FIXED LENGTH PENALTIES: ACCURACY–LENGTH TRADE-OFFS IN MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Length-aware reinforcement learning can shorten reasoning, but assessing adaptive rewards requires a strong account of what a simple fixed cost already achieves. We study a constant penalty on capped, normalized completion length in reinforcement learning with verifiable rewards. Across eight paired training seeds of DeepSeek-R1-Distill-Qwen-1.5B, a coefficient of 0.3 reduces mean generated tokens by 26.9%, 12.0%, 2.8%, and 7.8% on GSM8K, MATH-500, AIME, and AMC, with mean accuracy changes of -0.80, $+0.63$, $+0.21$, and $+2.11$ percentage points. Every seed pair yields shorter generations on every benchmark, and tokens per correct response fall by 26.2%, 12.7%, 3.8%, and 11.1%. Without problem-specific budgets, difficulty estimates, or online penalty adaptation, the fixed cost captures 84.9–97.3% of the largest mean token saving achieved by the tested adaptive controls. The largest absolute savings occur in intermediate-difficulty groups, while difficult AIME problems dominate the remaining generation volume. These results establish a simple and competitive accuracy–efficiency reference for evaluating adaptive length control.

1 INTRODUCTION

Extended reasoning has advanced mathematical problem solving through chain-of-thought generation and reinforcement learning with verifiable rewards (RLVR) (Wei et al., 2022; Shao et al., 2024; DeepSeek-AI et al., 2025). Inference-time scaling makes the computation spent on each answer an important design choice (Snell et al., 2025). Under correctness-only rewards, successful answers receive equal task scores regardless of their length. Studies of overthinking document repeated solutions and prolonged verification, while shorter sampled trajectories can remain highly accurate (Chen et al., 2025; Hassid et al., 2025). These observations motivate training objectives that account for both answer quality and generation cost.

Length-aware rewards provide a direct way to address this challenge. Prior work has shown that a scalar penalty can reduce response length and control the accuracy–length trade-off (Arora & Zanette, 2025). Other approaches specify token budgets or adapt length targets and penalties to sampled solutions, estimated difficulty, and online solve rates (Aggarwal & Welleck, 2025; Yi et al., 2025; Shen et al., 2025; Xiang et al., 2025). These designs reflect a central difficulty of efficient reasoning: problems vary in the amount of computation needed for a correct answer.

Assessing adaptive length control requires first establishing how much a shared cost can achieve on its own. Unlike methods that assign per-problem budgets, estimate difficulty, or adjust penalties online, a fixed-coefficient reward adds one transparent cost to the correctness objective. Its simplicity raises a central question: *Can a uniform length cost deliver stable compression and a competitive accuracy–efficiency trade-off, and how much additional value do adaptive rewards provide under a common training protocol?*

We answer this question with DeepSeek-R1-Distill-Qwen-1.5B and a fixed cost on capped, normalized completion length. Across eight paired training seeds, the cost shortens every benchmark in every seed pair, reduces mean token use by 2.8–26.9%, and produces positive mean accuracy changes on three of the four benchmarks; the remaining change is -0.80 percentage points on GSM8K. Tokens per correct response decrease on all four benchmarks. Comparisons under shared sampling and optimization settings further show that the fixed cost captures 84.9–97.3% of the largest mean token saving from the tested adaptive controls.

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.

Our contributions are:

1. **A simple fixed cost with strong accuracy–efficiency performance.** Across four mathematical benchmarks and eight paired training seeds, the fixed reward consistently shortens responses, produces positive mean accuracy changes on three benchmarks, and reduces tokens per correct response on all four.
2. **A competitive trade-off without adaptive reward machinery.** Without difficulty estimates, per-problem budgets, or online penalty adjustment, the fixed setting captures 84.9–97.3% of the largest mean token saving achieved by the tested adaptive controls, providing a strong and transparent reference for their additional value.
3. **A difficulty-resolved account of savings and residual cost.** Intermediate problems contribute the largest absolute token savings, easy problems show the largest proportional shortening, and difficult competition problems retain most of the remaining generation volume, identifying where further efficiency gains are available.

2 Related Work

2.1 Reasoning and Verifiable-Reward Optimization

Intermediate computation can be elicited through scratchpads, chain-of-thought examples, zero-shot instructions, and sequential problem decomposition (Nye et al., 2021; Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2022). Self-consistency and tree search organize multiple reasoning paths at inference time (Wang et al., 2022; Yao et al., 2023). Training approaches improve the reasoning policy itself: STaR learns from generated rationales, and mathematical self-improvement combines data generation, verification, and reinforcement learning (Zelikman et al., 2022; Yang et al., 2024; DeepSeek-AI et al., 2025; Kimi Team et al., 2025). Outcome and process supervision provide feedback at different granularities (Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2023). Our reward uses final-answer verification and a completion-level length cost.

Policy optimization determines how these rewards affect generation. PPO uses a clipped probability-ratio objective (Schulman et al., 2017); GRPO and REINFORCE-style methods estimate advantages from sampled responses (Shao et al., 2024; Ahmadian et al., 2024). DAPO introduces token-level loss aggregation, while Dr. GRPO and REINFORCE++ examine aggregation and normalization choices (Yu et al., 2025; Liu et al., 2025; Hu et al., 2025). Length-aware rewards interact with advantage normalization (Arora & Zanette, 2025). We specify both the group-normalized advantages and the token-aggregated surrogate used in our experiments.

2.2 Efficient Reasoning and Length Control

Adaptive computation allocates processing steps or model depth according to the input (Graves, 2016; Schuster et al., 2022). For reasoning models, efficiency methods additionally control the generated trajectory (Qu et al., 2025). TALE and s1 use prompt budgets or budget forcing; L1 learns requested length constraints, DAST adapts budgets to difficulty, and AdaptThink learns when to invoke extended thinking (Han et al., 2025; Muennighoff et al., 2025; Aggarwal & Welleck, 2025; Shen et al., 2025; Zhang et al., 2025a). Compression methods operate on the representation of reasoning: TokenSkip removes less important textual tokens, LightThinker compresses histories into gist tokens, and Coconut and Soft Thinking use continuous representations (Xia et al., 2025; Zhang et al., 2025b; Hao et al., 2024; Zhang et al., 2025c).

Reward-based length control provides a direct post-training route. O1-Pruner uses reference-performance estimates; ThinkPrune progressively tightens generation limits (Luo et al., 2025; Hou et al., 2025). Arora and Zanette normalize penalties using successful-response lengths, ShorterBetter derives a target from the shortest correct sample, and ALP scales the cost by online solve rate (Arora & Zanette, 2025; Yi et al., 2025; Xiang et al., 2025). Lazy penalties condition length pressure on correctness and training stage (Yuan et al., 2026). Preference optimization supplies another route to favor selected responses (Rafailov et al., 2023). Our study fixes the coefficient and normalization cap throughout training and compares this uniform cost with explicitly specified adaptive reward controls.

2.3 Reasoning Behavior and Evaluation

Behavioral studies connect reasoning length to repeated solutions, verification, and sample selection (Chen et al., 2025; Hassid et al., 2025). Repetition can be targeted explicitly through unlikelihood training (Welleck et al., 2019); SCoRe trains self-correction through multi-turn reinforcement learning (Kumar et al., 2024). ARLCP coordinates reflection and length penalties, and CLORE edits correct rollouts with a preference objective (Yu et al., 2026; Wu et al., 2026). We measure local repetition, lexical markers, and truncation alongside accuracy to characterize the outputs produced by a scalar length reward. GSM8K and MATH provide answer-verifiable mathematics tasks (Cobbe et al., 2021; Hendrycks et al., 2021); HumanEval and MBPP provide executable tests for the code extension (Chen et al., 2021; Austin et al., 2021). These evaluation settings support a joint analysis of answer success, generation volume, and observable response behavior.

3 Method

Figure 1 summarizes the training pipeline. For each problem, the policy generates a group of candidate solutions. We verify each final answer, measure the complete response length, and combine these signals into a correctness reward with a fixed length cost. Group-normalized rewards provide response advantages, which are broadcast to completion tokens for a clipped policy update. Repeating this sample–score–update loop trains the policy to jointly account for solution correctness and generation cost.

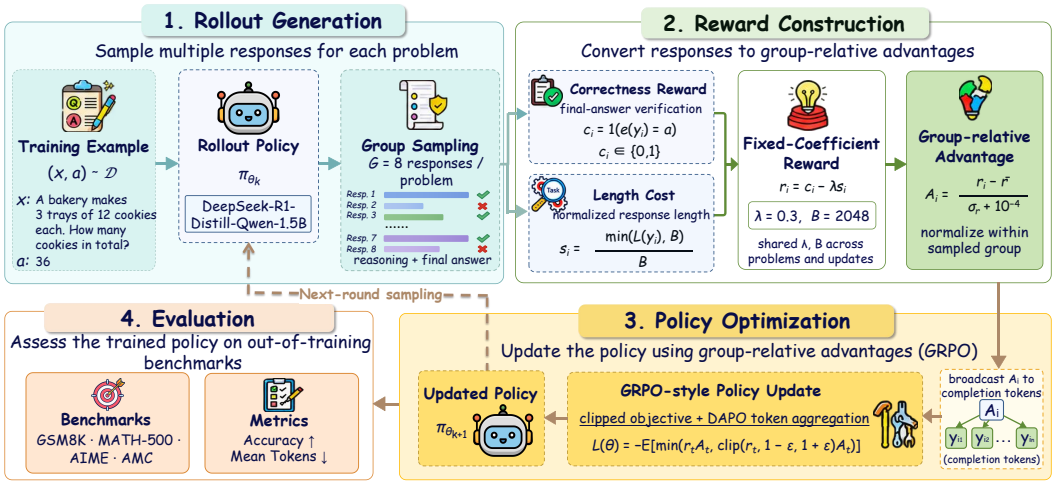


Figure 1: Training pipeline. Sampled responses receive a correctness reward and a fixed length cost. Group-relative advantages drive the clipped policy update with DAPO token aggregation; the updated policy generates the next batch. The displayed settings correspond to the primary mathematical experiment.

3.1 Problem Formulation

Let $(x, a) \sim \mathcal{D}$ denote a mathematical problem and its reference answer. An autoregressive policy π_θ generates a complete response $y = (y_1, \ldots, y_T)$, with $\pi_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t})$. The response contains the intermediate reasoning and final answer. Generation terminates at the end-of-sequence token or the configured completion limit, so the realized trajectory length varies across problems and sampled responses.

An answer extractor $e(y)$ identifies the final prediction, and a verifier returns $c(x, y) = \mathbb{1}\{e(y) \equiv a\}$, where equivalence follows the task-specific answer comparison. This supplies the outcome reward used in mathematical RL (Cobbe et al., 2021; Shao et al., 2024). We measure the generation cost $L(y)$ by tokenizing the complete decoded response with special tokens excluded. Correctness

and length are therefore evaluated on the same generated solution, including its intermediate reasoning.

The learning objective balances expected correctness with a normalized generation cost $s_B(y)$. With a nonnegative coefficient λ, we maximize

$$J_\lambda(\theta) = \mathbb{E}_{(x,a)\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[c(x,y) - \lambda s_B(y)]\,. \tag{1}$$

The next section specifies the bounded cost s_B. We evaluate the resulting policy through both answer accuracy and mean generated tokens, which describe its accuracy–length operating point.

3.2 Fixed-Coefficient Length Regularization

The reward assigns a constant marginal cost to response length below a fixed normalization cap $B > 0$:

$$s_B(y) = \frac{\min\{L(y), B\}}{B}, \qquad r_\lambda(x,y) = c(x,y) - \lambda\frac{\min\{L(y), B\}}{B}. \tag{2}$$

The same λ and B apply to every problem, correctness outcome, and training update. Normalization makes $s_B \in [0,1]$, and capping bounds the penalty by λ. Both successful and unsuccessful solutions incur the length cost. Below B, each additional token reduces the reward by λ/B; at and above B, the cost saturates. This construction gives a compact instance of length-aware reward optimization (Arora & Zanette, 2025; Xiang et al., 2025), with all problem-dependent behavior learned through the policy and its sampled groups.

For $0 \le \lambda < 1$, correct and incorrect responses have reward ranges $[1-\lambda, 1]$ and $[-\lambda, 0]$, respectively. Thus, the reward gap between any correct and incorrect response is at least $1 - \lambda > 0$. For $\lambda > 0$, a shorter capped length receives a higher reward among responses with equal correctness. These two properties encode a strict preference for correct answers and a secondary preference for concise solutions within each correctness class.

3.3 Policy Optimization

We optimize the objective in Equation (1) using GRPO-style group-relative advantages (Shao et al., 2024). For each prompt, the rollout policy samples $G > 1$ responses and computes $r_i = r_\lambda(x, y_i)$ from Equation (2). Using the sample standard deviation, we obtain

$$\bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j, \quad \sigma_r = \sqrt{\frac{1}{G-1}\sum_{j=1}^{G}(r_j - \bar{r})^2}, \quad A_i = \frac{r_i - \bar{r}}{\sigma_r + 10^{-4}}. \tag{3}$$

Each advantage A_i is held fixed during the update and broadcast to the response tokens. Let π_{old} denote the policy before the update. Let i index completions in an optimization batch, m_{it} indicate active tokens, and $\rho_{it}(\theta) = \pi_\theta(y_{it} \mid x_i, y_{i,<t})/\pi_{\text{old}}(y_{it} \mid x_i, y_{i,<t})$. We combine probability-ratio clipping (Schulman et al., 2017) with DAPO token aggregation (Yu et al., 2025):

$$\mathcal{L}(\theta) = -\frac{\sum_{i,t} m_{it} w_{it} \min[\rho_{it}(\theta)A_i, \text{clip}(\rho_{it}(\theta), 1-\epsilon, 1+\epsilon)A_i]}{\sum_{i,t} m_{it}}. \tag{4}$$

The detached weight w_{it} corrects the sampling-engine probability toward the old-policy probability and is capped at 2. The mask includes generated completion tokens through termination and excludes padding; truncated completions contribute their available tokens. Our experiments use $\epsilon = 0.2$, a zero KL coefficient, and one update iteration per generated batch. Appendix A gives the implementation and token accounting.

Centering the reward in Equation (2) exposes the length contribution to the update. Let $c_i = c(x, y_i)$ and $s_i = s_B(y_i)$, with group means $\bar{c}$ and $\bar{s}$. Then

$$r_i - \bar{r} = (c_i - \bar{c}) - \lambda(s_i - \bar{s}). \tag{5}$$

For $\lambda > 0$, below-average capped lengths receive positive centered rewards within an equal-correctness group. Group normalization rescales this signal by reward variation. Equal capped lengths cancel the length contribution; constant correctness then gives zero advantages. These properties motivate examining difficulty and truncation profiles in Section 4.4.

4 EXPERIMENTS

We evaluate the efficiency gains from a uniform length cost, the observed differences among adaptive rewards, and how compression varies with penalty strength and problem difficulty. Code and larger-model extensions examine the scope of these findings.

4.1 EXPERIMENTAL SETUP

Model and datasets. We fine-tune DeepSeek-R1-Distill-Qwen-1.5B on the 7,473 GSM8K training problems (DeepSeek-AI et al., 2025; Cobbe et al., 2021). Evaluation covers GSM8K test (1,319 problems), MATH-500 (500) (Hendrycks et al., 2021), and the AI-MO AIME (90) and AMC (83) collections. This combination spans grade-school mathematics and competition-style problems. Appendix A specifies the dataset identifiers, prompts, and answer verifier.

Training and decoding. All mathematical reward settings share full-parameter training for 400 updates, learning rate 10^{-5}, eight rollouts per prompt, and a 2,048-token completion limit. We evaluate the final checkpoint using eight samples per problem, temperature 0.6, top-p 0.95, and a limit of 3,072 tokens on GSM8K and 8,192 on the other benchmarks. The primary fixed penalty uses $\lambda = 0.3$ and $B = 2048$.

Baselines. Vanilla sets $\lambda = 0$ in Equation (2). Soft-budget assigns budgets from offline base-model solve rates and penalizes excess length (Equation (6)). Dual-gate conditions costs on offline difficulty and response correctness (Equation (7)). Online ALP scales raw-token costs by the current group solve rate, with floor $1/G$ (Equation (8)) (Xiang et al., 2025); we evaluate $\beta = 10^{-5}$ and 10^{-4}. These controls compare uniform, offline difficulty-conditioned, and online performance-conditioned costs under the same optimizer. Appendix B details their coefficients.

Metrics. With N problems and $K = 8$ samples each, accuracy is $\widehat{\mathrm{Acc}} = (NK)^{-1} \sum_{j,k} c(x_j, y_{jk})$ and mean length is $\widehat{L} = (NK)^{-1} \sum_{j,k} L_{\mathrm{eval}}(y_{jk})$, where L_{eval} counts returned generation token IDs. Token saving, in percent, is $100(1 - \overline{L}_{\mathrm{fixed}}/\overline{L}_{\mathrm{Vanilla}})$. Tokens per correct divides total generated tokens by total correct samples, measuring aggregate generation cost per success. For code, pass@1 is the mean success rate of one sampled program and pass@8 is the fraction of problems solved at least once among eight samples (Chen et al., 2021).

Replication and statistics. The primary Vanilla and fixed-length comparison uses eight paired training runs, and each adaptive arm in the main comparison also uses eight runs. Tables report run means and sample SD. Fixed-minus-Vanilla intervals use 5,000 nested paired bootstrap replicates, resampling training runs and then problems. Additional coefficient-sweep settings use two runs each and report their sample sizes explicitly. Appendix C gives paired tests and intervals; Appendix B lists complete run coverage.

4.2 MAIN RESULTS

A simple fixed cost delivers stable compression and a competitive accuracy–efficiency trade-off. At $\lambda = 0.3$, mean token use decreases by 26.9%, 12.0%, 2.8%, and 7.8% on GSM8K, MATH-500, AIME, and AMC (Table 1). Every one of the eight paired training runs shortens generations on every benchmark. Mean accuracy changes are -0.80, $+0.63$, $+0.21$, and $+2.11$ percentage points in the same order: three benchmarks improve on average, while GSM8K incurs a 0.80-point change. The paired uncertainty summaries appear in Table 6.

The cost per successful sample strengthens this result. Tokens per correct response fall by 26.2%, 12.7%, 3.8%, and 11.1% on GSM8K, MATH-500, AIME, and AMC, respectively (Table 6). On

Table 1: Mathematical performance under fixed and adaptive length costs. Every setting uses eight training seeds. The final row reports fixed-length token savings relative to paired Vanilla runs; paired intervals appear in Appendix C. Textual differences use unrounded means.

Method	**GSM8K**		**MATH-500**		**AIME**		**AMC**	
	Acc. ↑	Tokens ↓	Acc. ↑	Tokens ↓	Acc. ↑	Tokens ↓	Acc. ↑	Tokens ↓
Vanilla	$83.49_{\pm 0.26}$	$1273_{\pm 24}$	$75.45_{\pm 0.27}$	$3227_{\pm 37}$	$20.03_{\pm 0.33}$	$7156_{\pm 46}$	$56.68_{\pm 1.48}$	$5269_{\pm 61}$
Soft-budget	$\mathbf{83.82}_{\pm 0.87}$	$975_{\pm 37}$	$\mathbf{76.24}_{\pm 0.35}$	$\mathbf{2821}_{\pm 39}$	$19.65_{\pm 1.01}$	$6977_{\pm 54}$	$58.90_{\pm 0.84}$	$4866_{\pm 30}$
Dual-gate	$81.59_{\pm 2.83}$	$\mathbf{870}_{\pm 110}$	$75.95_{\pm 0.58}$	$2825_{\pm 43}$	$19.72_{\pm 1.54}$	$6990_{\pm 31}$	$\mathbf{59.26}_{\pm 0.90}$	$4851_{\pm 75}$
ALP (10^{-5})	$83.68_{\pm 0.96}$	$984_{\pm 44}$	$76.04_{\pm 0.45}$	$2856_{\pm 50}$	$20.26_{\pm 0.63}$	$\mathbf{6950}_{\pm 36}$	$58.58_{\pm 1.22}$	$\mathbf{4845}_{\pm 77}$
ALP (10^{-4})	$82.34_{\pm 2.73}$	$918_{\pm 148}$	$75.86_{\pm 0.52}$	$2871_{\pm 54}$	$\mathbf{20.62}_{\pm 1.28}$	$6952_{\pm 87}$	$58.92_{\pm 0.95}$	$4902_{\pm 116}$
Fixed-length	$82.69_{\pm 1.26}$	$931_{\pm 41}$	$76.09_{\pm 0.49}$	$2840_{\pm 32}$	$20.24_{\pm 0.91}$	$6956_{\pm 61}$	$58.79_{\pm 1.12}$	$4856_{\pm 76}$
Token saving	**26.9%**		**12.0%**		**2.8%**		**7.8%**	

Accuracy in %; subscripts: ± seed SD ($n = 8$). Bold: highest accuracy or shortest length in each column. The final row reports fixed-length token savings relative to Vanilla.

GSM8K, for example, this cost drops from 1,525 to 1,126 tokens despite the 0.80-point accuracy change. The fixed reward therefore reduces aggregate generation cost per success on all four benchmarks, including the setting with a modest accuracy trade-off.

Fixed-length attains 84.9–97.3% of the largest mean token saving from Vanilla among the tested adaptive controls, depending on the benchmark (Table 1; Appendix C). On MATH-500, soft-budget gains 0.16 pp and saves 19 tokens relative to fixed-length. On AIME, ALP at 10^{-4} gains 0.38 pp at a similar length; on AMC, dual-gate gains 0.47 pp and saves 5 tokens. GSM8K shows a clearer accuracy–length trade-off: soft-budget and ALP at 10^{-5} gain 1.13 and 1.00 pp while using 44 and 53 more tokens; dual-gate saves 61 tokens at a cost of 1.09 pp. These mean comparisons describe task-specific differences in accuracy, compression, and their balance across the tested reward settings.

4.3 Ablation and Sensitivity Analysis

Stronger nominal penalties yield diminishing compression, making accuracy and run-to-run variability important to the choice of reward. At $\lambda = 0.2$, GSM8K length falls from 1,273 to 994 tokens and MATH-500 from 3,227 to 2,858. Increasing λ to 0.5 gives 943 and 2,750 tokens; increasing it again to 0.8 gives 951 and 2,760 (Figure 2). Most shortening in this sweep therefore occurs when a length cost is introduced, while further coefficient increases mainly move the accuracy–length operating point.

Similar lengths can nevertheless accompany different accuracy outcomes. From $\lambda = 0.5$ to 0.8, AMC length changes from 4,740 to 4,765 tokens while accuracy rises from 59.04% to 59.71% (Table 5). The group-relative objective helps interpret why the nominal coefficient alone gives an incomplete description of the update: for equal-correctness groups, both centered rewards and reward SD scale with λ (Equations (3) and (5)). Normalization largely cancels this factor when reward SD is large relative to the numerical stabilizer. This algebra identifies group composition and normalization as relevant to effective length pressure, motivating joint measurement of accuracy and length across coefficient settings.

Cross-seed variability further separates rewards with similar average compression (Table 1). On GSM8K, fixed-length gives 931 ± 41 tokens, dual-gate 870 ± 110, and ALP at 10^{-4} 918 ± 148. The additional savings of 61 and 13 tokens accompany about 2.7 and 3.6 times the length SD. Soft-budget (975 ± 37) and ALP at 10^{-5} (984 ± 44) instead occupy the higher-accuracy, longer-response region with modest length variation. These results position the fixed setting as a simple intermediate choice and show that adaptive design changes accuracy, compression, and variability together.

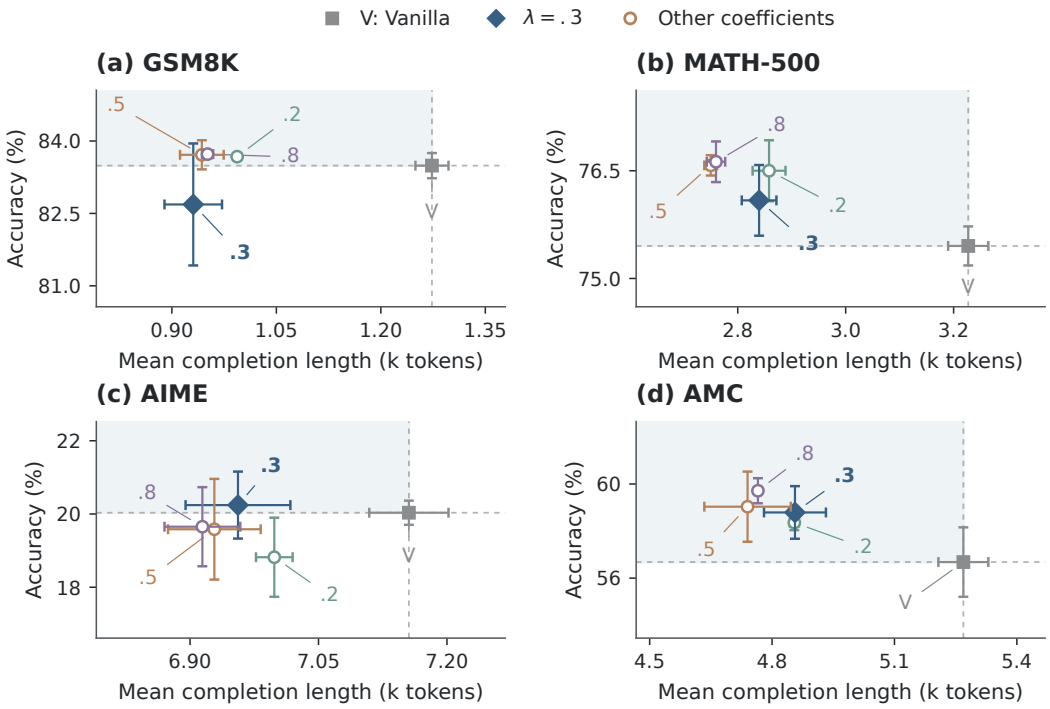


Figure 2: Accuracy–length operating points under fixed coefficients. Labels give λ; V denotes Vanilla. Filled markers use eight seeds and open markers use two. Horizontal and vertical whiskers show seed SD. Dashed lines mark the Vanilla means; shading indicates shorter generations with higher accuracy relative to those means.

4.4 Analysis of Reasoning Behavior

A shared cost produces a structured compression profile: the largest absolute savings occur on intermediate problems and the largest proportional savings on easy problems. Difficulty groups use Vanilla solve rates averaged over training seeds: easy at ≥ 0.75, hard at ≤ 0.25, and intermediate otherwise. All twelve benchmark–group means shorten. Intermediate groups save 462, 529, 432, and 520 tokens per response on GSM8K, MATH-500, AIME, and AMC; easy groups shorten by 29.0%, 16.9%, 8.6%, and 16.6% (Figure 3). Thus, difficulty-dependent compression is present even under a reward with a shared coefficient, providing a behavior reference for explicitly adaptive costs.

Hard problems account for most of the remaining generation volume on AIME. Its hard group contains 66 of 90 problems and still averages 7,765 tokens, close to the 8,192-token limit, after a 1.7% reduction. This group contributes 81.9% of post-training tokens, locating the main remaining cost behind the modest aggregate saving of 2.8%. Within the Vanilla-defined groups, intermediate accuracy changes range from +0.89 to +4.69 pp, while easy GSM8K and AIME change by -1.49 and -3.91 pp. These conditional outcomes motivate measuring where accuracy changes accompany savings; group measurements and selection effects are detailed in Table 7 and Appendix D.

Shorter responses also exhibit less local repetition and more frequent completion within the same decoding budget (Table 2). On GSM8K, repeated trigrams fall from 30.55% to 28.18%, word 5-grams from 9.67% to 8.56%, and marker density from 3.92 to 3.08 per 1,000 characters. Truncation falls from 8.26% to 4.50%, or 3.76 pp; reductions on MATH-500, AIME, and AMC are 2.16, 4.22, and 5.06 pp. Sentence repetition changes less, including 2.83% to 2.85% on AMC. The normalized repetition measures associate compression with differences in output composition as well as volume. Together with the difficulty profile, they specify concrete behaviors on which fixed and adaptive reward designs can be compared. Metric definitions and training dynamics appear in Appendix D.

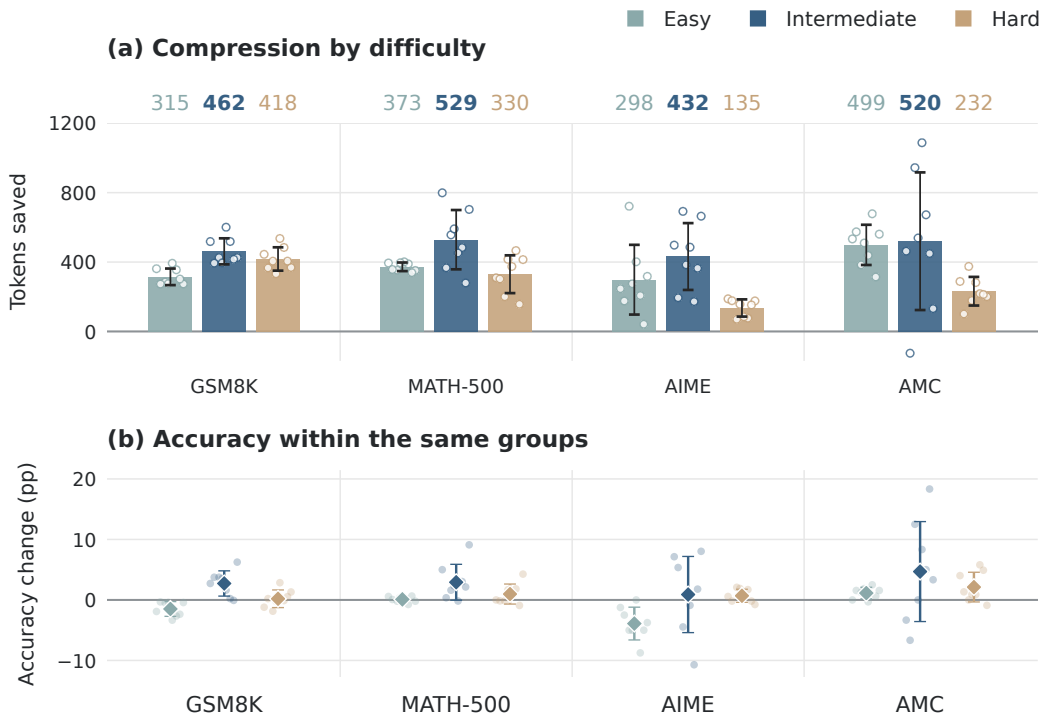


Figure 3: Difficulty-conditioned changes under the fixed penalty. (a) Mean tokens saved per response, with paired seed SD. (b) Mean accuracy change in pp, with paired seed SD. Small points show all eight seed pairs; colors identify shared Vanilla-defined difficulty groups. Intermediate groups have the largest absolute mean savings in every benchmark. Accuracy comparisons are conditional on the grouping protocol (Appendix D).

Table 2: Generation behavior under a fixed length cost. Repetition and truncation are percentages. Appendix D gives metric definitions and paired seed variation.

		Repetition (%)			per 1k chars	(%)
Benchmark	Method	3-gram	5-gram	Sentence	Markers	Trunc.
GSM8K	Vanilla	30.55	9.67	1.39	3.92	8.26
	Fixed-length	**28.18**	**8.56**	**1.16**	**3.08**	**4.50**
MATH-500	Vanilla	44.93	12.12	2.02	4.00	13.51
	Fixed-length	**43.74**	**11.86**	**1.96**	**3.67**	**11.34**
AIME	Vanilla	56.37	17.36	2.84	5.29	69.83
	Fixed-length	**55.96**	**17.10**	**2.76**	**5.09**	**65.61**
AMC	Vanilla	53.87	15.60	**2.83**	4.53	37.71
	Fixed-length	**53.05**	**15.30**	2.85	**4.21**	**32.64**

Means across eight paired seeds. Bold: lower mean within each benchmark–metric pair. Shading: fixed-length. Markers are counted per 1,000 characters.

4.5 Extensions to Code and Larger Models

The code extension shows positive cross-task transfer and greater objective sensitivity during task-specific post-training. We first evaluate the math-trained Vanilla and fixed-cost checkpoints directly on HumanEval and MBPP-Sanitized, then separately compare correctness-only, ungated length, and correctness-gated length rewards during MBPP-Sanitized post-training. Evaluation cov-

ers 164 HumanEval and 257 MBPP-Sanitized problems, with eight sampled programs per problem (Austin et al., 2021; Chen et al., 2021). Appendix E provides the training and evaluation protocols.

Table 3: Code transfer and task-specific post-training. Entries are mean success rates across eight training runs; the two phases use separate Vanilla references.

Setting	HumanEval		MBPP-Sanitized	
	pass@1 ↑	pass@8 ↑	pass@1 ↑	pass@8 ↑
Transfer: Vanilla	$36.89_{\pm 0.86}$	$75.38_{\pm 1.22}$	$34.08_{\pm 0.95}$	$64.25_{\pm 1.59}$
Transfer: Fixed	$\mathbf{38.95}_{\pm 0.86}$	$\mathbf{75.84}_{\pm 1.42}$	$\mathbf{35.91}_{\pm 1.17}$	$\mathbf{65.37}_{\pm 1.40}$
Post-train: Vanilla	$\mathbf{44.65}_{\pm 3.66}$	$\mathbf{78.43}_{\pm 2.42}$	$35.98_{\pm 7.71}$	$\mathbf{63.52}_{\pm 2.93}$
Post-train: Ungated	$0.44_{\pm 1.24}$	$2.44_{\pm 6.90}$	$0.46_{\pm 1.27}$	$2.14_{\pm 5.90}$
Post-train: Gated	$28.81_{\pm 8.22}$	$54.12_{\pm 13.39}$	$\mathbf{36.08}_{\pm 7.86}$	$56.37_{\pm 9.25}$

Entries: mean$_{\pm \mathrm{SD}}$ (%) across eight training runs. Transfer evaluates the math-trained checkpoints directly; post-train uses MBPP-Sanitized training. Bold denotes the larger mean within each phase.

The math-trained fixed-cost checkpoint transfers favorably: HumanEval and MBPP-Sanitized pass@1 increase by 2.06 and 1.82 percentage points while mean generation length decreases by 4.9% and 4.6%, respectively (Table 3). Pass@8 changes are smaller, localizing the main transfer benefit to single-sample success and generation cost. Task-specific post-training shows a different pattern. The ungated objective yields near-zero success on both benchmarks; correctness gating recovers MBPP pass@1 to 36.08% versus 35.98% for Vanilla, while pass@8 is 56.37% versus 63.52%. On HumanEval, gated pass@1/pass@8 are 28.81%/54.12%, compared with 44.65%/78.43% for Vanilla. The paired comparisons appear in Table 8. These results show that fixed-cost math training can transfer accuracy–efficiency benefits to code, while direct code post-training requires reward designs that preserve both correctness and sampling coverage.

At 7B scale, generation-budget saturation dominates both objectives: across 3,072–32,768-token evaluation budgets, about 99.5–99.9% of responses reach the limit under both Vanilla and fixed-length (Table 9). The persistence of this pattern across a tenfold budget range shows that increasing the inference allowance alone does not induce reliable termination. This boundary result motivates pairing length-aware rewards with termination-aware training at larger scales. Together with the code transfer gains, the extensions show where the fixed-cost principle carries over directly and where task-specific correctness and completion behavior become the central design targets.

5 Discussion and Conclusion

A simple fixed length cost delivers a strong accuracy–efficiency trade-off in the evaluated mathematical RLVR setting. Across four benchmarks and eight paired training seeds, it consistently shortens generations; mean accuracy improves on three benchmarks, while the remaining GSM8K change is -0.80 percentage points. Tokens per correct response decrease on all four benchmarks. These gains require only a scalar reward change within the common sampling and optimization procedure.

Without difficulty estimates, per-problem budgets, or online reward adjustment, the fixed cost captures 84.9–97.3% of the largest mean token saving achieved by the tested adaptive controls. Most observable compression therefore arises from a transparent uniform objective, while adaptive rewards provide task-specific refinements in accuracy, length, and variability. Difficulty analysis further localizes the opportunity for improvement: intermediate problems contribute the largest absolute savings, easy problems shorten most proportionally, and difficult AIME problems dominate the residual generation volume.

Limitations. Future work can extend the comparison to larger models, broader training corpora, and additional reasoning domains, and can combine a uniform cost with adaptive control when difficult problems require more targeted allocation. Our primary efficiency measure is generated tokens; deployment-oriented evaluation should additionally measure end-to-end latency, throughput, memory use, energy, and monetary cost. Appendix G documents the reproducibility materials.

AI Assistance Disclosure

Generative AI tools were used to improve the clarity and organization of the manuscript, support Chinese–English translation and literature organization, suggest limited code edits and debugging steps, and cross-check reported numbers against the authors' experimental records. The authors evaluated, implemented, and tested all code changes. The experiments were run by the authors, who made the final decisions on methodology, interpretation, and scientific claims and take full responsibility for the final content.

References

Pranjal Aggarwal and Sean Welleck. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *arXiv preprint arXiv:2503.04697*, 2025. doi: 10.48550/arXiv.2503.04697. URL https://arxiv.org/abs/2503.04697.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024. doi: 10.48550/arXiv.2402.14740. URL https://arxiv.org/abs/2402.14740.

Daman Arora and Andrea Zanette. Training Language Models to Reason Efficiently. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 60770–60808, 2025. doi: 10.52202/085713-2032. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/579b5b84311e122584dedab1d3b7613f-Paper-Conference.pdf.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*, 2021. doi: 10.48550/arXiv.2108.07732. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374. URL https://arxiv.org/abs/2107.03374.

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 Long Reasoning Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 9487–9499, 2025. URL https://proceedings.mlr.press/v267/chen25bx.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025. doi: 10.48550/arXiv.2501.12948. URL https://arxiv.org/abs/2501.12948.

Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *arXiv preprint arXiv:1603.08983*, 2016. doi: 10.48550/arXiv.1603.08983. URL https://arxiv.org/abs/1603.08983.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-Budget-Aware LLM Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training Large Language Models to Reason in a Continuous Latent Space. *arXiv preprint arXiv:2412.06769*, 2024. doi: 10.48550/arXiv.2412.06769. URL https://arxiv.org/abs/2412.06769.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. Don't Overthink it. Preferring Shorter Thinking Chains for Improved LLM Reasoning. *arXiv preprint arXiv:2505.17813*, 2025. doi: 10.48550/arXiv.2505.17813. URL https://arxiv.org/abs/2505.17813.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv preprint arXiv:2103.03874*, 2021. doi: 10.48550/arXiv.2103.03874. URL https://arxiv.org/abs/2103.03874.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. ThinkPrune: Pruning Long Chain-of-Thought of LLMs via Reinforcement Learning. *arXiv preprint arXiv:2504.01296*, 2025. doi: 10.48550/arXiv.2504.01296. URL https://arxiv.org/abs/2504.01296.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, 2021. doi: 10.48550/arXiv.2106.09685. URL https://arxiv.org/abs/2106.09685.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing Critic-Free Policy Optimization with Global Advantage Normalization. *arXiv preprint arXiv:2501.03262*, 2025. doi: 10.48550/arXiv.2501.03262. URL https://arxiv.org/abs/2501.03262.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, et al. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025. doi: 10.48550/arXiv.2501.12599. URL https://arxiv.org/abs/2501.12599.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. *arXiv preprint arXiv:2205.11916*, 2022. doi: 10.48550/arXiv.2205.11916. URL https://arxiv.org/abs/2205.11916.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, et al. Training Language Models to Self-Correct via Reinforcement Learning. *arXiv preprint arXiv:2409.12917*, 2024. doi: 10.48550/arXiv.2409.12917. URL https://arxiv.org/abs/2409.12917.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, et al. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*, 2023. doi: 10.48550/arXiv.2305.20050. URL https://arxiv.org/abs/2305.20050.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective. *arXiv preprint arXiv:2503.20783*, 2025. doi: 10.48550/arXiv.2503.20783. URL https://arxiv.org/abs/2503.20783.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-Pruner: Length-Harmonizing Fine-Tuning for O1-Like Reasoning Pruning. *arXiv preprint arXiv:2501.12570*, 2025. doi: 10.48550/arXiv.2501.12570. URL https://arxiv.org/abs/2501.12570.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, 2025. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, et al. Show Your Work: Scratchpads for Intermediate Computation with Language Models. *arXiv preprint arXiv:2112.00114*, 2021. doi: 10.48550/arXiv.2112.00114. URL https://arxiv.org/abs/2112.00114.

Xiaoye Qu, Yafu Li, Zhao-Chen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, et al. A Survey of Efficient Reasoning for Large Reasoning Models: Language, Multimodality, and Beyond. *arXiv preprint arXiv:2503.21614*, 2025. doi: 10.48550/arXiv.2503.21614. URL https://arxiv.org/abs/2503.21614.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv preprint arXiv:2305.18290*, 2023. doi: 10.48550/arXiv.2305.18290. URL https://arxiv.org/abs/2305.18290.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347. URL https://arxiv.org/abs/1707.06347.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident Adaptive Language Modeling. *arXiv preprint arXiv:2207.07061*, 2022. doi: 10.48550/arXiv.2207.07061. URL https://arxiv.org/abs/2207.07061.

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. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, et al. DAST: Difficulty-Adaptive Slow-Thinking for Large Reasoning Models. *arXiv preprint arXiv:2503.04472*, 2025. doi: 10.48550/arXiv.2503.04472. URL https://arxiv.org/abs/2503.04472.

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 *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b623663fd9b874366f3ce019fdfdd44-Paper-Conference.pdf.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, et al. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. doi: 10.48550/arXiv.2211.14275. URL https://arxiv.org/abs/2211.14275.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. *arXiv preprint arXiv:2312.08935*, 2023. doi: 10.48550/arXiv.2312.08935. URL https://arxiv.org/abs/2312.08935.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint arXiv:2203.11171*, 2022. doi: 10.48550/arXiv.2203.11171. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903*, 2022. doi: 10.48550/arXiv.2201.11903. URL https://arxiv.org/abs/2201.11903.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural Text Generation with Unlikelihood Training. *arXiv preprint arXiv:1908.04319*, 2019. doi: 10.48550/arXiv.1908.04319. URL https://arxiv.org/abs/1908.04319.

Yuyang Wu, Qiyao Xue, Guanxing Lu, Weichen Liu, Zihan Wang, Manling Li, and Olexandr Isayev. CLORE: Content-Level Optimization for Reasoning Efficiency. *arXiv preprint arXiv:2605.22211*, 2026. doi: 10.48550/arXiv.2605.22211. URL https://arxiv.org/abs/2605.22211.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. TokenSkip: Controllable Chain-of-Thought Compression in LLMs. *arXiv preprint arXiv:2502.12067*, 2025. doi: 10.48550/arXiv.2502.12067. URL https://arxiv.org/abs/2502.12067.

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. *arXiv preprint arXiv:2506.05256*, 2025. doi: 10.48550/arXiv.2506.05256. URL https://arxiv.org/abs/2506.05256.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, et al. Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement. *arXiv preprint arXiv:2409.12122*, 2024. doi: 10.48550/arXiv.2409.12122. URL https://arxiv.org/abs/2409.12122.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Jingyang Yi, Jiazheng Wang, and Sida Li. ShorterBetter: Guiding Reasoning Models to Find Optimal Inference Length for Efficient Reasoning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 39011–39043, 2025. doi: 10.52202/085713-1302. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/377ca194437916417e3dbe0dc67254f1-Paper-Conference.pdf.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025. doi: 10.48550/arXiv.2503.14476. URL https://arxiv.org/abs/2503.14476.

Zewei Yu, Lirong Gao, Yuke Zhu, Bo Zheng, Junbo Zhao, Sheng Guo, and Haobo Wang. Stop Unnecessary Reflection: Training LRMs for Efficient Reasoning with Adaptive Reflection and Length Coordinated Penalty. In *International Conference on Learning Representations*, volume 2026, pp. 4836–4854, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/08fe50bf209c57eecf0804f9f9ed639f-Paper-Conference.pdf.

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, 2026. doi: 10.18653/v1/2026.findings-acl.626. URL https://aclanthology.org/2026.findings-acl.626/.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning. *arXiv preprint arXiv:2203.14465*, 2022. doi: 10.48550/arXiv.2203.14465. URL https://arxiv.org/abs/2203.14465.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. AdaptThink: Reasoning Models Can Learn When to Think. *arXiv preprint arXiv:2505.13417*, 2025a. doi: 10.48550/arXiv.2505.13417. URL https://arxiv.org/abs/2505.13417.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, et al. LightThinker: Thinking Step-by-Step Compression. *arXiv preprint arXiv:2502.15589*, 2025b. doi: 10.48550/arXiv.2502.15589. URL https://arxiv.org/abs/2502.15589.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft Thinking: Unlocking the Reasoning Potential of LLMs in Continuous Concept Space. *arXiv preprint arXiv:2505.15778*, 2025c. doi: 10.48550/arXiv.2505.15778. URL https://arxiv.org/abs/2505.15778.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, et al. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. *arXiv preprint arXiv:2205.10625*, 2022. doi: 10.48550/arXiv.2205.10625. URL https://arxiv.org/abs/2205.10625.

A Optimization and Evaluation Protocol

A.1 Training Configuration

The primary reward comparison shares the sampling and optimization configuration in Table 4. Each single-device batch contains one prompt with eight completions and one optimization step. Rollouts use colocated vLLM in bfloat16, with full-parameter updates. Software versions are TRL 0.24.0, Transformers 4.57.3, PyTorch 2.8.0, Datasets 4.8.5, and Tokenizers 0.22.2. Every primary arm is evaluated at checkpoint 400.

Table 4: Primary mathematical training and evaluation protocol for the 1.5B model. Batch size counts completions. Training and evaluation use different sampling settings.

Item	Setting
Training	
Model / data	DeepSeek-R1-Distill-Qwen-1.5B / GSM8K train (7,473)
Updates / selected checkpoint	400 / 400
Learning rate	10^{-5}
Samples per prompt / batch / accumulation	8 / 8 / 1
Prompt / completion limit	512 / 2,048 tokens
Temperature / top-p	1.0 / 1.0
Optimization / reward scaling	DAPO token aggregation / group SD
KL coefficient / clipping / sampling correction	0 / 0.2 / enabled, capped at 2.0
Adaptation / precision	Full parameters / bfloat16
Truncated samples	Retained in the loss
Evaluation	
Checkpoint / samples per problem	Final checkpoint / 8
Temperature / top-p	0.6 / 0.95
GSM8K: problems / generation limit	1,319 / 3,072 tokens
MATH-500: problems / generation limit	500 / 8,192 tokens
AIME: problems / generation limit	90 / 8,192 tokens
AMC: problems / generation limit	83 / 8,192 tokens
Primary training seeds	45–52 for both Vanilla and fixed-length

Batch size counts completions. Evaluation uses a common decoding protocol across reward arms.

A.2 Optimization Objective

Equation (4) normalizes the summed token losses by the total number of active completion tokens in the optimization batch. Group scaling uses the sample reward standard deviation in Equation (3). These choices correspond to `loss_type=dapo` and `scale_rewards=group` in the saved configurations.

Here m_{it} masks padding, and ρ_{it} is the current-to-old policy probability ratio at token t. The detached vLLM correction is $w_{it} = \min\{2, \pi_{\text{old}}(y_{it} \mid x, y_{i,<t})/\pi_{\text{vLLM}}(y_{it} \mid x, y_{i,<t})\}$. Clipping uses $\epsilon = 0.2$, the KL coefficient is zero, and each generated batch receives one optimization iteration. Truncated completions are retained. These conventions follow the versioned trainer implementation.[1] The primary arms use the same token aggregation and sampling correction.

A.3 Data and Verification

Dataset identifiers are `openai/gsm8k` (configuration `main`), `HuggingFaceH4/MATH-500`, `AI-MO/aimo-validation-aime`, and `AI-MO/aimo-validation-amc`. AIME and AMC refer to these specific collections, whose `train` splits are used only for evaluation. Dataset revisions are not pinned in the launch scripts. Mathematical prompts use the model's chat template

[1] `https://github.com/huggingface/trl/tree/v0.24.0/trl/trainer`

and a shared instruction to reason step by step and provide a boxed final answer. Evaluation prompts are identical across reward arms, including soft-budget.

The mathematical verifier extracts boxed answers, with a last-number fallback for GSM8K, and accepts normalized string, numeric, or symbolic equivalence. Correctness is scored from the extracted answer, including for length-stopped completions. Reward length re-encodes the decoded full completion with special tokens excluded; evaluation length counts returned generation token IDs. These definitions specify the token accounting used at each stage. Accuracy averages all eight sampled correctness labels per problem and estimates single-sample success probability.

B Reward Designs and Coefficient Sensitivity

B.1 Difficulty-Dependent Rewards

Offline difficulty. Let $p_0(x)$ be the solve rate from eight base-model responses to each GSM8K training problem. Define $d(x)$ as easy when $p_0 \geq 0.75$, hard when $p_0 \leq 0.25$, and intermediate otherwise. These training estimates are separate from the post-training evaluation groups in Figure 3. Soft-budget assigns budgets $b_d = (512, 1024, 2048)$ in easy/intermediate/hard order and uses

$$r_{\text{soft}} = c - 0.1 \, \frac{\max(0, L - b_d)}{b_d}. \tag{6}$$

The experiment identifier `prompt_budget_matched` denotes this reward-based soft budget; all reward arms use the shared mathematical prompt. Dual-gate uses $\lambda_d = (0.5, 0.1, 0.1)$ in easy/intermediate/hard order and applies an excess-length cost to failed responses:

$$r_{\text{dual}} = c(1 - \lambda_d s_B) - (1 - c)\, 0.1 \max(0, s_B - 0.5). \tag{7}$$

B.2 Online Adaptive Length Penalty

Online solve rate. The ALP control uses $\hat{p} = G^{-1} \sum_j c_j$ from the current group and a floor of $1/G$ in its raw-token penalty:

$$r_{\text{ALP},i} = c_i - \beta_{\text{ALP}} L_i \max(\hat{p}, 1/G). \tag{8}$$

ALP weights raw token length by the current solve rate with a floor of $1/G$. The floor preserves a positive length coefficient in all-incorrect groups. Fixed-length uses a capped, normalized cost. Accordingly, each family has its own coefficient scale. Throughout the ALP comparisons, β denotes the length coefficient and the trainer KL coefficient remains zero.

B.3 Coefficient Coverage

The sweep includes both the no-cost ablation and stronger length pressure, with replication concentrated at the primary setting. Vanilla ($\lambda = 0$) and $\lambda = 0.3$ use seeds 45–52; $\lambda = 0.2, 0.5, 0.8$ use seeds 45–46. ALP uses one seed at 10^{-7}, three at 10^{-6}, and eight at each of 10^{-5} and 10^{-4}; the latter settings use seeds 42–49. Soft-budget and dual-gate use seeds 53–60. Table 5 reports all coefficient means with explicit seed counts.

The coefficient response depends on the benchmark and the reward scale. Within the fixed family, increasing the nominal penalty can increase or decrease the mean generated length. Equation (5) identifies two relevant quantities for interpreting this response: within-group length variation and the reward standard deviation. The sweep measures the resulting operating points with a common learning rate of 10^{-5}. The primary setting $\lambda = 0.3$ has the largest replication among the nonzero fixed coefficients; coefficient selection was exploratory and used no common held-out selection set.

Table 5: Complete coefficient sensitivity measurements. Accuracy is in percent; token counts are means. The seed count n applies to every benchmark in its row. Replication is concentrated at $\lambda = 0$ and 0.3 and at the two larger ALP coefficients.

		GSM8K		MATH-500		AIME		AMC	
Coefficient	n	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens
Fixed penalty									
$\lambda = 0$	8	83.49	1273	75.45	3227	20.03	7156	56.68	5269
$\lambda = 0.2$	2	83.68	994	76.50	2858	18.82	6998	58.36	4855
$\lambda = 0.3$	8	82.69	931	76.09	2840	20.24	6956	58.79	4856
$\lambda = 0.5$	2	83.71	943	76.58	2750	19.58	6928	59.04	4740
$\lambda = 0.8$	2	83.73	951	76.62	2760	19.65	6914	59.71	4765
Online ALP									
$\beta = 10^{-7}$	1	83.51	1222	75.60	3150	19.44	7129	56.78	5110
$\beta = 10^{-6}$	3	83.95	1115	75.96	2996	19.49	7052	57.73	5053
$\beta = 10^{-5}$	8	83.68	984	76.04	2856	20.26	6950	58.58	4845
$\beta = 10^{-4}$	8	82.34	918	75.86	2871	20.62	6952	58.92	4902

Accuracy in percent; n counts training seeds. Fixed λ multiplies capped, normalized length; ALP β multiplies raw length and a group-dependent factor.

C Statistical Analysis and Efficiency Measures

C.1 Paired Estimators and Uncertainty

The primary effect is the average within-seed difference, preserving the matched comparison at both the seed and problem levels. For each seed and problem, we first average the eight correctness labels and token counts. The benchmark estimate then averages problems within each seed and weights the eight training seeds equally. Accuracy differences are reported in percentage points, and relative token saving is computed from the unrounded grand means.

Nested paired resampling propagates variation across training seeds and evaluated problems. Each bootstrap replicate draws eight seeds with replacement; within each selected seed, it draws N problems with replacement and uses the same indices for the two arms. Each selected problem retains its eight-sample mean. We use 5,000 replicates and percentile 95% intervals, with random seed 20260908 plus the benchmark index in GSM8K, MATH-500, AIME, AMC order. Problem indices are drawn independently for each selected seed.

Accuracy effects are summarized by the mean paired difference, pointwise nested-bootstrap intervals, and paired-seed tests with Holm correction over the four benchmarks. The mean changes on GSM8K, MATH-500, AIME, and AMC are -0.80, $+0.63$, $+0.21$, and $+2.11$ pp; the corresponding Holm-adjusted p values are 0.172, 0.067, 0.507, and 0.052. Length effects are more uniform: every paired run is shorter on all four benchmarks, and all length intervals lie below zero (Table 6). Together, these summaries characterize a consistent reduction in generation cost and benchmark-dependent changes in accuracy.

C.2 Token Efficiency

Tokens per correct decreases on all four benchmarks, including GSM8K. This ratio divides the total generation volume by the total number of correct sampled completions. Its reduction indicates that aggregate token savings exceed the relative accuracy cost where that cost is positive. Together with mean length and accuracy, it provides a complementary summary of the sampled evaluation workload. Runtime, energy, and monetary efficiency require corresponding system-level measurements.

The share of adaptive token savings compares mean length reductions from the same Vanilla reference. Let $\mathcal{A}$ contain the soft-budget, dual-gate, and two ALP settings in Table 1. For each benchmark, we compute $100(\overline{L}_V - \overline{L}_F)/(\overline{L}_V - \min_{a\in\mathcal{A}} \overline{L}_a)$, expressed as a percentage. The shares are

Table 6: Paired effects, uncertainty, and aggregate token efficiency. Differences are fixed-length minus Vanilla. Pointwise 95% intervals use nested paired bootstrap; nominal tests use eight paired seed means. Tokens per correct is the ratio of grand mean length to grand mean accuracy.

	Accuracy (pp)			Tokens			Tokens / correct	
Benchmark	Δ	95% CI	p	Δ	95% CI	p	Vanilla	Fixed
GSM8K	-0.80	[-1.60, -0.01]	0.0861	-343	[-379, -310]	3.4×10^{-7}	1525	1126
MATH-500	+0.63	[+0.09, +1.19]	0.0224	-388	[-422, -350]	6.8×10^{-8}	4277	3732
AIME	+0.21	[-0.87, +1.28]	0.5072	-200	[-256, -145]	1.1×10^{-5}	35716	34362
AMC	+2.11	[+0.55, +3.80]	0.0131	-413	[-491, -333]	3.5×10^{-6}	9295	8260

Differences: fixed-length minus Vanilla. CI: pointwise hierarchical paired bootstrap; p: nominal two-sided paired seed t test ($n = 8$). Tokens per correct: ratio of grand means.

84.9%, 95.3%, 97.1%, and 97.3% on GSM8K, MATH-500, AIME, and AMC. This descriptive ratio summarizes mean token savings across the tested reward settings; accuracy and seed variation are evaluated separately in Table 1.

D Reasoning Behavior and Training Dynamics

D.1 Difficulty-Conditioned Measurements

Intermediate groups have the largest absolute token savings on every benchmark. For each problem, we average Vanilla correctness over all eight seeds and their eight samples. Easy problems have accuracy ≥ 0.75, hard problems have accuracy ≤ 0.25, and the remainder are intermediate. Both reward arms and every seed share these groups (Table 7). This grouping describes post-training evaluation behavior; the adaptive controls use separately estimated base-model training difficulty.

Table 7: Accuracy and mean generation length within difficulty groups, averaged across eight training seeds. N is the number of problems in each shared group. Accuracy differences are fixed-length minus Vanilla (pp); tokens saved are Vanilla minus fixed-length. Groups are defined from Vanilla outcomes, so conditional accuracy comparisons can reflect regression to the mean.

			Accuracy (%)			Mean tokens		
Benchmark	Difficulty	N	Vanilla	Fixed	Δ pp	Vanilla	Fixed	Saved
GSM8K	Easy	1035	96.52	95.02	-1.49	1085	771	315
	Intermediate	170	54.20	56.92	+2.72	1869	1408	462
	Hard	114	8.91	9.10	+0.19	2091	1673	418
MATH-500	Easy	348	96.79	96.89	+0.09	2201	1829	373
	Intermediate	70	53.12	56.05	+2.92	5670	5141	529
	Hard	82	3.94	4.92	+0.97	5496	5166	330
AIME	Easy	10	90.62	86.72	-3.91	3478	3180	298
	Intermediate	14	45.98	46.87	+0.89	6270	5839	432
	Hard	66	3.84	4.52	+0.69	7900	7765	135
AMC	Easy	40	94.77	95.90	+1.13	3014	2514	499
	Intermediate	15	44.79	49.48	+4.69	6585	6065	520
	Hard	28	8.65	10.77	+2.12	7785	7554	232

Bins are shared between arms and defined from Vanilla outcomes; accuracy changes are descriptive.

Group contributions to generation volume weight mean response length by the number of problems. With N_d problems and mean length $\overline{L}_{F,d}$ in group d, its share of fixed-length tokens is $N_d\overline{L}_{F,d}/\sum_{d'} N_{d'}\overline{L}_{F,d'}$. Each problem has the same sample and seed coverage. On AIME, the hard group contains 66 of 90 problems and contributes 81.9% of fixed-length generation tokens.

Weighted group means recover the benchmark mean, connecting the difficulty profile to aggregate token use.

Conditioning on Vanilla outcomes introduces selection effects into within-group accuracy comparisons. Selecting high-accuracy problems can produce a subsequent decrease relative to the selected baseline through regression to the mean; low-accuracy selection can produce the reverse pattern. The easy AIME group contains ten problems. We therefore interpret the groups as descriptive strata for locating compression. Independent difficulty annotations or a separate grouping sample would support a follow-up test of the conditional accuracy pattern.

D.2 Generation Metrics

Normalized repetition and marker frequencies decrease across the evaluated benchmarks. Table 2 reports lower repeated trigrams, word 5-grams, lexical markers, and truncation for every benchmark. Repeated-sentence frequency decreases on three benchmarks and rises slightly on AMC. Figure 4 displays the paired seed differences. The normalized measures describe the composition of generated text alongside its overall length.

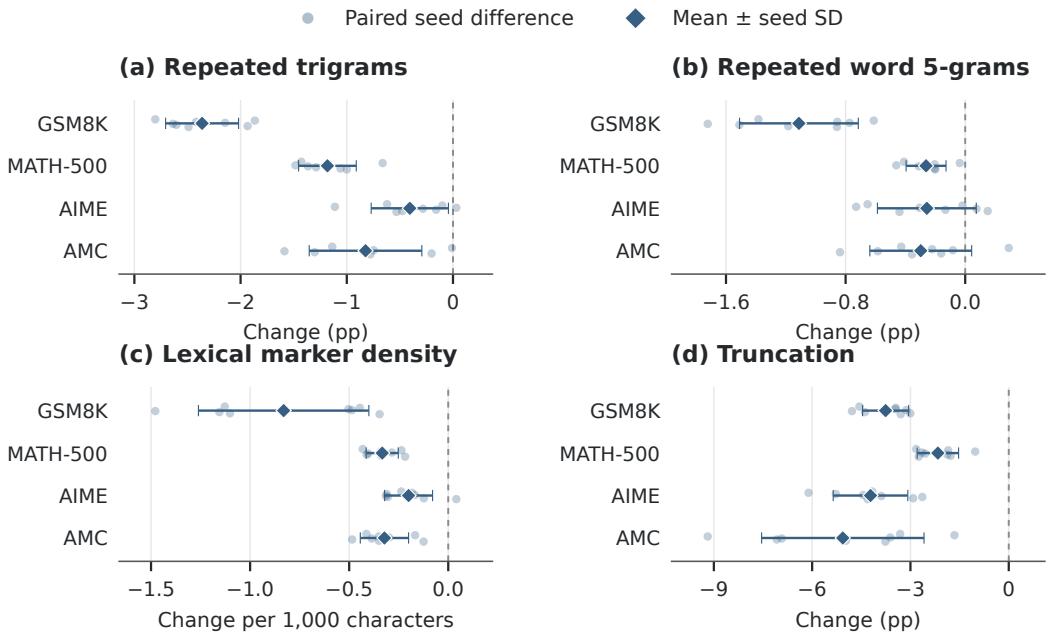


Figure 4: Paired changes in generation behavior across eight training seeds. Points show fixed-length minus Vanilla for each seed; diamonds and whiskers show the mean and sample SD. Repetition and truncation differences are in pp, and marker differences are per 1,000 characters. Table 2 reports the corresponding levels.

Repetition metrics count duplicate surface forms at three granularities. Trigram repetition is $1 - U_3/M_3$, where M_3 counts lowercased word-or-punctuation trigrams and U_3 counts distinct values. Word 5-gram repetition divides duplicate occurrences by all word 5-grams. Sentence repetition is the fraction of duplicate normalized segments longer than twenty characters, split at line breaks or sentence boundaries. Each metric is computed per completion before aggregation.

Lexical marker density measures the frequency of specified surface expressions. We count case-insensitive matches to wait, but, however, reconsider, alternatively, let me, double-check/double check, check again, and on the other hand, per 1,000 characters. The list includes ordinary discourse markers and expressions used during reconsideration, so its interpretation concerns lexical frequency. Truncation uses the generator's length-stop flag. Under shared evaluation caps, a decrease indicates that more responses terminate within the available generation budget.

D.3 Training Dynamics

Length-penalized training ends with shorter rollouts, lower entropy, and fewer truncated completions. Over updates 351–400, unsmoothed mean rollout lengths are 1,346 for Vanilla, 925 for fixed-length, 995 for ALP, and 870 for dual-gate; corresponding truncation rates are 22.7%, 10.1%, 10.5%, and 8.7%. Figure 5 shows the trajectories across eight seeds per reward under the 2,048-token training completion limit.

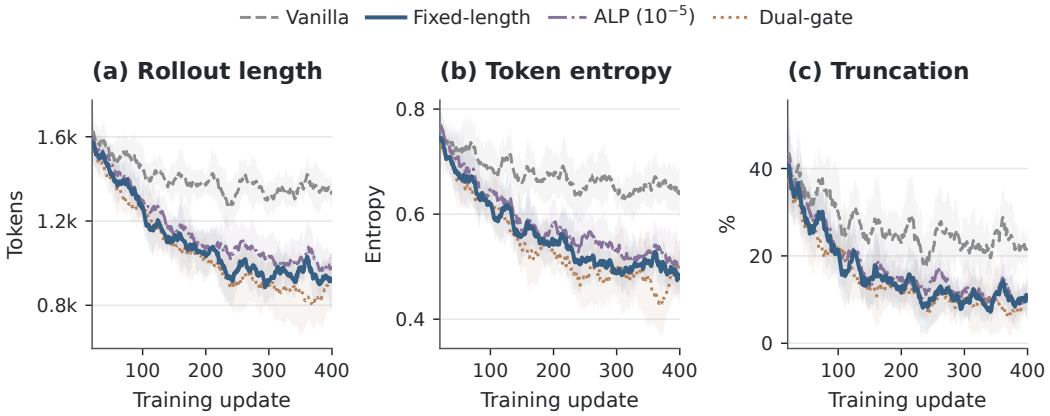


Figure 5: Training dynamics for four reward settings. Each seed is smoothed by a trailing 20-update mean; lines and bands show the mean and seed SD of the smoothed values. ALP uses 10^{-5}. The logged field `completions/clipped_ratio` is the proportion of completions truncated by the generation limit.

The trajectories show that the shorter final generations are accompanied by changes throughout policy optimization. Lower entropy describes a more concentrated token distribution, while lower truncation describes more frequent completion within the training budget. These observations complement the held-out length and correctness measurements. Connecting individual trajectory changes to downstream correctness is a direction for intervention-based analysis.

E Code Task Protocol and Additional Results

E.1 Training and Verification

The transfer study evaluates math-trained checkpoints from seeds 45–52 on HumanEval and MBPP-Sanitized. The task-specific study initializes from the original 1.5B model and trains on 120 MBPP-Sanitized problems for 400 updates, using learning rate 10^{-5}, batch size eight, eight rollouts per prompt, a 768-token prompt limit, a 2,048-token completion limit, and zero KL coefficient. Prompts request executable Python and include the canonical function signature. Evaluation draws eight samples per problem with temperature 0.6, top-p 0.95, and a 2,048-token completion limit. Training and evaluation use task-specific prompt wrappers, while all methods within each comparison are evaluated using the same prompt and execution protocol. The code study is included as an auxiliary extension under this common evaluation protocol.

The post-training comparison uses correctness-only c_{test}, ungated $c_{\text{test}} - 0.3s_{2048}$, and correctness-gated $c_{\text{test}}(1 - 0.3s_{2048})$ rewards. Length is measured on the complete sampled response. The verifier extracts the final Python program, appends task imports and assertions, and executes it in a fresh resource-limited subprocess. All configurations are scored with the same verifier. The ungated objective strongly suppresses generation but also collapses task success; correctness gating preserves substantially more functional behavior, motivating reward designs that jointly control generation cost and program correctness.

E.2 Success Metrics and Paired Tests

Table 8 reports paired-seed t tests and Holm-adjusted values over the complete 12-comparison code family. In transfer, the pass@1 gains of $+2.06$ pp on HumanEval and $+1.82$ pp on MBPP-Sanitized have adjusted p values of 0.0055 and 0.0287. In task-specific post-training, the gated objective retains Vanilla-level MBPP pass@1 ($+0.10$ pp) but lowers pass@8 by 7.15 pp; on HumanEval, pass@1 and pass@8 decrease by 15.83 and 24.31 pp. The contrast identifies cross-task transfer as the stronger use of the fixed-cost objective in the present code experiments.

Table 8: Paired-seed tests for code correctness in transfer and task-specific post-training. Differences are relative to the phase-matched Vanilla arm in percentage points. Holm correction covers all 12 code comparisons.

Setting	Benchmark	Metric	Δ pp	Paired t p	Holm p
Transfer: Fixed	HumanEval	pass@1	+2.06	0.0007276	0.005549
Transfer: Fixed	HumanEval	pass@8	+0.46	0.4758	0.9516
Transfer: Fixed	MBPP-Sanitized	pass@1	+1.82	0.005735	0.02868
Transfer: Fixed	MBPP-Sanitized	pass@8	+1.12	0.1556	0.4802
Post-train: Gated	HumanEval	pass@1	-15.83	0.000748	0.005549
Post-train: Gated	HumanEval	pass@8	-24.31	0.0006936	0.005549
Post-train: Gated	MBPP-Sanitized	pass@1	+0.10	0.973	0.973
Post-train: Gated	MBPP-Sanitized	pass@8	-7.15	0.1201	0.4802

Differences are relative to the phase-matched Vanilla arm. Tests pair eight training runs. Holm values use the complete 12-comparison code family.

E.3 Reproducibility of Code Results

The anonymous bundle contains sample-level rejudge-v1 correctness labels, generated lengths, and seed summaries for the five code configurations in Table 3. The accompanying scripts reproduce the reported means, standard deviations, and paired tests. Generated programs and raw verifier traces are retained by the authors but are not included in the anonymous bundle.

F Larger-Model Protocol and Budget Diagnostics

F.1 Adaptation Setting

The 7B study uses seed 42 and Unsloth/LoRA adaptation (Hu et al., 2021). The shared protocol specifies 150 updates, learning rate 10^{-5}, eight rollouts, batch size eight, a 3,072-token completion/normalization limit, and maximum sequence length 3,072. The resulting checkpoints are evaluated under progressively larger generation budgets to distinguish reward effects from budget saturation.

F.2 Evaluation Budget Diagnostics

Near-total truncation persists across the larger-model evaluations. Table 9 places the two rewards side by side at each observed budget and benchmark. Most responses exhaust the generation budget under both settings, making termination behavior the dominant signal in these checkpoints and motivating explicit optimization of completion behavior.

Budget labels in Table 9 follow the observed maximum generated token counts. The 16k GSM8K runs cover 500 problems and the base-budget runs cover 1,319. Comparisons across these rows consequently involve different evaluation subsets. The truncation statistics characterize the trained checkpoints under their respective generation and context configurations.

Equation (5) identifies a relevant diagnostic for budget-saturated training groups. A constant capped length cost cancels under centering, and constant correctness makes all advantages zero. Evaluating this condition requires group-level training rewards and lengths. Jointly examining those quantities,

Table 9: Exploratory 7B measurements from one training seed. Accuracy and truncation are percentages. Each row specifies its generation limit and evaluated problem count.

			Vanilla			**Fixed-length**		
Limit	Benchmark	N	Acc.	Tokens	Trunc.	Acc.	Tokens	Trunc.
3072	GSM8K	1319	2.08	3068	99.79	2.34	3071	99.93
8k	MATH-500	500	0.15	8186	99.90	0.10	8182	99.85
8k	AIME	90	0.00	8182	99.86	0.00	8184	99.86
8k	AMC	83	0.00	8185	99.85	0.15	8185	99.85
16k	GSM8K	500	1.68	16356	99.78	1.90	16373	99.92
16k	MATH-500	500	0.12	16368	99.80	0.20	16360	99.83
16k	AIME	90	0.00	16383	99.86	0.00	16348	99.72
16k	AMC	83	0.00	16325	99.55	0.00	16365	99.85
32k	AIME	90	0.00	32604	99.72	0.00	32570	99.58
32k	AMC	83	0.00	32652	99.85	0.30	32611	99.85

Accuracy and truncation in percent; one training seed. The 16k GSM8K subset contains 500 problems, versus 1,319 at the base limit.

context allocation, and termination tokens provides a concrete follow-up for the observed truncation regime.

G Evidence and Reproducibility

We retain the complete internal experimental records, including generated responses, verifier logs, training and evaluation code, configurations, and checkpoints. The anonymous bundle accompanying this submission contains de-identified per-problem aggregates, sample-level length and truncation records, seed-level summaries, code rejudge-v1 labels, training scalar traces, and scripts that reproduce the reported aggregate statistics and bootstrap intervals.

The bundle therefore supports independent numerical auditing of the reported results, while trace-level rejudging and end-to-end retraining rely on retained internal artifacts. The accompanying scripts recompute the primary benchmark–seed–arm summaries, paired tests, difficulty-group analyses, code summary statistics, and the four nested bootstrap intervals from the released numerical records. Seed counts and uncertainty definitions are stated explicitly throughout.