all of these methods, however, the learning signal remains a trajectory-level scalar. It can shorten a trace but not indicate which steps to cut—the model deletes the easiest steps, which need not be the redundant ones.

Distillation-based reasoning guidance. Length-based rewards determine how much a model reasons; distillation shapes how it reasons. Distillation offers supervision at several levels—output distributions, sequences, and rationales (Hinton et al., 2015; Kim & Rush, 2016; Hsieh et al., 2023)—and at the step level, process reward models score individual steps and improve both outcomes and intermediate-step quality (Lightman et al., 2024), while distillation reaches the same granularity without a separately trained reward model. On-policy distillation (OPD) applies teacher supervision to student-generated trajectories, reducing offline distribution mismatch and supporting different divergences (Agarwal et al., 2024). Building on OPD, CRISP applies token-level reverse KL with a self-teacher prompted to be concise (Sang et al., 2026), and BIRD iterates on-policy self-distillation from concise, correct trajectories (Dong et al., 2026).

These methods establish that token-level teacher supervision can reshape generation on the student's own trajectories. The supervision, however, is applied uniformly: it is not adapted to how much budget a problem needs, nor coupled with any trajectory-level control—leaving computation's amount and structure to be tuned separately.

Positioning of TAPER. To our knowledge, TAPER is the first framework to extend adaptive supervision over the reasoning process to both the trajectory and token levels: the trajectory-level objective determines how much budget each problem receives, while token-level supervision guides how that budget is used within the trace. At the trajectory level, TAPER scales a group-relative trajectory reward according to the group solve rate. Because it compares lengths only among trajectories sampled for the same problem, this objective requires neither an absolute length scale nor a predefined target length. At the token level, a fixed teacher provides region-aware distillation with asymmetric KL objectives: forward KL in the reasoning region and reverse KL in the answer region. Together, these two forms of supervision are optimized in a single on-policy update, allowing the model to learn jointly how to allocate each problem's budget and how to use it, rather than tuning the two separately.

3 Methodology: TAPER

This section formalizes TAPER. Given a problem x, the model generates a response $y = (y^{\text{think}}, y^{\text{ans}})$, where y^{think} is the reasoning trace and y^{ans} is the final answer; task performance is measured by final-answer correctness $r^{\text{acc}} \in \{0, 1\}$, and reasoning cost by thinking length $\ell = |y^{\text{think}}|$. Sections 3.1–3.2 instantiate the two granularities—how much to reason (an accuracy-

adaptive group-relative length objective) and how the budget is used (region-aware asymmetric distillation); Section 3.3 gives the joint objective.

3.1 Accuracy-Adaptive Group-Relative Length Objective

For each problem x, the current policy $\pi_{\theta_{\text{old}}}$ samples G responses $\{y_i\}_{i=1}^{G}$ with thinking lengths ℓ_i and correctness indicators $r_i^{\text{acc}} \in \{0, 1\}$. Since the benefit of additional reasoning varies with difficulty (Snell et al., 2025), length optimization should adapt across problems: instead of a uniform token budget, we compare each trajectory against the other samples for the same problem. Let $\ell_{\min}$, $\ell_{\max}$, and $\bar{\ell}$ be the group's minimum, maximum, and mean thinking lengths. The relative length reward is

$$r_i^{\text{len}} = \begin{cases} \dfrac{\bar{\ell} - \ell_i}{\bar{\ell} - \ell_{\min}}, & \ell_i \le \bar{\ell}, \\ \dfrac{\bar{\ell} - \ell_i}{\ell_{\max} - \bar{\ell}}, & \ell_i > \bar{\ell}. \end{cases} \tag{1}$$

By definition, $r_i^{\text{len}} \in [-1, 1]$: trajectories below the group mean are rewarded, and longer ones are penalized. However, applying the same compression pressure regardless of the policy's current ability can over-compress problems the model has not yet mastered. We therefore gauge the model's current competence on each problem by its group solve rate—the fraction of correct rollouts within the group:

$$p_x = \frac{1}{G} \sum_{i=1}^{G} r_i^{\text{acc}}, \tag{2}$$

and modulate the length reward with it:

$$R_i = \lambda_{\text{acc}} r_i^{\text{acc}} + \lambda_{\text{fmt}} r_i^{\text{fmt}} + \lambda_{\text{len}} p_x r_i^{\text{len}}, \tag{3}$$

where r_i^{fmt} is a binary format reward for output-format compliance, and the λ coefficients are reward weights. Compression thus weakens when the solve rate is low and strengthens as the policy improves, without annotated difficulty labels or a fixed length budget.

We center and normalize R_i within each group to obtain the group-relative advantage A_i, and optimize the student with the GRPO policy loss (Shao et al., 2024), denoted $\mathcal{L}_{\text{GRPO}}$. The full policy objective, reference-policy regularization, reward construction, and boundary-case handling are given in Appendix D.

3.2 Region-Aware Asymmetric On-Policy Distillation

Trajectory-level rewards supervise only the correctness and length of the whole trace and cannot guide token-level predictions during generation. We therefore distill token-level supervision from a stronger teacher model π_T along the student's own generated trajectories (Agarwal et al., 2024). For each sampled response y_i, the prefix state visited by the student at position t is $s_{i,t} = (x, y_{i,<t})$, with teacher and student distributions $q_{i,t} = \pi_T(\cdot \mid s_{i,t})$ and $p_{i,t} = \pi_\theta(\cdot \mid s_{i,t})$. The teacher is queried only at student-visited states and generates no trajectories of its own.

The two regions impose different supervision needs: the reasoning region should preserve multiple viable paths—the student must remain free to explore rather than be locked into the teacher's single trajectory—whereas the answer region must align strictly with the teacher, as the final answer must be exact. Forward and reverse KL match these two needs: forward KL penalizes missing mass on any continuation the teacher considers possible, preserving multiple candidate paths; reverse KL suppresses probability flow toward tokens the teacher disfavors, concentrating the student on the teacher's high-probability modes (Wu et al., 2025). We therefore apply forward KL in the thinking region and reverse KL in the answer region.

Let T_i^{think} and T_i^{ans} denote the token positions before and after the `</think>` boundary, respectively. To cut logit-transfer and KL costs, we keep only the top-20 teacher-probability tokens at each position and renormalize both distributions within this set, denoted $\tilde{q}_{i,t}$ and $\tilde{p}_{i,t}$. In the thinking region we use forward KL, $D_{\text{FKL}}^{i,t} = D_{\text{KL}}(\tilde{q}_{i,t} \parallel \tilde{p}_{i,t})$; in the answer region, reverse KL, $D_{\text{RKL}}^{i,t} = D_{\text{KL}}(\tilde{p}_{i,t} \parallel \tilde{q}_{i,t})$.

To prevent longer traces from gaining weight merely by containing more tokens, we normalize each region by its token count and average trajectories equally, yielding

$$\mathcal{L}_{\mathrm{OPD}} = \frac{1}{B}\sum_{i=1}^{B}\left[\frac{w_{\mathrm{think}}}{|T_i^{\mathrm{think}}|}\sum_{t\in T_i^{\mathrm{think}}} D_{\mathrm{FKL}}^{i,t} + \frac{w_{\mathrm{ans}}}{|T_i^{\mathrm{ans}}|}\sum_{t\in T_i^{\mathrm{ans}}} D_{\mathrm{RKL}}^{i,t}\right], \tag{4}$$

where $w_{\mathrm{think}} = 0.1$ and $w_{\mathrm{ans}} = 1.0$. The smaller thinking-region weight avoids over-constraining intermediate reasoning; the larger answer-region weight provides stronger supervision for the final answer. Student logits at the selected token IDs remain differentiable. This sparse objective matches relative probabilities only within the retained set and does not directly constrain the total probability mass outside it, which we find has no observable negative effect in practice. Details on sparse normalization, gradient propagation, and empty-region handling are given in Appendix E.

3.3 Joint Optimization

TAPER jointly optimizes the trajectory-level policy loss and the token-level distillation loss on the same freshly sampled student trajectories:

$$\mathcal{L}_{\mathrm{TAPER}} = \mathcal{L}_{\mathrm{GRPO}} + \eta_{\mathrm{OPD}}\mathcal{L}_{\mathrm{OPD}}, \qquad \eta_{\mathrm{OPD}} = 0.3. \tag{5}$$

The two objectives act on two granularities of the same problem: $\mathcal{L}_{\mathrm{GRPO}}$ adjusts how much to reason for each problem, while $\mathcal{L}_{\mathrm{OPD}}$ guides how to reason throughout compression.

4 Experiments

4.1 Experimental Setup

Models and tasks. We evaluate TAPER on Qwen3.5-4B and Qwen3.5-9B, with Qwen3.5-122B-A10B serving as the distillation teacher (Qwen Team, 2026). The teacher is kept frozen throughout training and is also evaluated under the same protocol as an accuracy–efficiency reference. Our experiments span mathematical, logical, and knowledge-intensive reasoning. For mathematics, models are trained on the MATH training set (Hendrycks et al., 2021) and evaluated on MATH500 (Lightman et al., 2024), AIME25 (Zhang & Math-AI Team, 2025), and AIME26 (Zhang & Math-AI Team, 2026). For ProofWriter (Tafjord et al., 2021) and MMLU-Pro (Wang et al., 2024), models are trained on 2,000 examples and evaluated on 500 disjoint examples, randomly drawn from the same task-specific source data pools. For training, eligible questions are filtered to fewer than 500 tokens; 2,100 eligible examples are selected per training task, with 2,000 used for optimization and 100 reserved for training-time monitoring. The final MATH500, MMLU-Pro, and ProofWriter evaluation subsets are not filtered by this training-time length rule. Full training configurations are provided in Appendix A.

Baselines. We compare against Base, standard GRPO (Shao et al., 2024), OPD (Agarwal et al., 2024), and three representative efficient-reasoning methods: GR^3 (Li et al., 2026), LASER-DE (Liu et al., 2026), and CRISP (Sang et al., 2026). Unless otherwise specified, all trained methods use the same student backbone, task-specific data, rollout budget, and optimization setup, differing only in their method-specific objectives or training mechanisms. Implementation details, hyperparameters, and prompts are provided in Appendix C.

Evaluation. We report task accuracy and average thinking tokens, where thinking tokens denote the number of tokens generated within the reasoning trace. For each test problem, we sample k independent completions, with $k = 16$ for AIME25/26 and $k = 4$ for all other benchmarks, and average both metrics across the sampled completions (avg@k). Training-time monitoring uses one completion per held-out example (avg@1); it is not used for checkpoint selection, and all reported results use the final checkpoint. All methods are evaluated on the same examples with identical decoding settings. Sampling protocols, evaluation subsets, decoding parameters, and grading procedures are detailed in Appendix B.

Table 1: Accuracy and thinking cost for 4B and 9B students. Each benchmark reports accuracy (Score, %) and average thinking tokens (#Tokens). MATH500, MMLU-Pro, and ProofWriter use avg@4; AIME25/26 use avg@16. The teacher is included as a capability reference.

Method	MATH500		AIME25		AIME26		MMLU-Pro		ProofWriter	
	Score↑	#Token↓	Score↑	#Token↓	Score↑	#Token↓	Score↑	#Token↓	Score↑	#Token↓
Qwen3.5-4B										
Base	84.0	4,452	43.3	14,209	46.7	15,910	68.0	2,330	82.6	2,479
GRPO	91.2	4,291	46.7	13,846	50.0	15,950	70.6	2,144	91.6	2,463
OPD	88.4	3,764	51.7	13,908	54.6	15,702	71.2	2,232	89.2	2,297
LASER-DE	88.6	1,023	48.3	8,739	49.2	9,955	72.2	1,179	91.8	899
GR^3	90.6	2,924	49.2	13,230	53.3	13,955	75.2	1,826	93.2	1,243
CRISP	79.6	2,211	49.0	13,001	46.7	12,612	68.6	2,233	82.4	2,073
TAPER	**92.4**	**1,443**	**54.6**	**11,292**	**56.5**	**10,991**	**75.6**	**1,565**	**93.6**	**1,034**
Qwen3.5-9B										
Base	87.8	3,312	50.6	14,143	55.8	13,890	76.6	1,861	89.8	1,892
GRPO	92.8	3,275	55.4	13,670	66.7	14,327	78.6	1,846	94.0	1,827
OPD	92.2	2,994	53.3	13,968	63.3	15,156	79.4	1,811	93.0	1,763
LASER-DE	90.2	1,208	48.8	7,910	59.6	8,079	78.6	1,103	94.8	666
GR^3	93.2	2,401	57.1	11,709	66.7	13,167	80.0	1,485	95.8	983
CRISP	89.4	1,962	49.8	10,219	57.1	11,755	78.0	1,732	92.2	1,493
TAPER	**94.7**	**1,330**	**59.2**	**9,461**	**69.0**	**10,168**	**80.2**	**1,183**	**96.4**	**639**
Teacher	98.8	2,368	72.1	11,890	77.9	10,911	82.6	1,908	99.2	1,240

4.2 Main Results

Table 1 summarizes the results across two model scales and five benchmarks. Overall, TAPER consistently improves both task performance and reasoning efficiency over standard GRPO. Across all 10 model–benchmark settings, TAPER improves accuracy by **1.2–7.9 percentage points** while reducing average thinking tokens by **18.4%–66.4%**. These gains are observed across both model scales and across mathematical, logical, and knowledge-intensive reasoning tasks.

Accuracy–efficiency trade-off. On Qwen3.5-9B MATH500, TAPER achieves **94.7% accuracy with 1,330 average thinking tokens**. Compared with GRPO, it improves accuracy by 1.9 percentage points while reducing thinking tokens by 59.4%. Compared with GR^3, TAPER gains 1.5 points in accuracy with 44.6% fewer thinking tokens. LASER-DE achieves slightly stronger compression, but TAPER uses only 10.1% more thinking tokens while improving accuracy by 4.5 points. Thus, on this benchmark, TAPER improves both metrics over GRPO and GR^3, while trading a higher token cost for higher accuracy relative to LASER-DE.

Consistency across tasks and model scales. The same trend holds across the remaining benchmarks and on the smaller 4B model. On Qwen3.5-9B, TAPER improves over GRPO by 3.8 and 2.3 percentage points on AIME25 and AIME26 while reducing thinking tokens by 30.8% and 29.0%, respectively. It also improves accuracy by 1.6 points on MMLU-Pro and 2.4 points on ProofWriter, with token reductions of 35.9% and 65.0%. On Qwen3.5-4B, TAPER likewise improves accuracy while reducing thinking tokens across all five benchmarks, with gains of 1.2–7.9 points and token reductions of 18.4%–66.4%. Among student models, TAPER achieves the highest accuracy in all 10 evaluation settings.

4.3 Ablation Study

We ablate the key components of TAPER on Qwen3.5-9B. **Absolute Length** replaces the relative length objective with an absolute length reward using a 300-token threshold chosen based on experience; **w/o Length Reward** removes the length objective and its gate; **w/o Gate** removes competence conditioning; **w/o OPD** removes teacher distillation; and **Uniform Reverse KL** applies reverse KL

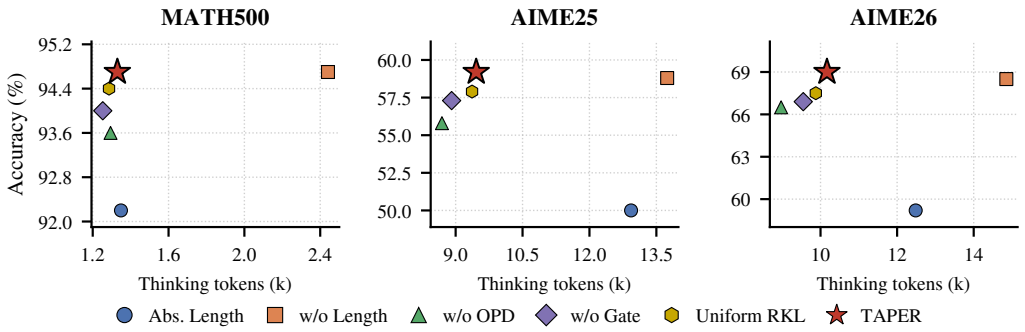

Figure 2: Accuracy–thinking-cost comparisons for the 9B ablation configurations. Each panel reports accuracy against average thinking tokens for MATH500 (avg@4), AIME25, and AIME26 (avg@16).

to both reasoning and answer regions. Figure 2 summarizes the main results, with full numerical results in Appendix F.

Relative length drives compression. Removing the length objective increases reasoning length substantially. Relative to w/o Length Reward, TAPER uses 45.5%, 31.2%, and 31.5% fewer thinking tokens on MATH500, AIME25, and AIME26, respectively; accuracy is unchanged on MATH500 and higher by 0.4 and 0.5 points on AIME25 and AIME26. The tested 300-token absolute-length configuration yields lower accuracy by 2.5, 9.2, and 9.8 points, respectively; relative to this configuration, TAPER uses 1.4%, 26.8%, and 18.6% fewer tokens. Overall, among the configurations tested, the relative-length objective achieves a better accuracy–length trade-off on all three benchmarks.

Competence conditioning moderates compression. The capability gate scales the relative-length reward according to the group solve rate, thereby controlling compression pressure. Compared with w/o Gate, Full TAPER uses 6.1–6.5% more thinking tokens across the three benchmarks, while improving accuracy by 0.7, 1.9, and 2.1 percentage points on MATH500, AIME25, and AIME26, respectively. This indicates that the gate moderates length reduction, resulting in somewhat longer reasoning under the full objective.

OPD improves observed accuracy under compression. Removing OPD reduces accuracy by 1.1, 3.4, and 2.5 points on MATH500, AIME25, and AIME26, respectively, while producing shorter reasoning traces. Full TAPER also exceeds Uniform Reverse KL by 0.3, 1.3, and 1.5 points on the three benchmarks, using 3.4%, 1.0%, and 2.9% more thinking tokens. Notably, the gains are larger on the two harder AIME benchmarks, consistent with our motivation: the harder the problem and the stronger the compression pressure, the more critical asymmetric distillation becomes for preserving essential reasoning behaviors. Section 5.2 examines the accompanying differences in reasoning behavior.

5 Analysis

We further analyze TAPER from three perspectives: how compression varies with problem difficulty, how reasoning behavior changes after compression, and how accuracy and reasoning length evolve during training.

5.1 Compression Across Difficulty Levels

Different problems require different reasoning budgets. We compare accuracy and average thinking length across MATH Easy (238 MATH500 Level 1–3 problems), MATH Hard (262 Level 4–5 problems), and AIME25/26 (60 problems) in Figure 3. TAPER shows different compression rates across these evaluation groups. Relative to Base, thinking length is reduced by 62.5% on MATH Easy, 58.0% on MATH Hard, and 30.0% on AIME25/26. TAPER achieves accuracies of 98.6%, 91.0%, and 64.1% in these groups, compared with 93.2%, 82.8%, and 53.2% for Base. The smaller

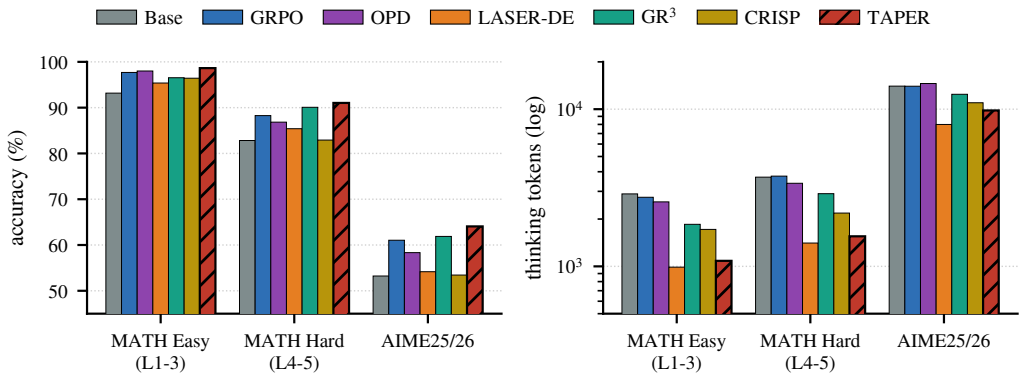

Figure 3: Accuracy and thinking length by evaluation group: MATH500 Level 1–3 (MATH Easy), MATH500 Level 4–5 (MATH Hard), and AIME25/26. The hatched bars denote TAPER.

relative length reduction on AIME25/26 is consistent with the intended competence-adaptive behavior. Since these groups come from different benchmarks and have different evaluation protocols, this comparison reflects both difficulty and benchmark differences and does not isolate the effect of the gate.

5.2 Reasoning Behavior Under Compression

Compression accompanies changes in reasoning behavior. We use an LLM annotator to classify reasoning segments into **Advance**, **Verify**, **Revise**, and **Redundancy** (Table 2; annotation prompt in Appendix I). Compared with Base, TAPER reduces Redundancy from 13.0% to 2.3% and Verify from 13.6% to 10.5%, while increasing Advance from 72.0% to 80.6% and Revise from 1.4% to 6.6%. GRPO shows a smaller reduction in redundancy, from 13.0% to 12.3%. Figure 4 uses the same final TAPER distribution in both panels. All trained checkpoints use evaluation traces: 360 were annotated at steps 5, 10, and 15, and 150 at later checkpoints (Table I3). We treat the fluctuating checkpoint trend as descriptive.

Behavior composition varies across evaluation groups. The difficulty-stratified analysis shows that TAPER has lower Verify and Redundancy shares than Base in all three groups, while Advance and Revise shares are higher. For TAPER, Verify accounts for 3.6% of tokens on MATH Easy, 10.9% on MATH Hard, and 10.8% on AIME25/26; Revise accounts for 5.4%, 3.2%, and 2.1%, respectively. Thus, verification is more prevalent in the two harder groups than in MATH Easy, while revision does not increase monotonically with group difficulty. Redundancy falls from 7.2% on MATH Easy to 6.8% on MATH Hard and 5.0% on AIME25/26. Taken together, these distributional shifts suggest that TAPER adapts the model's reasoning behavior to problems of different difficulty.

Stronger answer supervision reduces reasoning migration. As an additional finding, answer-region supervision helps mitigate reasoning migration beyond the `</think>` boundary, a potential shortcut of thinking-length optimization. In the 9B MATH500 weight sweep, we fix $w_{\text{think}} = 0.1$ and strengthen answer supervision by increasing w_{ans}. The observed migration rate decreases from 0.85% at $w_{\text{ans}} = 0.1$ to 0.60% at the default $w_{\text{ans}} = 1.0$, compared with 1.50% without OPD. These results support stronger answer-region supervision as a practical way to reduce this behavior in the evaluated setting; the full weight sweep is reported in Appendix E.

5.3 Training Dynamics of Accuracy and Thinking Length

Training gradually shifts from improving accuracy to compressing reasoning. We track the supplied rollout accuracy reward and average thinking length across 112 training steps, as shown in Figure 1 (right panel), with the complete per-step values provided in Appendix G. Accuracy reward rises from 77.78% at step 1 to the high-90s by roughly step 30 and reaches 100% at several later steps, although it remains noisy with occasional dips. Thinking length is also highly variable early

Table 2: Reasoning-behavior categories used in our analysis. We use GPT-5.6 Luna to annotate reasoning segments according to these definitions; the annotation prompt is provided in Appendix I.

Category	Definition
Advance	Adds new content to the solving state: deriving a relation, computing a result, introducing a case split, or setting a subgoal.
Verify	Confirms or refutes an existing conclusion by independent means—substituting values back, boundary checks, or recomputing via a different route—without changing the solving state.
Revise	Modifies the solving state itself: correcting an error, updating a value, or relaxing an assumption.
Redundancy	Restates previously established content without adding information and without performing a check.

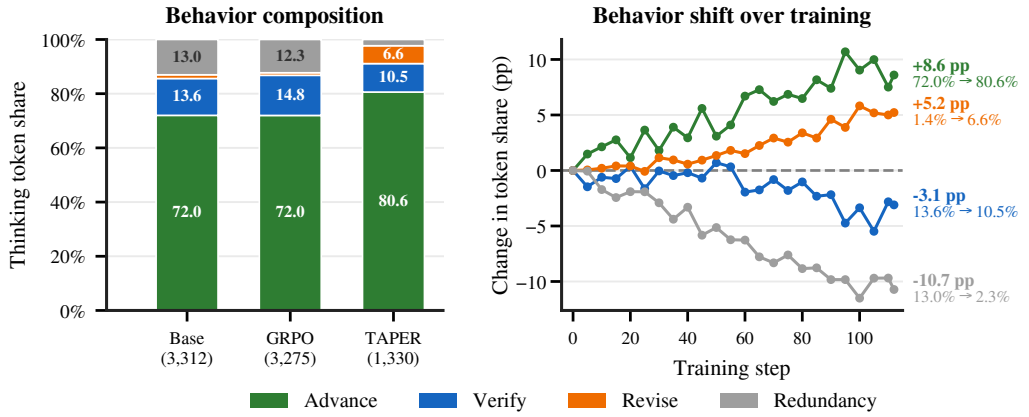

Figure 4: Reasoning-behavior token shares for Base, GRPO, and TAPER (left), and their changes during training (right). Parenthesized values are average thinking tokens.

in training, including long-rollout spikes up to 4,096.2 tokens, but falls from 902.9 tokens at step 1 to 46.4 tokens at step 112; the median over the final ten steps is 98.7 tokens. These measurements support an overall shift toward shorter rollouts as solving performance improves, while also showing that neither accuracy nor length changes monotonically at every step. The resulting dynamics are consistent with TAPER progressively moving from learning to solve effectively toward learning to solve more efficiently.

6 Conclusion

We present TAPER, which jointly optimizes problem-solving accuracy and thinking cost through a competence-conditioned relative length reward and region-aware asymmetric on-policy distillation. Across two model scales and five benchmarks, TAPER improves accuracy over standard GRPO while reducing thinking tokens. Further analyses show different compression rates across difficulty strata and changes in reasoning-behavior composition: relative to Base, TAPER has lower token shares for verification and redundancy and higher shares for advancement and revision.

7 Limitations and Future Work

Our characterization of behavioral changes is empirical rather than derived from principled analysis, and the mechanisms behind the observed behavioral shifts remain to be understood; our evaluation is also limited to relatively short-horizon mathematical, knowledge, and logical tasks. We plan to address both: developing a theoretical account of which reasoning behaviors can be safely removed under which conditions, and extending TAPER to longer-horizon settings such as coding, tool use, and agentic tasks to test its applicability at the frontier.

AI USE STATEMENT

Generative AI tools were used to support research ideation through discussions of the TAPER method, experimental design, and interpretation of results; to retrieve and discover related work and bibliographic information; and to draft, translate, edit, and polish manuscript text, including sections and appendices. AI assistance also contributed to writing and organizing code and to preparing and compiling the LaTeX source files. Experimental measurements were produced by the authors' runs; AI assistance was used for review and discussion, not as a substitute for the authors' verification. The authors reviewed the AI-assisted text and code and are responsible for the accuracy of the reported results, references, code, analyses, and final manuscript.

REPRODUCIBILITY STATEMENT

Reproducibility details, including training settings, data construction, baseline definitions, evaluation protocol, reward construction, and distillation implementation, are provided in the supplementary material (Appendices A–J) and the accompanying code.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations (ICLR 2024)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html. arXiv:2306.13649.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMs. *arXiv preprint arXiv:2412.21187*, 2024. URL https://arxiv.org/abs/2412.21187.

Leichao Dong, Dongxu Zhang, Yiding Sun, Qirui Wang, Yuhan Wang, Lin Chen, and Jihua Zhu. Better Starts, Better Ends: Bootstrapped Iterative Self-Reasoning Distillation for Compressed Reasoning. *arXiv preprint arXiv:2607.15736*, 2026. URL https://arxiv.org/abs/2607.15736.

Ruixiang Feng, Yuntao Wen, Silin Zhou, Ke Shi, Yifan Wang, Ran Le, Zhenwei An, Zongchao Chen, Chen Yang, Guangyue Peng, Yiming Jia, Dongsheng Wang, Tao Zhang, Lisi Chen, Yang Song, Shen Gao, and Shuo Shang. PACE: Prefix-Protected and Difficulty-Aware Compression for Efficient Reasoning. *arXiv preprint arXiv:2602.11639*, 2026. URL https://arxiv.org/abs/2602.11639.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015. URL https://arxiv.org/abs/1503.02531. NIPS 2014 Deep Learning Workshop; arXiv:1503.02531.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507/. arXiv:2305.02301.

Yoon Kim and Alexander M. Rush. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327. Association for Computational Linguistics, 2016. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/. arXiv:1606.07947.

Zichao Li, Jie Lou, Fangchen Dong, Zhiyuan Fan, Mengjie Ren, Hongyu Lin, Xianpei Han, Debing Zhang, Le Sun, Yaojie Lu, and Xing Yu. Tackling Length Inflation Without Trade-offs: Group Relative Reward Rescaling for Reinforcement Learning. *arXiv preprint arXiv:2603.10535*, 2026. URL https://arxiv.org/abs/2603.10535.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to Reason Efficiently with Adaptive Length-based Reward Shaping. In *The Fourteenth International Conference on Learning Representations (ICLR 2026)*, 2026. URL https://openreview.net/forum?id=hj9eKpqxQl. arXiv:2505.15612.

Rafael Pardinas, Ehsan Kamalloo, David Vazquez, and Alexandre Drouin. Apriel-Reasoner: RL Post-Training for General-Purpose and Efficient Reasoning. *arXiv preprint arXiv:2604.02007*, 2026. URL https://arxiv.org/abs/2604.02007.

Xiaoye Qu, Yafu Li, Zhao-Chen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-Sheng Hua, Bowen Zhou, and Yu Cheng. A Survey of Efficient Reasoning for Large Reasoning Models: Language, Multimodality, and Beyond. *arXiv preprint arXiv:2503.21614*, 2025. URL https://arxiv.org/abs/2503.21614.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. CRISP: Compressed Reasoning via Iterative Self-Policy Distillation. *arXiv preprint arXiv:2603.05433*, 2026. URL https://arxiv.org/abs/2603.05433.

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. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n. arXiv:2408.03314.

Jinyan Su and Claire Cardie. Thinking Fast and Right: Balancing Accuracy and Reasoning Length with Adaptive Rewards. *arXiv preprint arXiv:2505.18298*, 2025. URL https://arxiv.org/abs/2505.18298.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

Rui Wang, Hongru Wang, Boyang Xue, Yixia Li, Jianhui Pang, Shudong Liu, Yi Chen, Jiahao Qiu, Derek Fai Wong, Guanhua Chen, Heng Ji, and Kam-Fai Wong. Harnessing the Reasoning Economy: A Survey of Efficient Reasoning for Large Language Models. *arXiv preprint arXiv:2503.24377*, 2025. URL https://arxiv.org/abs/2503.24377.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, pp. 95266–95290, 2024. doi:

10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract.html. arXiv:2406.01574.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html. arXiv:2201.11903.

Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, Zhe Zhao, and Ngai Wong. Rethinking Kullback-Leibler divergence in knowledge distillation for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 5737–5755, Abu Dhabi, UAE, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.383/.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. Just Enough Thinking: Efficient Reasoning with Adaptive Length Penalties Reinforcement Learning. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. URL https://openreview.net/forum?id=oHR5d6tuXs. arXiv:2506.05256.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in LLMs. *arXiv preprint arXiv:2502.03373*, 2025. URL https://arxiv.org/abs/2502.03373.

Danlong Yuan, Tian Xie, Shaohan Huang, Huishuai Zhang, Zhuocheng Gong, Chong Luo, Furu Wei, and Dongyan Zhao. Shorten After You're Right: Lazy Length Penalties for Reasoning RL. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12864–12877, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.626. URL https://aclanthology.org/2026.findings-acl.626/. arXiv:2505.12284.

Xiaoyun Zhang, Jingqing Ruan, Xing Ma, Yawen Zhu, Haodong Zhao, Hao Li, Jiansong Chen, Ke Zeng, and Xunliang Cai. When to Continue Thinking: Adaptive Thinking Mode Switching for Efficient Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5808–5828, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.310. URL https://aclanthology.org/2025.findings-emnlp.310/. arXiv:2505.15400.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2026, 2026. URL https://huggingface.co/datasets/math-ai/aime26.

Shangziqi Zhao, Jiahao Yuan, Jinyang Wu, Zhenglin Wang, Guisong Yang, and Usman Naseem. Can pruning improve reasoning? revisiting long-CoT compression with capability in mind for better reasoning. *arXiv preprint arXiv:2505.14582*, 2025. URL https://arxiv.org/abs/2505.14582.

APPENDIX

A TRAINING AND IMPLEMENTATION DETAILS

The shared training configuration and method-specific settings are summarized together in Appendix C, Tables C1–C5. The decoding configuration used for training rollouts and evaluation is given in Table A1.

Data construction. For each task, training questions are filtered to fewer than 500 tokens, with 2,000 examples used for optimization and 100 held out for monitoring. MATH examples come from its training set. For ProofWriter and MMLU-Pro, the 2,000 optimization examples and 500 evaluation examples are randomly drawn from each task's source data pool used in this study, forming disjoint, study-specific splits. Monitoring examples are separate from both subsets. All methods use the same fixed subsets. Final evaluation uses MATH500, AIME25/26 (30 problems each), and the 500-example MMLU-Pro and ProofWriter subsets without the training-time length filter. Monitoring uses avg@1 and does not determine checkpoint selection; reported results use the final checkpoint and the avg@4/avg@16 protocol in Appendix B.

Rollout decoding. During training, the student samples $G = 8$ responses per problem using temperature 1.0, top-p 0.95, top-k 20, and repetition penalty 1.0. The maximum completion length is 6,144 tokens and the maximum training context length is 10,240 tokens. Evaluation uses a maximum completion length of 20,480 tokens and a maximum context length of 24,576 tokens. The decoding configurations are summarized in Table A1.

Table A1: Decoding configuration for training rollouts and evaluation.

Parameter	Training	Evaluation
Temperature	1.0	1.0
Top-p	0.95	0.95
Top-k	20	20
Repetition penalty	1.0	1.0
Presence penalty	1.5	1.5
Maximum completion length	6,144	20,480
Maximum context length	10,240	24,576

Correctness and format rewards. The correctness reward is binary:

$$r_i^{\text{acc}} \in \{0, 1\}, \tag{6}$$

where a response receives 1 if its extracted final answer is judged correct by the task-specific evaluator and 0 otherwise. The extraction and scoring rules are given in Appendix B.

The format reward is also binary:

$$r_i^{\text{fmt}} \in \{0, 1\}. \tag{7}$$

A response receives $r_i^{\text{fmt}} = 1$ if it contains a complete pair of `<think>` and `</think>` tags enclosing the reasoning trace and places the final result in `\boxed{...}` in the answer region after `</think>`. This format enables extraction of the final result for evaluation. If either requirement is not met, the format reward is 0. The format reward is a training signal; at evaluation time, a response without any thinking marker is still scorable, with thinking length zero, provided that it contains a valid boxed answer. An unclosed thinking block receives zero accuracy credit.

Truncated rollouts. Training rollouts that reach the maximum completion length are retained rather than discarded. They participate in the group length statistics and receive

$$r_i^{\text{acc}} = 0. \tag{8}$$

Consequently, truncated failures reduce the empirical solve rate p_x of their corresponding rollout group and therefore attenuate the compression pressure applied by the length term.

B Evaluation Protocol

- **Unified decoding.** The same system prompt as training; presence penalty 1.5, temperature 1.0, top-p 0.95, and top-k 20. Repeated samples use different random seeds, with base seed 42.
- **Thinking and generation limits.** Training rollouts are capped at 6,144 completion tokens with a 10,240-token context limit. For final benchmark evaluation, the thinking budget is 8,192 tokens for MATH500 and 16,384 tokens for AIME25/26; the maximum completion length is 20,480 tokens and the maximum model context length is 24,576 tokens.
- **Scoring and answer extraction.** The evaluator first identifies the final answer region. If a response contains one or more closing `</think>` markers, the text after the last marker is used; any answer appearing only inside the thinking region is ignored. If no thinking marker is present, the entire response is treated as the answer region and the thinking length is set to zero. An opening `<think>` marker without a matching `</think>` marker receives zero credit. In all cases, the final answer is extracted from the last `\boxed{...}` expression.
- **Benchmark evaluators.** For MATH500 and AIME25/26, `MathAccuracy` parses the boxed prediction and reference with `math_verify` and assigns credit when the mathematical expressions are equivalent. For MMLU-Pro, `BoxedChoiceAccuracy` uppercases the last boxed content and accepts only a single option in A–J that exactly matches the reference. For ProofWriter, `BoxedProofWriterAccuracy` removes non-alphabetic characters, normalizes case, maps the result to `True`, `False`, or `Unknown`, and compares it with the reference label.
- **Accuracy aggregation.** Accuracy for one completion is binary. For the final evaluation, we average over k independent completions per problem (avg@k), with $k = 4$ for MATH500, MMLU-Pro, and ProofWriter and $k = 16$ for AIME25/26. Training-time monitoring uses avg@1 on the 100 held-out examples and is not used for checkpoint selection. This differs from pass@k, which would count a problem as solved if any of its k completions is correct.
- **Evaluation sizes.** MATH500: 500; AIME25: 30; AIME26: 30; MMLU-Pro and ProofWriter: 500 randomly sampled examples per task from the task-specific source data pool, held out from the 2,000 optimization examples and the training-time monitoring examples. The same examples are used for all methods. For the stratified analysis, MATH500 is split into 238 Level 1–3 (MATH Easy) and 262 Level 4–5 (MATH Hard) problems; AIME25 and AIME26 are combined as a separate 60-problem group.
- **Samples per problem.** MATH500, MMLU-Pro, and ProofWriter: four independent samples (avg@4). AIME25/26: 16 (avg@16). Report mean accuracy and mean thinking tokens, not pass@k. The training-dynamics trace in Appendix G is a separate rollout diagnostic and is reported using the supplied per-step accuracy-reward percentages.
- **Metrics.** Accuracy (%) and average thinking tokens inside the thinking region; some tables additionally report answer tokens.

C BASELINES AND FAIR-COMPARISON ALIGNMENT

Tables C1–C5 report the shared training setup and the method-specific configurations used in our experiments. The method-specific entries describe our implementations; equations and terminology follow the cited papers where applicable.

Table C1: Shared training configuration.

Parameter	Setting
Student models	Qwen3.5-4B and Qwen3.5-9B
Training precision	bfloat16
Parameter-efficient tuning	LoRA; rank 64, alpha 32; all linear layers
Rollouts per prompt	$G = 8$
Global prompt batch size	18 prompt groups
Trajectories per rollout micro-batch	$18 \times 8 = 144$
Gradient accumulation	4 micro-batches
Trajectories per optimizer update	$144 \times 4 = 576$
Training duration	112 optimizer updates
Optimizer	Fused AdamW; weight decay 0.1
Adam parameters	$(\beta_1, \beta_2, \epsilon) = (0.9,\ 0.95,\ 10^{-8})$
Gradient clipping	1.0
Learning rate	1×10^{-5}
Learning-rate schedule	Constant with 6-step linear warmup
Random seed	42

Table C2: TAPER-specific configuration.

Parameter	Setting
Accuracy reward	Binary correctness reward; $\lambda_{\text{acc}} = 0.5$
Format reward	Requires a closed `<think>...</think>` block and a final answer in `\boxed{...}`; $\lambda_{\text{fmt}} = 0.1$
Relative-length reward	Piecewise within-group reward in $[-1, 1]$, scaled by group solve rate p_x; $\lambda_{\text{len}} = 0.5$
Teacher	Qwen3.5-122B-A10B
OPD coefficient	$\eta_{\text{OPD}} = 0.3$
Thinking-region distillation	Forward KL; $w_{\text{think}} = 0.1$
Answer-region distillation	Reverse KL; $w_{\text{ans}} = 1.0$
Teacher support	Top-20 teacher tokens

Table C3: GR^3-specific configuration.

Parameter	Setting
Reward rescaling	$\hat{R}_i = R_i / (1 + \alpha \ell_i / \bar{\ell})$, following GR^3 (Li et al., 2026)
Length definition	Full response length, following the original method
Rescaling coefficient	$\alpha = 0.33$
Advantage-aware calibration	Use the average-case criterion: a representative maximum-reward response at the group-mean length should retain nonnegative advantage; coefficient used here is 0.33
All-maximum-reward groups	Filtered online

Table C4: LASER-DE-specific configuration.

Parameter	Setting
Length-reward rule	$\hat{R} = R + \alpha\mathbb{I}[R=1]\mathbb{I}[L \le L_A] + \alpha\mathbb{I}[R=0]\mathbb{I}[L > L_A]$, following LASER-DE (Liu et al., 2026)
Length definition	Full response length
Length-reward coefficient	$\alpha = 0.5$
Initial target length	$L_T = 1{,}024$ tokens
Difficulty bins	Based on the correct count among $G = 8$ rollouts: hard 0–2, medium 3–5, and easy 6–8 (cutoffs at $G/3$ and $2G/3$)
Monitoring and update interval	500 monitoring examples; update every 20 training steps
Target-length search	1,024–8,192 tokens in 1,024-token increments; choose the smallest target with expected correct responses (ECR) ≥ 1
Incorrect over-length rollouts	Receive the exploration bonus $+0.5$

Table C5: CRISP-specific configuration and prompt used in our implementation.

Parameter	Setting
Self-teacher	Same model conditioned on the concise-reasoning prompt; refreshed from the student every 20 training steps
Distillation loss	Per-token reverse KL, $D_{\mathrm{KL}}(\pi_{\mathrm{student}} \| \pi_{\mathrm{teacher}})$
KL coefficient / support	Unit coefficient; full vocabulary
Teacher refresh interval	$M = 20$ training steps (the original paper reports $M = 50$ as its default (Sang et al., 2026))
Incorrect rollouts	Included in distillation without correctness filtering
Prompt	You are a precise math assistant. Your top priority is correctness.
	Think in compressed form: inside `<think>...</think>`, write only the key setup, equations, and transformations — one short line per step, no restating the problem, no prose explanations, no filler words.
	Before finishing, silently verify the result (plug it back in, or check sign and magnitude); if the check fails, redo the reasoning.
	After `</think>`, answer in exactly one short sentence containing `\boxed{...}`. Put only the final result inside the box — no units, no extra words.

- **Shared settings.** All methods use the same student base, task-specific training data, training steps, and evaluation protocol (Appendix B); the MATH training set is used for the mathematical task, while ProofWriter and MMLU-Pro use their respective 2,000-example training sets.
- **Base.** The original SFT student, evaluated directly with the same inference configuration and without RL from this training stage.
- **GRPO.** Standard group-relative policy optimization using only accuracy and format rewards, with no length reward or OPD.
- **GR^3 / LASER-DE.** Represent group-relative length regularization and dynamic length rewards, respectively; implementations follow the original papers and are aligned to the student base and data used here.
- **OPD.** On-policy distillation isolates teacher-guided distillation without the proposed efficiency objective.
- **CRISP.** Included as an efficient-reasoning reference in the scatter comparisons.

D Reward Construction and GRPO Optimization

This appendix provides additional details on the relative-length reward, competence-conditioned gating, and their interaction with group-relative reward normalization.

D.1 Relative Length Reward

For G rollouts sampled for the same problem, let ℓ_i denote the thinking length of trajectory i, and let

$$\ell_{\min} = \min_i \ell_i, \qquad \ell_{\max} = \max_i \ell_i, \qquad \bar{\ell} = \frac{1}{G}\sum_i \ell_i. \tag{9}$$

The relative length reward is

$$r_i^{\text{len}} = \begin{cases} \dfrac{\bar{\ell} - \ell_i}{\bar{\ell} - \ell_{\min}}, & \ell_i \le \bar{\ell}, \\ \dfrac{\bar{\ell} - \ell_i}{\ell_{\max} - \bar{\ell}}, & \ell_i > \bar{\ell}. \end{cases} \tag{10}$$

It maps the shortest trajectory to $+1$, the group mean to 0, and the longest trajectory to -1, without introducing a global target length.

When either normalization denominator is zero, the corresponding branch is set to zero. This includes degenerate groups in which all trajectories have identical thinking lengths.

D.2 Competence-Conditioned Gating

The empirical solve rate of problem x is

$$p_x = \frac{1}{G}\sum_{i=1}^{G} r_i^{\text{acc}}. \tag{11}$$

The effective relative-length reward becomes

$$\tilde{r}_i^{\text{len}} = p_x r_i^{\text{len}}, \tag{12}$$

which is equivalent to using an effective length coefficient

$$\lambda_{\text{eff}} = \lambda_{\text{len}} p_x. \tag{13}$$

The resulting trajectory reward is

$$R_i = \lambda_{\text{acc}} r_i^{\text{acc}} + \lambda_{\text{fmt}} r_i^{\text{fmt}} + \lambda_{\text{len}} p_x r_i^{\text{len}}. \tag{14}$$

The default reward coefficients are summarized in Table D1.

Table D1: Reward components.

Component	Symbol	Weight	Role
Correctness	r_i^{acc}	0.5	Rewards correct final answers
Format	r_i^{fmt}	0.1	Enforces the required response template
Relative length	r_i^{len}	0.5	Prefers shorter trajectories relative to the current rollout group
Competence gate	p_x	—	Modulates the relative-length term according to current solve rate

All trajectories participate in the relative-length statistics. This provides a dense efficiency signal without requiring length statistics to be estimated only from successful trajectories. When the current policy has a low solve rate, p_x automatically reduces the contribution of this efficiency signal.

D.3 Interaction Between Gating and Group Normalization

GRPO centers and normalizes the total reward within each rollout group to construct group-relative advantages:

$$A_i = \frac{R_i - \mu_R}{\sigma_R + \epsilon}, \tag{15}$$

where μ_R and σ_R are the group mean and standard deviation of R_i.

A common positive scaling applied to the entire reward would largely cancel under this normalization. However, the competence gate does not scale the entire reward. Instead, it selectively rescales the relative-length component before it is combined with correctness and format rewards. It therefore changes the relative contribution of efficiency variation compared with the other reward components.

The behavior of the gate can be summarized in Table D2.

Table D2: Effect of group solve rate.

Group state	p_x	Correctness variation	Length signal	Main effect
No rollout is correct	0	None	Disabled	No compression pressure from the length term
Partially solved group	$0 < p_x < 1$	Present	Partially active	Joint correctness–efficiency optimization
All rollouts are correct	1	None	Fully active	Optimization primarily distinguishes trajectories by efficiency and format

When all responses are incorrect, $p_x = 0$, so the length component contributes no advantage and correctness is constant within the group. Any remaining reward variation can therefore arise only from the format term.

Reference-policy KL regularization is retained during GRPO optimization with coefficient

$$\beta = 0.02. \tag{16}$$

E REGION-AWARE OPD IMPLEMENTATION

This appendix describes how the thinking and answer regions are constructed and how the sparse forward and reverse KL objectives are implemented.

E.1 REGION SEGMENTATION

Each student-generated response is divided using the explicit `</think>` marker.

Table E1: Region construction.

Response condition	Thinking region T_i^{think}	Answer region T_i^{ans}
Valid `</think>` marker	Tokens before the last boundary	Tokens after the last boundary
No thinking marker	Empty (length 0)	Entire generated sequence
Unclosed `<think>` marker	Generated tokens after the opening marker	Empty; scoring is zero
Empty thinking region	Empty	Remaining answer tokens

If either region is empty for a trajectory, its corresponding OPD loss term is set to zero for that trajectory.

E.2 REGION-AWARE ASYMMETRIC KL

For trajectory i at token position t, the teacher and student distributions conditioned on the student-generated prefix are

$$q_{i,t} = \pi_T(\cdot \mid x, y_{i,<t}), \qquad p_{i,t} = \pi_\theta(\cdot \mid x, y_{i,<t}). \tag{17}$$

We use different KL directions for the two generation regions, as summarized in Table E2. Here $\tilde{q}_{i,t}$ and $\tilde{p}_{i,t}$ denote distributions conditionally renormalized over the teacher top-20 support, defined below; these are the distributions used in the implemented loss.

Table E2: Region-aware distillation objectives.

Region	KL direction	Token-level loss	Weight	Intended role
Thinking	Teacher → Student	$D_{\text{KL}}(\tilde{q}_{i,t} \parallel \tilde{p}_{i,t})$	0.1	Encourage coverage of multiple teacher-supported continuations within the retained support
Answer	Student → Teacher	$D_{\text{KL}}(\tilde{p}_{i,t} \parallel \tilde{q}_{i,t})$	1.0	Discourage relative over-allocation to teacher-disfavored tokens within the retained support

The complete OPD objective is

$$\mathcal{L}_{\text{OPD}} = \frac{1}{B}\sum_{i=1}^{B}\left[\frac{w_{\text{think}}}{|T_i^{\text{think}}|}\sum_{t\in T_i^{\text{think}}} D_{\text{KL}}(\tilde{q}_{i,t} \parallel \tilde{p}_{i,t}) + \frac{w_{\text{ans}}}{|T_i^{\text{ans}}|}\sum_{t\in T_i^{\text{ans}}} D_{\text{KL}}(\tilde{p}_{i,t} \parallel \tilde{q}_{i,t})\right], \tag{18}$$

with

$$w_{\text{think}} = 0.1, \qquad w_{\text{ans}} = 1.0. \tag{19}$$

For an empty region, the corresponding normalized term is defined as zero. Region-wise normalization is performed separately for each trajectory before averaging over the B trajectories in the optimization batch. This prevents longer trajectories from receiving larger OPD weights merely because they contain more tokens.

E.3 SPARSE TOP-k KL APPROXIMATION

Transferring complete teacher logits over the full vocabulary is computationally expensive. We therefore retain only the teacher's top-20 tokens at each position:

$$\mathcal{S}_{i,t} = \text{Top20}(q_{i,t}). \tag{20}$$

Teacher and student probabilities are then conditionally renormalized over this support:

$$\tilde{q}_{i,t}(v) = \frac{q_{i,t}(v)}{\sum_{u \in \mathcal{S}_{i,t}} q_{i,t}(u)}, \qquad \tilde{p}_{i,t}(v) = \frac{p_{i,t}(v)}{\sum_{u \in \mathcal{S}_{i,t}} p_{i,t}(u)}, \qquad v \in \mathcal{S}_{i,t}. \tag{21}$$

Equivalently, let $z_{T,i,t}$ and $z_{\theta,i,t}$ denote the teacher and student logits before softmax. The support is selected from the teacher logits,

$$\mathcal{S}_{i,t} = \mathrm{Top20}(z_{T,i,t}), \tag{22}$$

and the sparse distributions are obtained by gathering the logits at those token IDs and applying softmax over the selected entries:

$$\tilde{q}_{i,t} = \mathrm{softmax}(z_{T,i,t}[\mathcal{S}_{i,t}]), \qquad \tilde{p}_{i,t} = \mathrm{softmax}(z_{\theta,i,t}[\mathcal{S}_{i,t}]). \tag{23}$$

Thus, the top-20 operation selects the teacher support, while the student values on that support come from the current student forward pass. The selected student logits remain in the autograd graph; only the teacher support indices are treated as fixed, since the teacher is frozen. The gradient path is

$$z_{\theta,i,t} \longrightarrow z_{\theta,i,t}[\mathcal{S}_{i,t}] \longrightarrow \tilde{p}_{i,t} \longrightarrow D_{\mathrm{KL}}. \tag{24}$$

The sparse KL terms are therefore

$$D^{i,t}_{\mathrm{FKL}} = D_{\mathrm{KL}}(\tilde{q}_{i,t} \,\|\, \tilde{p}_{i,t}) \tag{25}$$

for the thinking region and

$$D^{i,t}_{\mathrm{RKL}} = D_{\mathrm{KL}}(\tilde{p}_{i,t} \,\|\, \tilde{q}_{i,t}) \tag{26}$$

for the answer region.

For a selected token $v \in \mathcal{S}_{i,t}$, the gradients with respect to its student logit are

$$\frac{\partial D^{i,t}_{\mathrm{FKL}}}{\partial z_{\theta,i,t,v}} = \tilde{p}_{i,t}(v) - \tilde{q}_{i,t}(v), \tag{27}$$

and

$$\frac{\partial D^{i,t}_{\mathrm{RKL}}}{\partial z_{\theta,i,t,v}} = \tilde{p}_{i,t}(v) \left[\log \frac{\tilde{p}_{i,t}(v)}{\tilde{q}_{i,t}(v)} - D^{i,t}_{\mathrm{RKL}} \right]. \tag{28}$$

For logits outside $\mathcal{S}_{i,t}$, the direct gradient from this sparse KL term is zero. The selected logits depend on the student's hidden states and shared parameters, so their gradients still update the student model. Probability mass outside the teacher top-20 support is omitted from the KL approximation; KL computation is performed in fp32 for numerical stability.

E.4 Coupling with Policy Optimization

Both GRPO and OPD operate on the same freshly sampled student trajectories. The teacher does not generate independent trajectories and is queried only on student-visited prefix states.

The overall objective is written as a loss to be minimized, consistent with Section 3.3:

$$\mathcal{L}_{\mathrm{TAPER}} = \mathcal{L}_{\mathrm{GRPO}} + \eta_{\mathrm{OPD}} \mathcal{L}_{\mathrm{OPD}}, \qquad \eta_{\mathrm{OPD}} = 0.3. \tag{29}$$

Each gradient-accumulation micro-batch contains 18 prompt groups with 8 trajectories per prompt, for 144 trajectories per micro-batch. Gradients are accumulated across four such micro-batches before one optimizer update, so each of the 112 optimizer updates uses 576 trajectories in total. No repeated inner optimization epochs are performed over the same trajectories.

The teacher is used only during training and therefore introduces no additional inference-time cost.

E.5 Answer-Region Weight Sweep

To quantify reasoning migration across the `</think>` boundary, we vary the answer-region OPD weight while fixing $w_{\mathrm{think}} = 0.1$. All responses in this sweep are generated by the 9B student on MATH500, using four samples per problem and therefore 2,000 responses per setting. A separate large-language-model judge examines the answer region after the final

</think> marker and labels whether it contains an unfinished reasoning process, such as a continuing derivation, correction, or path exploration. The migration rate is the number of judged-positive responses divided by 2,000. The complete sweep is reported in Table E3 and in supplementary/answer_region_migration.csv.

Table E3: Answer-region supervision and reasoning migration.

Setting	w_{think}	w_{ans}	Migrating samples / total	Migration rate (%)
w/o OPD	0.1	—	30 / 2,000	1.50
OPD	0.1	0.1	17 / 2,000	0.85
OPD	0.1	0.4	16 / 2,000	0.80
OPD	0.1	0.7	14 / 2,000	0.70
OPD (Full)	0.1	1.0	12 / 2,000	0.60

F Ablation Configurations and Accuracy–Cost Results

The ablation study uses a 9B student on MATH500, AIME25, and AIME26. The main text visualizes the accuracy–thinking-cost trade-off; the corresponding numerical results are provided below. The configurations are Absolute Length, w/o Length Reward, w/o OPD, w/o Gate, and Full TAPER.

The Absolute Length variant replaces within-group relative statistics with a fixed 300-token threshold, chosen based on experience because a suitable absolute reasoning length is difficult to specify across problems with different reasoning demands. This is a heuristic choice, not a threshold established by a systematic search; the comparison is limited to this setting. Its binary length reward is

$$r_i^{\mathrm{abs}} = \begin{cases} 1, & \ell_i \le 300, \\ -1, & \ell_i > 300, \end{cases} \tag{30}$$

so responses at or below 300 thinking tokens receive the positive length signal, while longer responses receive the negative signal. The other ablations remove or modify only the component named in their labels; all share the training and evaluation settings in Appendices A–B. Evaluation follows the main protocol: avg@4 for MATH500 and avg@16 for AIME25/26.

Table F1: Component ablations with the 9B student. Each cell reports **Acc. (%) / Think.** (average thinking tokens). MATH500 uses avg@4; AIME25/26 use avg@16.

Method	**MATH500**	**AIME25**	**AIME26**
Absolute Length Reward	92.2 / 1,349	50.0 / 12,933	59.2 / 12,485
w/o Length Reward	94.7 / 2,441	58.8 / 13,742	68.5 / 14,852
w/o OPD	93.6 / 1,294	55.8 / 8,693	66.5 / 8,971
w/o Gate	94.0 / 1,253	57.3 / 8,910	66.9 / 9,549
Uniform Reverse KL	94.4 / 1,286	57.9 / 9,368	67.5 / 9,877
Full TAPER	**94.7 / 1,330**	**59.2 / 9,461**	**69.0 / 10,168**

G Training Dynamics and Per-Step Evaluation

The right panel of Figure 1 shows the supplied rollout accuracy-reward and thinking-token dynamics for 112 optimizer steps. Each optimizer update accumulates four micro-batches of 144 trajectories each, for 576 trajectories per update. The complete per-step accuracy reward, thinking-token, and OPD-loss values are listed in Table G1 and included as `supplementary/training_dynamics.csv`. These diagnostics are distinct from the final benchmark evaluation and from the held-out training-time evaluation described in Appendix A; they report the rollout statistics supplied with the training run.

The trace reaches the high-90s in accuracy reward by roughly step 30, while thinking length decreases substantially overall but remains noisy, with occasional long-rollout spikes. The table is provided to make the plotted trend auditable; it should not be interpreted as a monotonic learning curve or as evidence that reward magnitudes directly equal gradient contributions.

Table G1: Per-step rollout diagnostics for the TAPER training run. Accuracy reward is a percentage, thinking tokens are averaged over the rollout batch, and OPD loss is included as logged during optimization.

Step	Accuracy reward (%)	Thinking tokens	OPD loss
1	77.78	902.9	0.003916
2	79.17	2,079.3	0.003964
3	81.94	1,625.9	0.002300
4	77.78	2,067.2	0.002530
5	79.17	1,055.2	0.002165
6	68.06	1,081.3	0.004874
7	77.78	721.8	0.002673
8	86.11	3,037.8	0.004144
9	86.11	213.5	0.007185
10	80.56	1,283.8	0.002558
11	86.11	758.8	0.003316
12	90.28	1,326.3	0.005540
13	80.56	881.4	0.002883
14	86.11	734.3	0.007354
15	68.06	2,860.1	0.006304
16	93.06	2,227.2	0.006727
17	91.67	1,745.1	0.002807
18	90.28	2,140.8	0.002673
19	87.50	1,535.9	0.003242
20	88.89	2,107.8	0.001982
21	98.61	856.1	0.004084
22	87.50	909.4	0.002597
23	88.89	1,445.2	0.002638
24	98.61	554.7	0.003377
25	98.61	687.8	0.003471
26	95.83	583.7	0.004197
27	97.22	2,786.6	0.002974
28	90.28	1,161.4	0.003695
29	97.22	332.2	0.004341
30	93.06	2,579.9	0.004042
31	86.11	1,470.5	0.003099
32	97.22	659.1	0.003462
33	97.22	389.3	0.003052
34	97.22	144.5	0.003936
35	100.00	237.8	0.004228
36	97.22	1,238.3	0.003460
37	97.22	472.2	0.003731

Continued on the next page

Table G1: Per-step rollout diagnostics (continued).

Step	Accuracy reward (%)	Thinking tokens	OPD loss
38	97.22	1,067.3	0.002889
39	98.61	1,069.2	0.002969
40	100.00	204.0	0.004285
41	95.83	574.2	0.003585
42	94.44	1,224.7	0.008837
43	97.22	385.9	0.004102
44	87.50	4,096.2	0.003966
45	88.89	2,903.3	0.007481
46	100.00	460.1	0.003718
47	86.11	1,083.3	0.003143
48	98.61	1,850.8	0.002896
49	100.00	432.4	0.003541
50	98.61	147.4	0.005534
51	97.22	773.6	0.003500
52	98.61	470.9	0.003499
53	90.28	2,823.3	0.005842
54	95.83	277.9	0.003858
55	95.83	108.2	0.007606
56	97.22	706.1	0.003481
57	97.22	2,123.1	0.003522
58	100.00	557.7	0.003603
59	100.00	111.8	0.008677
60	100.00	764.7	0.003785
61	100.00	803.8	0.004643
62	93.06	1,746.2	0.005704
63	100.00	323.5	0.007503
64	100.00	479.6	0.003182
65	100.00	256.6	0.005280
66	87.50	1,825.8	0.006043
67	100.00	245.1	0.005053
68	95.83	175.2	0.006838
69	100.00	188.4	0.005686
70	98.61	318.2	0.005057
71	100.00	127.5	0.008190
72	100.00	188.5	0.010236
73	97.22	307.3	0.005863
74	95.83	146.0	0.005670
75	98.61	324.6	0.006175
76	100.00	184.2	0.006606
77	98.61	265.6	0.006890
78	94.44	1,219.8	0.004386
79	98.61	477.0	0.005367
80	100.00	133.0	0.008857
81	84.72	245.7	0.007117
82	95.83	337.2	0.005832
83	93.06	215.2	0.007562
84	97.22	78.6	0.012173
85	98.61	148.4	0.005881
86	98.61	247.3	0.007879
87	100.00	138.3	0.009969
88	97.22	124.4	0.010189
89	98.61	433.7	0.005723
90	93.06	1,060.4	0.005298
91	97.22	218.1	0.008172

Continued on the next page

Table G1: Per-step rollout diagnostics (continued).

Step	Accuracy reward (%)	Thinking tokens	OPD loss
92	97.22	115.2	0.007236
93	98.61	64.5	0.012430
94	98.61	71.6	0.013010
95	98.61	222.5	0.006745
96	98.61	148.5	0.014111
97	98.61	163.2	0.011522
98	97.22	63.5	0.017169
99	98.61	79.2	0.013653
100	97.22	91.5	0.012959
101	98.61	113.6	0.011199
102	98.61	780.5	0.007792
103	98.61	87.1	0.013535
104	100.00	110.2	0.016679
105	98.61	66.8	0.020750
106	100.00	181.2	0.017591
107	98.61	447.2	0.011246
108	90.28	53.0	0.014710
109	95.83	76.2	0.017920
110	100.00	318.2	0.013682
111	98.61	162.2	0.016485
112	100.00	46.4	0.018066

Data note. The supplied per-step accuracy percentages are rounded to two decimal places and fall on a 1.39-point grid, consistent with increments of two correct responses over a 144-trajectory diagnostic batch. This diagnostic denominator is distinct from the 576 trajectories accumulated across the four micro-batches for each optimizer update.

H ACCURACY–THINKING-COST COMPARISONS ACROSS TASKS AND SIZES

We provide an updated 5-by-2 scatter grid derived from Table 1 for Qwen3.5-4B and Qwen3.5-9B on MATH500, AIME25, AIME26, MMLU-Pro, and ProofWriter. Columns correspond to model sizes and rows to tasks; the horizontal axis is average thinking tokens in thousands and the vertical axis is accuracy (%). Distinct markers represent methods, with the teacher as a capability reference.

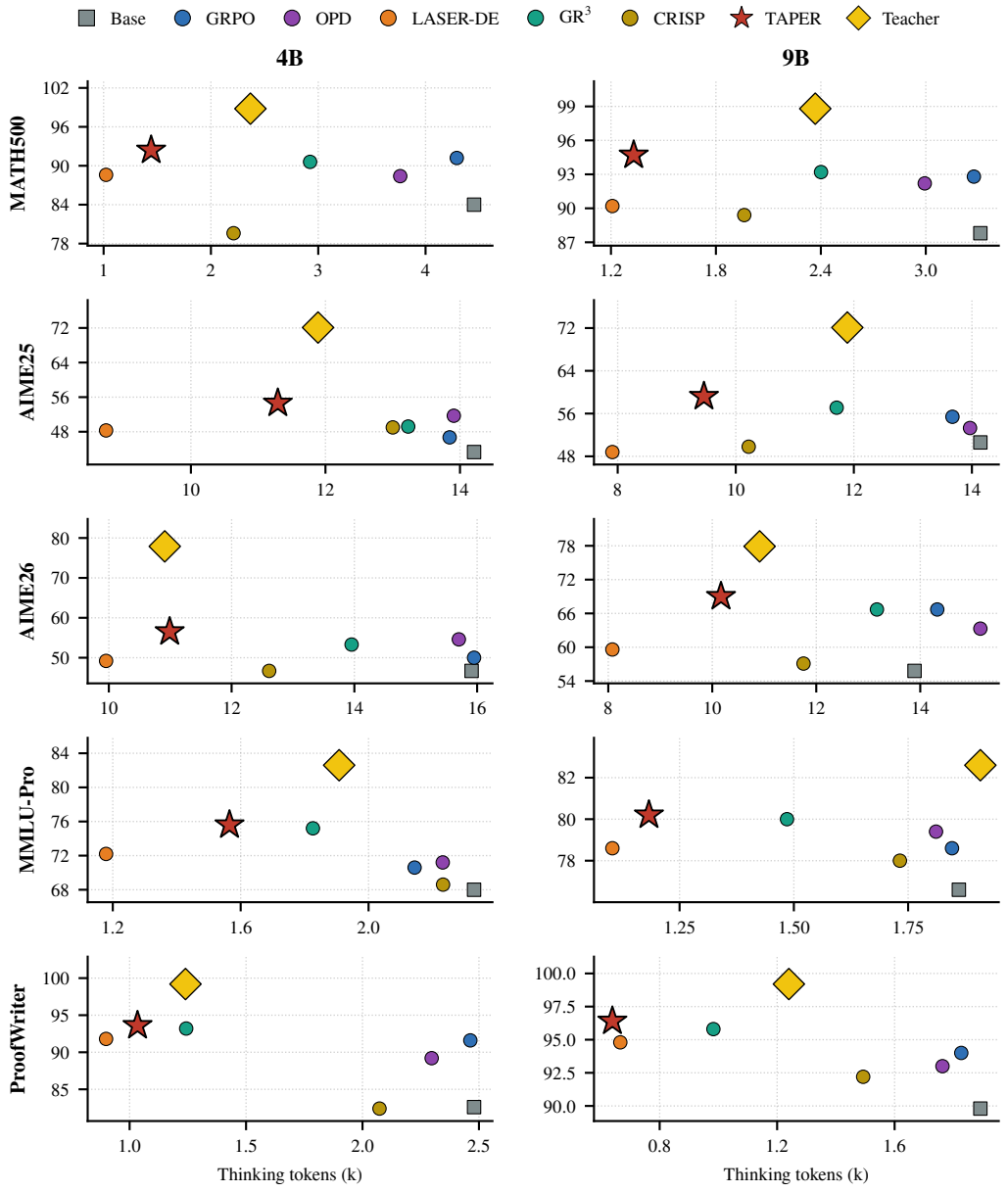

Figure H1: Cross-task and cross-size accuracy–thinking-cost comparisons derived from Table 1. Columns show the 4B and 9B students, rows show the five benchmarks, and the star denotes TAPER.

The numerical values used to draw the main-text scatter plots are reported below for direct cross-checking. The method label is shown as TAPER throughout.

Table H1: Numerical data for the left panel of Figure 1.

Model family	Method	Accuracy (%)	Thinking tokens
9B	Base	87.8	3,312
9B	GRPO	92.8	3,275
9B	OPD	92.2	2,994
9B	LASER-DE	90.2	1,208
9B	GR^3	93.2	2,401
9B	CRISP	89.4	1,962
9B	TAPER	94.7	1,330
122B-A10B	Teacher	98.8	2,368

Table H2: Numerical data for the difficulty-stratified accuracy and thinking-length plot in Figure 3.

Method	MATH Easy Acc.	Tok.	MATH Hard Acc.	Tok.	AIME Acc.	Tok.
Base	93.1723	2,890.7	82.8244	3,694.7	53.2292	14,016.5
GRPO	97.6891	2,751.7	88.2634	3,750.3	61.0417	13,998.5
OPD	98.0042	2,571.9	86.8321	3,377.5	58.3333	14,562.0
LASER-DE	95.3782	988.6	85.4008	1,407.3	54.1667	7,994.5
GR^3	96.5336	1,851.3	90.0763	2,900.3	61.8750	12,438.0
CRISP	96.4286	1,718.5	82.9198	2,183.2	53.4375	10,987.0
TAPER	98.6345	1,084.3	91.0305	1,553.2	64.0625	9,814.5

I REASONING-BEHAVIOR ANNOTATION

The reasoning-behavior analysis in Section 5.2 classifies segments of generated reasoning traces into four categories: **Advance**, **Verify**, **Revise**, and **Redundancy**.

The four categories are defined in Table 2 in the main text; the annotation prompt is provided below.

Figure I1 compares behavior composition across the MATH Easy, MATH Hard, and AIME25/26 evaluation groups for Base, GRPO, and TAPER. The behavior shares and evaluation counts are reported in Tables I1 and I2. Source values for the overall comparison and training series are included in `supplementary/panel_a_data.csv` and `supplementary/panel_b_data.csv`; the stratified values are in `supplementary/behavior_by_difficulty.csv`.

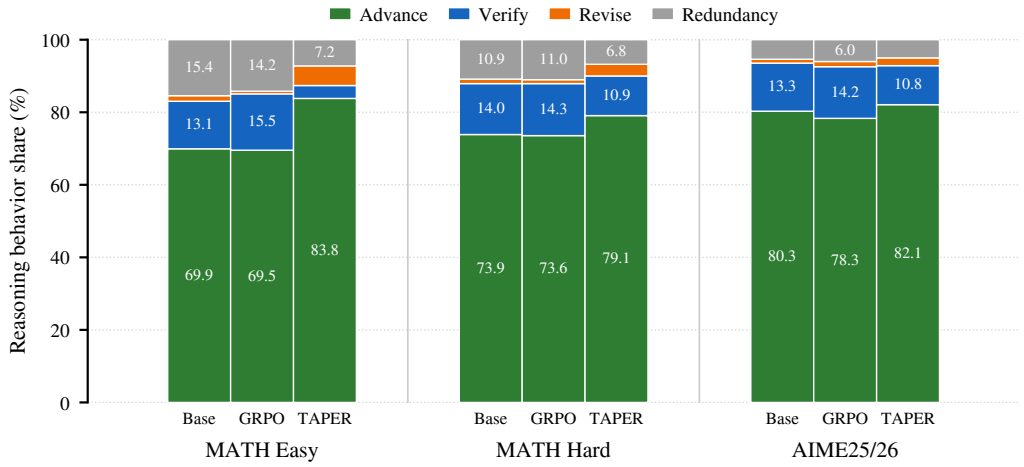

Figure I1: Reasoning-behavior composition across evaluation groups for the 9B student. Each panel compares Base, GRPO, and TAPER, with shares for Advance, Verify, Revise, and Redundancy.

Table I1: Behavior-category shares by evaluation group and method (percent).

Evaluation group	Method	Advance	Verify	Revise	Redundancy
MATH Easy	Base	69.9	13.1	1.5	15.4
MATH Easy	GRPO	69.5	15.5	0.7	14.2
MATH Easy	TAPER	83.8	3.6	5.4	7.2
MATH Hard	Base	73.9	14.0	1.2	10.9
MATH Hard	GRPO	73.6	14.3	1.1	11.0
MATH Hard	TAPER	79.1	10.9	3.2	6.8
AIME25/26	Base	80.3	13.3	1.1	5.4
AIME25/26	GRPO	78.3	14.2	1.4	6.0
AIME25/26	TAPER	82.1	10.8	2.1	5.0

Table I2: Evaluation and annotation counts and mean thinking length for the difficulty-stratified behavior analysis. A dash indicates a value not reported.

Evaluation group	Problems	Samples/item	Eval. samples	Judged/model	Base tokens	GRPO tokens	TAPER tokens
MATH Easy	238	4	952	—	2,890.7	2,751.7	—
MATH Hard	262	4	1,048	60	3,694.7	3,750.3	1,553.2
AIME25/26	60	16	960	60	14,016.5	13,998.5	9,814.5

Table I3: Per-step behavior shares and annotation counts used in the right panel of Figure 4. Step 0 is the Base anchor; all trained checkpoints use evaluation traces.

Step	Source	Advance	Verify	Revise	Redund.	Traces	Behav. tokens
0	Base	72.00	13.60	1.38	13.02	150	597,458
5	Eval	73.48	12.13	1.43	12.96	360	647,018
10	Eval	74.13	12.99	1.58	11.30	360	535,809
15	Eval	74.76	12.87	1.79	10.58	360	598,084
20	Eval	73.15	13.96	1.78	11.11	150	418,188
25	Eval	75.64	11.96	1.30	11.10	150	352,182
30	Eval	73.80	13.56	2.53	10.11	150	341,886
35	Eval	75.90	13.14	2.33	8.63	150	357,785
40	Eval	74.94	13.40	1.95	9.71	150	332,681
45	Eval	77.59	12.91	2.31	7.19	150	326,395
50	Eval	75.09	14.30	2.73	7.88	150	320,444
55	Eval	76.10	13.93	3.19	6.78	150	324,033
60	Eval	78.70	11.65	2.90	6.75	150	307,673
65	Eval	79.28	11.85	3.63	5.24	150	295,812
70	Eval	78.22	12.78	4.30	4.70	150	264,556
75	Eval	78.86	11.80	3.93	5.41	150	252,767
80	Eval	78.48	12.57	4.77	4.18	150	228,015
85	Eval	80.17	11.28	4.30	4.25	150	220,695
90	Eval	79.40	11.42	5.99	3.19	150	195,598
95	Eval	82.69	8.86	5.26	3.19	150	187,140
100	Eval	81.04	10.24	7.20	1.52	150	144,857
105	Eval	81.99	8.12	6.57	3.32	150	122,129
110	Eval	79.51	10.78	6.38	3.33	150	111,931
112	Final eval	80.60	10.50	6.60	2.30	150	79,665

The complete two-page annotation prompt used in our analysis is reproduced in Figure I2.

The annotator outputs the entire reasoning trace with every segment enclosed by one of the following tags:

```
<advance>...</advance>
<verify>...</verify>
<revise>...</revise>
<redundancy>...</redundancy>
```

No reasoning content is omitted during annotation. The resulting token spans are aggregated to calculate the proportion of each behavior category reported in Section 5.2.

The source prompt is also reproduced below for reference.

1. Prompt reasoning-behaviour segmentation

You are given a complete mathematical reasoning trace.

The four states (one XML tag each)

Advance Verify Revise Redundancy

<advance> establishes a new solving state, such as deriving a new relation, computing a new result, introducing a case split, or setting a concrete subgoal. It moves the solution forward rather than correcting earlier reasoning.

<verify> substantively checks an existing conclusion, such as substituting a result back into the original problem, checking boundary or edge cases, or independently recomputing a result. Merely stating that a conclusion is correct does not count as verification.

<revise> actually modifies a previous solving state, such as correcting an earlier computation, retracting an assumption, adding a missed condition, or abandoning a failed path and switching to a different approach.

<redundancy> only repeats or restates already established content. It introduces no new solving information, provides no substantive verification evidence, and makes no real modification to the current reasoning state.

Requirements

* Segment the complete reasoning trace according to meaningful changes in reasoning behavior.
* Every part of the original reasoning trace must appear exactly once.
* Preserve the original text exactly. Do not rewrite, summarize, delete, or add any content.
* Each segment must be wrapped in exactly one tag.
* Keep all text in its original order.
* Use the smallest meaningful segment that corresponds to a single reasoning behavior.
* If several consecutive sentences clearly belong to the same reasoning action, they may remain in the same segment.
* Do not output explanations, comments, Markdown, or any text outside the tags.

Output format

Output ONLY the fully tagged reasoning trace, with no additional text.

Figure I2: Prompt used for reasoning-behavior segmentation (first page).

2. Worked Example

illustrative · "How many divisors does 36 have?"

Input: a complete reasoning trace

Problem: How many positive divisors does 36 have?

Factor 36 into primes: $36 = 2^2 \cdot 3^2$. The number of positive divisors is the product of one more than each exponent, so $(2+1)(2+1) = 3 \cdot 3 = 6$.

Let me verify by listing every divisor: 1, 2, 3, 4, 6, 9, 12, 18, 36. Counting them gives 9, not 6.
So my multiplication was wrong: $(2+1)(2+1) = 9$.

So $36 = 2^2 \cdot 3^2$ and the divisor-count formula applies here.

Therefore 36 has 9 positive divisors.

Model output: the same text, fully tagged (verbatim)

<advance>Factor 36 into primes: 36 = 2^2 * 3^2. The number of positive divisors is the product of one more than each exponent, so (2+1)(2+1) = 3 * 3 = 6.</advance><verify>Let me verify by listing every divisor: 1, 2, 3, 4, 6, 9, 12, 18, 36. Counting them gives 9, not 6.</verify><revise>So my multiplication was wrong: (2+1)(2+1) = 9.</revise><redundancy>So 36 = 2^2 * 3^2 and the divisor-count formula applies here.</redundancy><advance>Therefore 36 has 9 positive divisors.</advance>

Interpreted: each segment coloured by its state

Factor 36 into primes: 36 = 2^2 * 3^2. The number of positive divisors is the product of one more than each exponent, so (2+1)(2+1) = 3 * 3 = 6.

Let me verify by listing every divisor: 1, 2, 3, 4, 6, 9, 12, 18, 36. Counting them gives 9, not 6.

So my multiplication was wrong: (2+1)(2+1) = 9.

So 36 = 2^2 * 3^2 and the divisor-count formula applies here.

Therefore 36 has 9 positive divisors.

advance derives a divisor count, verify lists the divisors and finds the count disagrees, revise fixes the arithmetic, redundancy merely restates a known fact, and a final advance states the answer.

J TAPER AND BASE CASE STUDY

TAPER trace (after optimization).

System Prompt

You are a helpful math assistant. Solve the problem correctly.
Use the shortest clear reasoning needed to reach the answer, and keep the final response as concise as possible.
Write the reasoning inside <think>...</think>. After </think>, answer directly in exactly one short sentence containing \boxed{...}. Put only the final result inside the box.

Query MATH500 #66 · level 5

The height (in meters) of a shot cannonball follows a trajectory given by $h(t) = -4.9t^2 + 14t - 0.4$ at time t (in seconds). As an improper fraction, for how long is the cannonball above a height of 6 meters?

TAPER (after optimization) thinking 230 tok · correct

Advance Verify Revise Redundancy

Thinking (actions colour-coded)

$-4.9t^2 + 14t - 0.4 > 6 \Longrightarrow -4.9t^2 + 14t - 6.4 > 0 \Longrightarrow 49t^2 - 140t + 64 < 0$.

Roots are $t = \frac{140 \pm \sqrt{19600 - 12544}}{98} = \frac{140 \pm \sqrt{7056}}{98} = \frac{140 \pm 84}{98}$.

$t_1 = \frac{56}{98} = \frac{4}{7}$, $t_2 = \frac{224}{98} = \frac{32}{14} = \frac{16}{7}$.

The time duration is $\frac{16}{7} - \frac{4}{7} = \frac{12}{7}$.

Final Answer

To find the duration the cannonball is above 6 meters, we set up the inequality:

$$h(t) > 6 \Longrightarrow -4.9t^2 + 14t - 0.4 > 6$$

$$-4.9t^2 + 14t - 6.4 > 0$$

Multiply by -10 to clear decimals and reverse the inequality sign:

$$49t^2 - 140t + 64 < 0$$

Find the roots of $49t^2 - 140t + 64 = 0$:

$$t = \frac{-(-140) \pm \sqrt{(-140)^2 - 4(49)(64)}}{2(49)} = \frac{140 \pm \sqrt{19600 - 12544}}{98} = \frac{140 \pm \sqrt{7056}}{98} = \frac{140 \pm 84}{98}$$

The roots are $t_1 = \frac{140 - 84}{98} = \frac{56}{98} = \frac{4}{7}$ and $t_2 = \frac{140 + 84}{98} = \frac{224}{98} = \frac{16}{7}$.

Since the parabola opens upward, the inequality holds between the roots. The duration is:

$$\frac{16}{7} - \frac{4}{7} = \frac{12}{7}$$

$\boxed{\frac{12}{7}}$

Base (before optimization). The first page of the five-page trace is shown below; the remaining pages follow on separate pages.

Base (before optimization) thinking 956 tok · correct

Advance Verify Revise Redundancy

Thinking (actions colour-coded)

The problem asks for the duration of time for which the cannonball is above a height of 6 meters.
The height function is $h(t) = -4.9t^2 + 14t - 0.4$.

We need to find the interval of time t such that $h(t) > 6$.

First, let's set up the inequality:
$-4.9t^2 + 14t - 0.4 > 6$

Rearranging the terms:
$-4.9t^2 + 14t - 0.4 - 6 > 0$

$-4.9t^2 + 14t - 6.4 > 0$

To make it easier, multiply by -1 and flip the inequality sign:
$4.9t^2 - 14t + 6.4 < 0$

Let's work with integers or simpler fractions if possible.
$4.9 = \frac{49}{10}$

$6.4 = \frac{64}{10} = \frac{32}{5}$

So, $\frac{49}{10}t^2 - 14t + \frac{32}{5} < 0$

Multiply the entire inequality by 10 to clear denominators:
$49t^2 - 140t + 64 < 0$

Now we need to solve the quadratic equation $49t^2 - 140t + 64 = 0$ to find the roots.

Using the quadratic formula $t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$:

$a = 49$, $b = -140$, $c = 64$.

Discriminant $\Delta = b^2 - 4ac$:

$\Delta = (-140)^2 - 4(49)(64)$

$\Delta = 19600 - 4(49)(64)$

$4 \times 64 = 256$

$\Delta = 19600 - 256(49)$

$256 \times 49 = 256(50 - 1) = 12800 - 256 = 12544$

$\Delta = 19600 - 12544 = 7056$

Now calculate $\sqrt{\Delta}$:

$\sqrt{7056}$. Let's check numbers ending in 4. 80^2 = 6400, 90^2 = 8100. So it's between 80 and 90.

Ends in 6 implies root ends in 4 or 6.
Try 84: $(80+4)^2 = 6400 + 640 + 16 = 7056$. Yes.

So $\sqrt{\Delta} = 84$.

Calculate the roots:
$t_1 = \frac{140-84}{2(49)} = \frac{56}{98}$

$t_2 = \frac{140+84}{2(49)} = \frac{224}{98}$

Simplify fractions:
$t_1 = \frac{56}{98}$. Both divisible by 14? $56 = 14 \times 4$, $98 = 14 \times 7$.

$t_1 = \frac{4}{7}$ seconds.

$t_2 = \frac{224}{98}$. Both divisible by 14? $224/14 = 16$, $98/14 = 7$.

$t_2 = \frac{16}{7}$ seconds.

The quadratic $49t^2 - 140t + 64$ opens upwards (coefficient of t^2 is positive).

Therefore, the expression is less than zero between the roots.
So the condition $h(t) > 6$ holds for $t \in (\frac{4}{7}, \frac{16}{7})$.

The duration is the difference between the upper and lower limits of this interval.
Duration $\Delta t = \frac{16}{7} - \frac{4}{7} = \frac{12}{7}$.

The question asks for an improper fraction. $\frac{12}{7}$ is an improper fraction.

Let me double check the calculations.
Equation: $49t^2 - 140t + 64 = 0$

Factorization attempt:
We need factors of $49 \times 64 = 3136$ that add up to 140.
Wait, splitting the middle term.
$49t^2 - 140t + 64$

Roots were $\frac{4}{7}$ and $\frac{16}{7}$.

So $49t^2 - 140t + 64 = 49(t - \frac{4}{7})(t - \frac{16}{7})$

$= 49(\frac{7t-4}{7})(\frac{7t-16}{7}) = (7t-4)(7t-16) = 49t^2 - 112t - 28t + 64 = 49t^2 - 140t + 64$. Correct.

Inequality: $49t^2 - 140t + 64 < 0$.

This means $(7t-4)(7t-16) < 0$.

Since parabola opens up, it's negative between roots.
$(7t-4) < 0 \implies t < 4/7$

$(7t-16) < 0 \implies t < 16/7$

Product is negative when one factor is positive and one is negative.
$7t - 4 > 0 \implies t > 4/7$ AND $7t - 16 < 0 \implies t < 16/7$.

So interval is $(4/7, 16/7)$.

Time duration is $t_{upper} - t_{lower} = \frac{16}{7} - \frac{4}{7} = \frac{12}{7}$.

Is there any constraint on t being positive?
The roots are both positive ($4/7 \approx 0.57$ s and $16/7 \approx 2.28$ s).
The shot starts at $t = 0$ usually, but here $h(0) = -0.4$, so it was underground or just started from below.
The ball goes up then down.
Since $4/7 > 0$, the part of the trajectory above 6m happens after launch.
Specifically, the ball is released at some t_{launch}. If we assume $t = 0$ is the time given in the formula, then for $t < 4/7$, height is less than 6. For t between $4/7$ and $16/7$, height is greater than 6.
So the total time spent above 6m is indeed the length of the interval.

The value requested is "As an improper fraction".
Result: $\frac{12}{7}$.

Let's re-read carefully.
"As an improper fraction, for how long is the cannonball above a height of 6 meters?"
"for how long" implies the duration, i.e., Δt.
Units are seconds.
Answer should be a number inside box.

Check arithmetic again.
$h(t) > 6$
$-4.9t^2 + 14t - 0.4 > 6$
$-4.9t^2 + 14t - 6.4 > 0$
$4.9t^2 - 14t + 6.4 < 0$
Multiply by 10: $49t^2 - 140t + 64 < 0$
Roots of $49t^2 - 140t + 64 = 0$
$t = \frac{140 \pm \sqrt{140^2 - 4(49)(64)}}{2(49)}$
$140^2 = 19600$
$4 \cdot 49 \cdot 64 = 196 \cdot 64 = 200 \cdot 64 - 4 \cdot 64 = 12800 - 256 = 12544$
$19600 - 12544 = 7056$
$\sqrt{7056} = 84$
$t = \frac{140 \pm 84}{98}$
$t_1 = \frac{56}{98} = \frac{4}{7}$
$t_2 = \frac{224}{98} = \frac{16}{7}$
Difference = $12/7$.

The result seems robust.

Final Answer Format:
Reasoning inside <think>...</think>.
Short sentence containing boxed answer.

The logic flow:
1. Set up inequality for height > 6.
2. Solve corresponding quadratic equation for equality.
3. Calculate difference between roots to find duration.
4. Verify steps and arithmetic.

Step 1: $-4.9t^2 + 14t - 0.4 > 6 \implies 49t^2 - 140t + 64 < 0$.

Step 2: Roots of $49t^2 - 140t + 64 = 0$ are $t = \frac{140 \pm 84}{98}$.

Step 3: Roots are $\frac{56}{98} = \frac{4}{7}$ and $\frac{224}{98} = \frac{16}{7}$.

Step 4: Duration is $\frac{16}{7} - \frac{4}{7} = \frac{12}{7}$.

The phrase "for how long" asks for the length of the time interval.
The units match (seconds).
The format requires an improper fraction. 12/7 fits.

Just to ensure no misinterpretation of "improper fraction". It means numerator > denominator. 12/7 is correct. If it were a mixed number like 1 5/7, that would likely not be preferred unless specified, but usually improper fraction means A/B.

Double check the discriminant calculation.
$b^2 - 4ac = (-140)^2 - 4(49)(64)$

$140^2 = 19600$.

$4 \times 49 = 196$.
$196 \times 64 = 12544$.
$19600 - 12544$:

19600
-12544

7056.
Correct.
Square root of 7056.
$80^2 = 6400$.

$90^2 = 8100$.

Last digit 6 -> 4 or 6.
$84^2 = (80+4)^2 = 6400 + 640 + 16 = 7056$.

$86^2 = (90-4)^2 = 8100 - 720 + 16 = 7396$. No.

So 84 is correct.

Roots:
$(140 - 84)/98 = 56/98 = 4/7$.

$(140 + 84)/98 = 224/98 = 16/7$.

Subtraction: $16/7 - 4/7 = 12/7$.

Looks good.

One detail: "improper fraction".
If the answer was, say, 2.5, it would be 5/2.
Here 12/7 is already improper.

Is there any ambiguity about "above a height of 6 meters"?
Strictly above ($>$) or at least ($\geq$)? Usually "above" implies strict inequality. Since it's a continuous function, the duration is the same whether it includes endpoints or not.
The wording "for how long" suggests a measure of time, which is a single point value regardless of inclusion of boundary instants (measure 0).

The solution seems complete.

Final check on the question text provided in prompt: "Write the reasoning inside ... After </think>, answer directly in exactly one short sentence containing $\boxed{\ldots}$. Put only the final result inside the box."

Okay.

Reasoning summary:
1. Determine when $h(t) > 6$.

2. Inequality simplifies to $49t^2 - 140t + 64 < 0$.

3. Find roots of $49t^2 - 140t + 64 = 0$.

4. Roots are $4/7$ and $16/7$.

5. Duration is the difference between roots, $12/7$ seconds.

Writing response.

TAPER reaches the same answer in 230 thinking tokens.

Final Answer

To find the duration for which the cannonball is above a height of 6 meters, we set the height function $h(t) = -4.9t^2 + 14t - 0.4$ strictly greater than 6.

$$-4.9t^2 + 14t - 0.4 > 6$$

Subtract 6 from both sides:

$$-4.9t^2 + 14t - 6.4 > 0$$

Multiplying the inequality by -1 reverses the inequality sign:

$$4.9t^2 - 14t + 6.4 < 0$$

To eliminate decimals, multiply by 10:

$$49t^2 - 140t + 64 < 0$$

Next, we find the roots of the corresponding quadratic equation $49t^2 - 140t + 64 = 0$ using the quadratic formula:

$$t = \frac{-(-140) \pm \sqrt{(-140)^2 - 4(49)(64)}}{2(49)}$$

$$t = \frac{140 \pm \sqrt{19600 - 12544}}{98}$$

$$t = \frac{140 \pm \sqrt{7056}}{98}$$

$$t = \frac{140 \pm 84}{98}$$

The two solutions are:

$$t_1 = \frac{140 - 84}{98} = \frac{56}{98} = \frac{4}{7}$$

$$t_2 = \frac{140 + 84}{98} = \frac{224}{98} = \frac{16}{7}$$

The quadratic $49t^2 - 140t + 64$ represents a parabola opening upwards, so it is negative (less than zero) between its roots. Thus, the cannonball is above 6 meters during the time interval $(\frac{4}{7}, \frac{16}{7})$.

The duration is the difference between these times:

$$\text{Duration} = \frac{16}{7} - \frac{4}{7} = \frac{12}{7} \text{ seconds}$$

The duration the cannonball is above 6 meters is $\boxed{\frac{12}{7}}$.