Shorter but not Worse: Frugal Reasoning via Easy Samples as Length Regularizers in Math RLVR

Abdelaziz Bounhar[1], **Hadi Abdine**[1], **Evan Dufraisse**[1], **Ahmad Chamma**[1], **Amr Mohamed**[1], **Dani Bouch**[1], **Michalis Vazirgiannis**[1,2], **Guokan Shang**[1]

[1]MBZUAI, [2]École Polytechnique

{abdelaziz.bounhar,guokan.shang}@mbzuai.ac.ae

Abstract

Large language models (LLMs) trained for step-by-step reasoning often become excessively verbose, raising inference cost. Standard Reinforcement Learning with Verifiable Rewards (RLVR) pipelines filter out "easy" problems for training efficiency, leaving the model to train primarily on harder problems that require longer reasoning chains. This skews the output length distribution upward, resulting in a **model that conflates "thinking longer" with "thinking better"**. In this work, we show that retaining and modestly up-weighting moderately easy problems acts as an implicit length regularizer. Exposing the model to solvable short-chain tasks constrains its output distribution and prevents runaway verbosity. The result is ***emergent brevity for free***: the model learns to solve harder problems without inflating the output length, **despite the absence of any explicit length penalization**. RLVR experiments using this approach on *Qwen3-4B-Thinking-2507* (with a 16k token limit) achieve baseline pass@1 AIME25 accuracy while generating solutions that are, on average, nearly twice as short.

1 Introduction

Recently, large language models (LLMs) have begun to rapidly advance the frontier of machine intelligence through test time scaling via step-by-step "reasoning". Scaling inference budget and training with RLVR have enabled models to achieve strong performance on competition-level mathematics and coding tasks by producing extended chains of thought. However, this progress often incurs at a cost: reasoning models tend to be overly verbose, generating excessively long solutions that increase inference latency and memory usage.

A common design choice in RLVR training pipelines is to filter out "easy" problems to maximize training efficiency, with training typically beginning in medium-difficulty samples and gradually shifting toward harder instances [Mistral-AI et al., 2025, Ji et al., 2025b,a]. This design choice is not arbitrary; it follows from the mechanics of Group Relative Policy Optimization (GRPO) [Shao et al., 2024], wherein groups with either all-correct or all-incorrect rollouts yield zero advantage and therefore provide no learning signal. Consequently, both easy and unsolvable hard problems are typically excluded, as they are unlikely to produce meaningful policy updates. This leaves the model to learn primarily from problems that inherently require longer reasoning chains. Over time, this imbalance shifts the output length distribution upward, leading the policy to reward verbosity even when many of the generated tokens are redundant. The outcome is a systematic drift towards unnecessarily long outputs, where models conflate **"thinking longer"** with **"thinking better"**. This phenomenon can also be understood from an information-theoretic perspective. Since *conditioning reduces entropy*, an autoregressive model can lower the expected uncertainty of the final answer simply by extending its output sequence, even if the additional tokens contribute little structure or

substance. In effect, **verbosity becomes a statistical shortcut to entropy reduction rather than a reflection of genuine reasoning**.

In this paper, we revisit the training-efficiency heuristic of discarding easy problems and instead argue for their value. We show that retaining, and upweighting moderately easy problems provides a natural counterbalance: they act as length regularizers. By exposing the policy to tasks that admit concise solutions and training under a limited context window, the model is implicitly pressured to maintain efficiency in its output distribution in order to obtain rewards on harder samples. Crucially, this yields *emergent brevity for free*: The model learns to solve harder problems without inflating the solution length, despite the absence of any explicit length-penalty term. Experiments on Qwen3-4B-Thinking-2507 with a 16k-token budget show that our method preserves baseline pass@1 accuracy on AIME25, while reducing solution length by nearly (2×). This demonstrates that concision and performance are not in opposition: carefully curating the training data is and has always been the key.

Our contributions are two fold:

- **Implicit length regularization:** We show that emphasizing moderately easy problems naturally regularizes output length, reducing verbosity without explicit reward shaping.
- **Empirical validation:** On *Qwen3-4B-Thinking-2507* with a 16k-token budget, our method preserves baseline pass@1 accuracy on AIME25 while reducing average solution length by nearly (2×).

Together, these findings highlight that data curation, not only reward design or model size, plays a critical role in shaping the efficiency of reasoning models.

2 Prelude

We consider an autoregressive language model parameterized by θ, defining a policy π_θ over token sequences. For a query x and a response $y = (y_1, \ldots, y_T)$, the likelihood under the policy is

$$\pi_\theta(y|x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t}). \tag{1}$$

Each response is evaluated by a verifier $r(x, y)$ defined over appropriate domains, which assigns a scalar reward indicating correctness[1]. RLVR seeks to optimize π_θ so as to maximize the expected verifier score $\mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta}[r(x, y)]$ where $\mathcal{D}$ is the training dataset.

Group Relative Policy Optimization (GRPO). Instead of relying on a value model as in PPO [Schulman et al., 2017], GRPO [Shao et al., 2024] uses groups of G responses $\{y_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|x)$ for the same query x sampled from a training dataset $\mathcal{D}$ to estimate the expected reward, a.k.a. the value function. The GRPO objective function is defined as

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, \{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min\left(w_{i,t}(\theta)A_i, \text{clip}(w_{i,t}(\theta), 1-\epsilon, 1+\epsilon)A_i\right) \right], \tag{2}$$

where each response receives an advantage computed relative to the group:

$$A_i = \frac{r(x, y_i) - \text{mean}\left(\{r(x, y_i)\}_{i=1}^{G}\right)}{\text{std}\left(\{r(x, y_i)\}_{i=1}^{G}\right)}, \tag{3}$$

and

$$w_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})} \tag{4}$$

is the *importance sampling weight* applied at the token level. Because rollouts are generated from the behavior policy $\pi_{\theta_{\text{old}}}$, while the objective is to optimize expectations under the updated policy π_θ, a correction is required. Importance sampling provides this correction by reweighting each sampled

[1]Throughout this work, we use binary verifiable rewards $r(x, y) \in \{0, 1\}$, where 1 denotes a correct solution and 0 a failure.

token according to how much more or less likely it is under the new policy than under the old one. Formally, this ensures that expectations with respect to π_θ can be estimated from samples drawn under $\pi_{\theta_{\text{old}}}$:

$$\mathbb{E}_{y\sim\pi_\theta}[f(y)] = \mathbb{E}_{y\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(y)}{\pi_{\theta_{\text{old}}}(y)} f(y)\right]. \tag{5}$$

Applied at the token level, $w_{i,t}(\theta)$ rescales the contribution of each sampled token so that the update direction reflects the current policy. Since these ratios can vary dramatically, GRPO follows PPO in applying clipping, i.e.,

$$\min(w_{i,t}(\theta)A_i,\ \text{clip}(w_{i,t}(\theta), 1-\epsilon, 1+\epsilon)A_i)\,, \tag{6}$$

as a mean of stability.

Difficulty and vanishing advantage. By construction, if all G rollouts are correct ($r = 1$ for all) or all are incorrect ($r = 0$ for all), then $A_i = 0$ for every i, see (3). Such groups do not provide a gradient signal. Consequently, "easy" problems (solved with probability ≈ 1) and "hard" problems (solved with probability ≈ 0) are systematically excluded from the RLVR pipelines. Training is therefore efficient when done on samples that satisfy

$$0 < \Pr[r(x, y) = 1] < 1. \tag{7}$$

Length bias from difficulty imbalance. Medium and hard problems inherently require longer reasoning chains. Filtering out easy problems therefore biases the effective training distribution toward longer outputs. Over successive updates, the policy may learn that reward is typically associated with extended completions, skewing the output length distribution upward. Empirically, this manifests as models producing unnecessarily long solutions, even when concise reasoning would suffice.

Information-Theoretic view. Let $x \in \mathcal{X}$ be a query, Y denote the (final) answer token (or a deterministic function of the full response), and $Z_t \triangleq (Y_1, \ldots, Y_t)$ be the length-t prefix produced by the autoregressive policy. We treat (x, Y, Z_t) as jointly distributed under the rollout process. By the chain rule of entropy,

$$\mathcal{H}(Y \mid X, Z_t) - \mathcal{H}(Y \mid X, Z_{t+1}) = I(Y; Y_{t+1} \mid X, Z_t) \geq 0, \tag{8}$$

so $\mathcal{H}(Y \mid X, Z_{t+1}) \leq \mathcal{H}(Y \mid X, Z_t)$ for all t. That is, *conditioning on a longer prefix reduce the conditional entropy of the final answer*. This statement holds irrespective of how Y_{t+1} is generated (it may even be semantically vacuous), as it is simply a property of conditional entropy and mutual information.

Implication for RLVR. In the absence of any explicit penalty on output length or token semantics, and with rewards depending solely on the correctness of the final answer, a policy can (weakly) lower the uncertainty of its own final prediction by emitting additional tokens i.e., by increasing t before committing to Y (given a sufficiently large context length). On medium/hard instances where reward is sparse and longer chains are more common, this creates a systematic incentive for longer completions: reducing $\mathcal{H}(Y \mid X, Z_t)$ via extra tokens is never penalized *and may correlate* with higher success rates. Over training, this induces an upward drift in output length, leading models to *conflate "thinking longer" with "thinking better"* even when many added tokens are redundant. In effect, verbosity becomes a statistical shortcut for reducing uncertainty in the final boxed answer, rather than a genuine indicator of deeper reasoning.

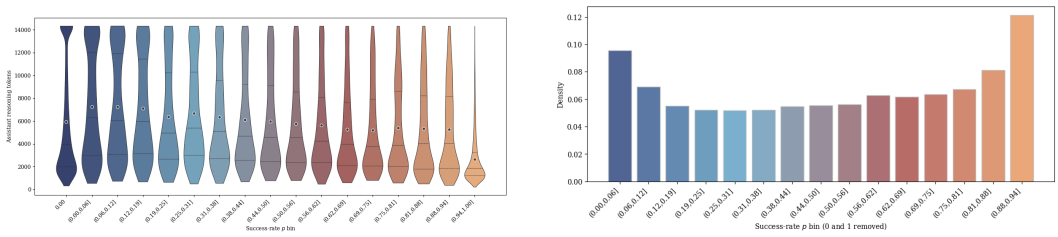
3 Methodology

We focus only on math data.

Length regularization via moderately easy problems. Consider a problem with a success-rate parameter p defined as

$$p = \Pr[r(x, y) = 1 \mid y \sim \pi_\theta]\,. \tag{9}$$

The easy problems correspond to $p \approx 1$, and the hard ones to $p \approx 0$. In standard RLVR pipelines, trivially easy and hard problems are often discarded for efficiency: easy problems have a high success



(a) Token count distribution as a function of empirical success rate p.

(b) Distribution $\rho(p)$ after filtering out trivial and hard cases with $p \in \{0, 1\}$.

Figure 1: Empirical success-rate analysis.

rate probability, while very hard ones rarely contribute useful gradients, as their lengthy solutions either exceed the context window or fail verification, yielding zero reward. Consequently, the effective training signal is dominated by medium-to-hard instances, which inherently require longer reasoning chains.

Reward shaping. We adopt a binary verifiable reward based on exact string matching of the extracted final answer. The model encloses its final prediction within `\boxed{}`, allowing deterministic parsing and verification. Let $\hat{a}(x)$ denote the model's predicted answer extracted from its output for a given query x. A reward of 1 is assigned if the normalized prediction matches the normalized ground truth y, and 0 otherwise:

$$r(x, y) = \begin{cases} 1, & \text{if } \hat{a}(x) = y, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Data Curation.

Stage 1 — Emergent brevity. Although batches are sampled uniformly at random, with mixed difficulty[2], the global dataset distribution is deliberately imbalanced, containing a higher proportion of moderately easy problems. Our design departs from standard RLVR pipelines, which typically discard trivially easy samples for efficiency. Because short, solvable problems provide stable positive rewards associated with structured and concise reasoning traces, they tend to dominate the effective reward signal. Very hard problems, by contrast, contribute little due to sequence truncation or verification failure when the policy cannot yet solve them. Over time, this inductive bias in the reward signal implicitly encourages shorter, more efficient reasoning traces.

Formally, the expected reward over the training distribution $\mathcal{D}$ can be expressed as

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[r(x, y)] = \int_0^1 p\, \rho(p)\, dp, \tag{11}$$

where $\rho(p)$ denotes the empirical density of problems with success probability p. In our dataset, $\rho(p)$ is intentionally skewed toward easy problems, so these samples dominate the reward signal. During RL, the gradient of $J_{\text{GRPO}}(\theta)$ therefore receives stronger, more stable updates from solvable problems within the token limit, while hard problems often yielding long or truncated completions contribute negligible gradients. This imbalance constrains the learned output distribution: since rewards arise predominantly from shorter, solvable trajectories, verbosity ceases to be a profitable strategy. The policy instead optimizes for correctness within limited context, giving rise to what we term *emergent brevity for free*, the model maintains accuracy on hard problems while producing substantially shorter solutions. In effect, easy samples function as implicit *length regularizers*, shaping the model's reasoning behavior toward concise and efficient thought.

The empirical success-rate distribution $\rho(p)$ computed using our target model `Qwen3-4B-Thinking-2507` based on 16 rollouts per-prompt exhibited a bimodal pattern, with a large mass at $p = 0$ and $p = 1$. This pattern indicates that many problems are either trivially solved or currently unsolved by the base policy (given a budget of 16k tokens), while relatively few

[2]Curriculum learning is applied only in the second stage.

lie in the intermediate difficulty range where learning gradients are most informative. Figure 1b isolates that central region by excluding samples with $p \in \{0, 1\}$, highlighting the subset that drives effective RLVR optimization. Additionally, Figure 1a shows that reasoning length varies systematically with difficulty, with easy problems requiring a small number of tokens.

Remark 1. *In our experiments, we do not filter out the samples with $p = 0$, rather only those with $p = 1$, since such problems can become solvable if the model is given a larger reasoning window or improved intermediate steps. Indeed, as shown in Appendix A Figure 4, when sampling 256 of these initially unsolved problems and allowing up to a 42k tokens context, a significant fraction becomes solvable.*

Stage 2 — Improvement via Curriculum RLVR. Having obtained a concise and efficient policy after *Stage 1*, we proceed with a second reinforcement phase based on Curriculum RLVR. Our goal in this stage is to enrich the model's knowledge and reasoning capabilities on a wider domain of mathematical problems while maintaining the same 16k-token limit. Training is conducted on a filtered subset of the `DeepMath-103` dataset [He et al., 2025], which contains problems grouped by difficulty. We follow the difficulty annotations provided by the authors to structure a progressive curriculum: training begins with moderately solvable instances and gradually incorporates harder problems as the policy's competence improves.

We filtered the `DeepMath-103` dataset through a multi-step process. First, we removed samples already present in the Stage 1 dataset. We then retained only those with the correct format and sufficient difficulty.

Following the pre-filtering procedure from this repository, we excluded Multiple-Choice, True/False, Yes/No, and formal Proof-based questions, as in [Mistral-AI et al., 2025]. We also discarded examples with inconsistent answers across the three DeepSeek-R1 generations, those asking for counterexamples or lacking a single correct answer, and those that were ill-posed or underspecified. We adopted the annotations from the same repository, produced using `gpt-5-mini-2025-08-07` with `verbosity="medium"` and `reasoning_effort="minimal"`. After this stage, the dataset was reduced from 103k to 57k samples.

Next, we filtered by difficulty. For each of the nine second-level math domains in the DeepMath dataset, we sampled around 30 examples for difficulty levels 5 to 9 and evaluated the pass@1 performance across these levels. Model accuracy varied across domains: since training datasets typically overrepresent precalculus, calculus, and algebra problems, performance was higher in those areas. For each domain, we retained all difficulty levels starting from the one with less than 75% success, yielding a final set of 14.5k samples (see Table 3 for cut details).

Empirically, we find that performing curriculum RLVR on this subset gives us the best performances compared to random shuffling.

4 Experiments

4.1 Experimental Setup

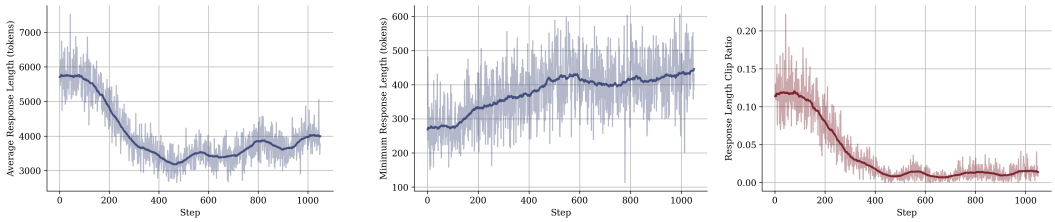
We fine-tune `Qwen3-4B-Thinking-2507` using GRPO with a verifiable binary reward function as defined in (10). The training data consist of verifiable mathematical problems curated and filtered as described in Section 3. Our RLVR implementation is based on the `verl` framework [Sheng et al., 2025].

Training is performed in two stages:

- **Stage 1** (emergent brevity) runs for 1,050 optimization steps, corresponding to a single epoch over the curated dataset.
- **Stage 2** (curriculum RLVR) continues training for 255 additional steps, corresponding to two curriculum epochs over our filtered subset of `DeepMath-103`.

All stages use a fixed 16k-token generation limit and share identical optimization hyperparameters[3]. A summary of the main hyperparameters is provided in Appendix B Table 4.

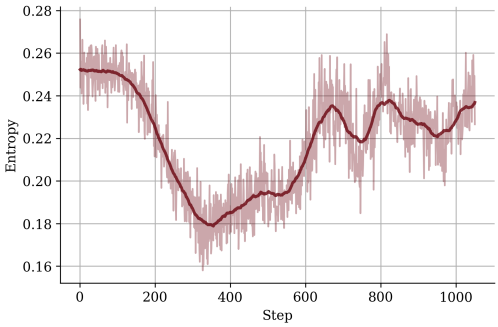
[3] Stage 2 also includes a short warm-up phase before resuming full-rate training.



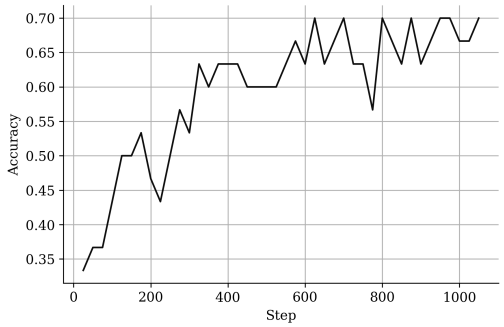
(a) Average response length.

(b) Minimum response length.

(c) Response length clip ratio.



(d) Policy entropy.



(e) Accuracy on AIME25.

Figure 2: Training dynamics during Stage 1 (emergent brevity). Early training is dominated by overly long, truncated generations with high entropy and low accuracy. As learning progresses, average response length and clip ratio decrease sharply, entropy stabilizes, and validation accuracy on AIME25 improves steadily—showing that conciseness and correctness co-emerge.

4.2 Training Dynamics

Figure 2 summarizes the evolution of key metrics during training *Stage 1*. At the beginning of optimization, the model displays pronounced verbosity, reflected by a response-length clipping ratio exceeding 15%, indicating that many generations are prematurely truncated at the 16k-token limit. As training progresses, the average response length steadily decreases while the minimum length increases, suggesting that the model learns to produce more compact yet complete reasoning traces. This reduction in verbosity coincides with a sharp decline in the response-clipping ratio, confirming that the policy increasingly completes its reasoning within the available context budget.

Entropy dynamics provide additional insight into this transition. Entropy decreases sharply in the early phase as the policy shifts from exploration to exploitation, stabilizing around consistent reasoning patterns that yield reliable verifier rewards. Around mid-training (steps 400–600), entropy rises slightly again, indicating renewed exploration which can indicate that the model begins tackling more diverse or harder samples, yet without reverting to the excessively long outputs observed initially. This interplay between entropy and response length supports the interpretation of *emergent brevity* as a stable equilibrium: the policy reduces uncertainty through more efficient reasoning rather than through longer sequences.

Validation accuracy on AIME25 (bottom-right panel) increases steadily from roughly 33%[4] to about 70% throughout Stage 1, showing that conciseness and reasoning competence improve in tandem rather than in opposition. By the end of Stage 1, the policy achieves strong accuracy while maintaining concise, self-terminating outputs, consistent with the intended effect of implicit length regularization. During Stage 2 (Curriculum RLVR), the overall behavior remains qualitatively similar: the model continues to generate short, efficient reasoning traces, although the minimum response length increases to $\approx$ 1,200 tokens due to the increased difficulty.

[4]Evaluation conducted under a 16k-token generation budget.

5 Evaluation

We evaluate our method on verifiable mathematical reasoning tasks, focusing on the efficiency–accuracy trade-off induced by easy-sample regularization and curriculum RLVR. Experiments are conducted on `Qwen3-4B-Thinking-2507`, chosen for its open availability and strong baseline reasoning performance. All experiments use verifiable binary rewards and follow the GRPO training procedure described in Section 3. We report both standard accuracy and our proposed Efficiency-Adjusted Accuracy (EAA; Definition 1) to jointly assess performance and output conciseness.

5.1 Reasoning Benchmarks

We evaluate models on diverse reasoning benchmarks spanning mathematics, STEM, and instruction following.

- `Mathematics`
  - **AIME25** [Ye et al., 2025]: The 2025 American Invitational Mathematics Examination, containing 30 integer-answer problems.
  - **Omni-MATH-Hard** [Gao et al., 2024]: The hardest subset of Omni-MATH, retaining only Olympiad-level problems rated 9–10 in difficulty (100 problems total).
  - **MATH-500**[5]: A held-out set of 500 problems from the original MATH benchmark introduced in "Let's Verify Step by Step" [Lightman et al., 2023].
  - **GSM-Plus** [Li et al., 2024]: A robustness extension of GSM8K with controlled perturbations (e.g., rewording, distractors, numerical changes) to assess consistency under input variations.
- `STEM`
  - **GPQA-Diamond** [Rein et al., 2023]: 198 expert-written, "Google-proof" multiple-choice questions across biology, physics, and chemistry.
- `Instruction Following`
  - **IFEval** [Zhou et al., 2023]: 500 prompts designed to test precise adherence to explicit textual instructions with verifiable outcomes.

5.2 Metrics

Definition 1 (Efficiency Adjusted Accuracy (EAA))**.** *To jointly evaluate reasoning accuracy and conciseness, we define the* Efficiency Adjusted Accuracy *(EAA) metric. Let $a \in [0, 1]$ denote the pass@k (or accuracy) and $L \in [L_{\min}, L_{\max}]$ the mean output length in tokens of a model on a given benchmark. For a tunable penalty exponent $\gamma > 0$, we define*

$$\mathrm{EAA}_\gamma(a, L) = a \cdot \exp\left[-\gamma \cdot \left(\frac{L - L_{\min}}{L_{\max} - L_{\min}}\right)\right]. \tag{12}$$

This formulation linearly rescales output length to the unit interval, so that shorter completions ($L \approx L_{\min}$) preserve the full score, whereas longer ones ($L \approx L_{\max}$) are increasingly penalized depending on γ.

Interpretation. $\mathrm{EAA}\gamma$ remains bounded in $[0, 1]$ and decreases monotonically with output length. The exponent γ controls the strength of the penalty: $\gamma = 1$ yields a linear trade-off between accuracy and brevity, $\gamma > 1$ enforces a sharper preference for concise completions, and $0 < \gamma < 1$ applies a milder adjustment.

5.3 Results

We refer to our Stage 1 and Stage 2 models as Frugal-Math-4B-Stage1, and Frugal-Math-4B-Stage2 respectively.
Table 1 summarizes the reasoning performance of models ranging from 3B to 30B parameters with similar scale active number of parameters under a 42k-token decoding limit, while Table 2 reports

[5] `https://huggingface.co/datasets/HuggingFaceH4/MATH-500`

their corresponding average output lengths. Each cell in Table 1 contains two metrics: the left value is standard Pass@1 (accuracy using LLM-as-a-Judge for Omni-Hard and average scores for IFEval), while the right value is EAA (accuracy normalized by output length) using γ=3.0.

Table 1: Reasoning benchmark performance (similar active number of parameters models, max output 42 k tokens).

Model	Size	GPQA Diamond	AIME25	Omni-Hard	GSM_PLUS	IFEVAL	MATH_500	Average
Qwen3-30B-A3B-Thinking-2507	30B	**70.71**|25.26	**86.67**|09.79	08.09|00.63	**90.29**|**90.29**	41.35|**41.35**	**97.80**|08.15	65.82|29.25
SmolLM3-3B	3B	27.78|01.38	30.00|11.44	35.26|14.20	83.48|29.39	**71.21**|03.55	90.80|45.35	56.42|17.55
Phi-4-mini-reasoning	4B	30.30|03.05	40.00|12.83	32.37|18.39	87.10|61.12	51.58|22.05	90.80|44.21	55.36|26.94
Qwen3-4B-Thinking-2507	4B	67.17|03.68	73.33|03.65	04.62|00.23	89.05|16.71	38.57|20.79	97.60|04.86	61.72|08.32
Frugal-Math-4B-Stage1 (ours)	4B	63.64|31.22 (-03.53)|(+27.54)	60.00|43.73 (-13.33)|(+40.08)	35.84|31.54 (+31.22)|(+31.31)	89.24|04.44 (+00.19)|(-12.27)	39.91|22.43 (+01.34)|(+01.64)	95.00|55.51 (-02.60)|(+50.65)	63.94|31.48 (+02.22)|(+23.16)
Frugal-Math-4B-Stage2 (ours)	4B	70.20|**70.20** (+03.03)|(+66.52)	70.00|**70.00** (-03.33)|(+66.35)	**47.40**|**47.40** (+42.78)|(+47.17)	89.00|11.15 (-00.05)|(-05.56)	39.49|23.20 (+00.92)|(+02.41)	95.20|**95.20** (-02.40)|(+90.34)	68.55|52.86 (+06.83)|(+44.56)

At this maximum context length, Frugal-Math-4B-Stage2 achieves an average accuracy of 68.55% and an EAA of 52.86, outperforming its base model (61.72 / 8.32) by +6.83 and +44.54, respectively. The Stage 1 variant also improves to 63.94 / 31.48, showing that our Stage 1 fine-tuning yields substantially better token efficiency. Compared to larger or similar-sized baselines, the 30B Qwen3-A3B model achieves 65.82 / 29.25, while Phi-4-mini-reasoning and SmolLM3-3B trail behind at 55.36 / 26.94 and 56.42 / 17.55, confirming that the Frugal models preserve or slightly improve overall accuracy while delivering far better efficiency.

The length analysis in Table 2 reinforces this efficiency narrative. While Qwen3-4B-Thinking-2507 generates on average 11491 tokens per sample, our Frugal-Math-4B variants drastically reduce this to 6270 for Stage 1 and 5712 for Stage 2. The efficiency gains are most pronounced on harder mathematical reasoning tasks, notably AIME25 and Omni-Hard, where solution chains are typically long. On these benchmarks, Stage 2 achieves comparable or higher accuracy using 55–61% fewer tokens, indicating that it learns to reason more efficiently, while still reaching correct final answers. In contrast, for easier arithmetic problems such as GSM_PLUS, where all models already reach high accuracy with short outputs, the advantage is less pronounced; Stage 2's generations are slightly longer (+5.6%) and EAA shows a small regression. This pattern suggests that the Frugal-Math-4B models allocate reasoning effort adaptively—compressing complex reasoning when needed but not over-optimizing brevity on tasks that are inherently simple.

Table 2: Average output length (tokens) per benchmark under the 42 k-token decoding budget.

Model	Size	GPQA Diamond	AIME25	Omni-Hard	GSM_PLUS	IFEVAL	MATH_500	Average Length
Qwen3-30B-A3B-Thinking-2507	30B	7208.61	17887.8	26960.1	1373.03	**1179.44**	5069.94	9946.49
SmolLM3-3B	3B	8966.65	13136.2	17076.9	1634.25	5521.41	3695.5	8338.48
Phi-4-mini-reasoning	4B	8338.75	13811.7	15009.4	**1461.65**	2409.01	3714.96	7457.58
Qwen3-4B-Thinking-2507	4B	8882.41	21090.1	29642.2	1791.69	2073.93	5465.89	11491.04
Frugal-Math-4B-Stage1 (ours)	4B	6925.65	10604.1	12380.3	2123.68	2013.00	3574.92	6270.28
Frugal-Math-4B-Stage2 (ours)	4B	**6290.44**	**9367.67**	**11611.9**	1892.89	1949.61	**3162.40**	**5712.49**

The scaling curves in Figure 3 illustrate how different models behave under increasing generation budgets (8k $\rightarrow$ 16k $\rightarrow$ 32k $\rightarrow$ 42k). For the hard reasoning tasks, AIME25 and Omni-Hard, our Frugal-Math-4B models, particularly Stage 2, demonstrate superior efficiency at lower budgets. At 8k and 16k tokens, they already achieve accuracy levels close to or exceeding larger models, while maintaining a higher EAA across all budgets. This indicates that the Frugal models can solve complex, multi-step mathematical problems correctly with much shorter reasoning chains. In contrast, Qwen3-4B-Thinking-2507 and Qwen3-30B-A3B-Thinking-2507 continue to improve with larger decoding budgets (32k and 42k) achieving better accuracies, but their EAA remains consistently lower, suggesting that their accuracy gains rely on significantly longer outputs. For simpler arithmetic tasks like GSM_PLUS, where all models rapidly converge to high accuracy even under small budgets, our method provides limited benefits. The Frugal-Math models produce slightly longer outputs and show nearly equivalent performance, reflecting that these tasks already require minimal reasoning depth and offer little room for further compression. Overall, these scaling results highlight the strength of our approach on reasoning-intensive tasks, where Frugal-Math-4B models maintain a more favorable accuracy-per-token ratio and deliver strong performance even under tight output constraints.

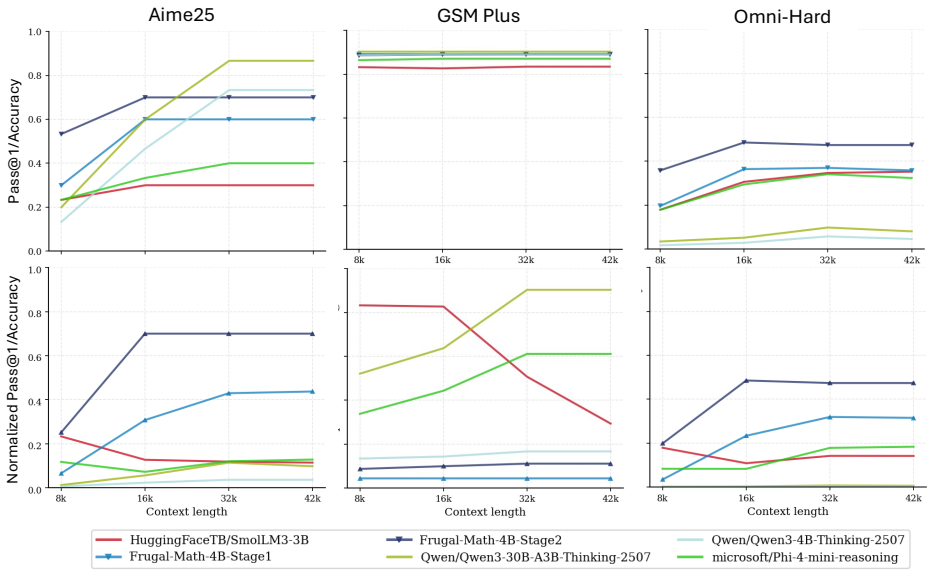


Figure 3: Scaling behavior under varying generation budgets (8 k $\rightarrow$ 16 k $\rightarrow$ 32 k $\rightarrow$ 42 k). The top panels show Pass@1 accuracy and the bottom panels show Efficiency-Adjusted Accuracy for the three benchmarks; AIME25, GSM Plus, and Omni-Hard.

6 Outlooks

Our findings show that retaining moderately easy problems naturally regularizes reasoning length without explicit penalties. Future work may extend this idea to other domains such as coding or logical reasoning, explore adaptive curricula balancing easy–hard samples, and combine implicit and explicit regularization for finer control of brevity.

7 Limitations

Our study focuses on math reasoning with verifiable rewards and a single 4B model. Generalization to open-ended or larger-scale settings remains unexplored and is part of ongoing research. Moreover, brevity is observed empirically rather than theoretically explained, and its understanding warrants further study.

References

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models, 2024. URL `https://arxiv.org/abs/2410.07985`.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

Yunjie Ji, Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yiping Peng, Han Zhao, and Xiangang Li. Am-thinking-v1: Advancing the frontier of reasoning at 32b scale, 2025a. URL `https://arxiv.org/abs/2505.08311`.

Yunjie Ji, Sitong Zhao, Xiaoyu Tian, Haotian Wang, Shuaiting Chen, Yiping Peng, Han Zhao, and Xiangang Li. How difficulty-aware staged reinforcement learning enhances llms' reasoning

capabilities: A preliminary experimental study, 2025b. URL https://arxiv.org/abs/2504.00829.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers, 2024. URL https://arxiv.org/abs/2402.19255.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Mistral-AI, :, Abhinav Rastogi, Albert Q. Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, Léonard Blier, Lucile Saulnier, Matthieu Dinot, Maxime Darrin, Neha Gupta, Roman Soletskyi, Sagar Vaze, Teven Le Scao, Yihan Wang, Adam Yang, Alexander H. Liu, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Andy Ehrenberg, Anmol Agarwal, Antoine Roux, Arthur Darcet, Arthur Mensch, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Chris Bamford, Christian Wallenwein, Christophe Renaudin, Clémence Lanfranchi, Darius Dabert, Devon Mizelle, Diego de las Casas, Elliot Chane-Sane, Emilien Fugier, Emma Bou Hanna, Gauthier Delerce, Gauthier Guinet, Georgii Novikov, Guillaume Martin, Himanshu Jaju, Jan Ludziejewski, Jean-Hadrien Chabran, Jean-Malo Delignon, Joachim Studnia, Jonas Amar, Josselin Somerville Roberts, Julien Denize, Karan Saxena, Kush Jain, Lingxiao Zhao, Louis Martin, Luyu Gao, Lélio Renard Lavaud, Marie Pellat, Mathilde Guillaumin, Mathis Felardos, Maximilian Augustin, Mickaël Seznec, Nikhil Raghuraman, Olivier Duchenne, Patricia Wang, Patrick von Platen, Patryk Saffer, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Romain Sauvestre, Rémi Delacourt, Sanchit Gandhi, Sandeep Subramanian, Shashwat Dalal, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Thibault Schueller, Thibaut Lavril, Thomas Robert, Thomas Wang, Timothée Lacroix, Valeriia Nemychnikova, Victor Paltz, Virgile Richard, Wen-Ding Li, William Marshall, Xuanyu Zhang, and Yunhao Tang. Magistral, 2025. URL https://arxiv.org/abs/2506.10910.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

Yixin Ye, Yang Xiao, Tiantian Mi, and Pengfei Liu. Aime-preview: A rigorous and immediate evaluation framework for advanced mathematical reasoning, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A Context-Length Effects on Data Difficulty Filtering

To ensure a meaningful analysis of model competence, we filter the dataset based on the empirical success rate

$$p = \Pr[r(x, y) = 1 \mid y \sim \pi_\theta],$$

computed over multiple stochastic generations per prompt. We re-evaluate 256 of the initially unsolved problems at 16k tokens budget by allowing up to a 42k tokens context. As shown in

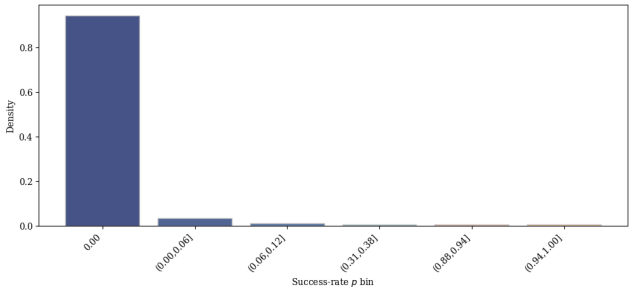


Figure 4: Distribution $\rho(p)$ after scaling maximum response length to 42k tokens.

Figure 4, the resulting distribution $\rho(p)$ remains dominated by low success-rate probability cases, but a small yet non-trivial mass appears at higher p-values. This confirms that a subset of previously unsolved tasks can be recovered through longer reasoning chains rather than fundamental limitations in model ability.

Table 3: Stage 2 Filtering - Difficulty level (start included) for each second-level math domain.

Math Domain	**Starting Difficulty Level**
Algebra	7
Calculus	7
Precalculus	7
Discrete Mathematics	7
Number Theory	6
Geometry	7
Other	5
Applied Mathematics	6

B Training Hyperparameters

Table 4: Hyperparameters and system configuration for RL fine-tuning.

Parameter	**Value**
Base model	`Qwen3-4B-Thinking-2507`
RL algorithm	GRPO
Reward type	Verifiable binary reward (exact match)
Rollout group size (G)	16
Clipping thresholds $(1-\epsilon, 1+\epsilon)$	(0.8, 1.28)
Maximum completion length	16,384 tokens
Batch size (per step)	128
Learning rate	1×10^{-6}
Warmup schedule	Linear, first 5% of steps
Optimizer	AdamW
Hardware	250 NVIDIA H200 GPU days