LEAD: Length-Efficient Adaptive and Dynamic Reasoning for Large Language Models

Songtao Wei[1], **Yi Li**[1], **Zhikai Li**[2], **Xu Hu**[1], **Yuede Ji**[4], **Guanpeng Li**[5], **Feng Chen**[1]
Carl Yang[2], **Zhichun Guo**[3], **Bingzhe Li**[1]

[1]University of Texas at Dallas [2]Emory University
[3]Individual Researcher [4]University of Texas at Arlington
[5]University of Florida

Abstract

Large reasoning models, such as OpenAI o1 and DeepSeek-R1, tend to become increasingly verbose as their reasoning capabilities improve. These inflated Chain-of-Thought (CoT) trajectories often exceed what the underlying problems require, wasting compute, latency, and context budgets. While introducing length-based efficiency rewards during reinforcement learning offers a natural remedy, existing methods struggle with two fundamental challenges: the optimal balance between correctness and efficiency is non-stationary throughout training, and intrinsic reasoning budgets vary drastically across problems. Relying on static reward weights and global length constraints inevitably forces a compromise between degraded accuracy and unrealized compression. To overcome these limitations, we propose LEAD (**L**ength-**E**fficient **A**daptive and **D**ynamic reasoning), a method that replaces static heuristics with online, self-adaptive mechanisms. LEAD dynamically calibrates the correctness–efficiency trade-off at each step using a Potential-Scaled Instability, directing optimization capacity to the most informative learning signal. Furthermore, it estimates an adaptive per-problem target length online based on the model's own correct rollouts, applying a symmetric efficiency reward that penalizes both overthinking and over-compression. Evaluated on five mathematical reasoning benchmarks, LEAD achieves the highest accuracy and Accuracy-Efficiency Score among RL-trained efficient-reasoning methods in our matched comparisons while producing substantially shorter outputs than the base model.

1 Introduction

Chain-of-thought (CoT) prompting [1] shows that large language models (LLMs) can improve complex problem solving through explicit intermediate reasoning, inspiring many subsequent reasoning and tool-use methods [2, 3, 4, 5, 6, 7]. More recently, reinforcement learning (RL) has further strengthened reasoning models such as OpenAI o1 [8] and DeepSeek-R1 [9], producing long and elaborate reasoning traces that improve performance on challenging tasks. However, this emergent reasoning comes at a cost: reasoning models are verbose by default. As models improve, their solutions grow longer, consuming compute, latency, and context budget on reasoning steps that are often unnecessary for the problem at hand [10, 11]. A competition-level math problem may legitimately require thousands of reasoning tokens; a single-step arithmetic query should not. Yet models trained solely to maximize correctness learn to "think longer to think better," producing responses whose length is largely decoupled from the complexity of the underlying task.

Making LLM reasoning *efficient* has therefore become a central research question [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. The standard recipe is to augment the RL training loop with a length-based efficiency signal in addition to the correctness signal, either through reward shaping [12,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

16, 17, 22, 19], multi-objective reinforcement learning [18, 23, 14, 24, 21, 25], or trajectory-level constraints [26, 27, 15, 20, 28]. In principle, this signal should encourage the model to remove redundant reasoning while preserving the reasoning needed for correctness. In practice, however, this goal depends on two questions that static length-control schemes do not answer well: *when* should the optimizer prioritize brevity during training, and *how much* reasoning should each problem be allowed to use?

These questions expose two challenges that efficient reasoning methods must address. **The first challenge is to dynamically balance reward contributions over training.** The relative usefulness of rewards for correctness and efficiency changes as the policy improves. Early in training, correctness-oriented exploration is essential, and excessive length pressure can suppress reasoning needed to discover valid solutions. As training progresses and some prompts become reliably solvable, the efficiency signal becomes more useful for removing redundant reasoning from those solved trajectories. Thus, a fixed reward ratio $(\lambda_c, \lambda_\ell)$ is unlikely to remain appropriate throughout training. **The second challenge is adaptive efficiency across problem difficulties.** Different prompts require different amounts of reasoning, so a single target length should not be applied uniformly across all problems. A simple arithmetic problem may be solved concisely, whereas an Olympiad-level problem may require many intermediate steps. A global budget either over-compresses hard problems, hurting correctness, or under-compresses easy problems, wasting tokens. Together, these challenges call for a framework that *dynamically* balances reward contributions throughout training while *adaptively* calibrating the target length for each prompt.

We propose **LEAD** (**L**ength-**E**fficient **A**daptive and **D**ynamic reasoning), a framework that addresses both challenges through online self-calibration. LEAD combines two mechanisms. *First*, it dynamically adjusts the correctness–efficiency trade-off during training. Rewards are normalized separately to prevent scale dominance, and their weights are updated online according to which signal remains informative. This creates a transient curriculum in which length efficiency guides early compression, while optimization gradually shifts toward correctness as the efficiency signal saturates. *Second*, LEAD replaces a global length budget with a per-prompt target L_q^* estimated from the model's current correct rollouts. This target adapts to both problem difficulty and model capability, allowing hard prompts to retain the necessary reasoning while encouraging easy prompts to be concise. A symmetric efficiency reward around L_q^* penalizes both overthinking and over-compression.

We evaluate LEAD on five math reasoning benchmarks using different LLM models. LEAD matches or exceeds baseline accuracy while significantly reducing solution length, outperforming recent efficient-reasoning methods (DRPO [18], ShorterBetter [16]) on the accuracy–efficiency score. Our contributions are:

- We identify two algorithm-agnostic challenges in efficient-reasoning RL: dynamic reward balancing over training and adaptive efficiency across problem difficulties, and show they are difficult to resolve reliably with a static coefficient without task- and model-specific tuning.
- We propose LEAD, which combines online instability-driven reward weighting with per-problem target-length calibration, requiring no manual coefficient scheduling.
- We validate LEAD across five math benchmarks and on 1.5B- and 7B-sized models, showing consistent improvements in the accuracy–efficiency trade-off over state-of-the-art baselines. The code is released [1].

2 Related Work

Reinforcement Learning for LLM Reasoning. Outcome-based reinforcement learning is the dominant paradigm for training large reasoning models such as OpenAI o1 [8], DeepSeek-R1 [9], Kimi-k1.5 [22], and Qwen-QwQ [29], all of which scale test-time chain-of-thought to deliver substantial gains on complex reasoning tasks. The most widely used algorithm in this setting is GRPO [9], which samples multiple rollouts per prompt and computes group-relative advantages under a clipped policy-gradient objective without a critic. DAPO [27] extends GRPO with dynamic sampling, token-level policy gradients, and overlong reward shaping for large-scale stability. More generally, optimizing reasoning for both correctness and efficiency is a multi-objective RL problem,

[1] `https://github.com/CrazyMint/LEAD`.

where simple scalarization can obscure trade-offs between competing objectives [30]. When multiple reward signals are combined, GDPO [24] identifies reward-advantage collapse in GRPO's combine-then-normalize design, where the higher-variance signal dominates after normalization, and mitigates it by normalizing each reward separately before combining them with static weights.

Efficient Reasoning. A growing body of work addresses the verbosity problem in reasoning models, namely the tendency to generate unnecessarily long solutions when optimized primarily for correctness. Several methods introduce length penalties, pruning objectives, or budget constraints during training. L1 [14] trains reasoning models to follow user-specified length constraints, O1-Pruner [15] uses length-harmonizing fine-tuning to reduce redundant long-thought reasoning, and DRPO [18] decouples the learning signals for correct and incorrect rollouts to avoid penalizing valid long reasoning. LASER [19] formulates efficient reasoning through adaptive length-based reward shaping, while GFPO [21] encourages concise reasoning by filtering sampled rollouts according to length and reward-per-token efficiency. LASER uses a target-length step reward, and LASER-D additionally adapts targets to training progress and difficulty. LEAD combines an online per-prompt mean-correct target, a symmetric deviation reward, and reward-conditioned weighting of separately normalized advantages. Other methods estimate or impose problem-dependent budgets: ShorterBetter [16] uses the shortest correct rollout as a Sample Optimal Length, SmartThinker [17] calibrates reasoning length through a distributional estimate, SelfBudgeter [20] predicts query-specific token budgets before generation, and e1 [31] learns adaptive control of reasoning effort through an inference-time effort parameter. A complementary line studies test-time compute allocation rather than training-time reward optimization: s1 [28] uses budget forcing for test-time scaling, Plan-and-Budget [32] allocates token budgets across decomposed subproblems, and Agarwal et al. [33] show that the best test-time scaling strategy depends on model type, problem difficulty, and compute budget.

Trajectory editing and learned reasoning control. LC-R1 [34] combines a length reward with a compression reward aimed at redundant thinking after an answer is found. JET [35] uses truncated training rollouts and a quality-controlled length reward. Step-GRPO [36] combines dynamic rollout truncation with semantic-step-relative rewards to internalize early exit. These methods modify the exploration trajectories or the unit of length control, whereas LEAD calibrates a token-length target from ordinary capped rollouts. Think-How-to-Think [37] uses staged training for difficulty and redundancy awareness, and Correct, Concise and Complete [38] combines supervised preparation with RL that penalizes continuation beyond the first correct answer. TIME [39] instead learns context-triggered reasoning in temporally structured dialogue. These approaches offer complementary ways to learn when reasoning is needed. LEAD focuses on online reward allocation and behavior-relative target estimation within verifiable mathematical reasoning.

3 Limitations of Static Length Control

Notation. We consider a reasoning policy π_θ trained on a dataset $\mathcal{D} = \{q_i\}_{i=1}^{N}$ of prompts. Following GRPO [9], for each prompt q we sample a group of G rollouts $\{o_{q,j}\}_{j=1}^{G}$ from the old policy $\pi_{\theta_{\text{old}}}$, each with token length $\ell_{q,j} = |o_{q,j}|$. Let $r_c(o,q) \in \{0,1\}$ denote the binary correctness reward and $r_\ell(o,q) \in \mathbb{R}$ a length-based efficiency reward. In standard GRPO, the final reward is a scalar combination $r(o,q) = \lambda_c\, r_c(o,q) + \lambda_\ell\, r_\ell(o,q)$ with non-negative weights λ_c, λ_ℓ (their relative ratio $\rho = \lambda_\ell/\lambda_c$ controls how much the optimizer listens to length), and the group-relative advantage $A_{q,j}$ is shared across all tokens t of rollout j:

$$A_{q,j} \;=\; \frac{r(o_{q,j},q) - \mu_q}{\sigma_q + \epsilon}, \qquad \mu_q = \tfrac{1}{G}\sum_j r(o_{q,j},q), \;\; \sigma_q = \mathrm{Std}_j\, r(o_{q,j},q). \tag{1}$$

The policy is then updated by minimizing a loss that is the negative of the clipped PPO-style surrogate over $A_{q,j}$ plus a KL regularizer (full objective deferred to Appendix B). While this formulation works well for a single reward, the combined-then-normalized structure of Eq. (1) introduces structural pathologies when applied to jointly optimize accuracy and efficiency. We identify two such pathologies below, both of which motivate our method.

3.1 Reward Collapse under Static Weighting

The group normalization in Eq. (1) is applied *after* the two reward components have already been combined. Consider a group in which all G rollouts are correct (r_c=1) and differ only in length. The combined reward reduces to $r(o_{q,j}, q) = \lambda_c + \lambda_\ell\, r_\ell(o_{q,j}, q)$, so $\mu_q = \lambda_c + \lambda_\ell\, \mu_q^{(\ell)}$ and $\sigma_q = \lambda_\ell\, \sigma_q^{(\ell)}$. Substituting into Eq. (1), for any $\lambda_\ell > 0$ and ignoring the numerical regularizer ϵ, the static trade-off coefficient cancels in numerator and denominator and the advantage reduces to $A_{q,j} \approx (r_\ell(o_{q,j}, q) - \mu_q^{(\ell)})/\sigma_q^{(\ell)}$: the length penalty drives the gradient at full normalized magnitude regardless of the practitioner's intended λ_ℓ. Conversely, in an all-incorrect group (r_c=0), the same cancellation means the efficiency signal drives the entire advantage, even though there is no correctness to preserve. In mixed groups, the scale mismatch between binary correctness and continuous length rewards causes the higher-variance component to dominate after normalization, while the other becomes noise. Tuning static weights cannot fully solve this, because the useful balance changes over training. Length rewards are informative while the model is learning to compress, but their within-group variance collapses once responses cluster, whereas correctness often remains informative on hard prompts. Thus, a fixed pair $(\lambda_c, \lambda_\ell)$ either over-compresses before solving is learned or underuses length feedback after accuracy stabilizes.

3.2 Global Length Budget Ignores Problem Difficulty

A second limitation is how the efficiency reward itself is shaped. A common strategy applies a global length budget B to all prompts [27, 18, 26, 22], e.g., $r_\ell = \min(0, 1 - \ell/B)$ once the response exceeds the budget.

This ignores the heterogeneity of reasoning difficulty. For example, easy arithmetic and olympiad-level problems should not share the same target length. When B is set aggressively to drive compression, the model is forced to truncate its reasoning on hard problems that genuinely require more steps, producing short but often incorrect outputs. This is a well-documented accuracy regression in prior efficient-reasoning methods [12, 23, 18]. When B is set loosely to preserve accuracy, the penalty rarely fires on easy problems, and the compression benefit vanishes. Thus, a fixed global budget cannot simultaneously respect problem-dependent reasoning requirements and exploit compression opportunities when they exist. Both failure modes arise from the same mismatch: a single global budget cannot reflect the heterogeneous reasoning demands of different prompts.

4 Method

LEAD has two key components: **dynamic reward weighting with decoupled group normalization** (Section 4.1), which combines per-reward normalized advantages under online, instability-driven weights instead of the scalar-combined advantage of Eq. (1); and **per-problem online target-length calibration** (Section 4.2), which replaces the global length budget with a per-problem target L_q^* estimated from the model's own correct rollouts. Figure 1 shows the full pipeline.

4.1 Dynamic Reward Weighting with Decoupled Group Normalization

Decoupled group normalization. Following GDPO [24], we normalize each reward in its own group before aggregation, which prevents the reward-advantage collapse of Section 3.1. For each reward $k \in \{c, \ell\}$,

$$A_{q,j}^{(k)} \;=\; \frac{r_k(o_{q,j}, q) - \mu_q^{(k)}}{\sigma_q^{(k)} + \epsilon}, \qquad \mu_q^{(k)} = \tfrac{1}{G}\sum_j r_k(o_{q,j}, q),\;\; \sigma_q^{(k)} = \mathrm{Std}_j\, r_k(o_{q,j}, q), \tag{2}$$

and the components are combined under a weight vector $\boldsymbol{\lambda} = (\lambda_c, \lambda_\ell)$ with $\lambda_k \geq 0$, $\sum_k \lambda_k = 1$:

$$\tilde{A}_{q,j} \;=\; \lambda_c\, A_{q,j}^{(c)} \;+\; \lambda_\ell\, A_{q,j}^{(\ell)}, \qquad A_{q,j} \;=\; \mathrm{BatchWhiten}(\tilde{A}_{q,j}) \;=\; \frac{\tilde{A}_{q,j} - \bar{\mu}}{\sqrt{\bar{v} + \epsilon}}, \tag{3}$$

where $\bar{\mu}$ and $\bar{v}$ are the mean and sample variance of $\tilde{A}$ over valid response-token positions after broadcasting each rollout advantage to its tokens (Appendix F.7). Because each $A^{(k)}$ is already

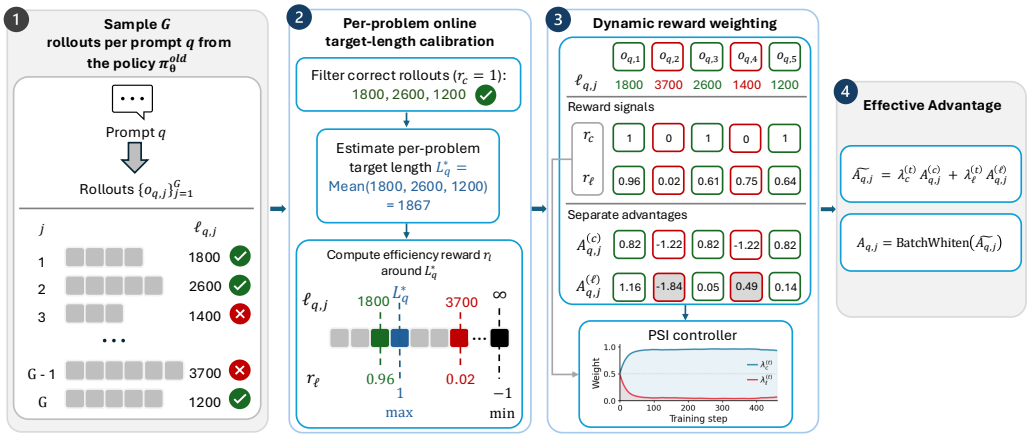


Figure 1: **LEAD framework. (1)** Sample G rollouts per prompt q from the old policy $\pi_{\theta_{\text{old}}}$ and score each by correctness r_c. **(2)** Per-problem online target-length calibration: filter to correct rollouts $\mathcal{C}_q$, set L_q^* to their mean, and compute the symmetric efficiency reward r_ℓ, which peaks at $\ell{=}L_q^*$ and decays linearly to -1 on either side. **(3)** Dynamic reward weighting: each reward is group-normalized separately to produce decoupled advantages $A_{q,j}^{(c)}$ and $A_{q,j}^{(\ell)}$, and the PSI controller adapts the weights $(\lambda_c^{(t)}, \lambda_\ell^{(t)})$ online from per-reward instability and headroom. **(4)** The two advantage channels are linearly combined under the EMA-smoothed weights and batch-whitened to obtain the effective advantage $A_{q,j}$ used in the GRPO objective. Blue-framed components are LEAD's contributions; gray-framed components are inherited from GRPO.

group-centered, $\bar{\mu}$ is close to zero and BatchWhiten mainly rescales the combined advantage to unit variance across the batch. We keep the explicit centering for numerical robustness. Decoupled normalization addresses only the scale-mismatch half of the pathology in Section 3.1: it prevents the reward with the larger within-group variance from drowning out the other, but it inherits GDPO's assumption that a fixed $(\lambda_c, \lambda_\ell)$ is appropriate throughout training. The non-stationary half remains, since the relative learnability of the two rewards drifts as one saturates faster than the other. We close this gap with online dynamic weighting.

Dynamic weighting via the Potential-Scaled Instability (PSI). With scale mismatch already removed by decoupled normalization, the remaining question is which reward still provides a usable learning signal at the current training step. We adapt $\boldsymbol{\lambda}$ online from two statistics of each reward: its *instability* (a reward still changing rapidly carries a gradient signal) and its *headroom* (a reward near its ceiling cannot improve further). At each training step, from the current batch of M prompts (G rollouts each), the Law of Total Variance gives the raw-reward mean and standard deviation as

$$\mu_k = \frac{1}{M}\sum_{q=1}^{M} \mu_q^{(k)}, \qquad \sigma_k = \sqrt{\frac{1}{M}\sum_{q=1}^{M} \mathrm{Var}_j\, r_k(o_{q,j}, q) \;+\; \mathrm{Var}_q\, \mu_q^{(k)}} \;+\; \epsilon, \tag{4}$$

where Var_j and Var_q are population variances over the rollouts of a prompt and over the prompt means of the batch, and the coefficient of variation $\mathrm{CV}_k = \sigma_k/(|\mu_k|+\epsilon)$ measures instability relative to magnitude.[2] The **potential** P_k measures headroom to the reward's ceiling, given the reward's

[2]For the efficiency reward ($k{=}\ell$), the per-prompt mean and variance entering Eq. (4) are restricted to correct rollouts $\mathcal{C}_q$, and prompts with $|\mathcal{C}_q|{=}0$ are dropped from the outer average, since incorrect rollouts carry no usable efficiency signal. The per-rollout advantage in Eq. (2) continues to use all G rollouts. The $\epsilon{=}10^{-8}$ regularizer in the CV denominator handles transient zero-crossings during early warmup.

range $[R_k^{\min}, R_k^{\max}]$ ($[0,1]$ for correctness; $[-1,1]$ for our symmetric length reward):

$$P_k \;=\; \left(1 - \frac{\mu_k - R_k^{\min}}{R_k^{\max} - R_k^{\min}}\right)^{\alpha}, \tag{5}$$

where α controls the decay sharpness near the ceiling. The combined **potential-scaled instability (PSI)** is

$$\Psi_k \;=\; \widetilde{\mathrm{CV}}_k \cdot P_k, \qquad \widetilde{\mathrm{CV}}_k \;=\; \frac{\mathrm{CV}_k}{\sum_{k'} \mathrm{CV}_{k'} + \epsilon}, \tag{6}$$

which is large when the reward k is noisy and far from the ceiling; small when stable or saturated.

Why $\widetilde{\mathrm{CV}}_k \cdot P_k$. After decoupled normalization removes scale mismatch, a reward should receive high weight only if it remains both informative and improvable. $\widetilde{\mathrm{CV}}_k$ measures relative reward variability, while P_k measures remaining headroom to the reward ceiling. The two factors capture orthogonal failure modes: a reward can have ample variance yet sit near its ceiling on most prompts, or have substantial headroom but little usable variation across rollouts. Their product is large only when both conditions hold and small when either fails. Unlike GradNorm [40] or uncertainty weighting [41], which balance raw gradient or loss scales, PSI uses inexpensive batch statistics of the raw rewards: group normalization assigns relative credit within each prompt, while PSI sets the ratio between the two reward channels from their batch-level variability and headroom.

Per-batch Ψ_k values are noisy, so we normalize and EMA-smooth them into the target weights:

$$\hat{\lambda}_k^{(t)} = \frac{\Psi_k}{\sum_{k'} \Psi_{k'} + \epsilon}, \qquad \boldsymbol{u}^{(t)} \;=\; \beta_{\mathrm{ema}}\, \boldsymbol{u}^{(t-1)} \;+\; (1 - \beta_{\mathrm{ema}})\, \hat{\boldsymbol{\lambda}}^{(t)}, \qquad \boldsymbol{u}^{(0)} = \mathbf{1}/K, \tag{7}$$

with $\beta_{\mathrm{ema}} = 0.95$ (effective horizon of roughly 20 steps). The smoothed state $\boldsymbol{u}^{(t)}$ is stored before any floor is applied, and the applied weights enforce a floor $\lambda_c \geq \lambda_{\min}$ by setting $\lambda_c^{(t)} = \max(u_c^{(t)}, \lambda_{\min})$ and $\lambda_\ell^{(t)} = 1 - \lambda_c^{(t)}$, which preserves $\sum_k \lambda_k = 1$. This prevents the correctness signal from being fully dampened by a transiently stable batch. Fixed reward bounds keep the potential well defined when every rollout has the same reward, and Appendix F.8 describes the all-correct and all-wrong cases. The only added state beyond GRPO is $\boldsymbol{u}^{(t)}$ (two scalars). The full procedure is summarized in Algorithm 1.

4.2 Per-problem Online Target-Length Calibration

The second component replaces the global budget B with a per-problem target length L_q^* estimated per prompt from the model's own correct rollouts, addressing the heterogeneity and over-compression issues of Section 3.2.

Online target-length estimation. Let $\mathcal{C}_q = \{j : r_c(o_{q,j}, q) = 1\}$ be the indices of correct rollouts for prompt q. We define L_q^* as the mean length of $\mathcal{C}_q$, clamped to a permissible range:

$$L_q^* \;=\; \begin{cases} \mathrm{clip}\Big(\frac{1}{|\mathcal{C}_q|} \sum_{j \in \mathcal{C}_q} \ell_{q,j},\; L_{\min},\; B_{\max}\Big) & \text{if } |\mathcal{C}_q| \geq 1, \\ B_{\max} & \text{if } |\mathcal{C}_q| = 0, \end{cases} \tag{8}$$

where $L_{\min}$ keeps the reward well-conditioned for very short solutions and $B_{\max}$ is the training-time max response length, doubling as the upper clamp and the sentinel value for unsolved prompts. When $|\mathcal{C}_q| = 0$, setting $L_q^* = B_{\max}$ makes Eq. (9) reduce to $r_\ell = \ell_{q,j}/B_{\max}$, which after group normalization places the longest rollouts in the group at positive efficiency advantage and the shortest at negative. We accept this expansion pressure on unsolved prompts as a deliberate trade-off: correctness on those prompts is what matters first, so encouraging longer reasoning while the model is still searching for a solution is consistent with the long-reasoning behavior already present in the base model. Its contribution to the policy gradient is small in practice because λ_ℓ is small in steady state (Appendix E.2 reports $\lambda_\ell \approx 0.07$ post-warmup) and the fraction of unsolved prompts diminishes as training progresses. L_q^* adapts to both prompt and model: harder prompts produce longer correct rollouts and larger L_q^*, and as the model learns to solve a problem more concisely, L_q^* tightens automatically, sustaining compression without a manual curriculum. Using the *mean* rather than the *minimum* of correct lengths (contrast with ShorterBetter's SOL [16]) prevents a single anomalously short rollout from setting an unrealistically aggressive target. Appendix F.9 examines sparse-correct groups and the zero-correct fallback.

Symmetric efficiency reward. Given L_q^*, the efficiency reward is symmetric around the target:

$$r_\ell(o_{q,j}, q) \;=\; \max\left(-1,\; 1 - \frac{\left|\ell_{q,j} - L_q^*\right|}{L_q^*}\right). \tag{9}$$

The reward equals 1 at $\ell_{q,j} = L_q^*$, decreases linearly with deviation, and is clipped at -1. Penalizing under-length is intentional: an over-short "correct" solution often signals a shortcut (pattern-matched answer) rather than reasoning, and rewarding it would reintroduce the over-compression pathology of Section 3.2. Because L_q^* is recomputed each batch from the current correct rollouts, the penalty on a genuinely short-but-valid solution is transient rather than permanent: if the policy actually discovers shorter solutions on prompt q, those rollouts pull L_q^* downward in subsequent updates, and the symmetric form tracks the new optimum. Appendix F.10 examines this target evolution and its interaction with the training cap.

Rollout masks for advantages and controller statistics. The per-rollout efficiency advantage $A_{q,j}^{(\ell)}$ in Eq. (2) and the batch-level controller statistics in Eq. (4) use different masks, which we list explicitly. *(i) Per-rollout advantages.* For every prompt q, r_ℓ is computed for all G rollouts from the prompt's L_q^*, and $\mu_q^{(\ell)}, \sigma_q^{(\ell)}$ are taken over the full group of G. An incorrect rollout whose length is near L_q^* therefore receives a non-trivial efficiency advantage, which the correctness channel offsets through its negative correctness advantage. Using all G rollouts also avoids the singular case $|\mathcal{C}_q|{=}1$, where a within-correct-only standard deviation would be zero. *(ii) Controller statistics.* For the PSI controller only, the per-prompt $\mu_q^{(\ell)}, \sigma_q^{(\ell)}$ are computed over $\mathcal{C}_q$ (a single correct rollout contributes its reward as $\mu_q^{(\ell)}$ with $\sigma_q^{(\ell)}{=}0$), and the outer average runs over the $M' \le M$ prompts with $|\mathcal{C}_q|{\ge}1$. Prompts with $|\mathcal{C}_q|{=}0$ are dropped because their efficiency reward $r_\ell = \ell_{q,j}/B_{\max}$ carries no usable signal about target-length tracking. This mask affects only the global controller weight λ_ℓ, not the per-rollout advantage. Without it, unsolved prompts would inflate CV_ℓ early in training and spuriously up-weight efficiency before correctness is achieved (edge cases are listed in Appendix F.8).

5 Experiment

5.1 Experimental Setup

Dataset and Models. We train all methods on the level 3–5 split of the MATH dataset [42], comprising $8{,}521$ problems. We use DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B [9] as base models. The MATH-500 benchmark used at evaluation time is held out and disjoint from the training pool.

Baselines. We evaluate LEAD against state-of-the-art RL baselines for LLM reasoning and correctness–length optimization. **GRPO [9]** aggregates rewards into a single scalar before group-wise normalization, serving as our foundation for standard and statically-weighted multi-objective training. **GDPO [24]** mitigates scale dominance by normalizing objectives independently before aggregation, but relies on a static combination of reward weights. **DRPO [18]** decouples learning signals via a discriminative framework, applying a fixed, global length penalty exclusively to correct responses to encourage conciseness without inverting validity. **ShorterBetter [16]** penalizes deviation from a dynamic Sample Optimal Length (SOL) target, but uses fixed-weight GRPO, lacking the ability to dynamically shift optimization focus as accuracy stabilizes.

Implementation. We implement LEAD on top of the Verl framework [43] with vLLM [44] as the rollout engine. Training hyperparameters (optimizer, batch sizes, KL, PPO clip, G, epochs) and LEAD's controller settings are listed in Appendix E.1, Table 4. The 7B runs use smaller batches and a lower learning rate than the 1.5B runs but share the rollout, clip, and LEAD blocks. We set $L_{\min}{=}1{,}000$ because the training distribution is MATH Level 3–5, where even the easiest correct rollouts span several hundred to a few thousand tokens. Other method-specific hyperparameters follow the original baselines, and Appendix F.1 records the baseline optimizer settings.

Evaluation. We evaluate all models with five math reasoning benchmarks: AIME 2024, AIME 2025, AMC 2023, MATH-500 [42], and OlympiadBench [45]. Following the sampling setup of

Table 1: Performance comparison across methods on DeepSeek-R1-Distill-Qwen-1.5B and -7B trained with a 4K maximum response length. AES denotes the Accuracy-Efficiency Score. For each model, **bold** marks the best Acc / AES among trained methods and underline marks the second best.

Model	AIME24		AIME25		AMC23		MATH		OlyBch		Average		AES ↑
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	
DeepSeek-R1-Distill-Qwen-1.5B													
Base	29.33	13018	24.00	13126	69.25	7472	84.93	4080	51.75	8370	51.85	9213	–
GRPO	23.67	2723	19.67	2227	65.75	1634	82.27	903	49.23	1747	48.12	1847	0.08
GDPO	20.00	2232	16.67	1837	61.25	1545	80.80	957	48.74	1645	45.49	1643	−0.40
ShorterBetter	24.67	5361	18.67	5649	65.00	2685	80.40	1466	44.84	3159	46.72	3664	−0.39
DRPO	27.33	3658	21.00	3745	**70.25**	2173	83.27	1254	51.85	2404	50.74	2647	0.50
LEAD (Ours)	**35.00**	5133	**24.33**	4550	67.50	3336	**85.47**	2100	**54.52**	3450	**53.36**	3714	**0.68**
DeepSeek-R1-Distill-Qwen-7B													
Base	53.67	9444	39.00	10263	87.75	4552	93.53	2717	67.06	6181	68.20	6631	–
GRPO	37.67	4245	27.67	3421	83.25	1714	88.93	979	59.01	2049	59.31	2482	−0.68
GDPO	41.33	3159	27.67	2871	80.25	1551	89.73	940	60.05	2335	59.81	2171	−0.56
ShorterBetter	**46.33**	4765	31.33	5345	80.75	1583	83.53	741	60.49	2489	60.49	2985	−0.58
DRPO	44.00	5090	30.67	5232	88.00	2255	92.07	1334	**65.23**	3420	63.99	3466	−0.14
LEAD (Ours)	44.67	6631	**35.00**	7090	**88.75**	2662	**92.27**	1705	65.14	3997	**65.17**	4417	**−0.11**

Sober Reasoning [46], we use temperature 0.8, top-$p = 0.9$, and $n = 10$ decodes per question for AIME24/25 and AMC23, or $n = 3$ for MATH-500 and OlympiadBench. **Accuracy** is avg@n: the mean correctness of the n decodes, not the probability that at least one succeeds (pass@n). **Average Length** includes all responses, correct or incorrect, and excludes prompt tokens. Both headline metrics weight the five benchmark means equally. Evaluation uses a 32,768-token context setting, distinct from the 4,000/8,000-token training caps. Generation and truncation details appear in Appendix F.1. **AES** [15] combines relative accuracy and length changes against the same-scale base, using the coefficients of DRPO [18] (Appendix D).

5.2 Results

Math Reasoning Results. Table 1 reports per-benchmark accuracy, response length, and AES for models trained with a max response length of $4,000$. Across both 1.5B and 7B scales, LEAD achieves the highest average accuracy and AES among all trained methods in Table 1. For the 1.5B model, LEAD is the only method that improves over the base model while reducing average length, reaching 53.36 accuracy and 0.68 AES. Compared with the strongest baseline DRPO, it improves accuracy by 2.62 points and AES by 0.18. For the 7B model, all RL-trained methods reduce length but regress from the base model in accuracy. LEAD has the smallest accuracy drop and the best AES, reaching 65.17 accuracy and -0.11 AES. Although LEAD is not the shortest method, its stronger AES indicates a better accuracy–efficiency trade-off: it preserves more reasoning tokens when they are useful, rather than uniformly compressing all prompts. This is consistent with the per-problem target design in Section 4.2 and the difficulty-based allocation analysis in Appendix E.3.

5.3 Ablation Study

We isolate two central design choices of LEAD with controlled ablations on DeepSeek-R1-Distill-Qwen-1.5B trained on MATH (Level 3–5) under the same evaluation protocol as Section 5: (i) static vs. dynamic reward weighting, paired with the orthogonal scalarized vs. decoupled normalization choice, and (ii) the choice of aggregator for the per-problem target length L_q^*.

Static vs. Dynamic reward weighting. We compare scalarized GRPO, static decoupled weighting, and LEAD's dynamic weighting by sweeping six fixed ratios with $\lambda_c + \lambda_\ell = 1$ and reporting dynamic LEAD as an independent reference. Table 2 shows three trends. First, scalarized GRPO is highly sensitive to the reward ratio: its best AES is only 0.08 at 1:1, with most ratios at or below zero, indicating that combine-then-normalize aggregation makes the length signal difficult to control. Second, every static LEAD ratio achieves AES ≥ 0.27, outperforming the best GRPO setting. Besides separately normalizing the two rewards, static LEAD applies the final token-masked BatchWhiten, which remains active even at 1:0, and the GRPO sweep uses the baseline's optimizer

Table 2: Static vs. dynamic weighting on DeepSeek-R1-Distill-Qwen-1.5B (4K). Both **GRPO** and **LEAD** (static) use static combination $(\lambda_c, \lambda_\ell)$, and **LEAD (dynamic)** adopts $\lambda^{(t)}$ online (i.e., dynamic combination of $(\lambda_c, \lambda_\ell)$).

λ_c:λ_ℓ	**GRPO**			**LEAD (static)**			**LEAD (dynamic)**		
	Acc	Len	AES ↑	Acc	Len	AES ↑	Acc	Len	AES ↑
1 : 0	44.59	2320	−0.65	50.60	3141	0.42			
4 : 1	47.87	1802	0.04	52.74	4013	0.62			
2 : 1	47.24	1689	−0.07	52.16	3586	0.63			
1 : 1	48.12	1847	0.08	51.46	5177	0.36	**53.36**	3714	**0.68**
1 : 2	46.11	1527	−0.27	52.26	5819	0.39			
1 : 4	47.27	1670	−0.07	51.08	5322	0.27			

Table 3: Choice of aggregator for L_q^*. All variants restrict to correct rollouts $\mathcal{C}_q$ except *Mean of all rollouts*, the unfiltered baseline.

Aggregator	Definition	Avg Acc	Avg Len	AES ↑
Min of correct [16]	$\min_{j\in\mathcal{C}_q} \ell_{q,j}$	49.99	2611	0.36
Median of correct	$\mathrm{med}_{j\in\mathcal{C}_q} \ell_{q,j}$	50.13	2990	0.34
Mean of all rollouts	$\mathrm{mean}_j \ell_{q,j}$	51.47	3230	0.58
Mean of correct (LEAD)	$\mathrm{mean}_{j\in\mathcal{C}_q} \ell_{q,j}$	**53.36**	3714	**0.68**

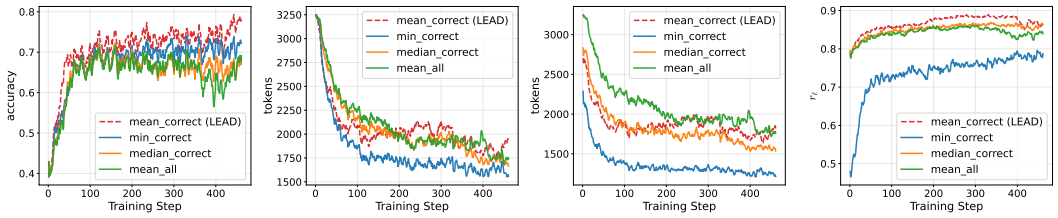
settings, so the gap at 1:0 reflects these pipeline differences rather than reward decoupling alone (Appendices F.7 and F.1). Third, dynamic LEAD achieves the best overall trade-off without manually selecting a ratio, reaching AES 0.68 and accuracy 53.36, above the best static setting. The dynamic weights explain this behavior: λ_ℓ starts near 0.5 and decays to ≈ 0.07 as the efficiency signal saturates, allowing LEAD to first learn compression and then shift optimization toward correctness (shown in Appendix E.2). Section 5.4 compares PSI with hand-designed schedules under a matched configuration.

Choice of L_q^* aggregator. We vary the aggregator: the mean of correct rollouts (LEAD), the minimum (ShorterBetter's SOL [16]), the median, and a degenerate baseline averaging over *all* rollouts regardless of correctness. As shown in Table 3, LEAD wins on both Acc (53.36) and AES (0.68). Min (Min of correct) compresses most aggressively, but a single short outlier sets an unrealistically tight target, causing accuracy to drop by over 3 points. Median (Median of correct) yields the worst AES (0.34): its length sits between Min and Mean-of-all, yet its accuracy is no better than Min's. Mean-of-all is the closest competitor, but trails LEAD by 1.9 accuracy points because the inflation from incorrect rollouts is absorbed by off-track or truncated trajectories rather than valid reasoning. Figure 2 explains this behavior through the training dynamics. LEAD pulls ahead in batch accuracy in (a) while Min lags. Rollout lengths in (b) converge for Median, LEAD, and Mean-of-all, while Min compresses further. The targets L_q^* in (c) stay ordered Min<Median<LEAD<Mean-of-all. Mean-of-all's target runs ∼150–200 tokens above LEAD's, even though their rollout lengths converge in (b), so the gap is absorbed by incorrect rollouts rather than reshaping the target on solvable problems. The diagnostic is (d): Median, LEAD, and Mean-of-all all reach $r_\ell \approx 0.85$, so correct trajectories land near L_q^*. Min plateaus at ≈ 0.78, since even its own correct rollouts cannot meet such a tight target, so the efficiency penalty fights correctness, explaining Min's lag in (a).

5.4 Additional Results

Full tables and protocol details for the following experiments are in Appendix F.

Training-run variability. Re-training GRPO, GDPO, DRPO, and LEAD on DeepSeek-R1-Distill-Qwen-1.5B (4K) with three data-order seeds gives LEAD a mean accuracy of 52.93 ± 0.91 and AES of 0.67 ± 0.05, against 50.82 ± 0.67 and 0.50 ± 0.12 for DRPO, the strongest baseline (Table 6). LEAD's worst seed exceeds DRPO's three-seed mean. Paired question-bootstrap intervals for the Table 1 models exclude zero at 1.5B but include zero at 7B (Appendix F.2).



(a) Batch Accuracy (b) Mean Response Length (c) Mean L_q^* (solvable prompts) (d) Mean r_ℓ (correct rollouts)

Figure 2: **Training trajectories of the four aggregator variants on DeepSeek-R1-Distill-Qwen-1.5B. (a)** On-policy batch accuracy. **(b)** Mean response length on the rollout batch. **(c)** Per-problem target L_q^* averaged over solvable prompts. **(d)** Symmetric efficiency reward r_ℓ averaged over correct rollouts.

Comparison with pre-defined weighting schedules. We compare our PSI controller with some pre-defined weighting schedule settings. A linear decay of λ_ℓ from 0.5 to 0.05 over the first 50 updates is competitive with PSI at step 300 and worse at step 450. A two-phase schedule that switches from equal weights to correctness only after 50 updates is also included in the comparison and is shown as the worst setting. PSI therefore exceeds a well-chosen schedule without specifying it in advance and continues to improve both accuracy and length later in training.

Comparison with released LASER-family checkpoints. Under the same evaluation protocol, LEAD is more accurate and shorter than all nine released 1.5B LASER, LASER-D, and LASER-DE checkpoints (best LASER AES 0.53 versus 0.68). At 7B, the two released adaptive checkpoints are more accurate and shorter than LEAD. Since the training protocols differ (LASER uses the larger DeepScaleR training dateset and a larger 16K cap), so these are comparisons of released models rather than matched training runs (Appendix F.4). However, LEAD is still competitive with more restricted training setups.

Cross-family transfer with LoRA. We compare LEAD-LoRA and GRPO-LoRA on Qwen3-4B [47]. GRPO combines correctness and its scalar length reward at a $1{:}1$ ratio. Both LEAD-LoRA and GRPO-LoRA share the same 4,000-token training cap and training horizon. In Table 14, LEAD retains more accuracy than GRPO ($+3.06$ points) and achieves higher AES, while producing longer responses than GRPO and shortening the base model's responses by 30.5%. More details are shown in Appendix F.5.

6 Conclusion

We introduced **LEAD**, an online self-calibrating framework for length-efficient reasoning motivated by two challenges in RL-based reasoning optimization: the changing correctness–efficiency trade-off during training and the heterogeneous reasoning demands of different prompts. LEAD combines a Potential-Scaled Instability controller, which adapts $(\lambda_c, \lambda_\ell)$ from per-reward instability and remaining headroom, with a per-problem target length L_q^* estimated from the model's own correct rollouts under a symmetric efficiency reward. Across five mathematical reasoning benchmarks, LEAD achieves the best Accuracy-Efficiency Score among GRPO, GDPO, DRPO, ShorterBetter, and their length-control variants, without requiring a hand-tuned $(\lambda_c, \lambda_\ell)$ schedule. Ablations show that dynamic weighting improves over the evaluated static reward ratios, and the mean-of-correct target outperforms the minimum, median, and correctness-blind alternatives. Overall, LEAD is a step toward reasoning models that adapt their computational footprint online to both problem difficulty and optimization progress, improving efficiency without sacrificing the reasoning performance.

7 Acknowledgment

This work was partially supported by NSF 2343863, 2413520, 2417747 and 2440611. Any opinions, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[2] 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.

[3] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[4] 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. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[5] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690, 2024.

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[7] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

[8] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[10] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

[11] Junyi Chen, Chuheng Du, Renyuan Liu, Shuochao Yao, Dingtian Yan, Jiang Liao, Shengzhong Liu, Fan Wu, and Guihai Chen. Tokenflow: Responsive llm text streaming serving under request burst via preemptive scheduling. *arXiv preprint arXiv:2510.02758*, 2025.

[12] Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

[13] 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.

[14] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[15] 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.

[16] Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *arXiv preprint arXiv:2504.21370*, 2025.

[17] Xingyang He, Xiao Ling, and Jie Liu. Smartthinker: Learning to compress and preserve reasoning by step-level length control. *arXiv preprint arXiv:2507.04348*, 2025.

[18] Gang Li, Yan Chen, Ming Lin, and Tianbao Yang. Drpo: Efficient reasoning via decoupled reward policy optimization. *arXiv preprint arXiv:2510.04474*, 2025.

[19] Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. In *International Conference on Learning Representations*, 2026.

[20] Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. Selfbudgeter: Adaptive token allocation for efficient llm reasoning. *arXiv preprint arXiv:2505.11274*, 2025.

[21] Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, 2025.

[22] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[23] Chengyu Huang et al. HAPO: History-aware policy optimization for efficient reasoning. *arXiv preprint arXiv:2505.11225*, 2025.

[24] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

[25] Yining Lu, Zilong Wang, Shiyang Li, Xin Liu, Changlong Yu, Qingyu Yin, Zhan Shi, Zixuan Zhang, and Meng Jiang. Learning to optimize multi-objective alignment through dynamic reward weighting. *arXiv preprint arXiv:2509.11452*, 2025.

[26] 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.

[27] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[28] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, 2025.

[29] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[30] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(2):26, 2022.

[31] Michael Kleinman, Matthew Trager, Alessandro Achille, Wei Xia, and Stefano Soatto. e1: Learning adaptive control of reasoning effort. *arXiv preprint arXiv:2510.27042*, 2025.

[32] Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. Plan and budget: Effective and efficient test-time scaling on large language model reasoning. *arXiv preprint arXiv:2505.16122*, 2025.

[33] Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. The art of scaling test-time compute for large language models. *arXiv preprint arXiv:2512.02008*, 2025.

[34] Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. Optimizing length compression in large reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3232–3250. Association for Computational Linguistics, 2026.

[35] Jinyi Han, Ying Huang, Ying Liao, Haiquan Zhao, Zishang Jiang, Xinyi Wang, Xikun Lu, Guanghao Zhou, Sihang Jiang, Jiaqing Liang, Weikang Zhou, Zeye Sun, Fei Yu, and Yanghua Xiao. Your models have thought enough: Training large reasoning models to stop overthinking. In *International Conference on Learning Representations*, 2026.

[36] Benteng Chen, Weida Wang, Shufei Zhang, Mingbao Lin, and Min Zhang. Step-GRPO: Internalizing dynamic early exit for efficient reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21710–21724. Association for Computational Linguistics, 2026.

[37] Yongjiang Liu, Haoxi Li, Xiaosong Ma, Jie Zhang, and Song Guo. Think how to think: Mitigating overthinking with autonomous difficulty cognition in large reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 38105–38126. Association for Computational Linguistics, 2026.

[38] Nathanaël Carraz Rakotonirina, Ren Pang, Neha Anna John, Michael Bohlke-Schneider, and Momchil Hardalov. Correct, concise and complete: Multi-stage training for adaptive reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 12798–12810. Association for Computational Linguistics, 2026.

[39] Susmit Das. TIME: Temporally intelligent meta-reasoning engine for context-triggered explicit reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 38872–38906. Association for Computational Linguistics, 2026.

[40] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.

[41] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[42] 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.

[43] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[44] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[45] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, 2024.

[46] Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *arXiv preprint arXiv:2504.07086*, 2025.

[47] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

A Limitations and Broader Impact

A.1 Limitations

LEAD is designed for reinforcement learning settings where correctness can be reliably evaluated and multiple rollouts can be sampled per prompt. This makes it well suited to mathematical reasoning, where answer verification is relatively precise, but extending the same framework to open-ended generation, instruction following, or subjective preference tasks may require task-specific reward models or validators. We view this as a natural extension rather than a limitation of the core mechanism, since the dynamic weighting and per-problem calibration components are agnostic to the particular reward source.

LEAD estimates its target length from the model's own correct rollouts. This design intentionally prioritizes correctness before compression: prompts that the model cannot yet solve do not receive target-based compression pressure until at least one correct trajectory appears. As a result, efficiency gains on very hard prompts may emerge later in training than on easier prompts. However, this behavior is desirable for reasoning tasks, since premature compression on unsolved problems can suppress the exploration needed to discover valid solutions.

A related limitation is that the target is self-referential. Because it is estimated from the policy's own correct rollouts, it can stabilize above an unknown minimum sufficient reasoning length, particularly when successful trajectories cluster tightly around a behavioral floor. Within-group length variance may then shrink, PSI may down-weight the efficiency channel, and further compression can stall. Appendix F.11 measures how often such tight high-length clusters occur in our runs. They are uncommon in the evaluated setting, but the possibility should be kept in mind when LEAD is applied to other models or domains. Targets estimated from only one or two correct rollouts can also be noisy (Appendix F.9), and PSI uses reward variability and headroom as heuristic proxies for learnability rather than exact measures of policy-gradient utility. Our cross-family evidence is a Qwen3-4B LoRA study in which both trained methods lose accuracy relative to the base under the 4K cap, and transfer to larger models such as Qwen3-32B remains future work.

The current formulation uses a single scalar target length for each prompt, which is appropriate for math reasoning where correct solutions often cluster around a problem-dependent reasoning budget. Some tasks may admit multiple valid solution styles with substantially different lengths, such as concise direct answers and longer explanatory derivations. Extending LEAD from a single target to a distributional or multi-target length model is an interesting direction for future work.

Finally, LEAD improves length efficiency through training-time policy optimization rather than enforcing a hard inference-time token budget. Therefore, it should be viewed as a method for learning more efficient reasoning behavior, not as a replacement for deployment-time budget controllers when strict latency or cost constraints are required. Combining LEAD with test-time budget allocation may further improve efficiency under fixed resource limits.

A.2 Broader Impact

LEAD aims to reduce unnecessary reasoning tokens while preserving task performance. By shortening redundant reasoning trajectories, it can lower inference cost, latency, and energy use in deployments where reasoning models are queried at scale. Because LEAD is a reward-aggregation and length-calibration method, it does not directly introduce new task capabilities beyond those induced by the underlying RL training objective. At the same time, improving reasoning efficiency may change the form of model outputs: shorter responses can be harder for users to inspect when detailed explanations are needed. For applications where transparency, debugging, or human verification is important, we recommend retaining longer reasoning traces, auxiliary logs, or configurable verbosity settings in controlled settings. More broadly, LEAD should be deployed with the same safeguards as the base

Algorithm 1 LEAD advantage computation at training step t

Require: Rollouts $\{o_{q,j}\}$, correctness $\{r_c(o_{q,j}, q)\}$, lengths $\{\ell_{q,j}\}$, prior pre-floor EMA state $\boldsymbol{u}^{(t-1)}$

1: ***Phase 1: Online Target-Length Calibration & Decoupled Normalization***
2: **for** each prompt q in the current batch **do**
3: Compute per-prompt target L_q^* from correct rollouts via Eq. (8)
4: Compute symmetric efficiency rewards $r_\ell(o_{q,j}, q)$ for all $j \in \{1 \dots G\}$ via Eq. (9)
5: Compute decoupled group-relative advantages $A_{q,j}^{(c)}$ and $A_{q,j}^{(\ell)}$ via Eq. (2)
6: **end for**
7: ***Phase 2: Dynamic PSI Controller***
8: Compute batch-level reward statistics μ_k, σ_k via Eq. (4) (for $k{=}\ell$, restrict to correct rollouts $\mathcal{C}_q$ and drop prompts with $|\mathcal{C}_q|{=}0$; the per-rollout advantage in line 5 is unchanged and uses all G rollouts)
9: Compute instability CV_k, potential headroom P_k, and PSI Ψ_k via Eqs. (5)–(6)
10: Derive target weights $\hat{\boldsymbol{\lambda}}^{(t)}$, update and retain $\boldsymbol{u}^{(t)}$ via Eq. (7)
11: Apply $\lambda_c^{(t)} = \max(u_c^{(t)}, \lambda_{\min})$, $\lambda_\ell^{(t)} = 1 - \lambda_c^{(t)}$
12: ***Phase 3: Advantage Aggregation***
13: Combine signals into scalar advantage $\tilde{A}_{q,j} = \lambda_c^{(t)} A_{q,j}^{(c)} + \lambda_\ell^{(t)} A_{q,j}^{(\ell)}$
14: **return** Final advantages $A_{q,j} = \mathrm{BatchWhiten}(\tilde{A}_{q,j})$

reasoning model, since efficiency improvements can also make both beneficial and harmful uses cheaper to run.

B Full GRPO Objective

Following the notation of Section 3, the policy is updated by *minimizing* the loss $\mathcal{L}_{\mathrm{GRPO}}$ defined below, which equals the negative of the clipped PPO-style surrogate over the group-relative advantage $A_{q,j}$ of Eq. (1) plus a KL-divergence regularizer against a reference policy π_{ref}. Minimizing $\mathcal{L}_{\mathrm{GRPO}}$ therefore maximizes the surrogate while penalizing deviation from π_{ref}:

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) \;=\; -\,\mathbb{E}_{q\sim\mathcal{D},\,\{o_{q,j}\}\sim\pi_{\theta_{\text{old}}}}\left[\frac{1}{G}\sum_{j=1}^{G}\frac{1}{|o_{q,j}|}\sum_{t=1}^{|o_{q,j}|}\mathcal{J}_{q,j,t}\right] + \beta\,\mathbb{D}_{\mathrm{KL}}\big[\pi_\theta \,\|\, \pi_{\text{ref}}\big], \tag{10}$$

$$\begin{aligned}\mathcal{J}_{q,j,t} \;&=\; \min\Big(\rho_{q,j,t}(\theta)\,A_{q,j},\;\; \mathrm{clip}\big(\rho_{q,j,t}(\theta),\,1-\varepsilon,\,1+\varepsilon\big)\,A_{q,j}\Big),\\ \rho_{q,j,t}(\theta) \;&=\; \frac{\pi_\theta(o_{q,j,t}\mid q, o_{q,j,<t})}{\pi_{\theta_{\text{old}}}(o_{q,j,t}\mid q, o_{q,j,<t})},\end{aligned} \tag{11}$$

where $\rho_{q,j,t}(\theta)$ is the per-token importance ratio, ε is the clipping threshold, and β controls the KL penalty strength. LEAD inherits this objective unchanged; the only modification is the construction of $A_{q,j}$ described in Section 4.

C LEAD Algorithm

Algorithm 1 summarizes the LEAD advantage computation at a single training step, combining the per-reward decoupled normalization (Section 4.1), the PSI-driven dynamic weighting (Eqs. (6)–(7)), and the per-problem online target length (Section 4.2).

D Accuracy-Efficiency Score

The Accuracy-Efficiency Score [15] jointly measures accuracy preservation and length reduction relative to a reference model (the base model before RL training):

$$\mathrm{AES} = \begin{cases} \alpha\cdot\Delta_{\text{Len}} + \beta\cdot|\Delta_{\text{Acc}}|, & \text{if } \Delta_{\text{Acc}} \geq 0,\\ \alpha\cdot\Delta_{\text{Len}} - \gamma\cdot|\Delta_{\text{Acc}}|, & \text{if } \Delta_{\text{Acc}} < 0,\end{cases} \tag{12}$$

where $\Delta_{\text{Len}} = \frac{L_{\text{ref}} - L_{\text{model}}}{L_{\text{ref}}}$ measures relative length reduction, $\Delta_{\text{Acc}} = \frac{A_{\text{model}} - A_{\text{ref}}}{A_{\text{ref}}}$ measures relative accuracy change, and we use α=1, β=3, γ=10 following [18], with a large γ to emphasize the importance of minimizing accuracy degradation. Higher AES indicates a better accuracy–efficiency trade-off.

E Additional Experiment

This appendix collects full hyperparameter specifications and additional empirical results, including training dynamics and complementary results.

E.1 Full Hyperparameter Specification

Table 4 records the reported LEAD configurations. It is not a universal configuration for all baseline rows: baseline optimizer, filtering, and horizon choices differ. Appendix F.1 distinguishes original comparisons from the matched schedule runs.

Table 4: Reported LEAD training configurations for DeepSeek-R1-Distill-Qwen-1.5B and -7B. "Shared" refers to these two columns, not all baseline methods.

Setting	**1.5B**	**7B**
Optimizer / rollout		
Learning rate	2×10^{-6}	1×10^{-6}
Training batch size (prompts)	128	32
PPO mini-batch size	16	8
PPO micro-batch size	16	4
KL loss coefficient	5×10^{-4}	1×10^{-3}
Total epochs / steps	7 / 462	2 / $\sim$532
Max response length $B_{\max}$	4,000 (4K) / 8,000 (8K)	4,000
Rollouts per prompt G	8 (shared)	
Rollout temperature	1.0 (shared)	
PPO epochs per update	1 (shared)	
PPO clip ratio ε	0.2 (Verl default; ratio range $[0.8, 1.2]$)	
Max prompt length	1,024 (shared)	
Outer KL controller coefficient	1×10^{-3} (shared, low-variance KL)	
LEAD-specific (shared across scales)		
Potential exponent α	1.0	
EMA smoothing β_{ema}	0.95	
Numerical regularizer ϵ	10^{-8}	
Correctness floor $\lambda_{\min}$	0.3	
Target floor $L_{\min}$	1,000	
Initial weights $\boldsymbol{\lambda}^{(0)}$	$(0.5, 0.5)$	

E.2 Training dynamics on DeepSeek-R1-Distill-Qwen-1.5B

Figure 3 reports three signals over the course of a single LEAD run on DeepSeek-R1-Distill-Qwen-1.5B (290 logged steps): (a) the dynamic weights $\lambda_c^{(t)}, \lambda_\ell^{(t)}$, (b) the per-prompt L_q^* statistics (mean and min–max range across solvable prompts, plus the count of unsolved prompts assigned the sentinel $B_{\max}$), and (c) the rolling mean response length and validation accuracy on MATH-500. Together they show that LEAD behaves as designed: the controller smoothly reweights the two objectives without manual scheduling, L_q^* tightens online as the model improves, and length drops while accuracy keeps rising.

The trajectory of λ_ℓ in Figure 3(a) reveals a key mechanism: *the efficiency reward acts as a transient curriculum signal*. From the uniform initialization $(\lambda_c, \lambda_\ell)$=$(0.5, 0.5)$ the controller shifts rapidly toward correctness in the early "learn-to-solve" phase, reaching $\lambda_\ell \approx 0.08$ by step 50 and plateauing near $\lambda_c \approx 0.93$, $\lambda_\ell \approx 0.07$ for the remainder of training. This early window is precisely when the bulk of length compression occurs in (c) and when L_q^* tightens most rapidly in (b). Once a concise

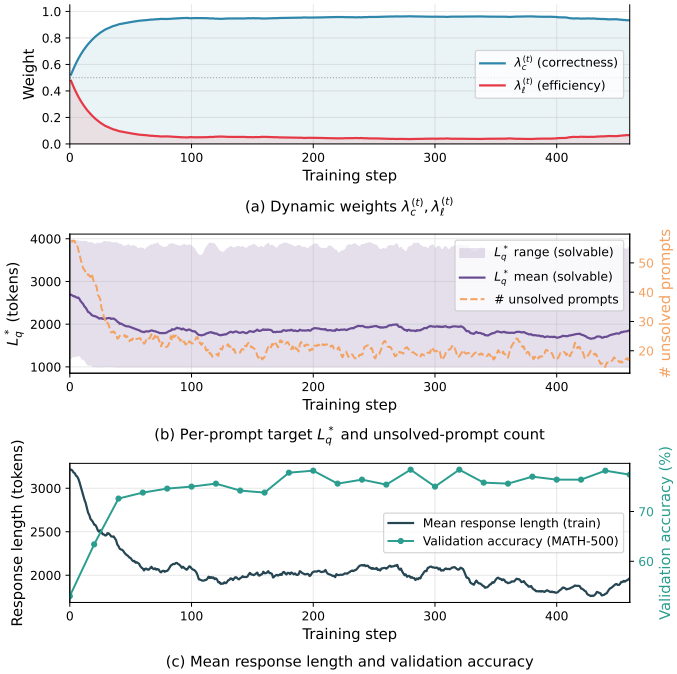
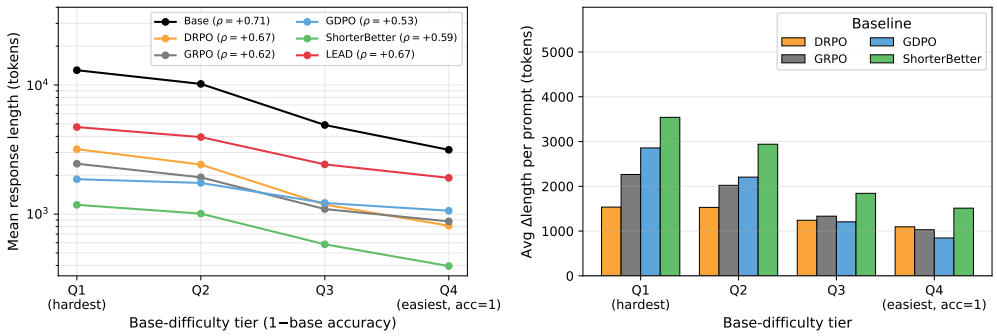


Figure 3: **Training dynamics of a LEAD run on DeepSeek-R1-Distill-Qwen-1.5B** (4K budget, $B_{\max}$=4,000). **(a)** Dynamic weights $\lambda_c^{(t)}, \lambda_\ell^{(t)}$. **(b)** Per-prompt L_q^* statistics across solvable prompts (mean and min–max range) and the count of unsolved prompts (assigned $B_{\max}$). **(c)** Rolling mean response length on the rollout batch and validation accuracy on MATH-500.

reasoning style is established and the efficiency reward saturates (its within-group CV collapses), the controller automatically holds gradient capacity on correctness, where signal still remains. Crucially, this is not equivalent to training without a length reward: if we had ablated the efficiency term entirely from the start, the model would never have learned the early length-compression behavior that makes the late-phase correctness-only optimization possible.

E.3 Token allocation by prompt difficulty

We probe whether LEAD allocates its token budget non-uniformly across prompts, as the per-problem L_q^* design predicts. We define the difficulty of an evaluation prompt q as $\mathrm{diff}(q) = 1 - \mathrm{acc}_{\mathrm{base}}(q)$, where $\mathrm{acc}_{\mathrm{base}}(q)$ is the avg@n accuracy of the *unmodified base* DeepSeek-R1-Distill-Qwen-1.5B on q (using the same n and decoding settings as the main evaluation: n=10 for AIME 2024/25 and AMC 2023, n=3 for MATH-500 and OlympiadBench, temperature 0.8, top-p 0.9). Difficulty is defined externally via the base model so it does not depend on which method we are scoring. We pool the 1,275 prompts across the five benchmarks, rank by base accuracy (with ties broken by first-appearance ordering), and partition into four ordered tiers: Q_1 (hardest, base pass-rate 0, 255 prompts) through Q_3, plus Q_4 (easiest, base pass-rate 1, 510 prompts). We collapse the perfect-pass tier into a single bin because avg@n is discrete and roughly 40% of prompts hit acc=1, so a fifth quantile bin would split tied prompts arbitrarily by benchmark composition rather than by difficulty. Figure 4(a) plots the mean response length per tier (log scale). The base model's strong positive dependence between difficulty and length (Spearman $\rho = +0.71$, computed per-prompt) reflects the natural fact that harder problems require more reasoning. LEAD preserves this structure ($\rho = +0.67$), staying closer to the base curve than any compression baseline; GDPO ($\rho = +0.53$) flattens the curve the most. Figure 4(b) makes the allocation gap explicit: for each baseline B we plot the average per-prompt $\Delta\ell(q) = \ell_{\mathrm{LEAD}}(q) - \ell_B(q)$ within each tier. The extra tokens LEAD spends are concentrated on the hardest prompts (+1,540 to +3,540 tokens on Q_1 vs. baselines, dropping to +850 to +1,510 on Q_4). The skew toward hard prompts is most pronounced against the strongest compression baselines (GDPO 3.4×, ShorterBetter 2.3×, GRPO 2.2×); against DRPO, which already preserves length on hard problems, LEAD's extra spend is closer to uniform (1.4×). This is the mechanism behind



(a) Length grows with difficulty

(b) LEAD's extra spend vs each baseline

Figure 4: **Per-prompt token allocation by base-difficulty tier** on the pooled 5-benchmark eval set (1,275 prompts). Difficulty is $1 - \mathrm{acc}_{\mathrm{base}}(q)$ from the unmodified base model. Prompts are grouped into four tiers by base pass-rate; Q_4 collapses all $\mathrm{acc}{=}1$ prompts (510 prompts) into one bin to avoid an arbitrary task-driven split of tied perfect-pass prompts. **(a)** Mean response length per tier, with Spearman ρ between difficulty and length in the legend (higher ρ = greater difficulty-sensitivity). LEAD ($\rho{=}+0.67$) tracks the base model ($\rho{=}+0.71$) most closely; GDPO ($\rho{=}+0.53$) compresses most uniformly. **(b)** Average per-prompt extra tokens spent by LEAD over each baseline, $\Delta\ell(q) = \ell_{\mathrm{LEAD}}(q) - \ell_B(q)$, averaged within each tier. Against the strongest compression baselines (GDPO, ShorterBetter, GRPO), the gap on the hardest tier is 2.2–3.4$\times$ larger than on the easiest, showing that LEAD's extra tokens go preferentially to harder prompts. Note: this panel aggregates per-prompt across the 1,275-prompt evaluation set (so MATH-500 and OlympiadBench, which are larger, dominate the overall mean), whereas Table 1's "Average Length" is averaged across the five benchmarks with equal weight; the two aggregations can disagree on which method is shorter overall when per-task mean lengths differ widely.

LEAD's higher AES despite its longer average length: LEAD does not blanket-compress; it rations its budget by per-problem difficulty.

Computation of ρ. The Spearman ρ values shown in the legend of Figure 4(a) are computed at *prompt granularity* across the 1,275-prompt evaluation set, not from the four tier means in the plot (the tiers are display-only). For each method m, we rank both the per-prompt difficulty vector $\mathrm{diff}(q)$ and the per-prompt mean-length vector $\bar{\ell}_m(q) = \frac{1}{n_q}\sum_j \ell_{m,q,j}$ (ties broken by first-appearance order), and take the Pearson correlation between the two rank vectors, which equals the standard Spearman rank correlation. ρ is invariant under monotone rescaling of either axis, so it is unaffected by the log-axis in the plot. $\rho{=}+1$ means the method's per-prompt response length perfectly tracks base difficulty; $\rho{=}0$ indicates no monotonic rank association and does not by itself imply uniform response lengths.

E.4 Results at 8K budget on DeepSeek-R1-Distill-Qwen-1.5B

We additionally train and evaluate every method on DeepSeek-R1-Distill-Qwen-1.5B with the training-time max response length raised from 4K to 8K, keeping all other hyperparameters and the evaluation protocol identical to Section 5.1. As shown in Table 5, the same conclusions as the 4K main results hold: LEAD again posts the highest accuracy (54.44, +2.59 over the base) and the highest AES (0.54). GDPO closes the AES gap to 0.01 but still trails LEAD by 3.17 accuracy points. ShorterBetter remains the weakest, with a 7.3-point accuracy regression and the worst AES among baselines, as forcing every correct rollout toward a single minimum length still over-compresses on hard problems. Appendix F.2 reports accuracy confidence intervals for these rows, and Appendix F.6 re-evaluates the public 8K checkpoint.

Table 5: Performance comparison across methods on DeepSeek-R1-Distill-Qwen-1.5B at the 8K training-time max response length. AES denotes the Accuracy-Efficiency Score. **Bold** marks the best Acc / AES among trained methods and underline marks the second best (the Base row is a reference and excluded from ranking).

Model	AIME24		AIME25		AMC23		MATH		OlyBch		Average		AES ↑
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	
Base	29.33	13018	24.00	13126	69.25	7472	84.93	4080	51.75	8370	51.85	9213	–
GRPO	22.67	4275	20.00	3766	71.50	2607	84.80	1572	51.16	2813	50.03	3007	0.32
GDPO	29.33	4736	21.33	4058	67.00	2795	85.20	1665	53.48	3005	51.27	3252	0.53
ShorterBetter	22.67	4305	14.67	3934	62.00	1948	78.93	1109	44.69	2271	44.59	2713	−0.69
DRPO	28.00	4472	22.33	4779	70.00	3665	83.13	6005	53.23	4255	51.34	4635	0.40
LEAD (Ours)	**30.33**	8219	**26.00**	7874	**73.50**	4368	**86.53**	2624	**55.85**	5015	**54.44**	5620	**0.54**

F Additional protocol details and robustness evidence

F.1 Evaluation and training-comparison conventions

Accuracy and response length. For benchmark b with N_b questions, let $y_{bis} \in \{0, 1\}$ denote correctness of decode s for question i. We compute

$$a_{bi} = \frac{1}{n_b} \sum_{s=1}^{n_b} y_{bis}, \qquad A_b = \frac{100}{N_b} \sum_{i=1}^{N_b} a_{bi}, \qquad A = \frac{1}{5} \sum_{b=1}^{5} A_b. \tag{13}$$

This is avg@n, not pass@n. The five tasks are AIME24, AIME25, AMC23, MATH-500, and OlympiadBench. Minerva outputs generated by the shared harness are excluded from the paper aggregate. Length is averaged across all scored response texts within a benchmark, then equally across benchmarks. The aggregation script re-tokenizes response text using the DeepSeek-R1-Distill-Qwen-1.5B tokenizer with special-token insertion disabled. It excludes the prompt and includes incorrect responses. This text-token count is distinct from the training response-mask length.

Evaluation budget and scoring. The frozen evaluator uses temperature 0.8, top-$p = 0.9$, seed 0, and nominal settings of 32,768 for both maximum new tokens and model context length. The context budget includes prompt tokens, which reduce the available completion budget. Generation also stops at the model's termination condition. Responses reaching the length limit are retained and graded from their emitted text using the same answer-extraction and verification pipeline. We retain the runtime configuration, tokenizer, prompt template, and generated outputs for reproducibility.

Training configurations. The original training comparisons share data, base models, and nominal training-response caps, with method-specific reward and optimization settings. The 1.5B scalar-reward GRPO sweep uses learning rate 10^{-6}, upper PPO clip parameter 0.28, group filtering, and a fixed-budget indicator reward. Dynamic LEAD uses learning rate 2×10^{-6}, upper clip 0.2, no group filtering, and the symmetric target reward. The matched Linear decay schedule/PSI pair uses the same data, seed 42, four-L40S topology, learning rate 2×10^{-6}, clip parameters $0.2/0.2$, no group filtering, 128 prompts, $G = 8$, 4,000-token cap, and seven-epoch training configuration. Only the channel-weight controller changes in this pair.

F.2 Data-order-seed variation and question uncertainty

Three-run summaries. Table 6 summarizes the three runs and Table 7 gives the individual runs. The varied seed controls training data order. Seed-42 rows retain their original precision, and new aggregates use the five per-benchmark results before final table rounding. AES is computed per run against $51.85/9213$ and then averaged. GRPO's mean AES is slightly negative before rounding (-0.002).

We use sample SD with denominator $n - 1$ and the Student-t interval $\bar{x} \pm t_{0.975,2} s/\sqrt{3}$, where $t_{0.975,2} = 4.303$. Table 8 reports the resulting 95% confidence intervals. With three runs, these intervals are wide and depend on the Student-t assumptions. LEAD's mean accuracy exceeds DRPO's by 2.11 points.

Table 6: Results over three data-order seeds on DeepSeek-R1-Distill-Qwen-1.5B with a 4K training cap, reported as mean ± sample SD. **Bold** marks the best mean Acc / AES and underline marks the second best.

Method	Avg Acc	Avg Len	AES
GRPO	47.82 ± 1.25	2078 ± 219	0.00 ± 0.24
GDPO	46.25 ± 1.26	1598 ± 195	-0.25 ± 0.23
DRPO	$\underline{50.82 \pm 0.67}$	2752 ± 124	$\underline{0.50 \pm 0.12}$
LEAD	$\mathbf{52.93 \pm 0.91}$	3589 ± 108	$\mathbf{0.67 \pm 0.05}$

Table 7: Individual 1.5B/4K data-order-seed runs. Each cell is Avg Acc / Avg Len / AES.

Method	Seed 42 (submitted)	Seed 43	Seed 44
GRPO	48.12 / 1847 / 0.080	48.89 / 2282 / 0.182	46.46 / 2104 / -0.269
GDPO	45.49 / 1643 / -0.405	45.56 / 1385 / -0.364	47.71 / 1767 / 0.009
DRPO	50.74 / 2647 / 0.499	51.53 / 2889 / 0.624	50.19 / 2719 / 0.384
LEAD	53.36 / 3714 / 0.684	51.88 / 3526 / 0.619	53.55 / 3528 / 0.715

Paired question bootstrap. For each comparison, we resample questions separately within each benchmark, retaining the complete avg@n score of each sampled question. We use the same sampled question indices for the two methods, compute five benchmark differences, and average them equally. The 2.5th and 97.5th percentiles of 10,000 resamples (seed 0) give Table 9. These intervals quantify question-resampling uncertainty for the evaluated models and sampled decodes, complementing the training-run summaries above. The 7B difference is rounded from per-question results and can differ by 0.01 from subtracting displayed table cells.

Both 1.5B intervals exclude zero, supporting LEAD's accuracy gains over these baselines under question resampling. The 7B interval includes zero.

Table 10 provides the marginal accuracy intervals for the original 4K and 8K comparisons. These accompany the submitted point estimates. Recomputing from per-question scores can change a displayed average by 0.01 through rounding.

F.3 Matched schedule comparisons

The Linear decay schedule sets $\lambda_\ell(t) = 0.5 - 0.45 \min((t-1)/49, 1)$ for training updates $t \geq 1$ and $\lambda_c(t) = 1 - \lambda_\ell(t)$: the efficiency weight is 0.5 at update 1, reaches 0.05 at update 50, and stays there. All remaining target, reward, normalization, and optimizer settings match PSI. The Two-phase schedule keeps equal weights for the first 50 updates and switches to correctness only afterward; its run uses learning rate 10^{-6}, upper clip 0.28, and group filtering. Table 11 reports the three controllers. Linear decay is competitive with PSI at step 300, PSI improves both accuracy and length relative to it at step 450, and the Two-phase schedule is below both.

F.4 Released LASER-family checkpoint comparison

Table 12 compares LEAD with all nine released 1.5B LASER/LASER-D/LASER-DE checkpoints. Table 13 provides per-benchmark results at 1.5B and for two additional 7B adaptive checkpoints. The numeric suffix denotes the released checkpoint's target parameter. All checkpoints use our shared evaluation protocol. Their training protocols differ: LASER uses DeepScaleR (approximately 40K

Table 8: Across-run 95% Student-t confidence intervals for the means of the three data-order-seed runs. These are distinct from question-bootstrap intervals.

Method	Avg Acc	Avg Len	AES
GRPO	[44.73, 50.92]	[1534, 2621]	[−0.589, 0.585]
GDPO	[43.12, 49.38]	[1115, 2082]	[−0.820, 0.313]
DRPO	[49.14, 52.49]	[2443, 3060]	[0.204, 0.801]
LEAD	[50.66, 55.20]	[3321, 3857]	[0.551, 0.795]

Table 9: Paired 95% question-bootstrap intervals for average-accuracy differences, in percentage points.

Comparison	Difference	95% CI
1.5B/4K: LEAD − DRPO	$+2.62$	$[+0.79, +4.52]$
1.5B/8K: LEAD − GDPO	$+3.17$	$[+1.31, +5.33]$
7B/4K: LEAD − DRPO	$+1.17$	$[-0.65, +3.09]$

Table 10: Marginal 95% question-bootstrap intervals for accuracy of the original models (10,000 resamples). The average gives equal weight to the five benchmarks. The intervals quantify question-resampling uncertainty for these fixed models.

Setting / method	AIME24	AIME25	AMC23	MATH-500	OlympiadBench	Avg Acc
1.5B-4K Base	[18.00, 41.33]	[12.00, 37.33]	[59.75, 78.75]	[82.40, 87.47]	[48.49, 55.01]	[47.81, 55.98]
1.5B-4K GRPO	[14.00, 34.67]	[8.67, 31.67]	[55.49, 75.75]	[79.47, 85.00]	[45.88, 52.44]	[44.35, 52.01]
1.5B-4K GDPO	[9.67, 31.67]	[6.33, 28.33]	[49.75, 72.00]	[77.80, 83.73]	[45.38, 52.00]	[41.65, 49.46]
1.5B-4K ShorterBetter	[14.67, 35.33]	[8.67, 30.33]	[54.75, 75.00]	[77.53, 83.20]	[41.63, 47.95]	[43.04, 50.46]
1.5B-4K DRPO	[15.67, 40.00]	[10.00, 33.67]	[60.00, 80.25]	[80.53, 86.00]	[48.49, 55.11]	[46.76, 54.86]
1.5B-4K LEAD	[22.67, 48.33]	[12.33, 37.67]	[56.00, 78.00]	[82.87, 88.00]	[51.16, 57.78]	[49.14, 57.71]
7B-4K Base	[39.66, 67.67]	[26.00, 52.33]	[80.00, 94.50]	[91.73, 95.27]	[63.90, 70.12]	[63.93, 72.37]
7B-4K GRPO	[25.67, 50.33]	[15.67, 40.33]	[73.75, 91.51]	[86.60, 91.13]	[55.70, 62.17]	[55.34, 63.32]
7B-4K GDPO	[27.33, 56.00]	[14.67, 41.67]	[70.25, 89.25]	[87.33, 91.93]	[56.64, 63.36]	[55.40, 64.38]
7B-4K ShorterBetter	[33.67, 59.00]	[18.33, 45.00]	[72.25, 88.25]	[80.87, 86.13]	[57.33, 63.70]	[56.41, 64.68]
7B-4K DRPO	[30.00, 58.67]	[17.67, 44.67]	[80.50, 94.25]	[90.00, 94.07]	[61.88, 68.44]	[59.81, 68.24]
7B-4K LEAD	[31.67, 58.00]	[22.00, 48.67]	[80.50, 95.75]	[90.33, 94.13]	[61.83, 68.30]	[61.09, 69.31]
1.5B-8K LEAD	[18.67, 43.00]	[13.67, 39.33]	[63.25, 83.00]	[83.87, 89.00]	[52.49, 59.16]	[50.25, 58.74]
1.5B-8K GDPO	[18.33, 41.33]	[9.67, 34.33]	[56.25, 77.25]	[82.67, 87.67]	[50.12, 56.79]	[47.26, 55.32]
1.5B-8K GRPO	[12.00, 34.67]	[9.00, 32.33]	[61.00, 81.25]	[82.13, 87.40]	[47.75, 54.47]	[46.21, 54.10]
1.5B-8K DRPO	[17.00, 42.67]	[11.33, 35.33]	[61.75, 81.00]	[79.07, 84.33]	[47.56, 53.98]	[47.23, 55.45]

problems) and a 16K training-response cap, while LEAD uses 8,521 MATH Level 3 to 5 problems and a 4K cap.

At 1.5B, LEAD is more accurate and shorter on aggregate than each of the nine evaluated checkpoints. Against LASER-D-2048, the highest-AES LASER checkpoint, LEAD gains 1.31 accuracy points and uses approximately 713 fewer tokens (16.1% shorter). All lengths are measured under our shared decoding and scoring configuration.

Additional 7B comparison. LASER-D-4096 obtains $68.71/3989/0.42$ (Acc / Len / AES), and LASER-DE-4096 obtains $68.19/3780/0.43$, compared with LEAD's $65.17/4417/-0.11$. Both LASER variants are more accurate and shorter at this scale under the shared evaluator. AES uses the 7B base accuracy and length of $68.20/6631$, and the training-protocol differences described above apply at both scales.

F.5 Qwen3-4B LoRA transfer results

Table 14 summarizes the Qwen3-4B LoRA comparison and Table 15 reports per-benchmark results. The base and both trained methods use the same evaluation configuration, with $n = 10$ sampled decodes per question for AIME24, AIME25, and AMC23, and $n = 3$ for MATH-500 and Olympiad-

Table 11: Weighting-controller comparison on DeepSeek-R1-Distill-Qwen-1.5B (4K), seed 42. This PSI run is distinct from the LEAD run in Table 1.

Step	Controller	Avg Acc	Avg Len	AES
300	Linear decay schedule	53.97	3640	0.727
300	PSI	53.53	3561	0.710
450	Linear decay schedule	52.29	3637	0.631
450	Two-phase schedule	50.76	3169	0.446
450	PSI	53.89	3343	0.755

Table 12: Comparison with released LASER models on DeepSeek-R1-Distill-Qwen-1.5B. AES uses the base accuracy and length of $51.85/9213$. **Bold** marks the best Acc / AES and underline marks the second best.

Method	Avg Acc	Avg Len	AES
LASER-2048	49.83	4459	0.13
LASER-4096	51.93	5716	0.38
LASER-8192	51.92	5898	0.36
LASER-D-1024	49.39	4256	0.06
LASER-D-2048	52.05	4427	<u>0.53</u>
LASER-D-4096	52.06	5009	0.47
LASER-DE-1024	49.74	4187	0.14
LASER-DE-2048	51.05	4479	0.36
LASER-DE-4096	<u>52.07</u>	5319	0.44
LEAD	**53.36**	3714	**0.68**

Table 13: Per-benchmark released LASER results. Each benchmark cell is Acc / Len. LEAD's corresponding values are in Table 1.

Scale	Checkpoint	AIME24	AIME25	AMC23	MATH-500	OlympiadBench
1.5B	LASER-2048	26.67 / 6657	19.00 / 6235	68.00 / 3515	82.80 / 1923	52.69 / 3962
	LASER-4096	31.33 / 8476	21.33 / 7935	71.50 / 4598	84.07 / 2541	51.41 / 5029
	LASER-8192	31.33 / 7887	23.00 / 7753	71.25 / 5136	83.67 / 3143	50.37 / 5570
	LASER-D-1024	26.67 / 6506	20.33 / 5876	68.00 / 3452	82.00 / 1759	49.93 / 3685
	LASER-D-2048	30.33 / 6564	23.33 / 6161	73.00 / 3469	83.07 / 1969	50.52 / 3975
	LASER-D-4096	30.67 / 7293	20.67 / 6888	73.00 / 3993	83.40 / 2338	52.54 / 4532
	LASER-DE-1024	30.00 / 6205	20.00 / 5713	68.50 / 3430	82.07 / 1822	48.15 / 3765
	LASER-DE-2048	31.00 / 6674	20.67 / 6369	68.50 / 3517	83.87 / 1868	51.21 / 3965
	LASER-DE-4096	32.00 / 7751	21.33 / 7388	72.25 / 4162	82.13 / 2462	52.64 / 4833
7B	LASER-D-4096	53.67 / 5840	37.67 / 6306	90.75 / 2772	93.60 / 1622	67.85 / 3405
	LASER-DE-4096	50.00 / 5506	39.00 / 5833	90.75 / 2515	93.60 / 1612	67.60 / 3436

Bench. Aggregate accuracy is computed from unrounded question-level scores with equal benchmark weighting.

Table 14: Qwen3-4B LoRA transfer study. Acc and Len average the five benchmarks equally. AES uses the unmodified Qwen3-4B base. Bold marks the best Acc / AES among the trained methods.

Method	Avg Acc	Avg Len	AES
Qwen3-4B base	78.90	9945	0.00
GRPO-LoRA	70.95	4776	-0.49
LEAD-LoRA	**74.00**	6908	$\mathbf{-0.32}$

Both arms use Qwen3-4B with thinking enabled, the same 8,521 MATH training problems, data-order seed 42, 128 prompts per batch, and $G = 8$ rollouts per prompt. The prompt and response caps are 2,048 and 4,000 tokens. LoRA uses rank 64, scaling parameter 32, and all linear layers. The learning rate is 3×10^{-6}, constant without warmup. Both arms use PPO mini-batches of 16 prompts, one PPO epoch, clipping thresholds of 0.2, KL coefficient 0.0005, and entropy coefficient 0.001. The GRPO scalar length reward is a binary threshold at 2,000 character-estimated tokens, computed as the combined prompt and response character count divided by four. LEAD uses its symmetric response-token reward with $L_{\min} = 1000$, $B_{\max} = 4000$, $\beta = 0.95$, and correctness-weight floor 0.3.

Training uses verl v0.7.0 with the LEAD port and vLLM 0.11.0. LEAD runs on four L40S GPUs and GRPO on two H200 GPUs, with identical global batch and optimizer settings. All reported models are evaluated on an H200 using the same evaluation environment and decoding configuration. The confidence intervals use 10,000 paired question-bootstrap resamples and quantify question-sampling uncertainty rather than training-run variability.

For Table 14, the 95% accuracy intervals are $[75.01, 82.67]$ for the base, $[66.68, 75.11]$ for GRPO, and $[69.77, 78.01]$ for LEAD. The paired LEAD-minus-GRPO difference is $+3.06$ points ($[+1.04, +5.13]$).

Table 15: Per-benchmark Qwen3-4B LoRA results. Each benchmark cell reports Acc / Len.

Method	AIME24	AIME25	AMC23	MATH-500	OlympiadBench
Qwen3-4B base	71.00 / 13189	61.33 / 15902	93.50 / 7000	95.40 / 4207	73.28 / 9428
GRPO-LoRA	59.33 / 7083	44.00 / 7852	89.75 / 3072	93.20 / 1832	68.44 / 4040
LEAD-LoRA	66.00 / 10147	48.67 / 11315	90.50 / 4304	93.73 / 2741	71.11 / 6035

F.6 Public-checkpoint re-evaluation

The public 8K LEAD checkpoint, `Kotom1/math_lead_8k_deepseek-r1-1.5b` at revision `f515d452`, obtains $54.93/5824/0.546$ under the frozen evaluator, compared with the original Table 5 row $54.44/5620/0.54$. Its per-benchmark accuracies are $34.33, 23.33, 73.25, 88.13, 55.60$ in table order. We report this checkpoint re-evaluation separately from the original result. Re-evaluating the identical 4K checkpoint gives 53.16 accuracy compared with the original 53.36, illustrating variation in sampled evaluation across executions.

F.7 BatchWhiten and the correctness-only comparison

Let $m_{qjt} \in \{0, 1\}$ mask valid response-token positions and let $N = \sum_{qjt} m_{qjt}$. Broadcasting the rollout advantage $\tilde{A}_{q,j}$ across its response tokens, the implementation computes

$$\bar{\mu} = \frac{1}{N} \sum_{qjt} m_{qjt} \tilde{A}_{q,j}, \qquad \bar{v} = \frac{1}{N-1} \sum_{qjt} m_{qjt} (\tilde{A}_{q,j} - \bar{\mu})^2, \tag{14}$$

and returns $m_{qjt}(\tilde{A}_{q,j} - \bar{\mu})/\sqrt{\bar{v} + 10^{-8}}$. The training batches have more than one valid token. The utility rejects masks with zero or one valid position. Group normalization uses a sample standard deviation plus ϵ, whereas PSI's pooled statistics use population variances.

At zero efficiency weight, reward-wise and scalar normalization coincide for correctness, while static LEAD additionally applies final BatchWhiten. All-equal correctness groups have zero correctness advantage before this operation. Their prevalence and response lengths affect the token-weighted batch moments. Whitening removes the batch offset and standardizes the surviving correctness contrasts, changing their scale relative to other loss terms while leaving the PPO clipping threshold unchanged. In a matched first-batch diagnostic, advantage mean/variance and gradient norm are $-0.048/0.439/0.628$ without final whitening and approximately $0/1/0.942$ with it. This additional normalization step is part of the advantage-processing difference in Table 2.

F.8 PSI edge cases and usable ranking contrasts

For correctness, near-failure has $\mu_c \approx 0$ and $P_c \approx 1$, whereas near-saturation has $\mu_c \approx 1$ and $P_c \approx 0$. The headroom factor distinguishes these reward means even when both variances are small. Within an all-wrong or all-correct prompt, correctness advantages are zero because all rewards are equal. The implementation adds 10^{-8} before taking the controller standard deviation, providing numerical stabilization even for constant rewards. If the whole batch has no correct rollout, the efficiency controller moments are set to zero. If both PSI scores vanish, their stabilized target ratios are zero and the pre-floor EMA decays. The correctness floor and complementary efficiency weight determine the applied pair. Fixed reward bounds keep the potential denominator nonzero.

F.9 Sparse-correct targets and the zero-correct fallback

Table 16 reports correct-count histograms from the matched seed-42 PSI and *Linear decay schedule* runs. These additional runs share the target construction and 128×8 rollout geometry. We use identical windows, with 128 prompt groups per update. The PSI run resumed from step 100, and

Table 16: Correct-count distributions in the additional matched seed-42 runs. Percentages are computed over prompt groups and can differ from 100% by rounding. The last row for each controller is a 100-update late-training window.

Controller	Steps	$\lvert\mathcal{C}_q\rvert = 0$	1-2	3-5	6-8
PSI	1-50	13.33%	8.95%	11.94%	65.78%
	51-300	10.05%	6.65%	10.96%	72.34%
	301-460	8.62%	4.62%	7.49%	79.27%
	361-460	8.34%	4.41%	7.16%	80.09%
Linear decay	1-50	13.84%	7.80%	13.36%	65.00%
	51-300	8.81%	5.61%	9.59%	76.00%
	301-460	7.39%	4.59%	7.56%	80.46%
	361-460	7.06%	4.42%	7.39%	81.13%

repeated steps 101 to 113 use the resumed trajectory. Groups with one or two correct rollouts are a minority in both runs and become less frequent as training progresses.

The target lies in $[1000, B_{\max}]$, raw efficiency rewards lie in $[-1, 1]$, and advantage normalization uses all eight rollout rewards with numerical stabilization. These safeguards keep raw values bounded and provide a full group for normalization. Correct rollouts supply positive-versus-negative correctness contrast and examples of successful reasoning lengths. With one correct sample, mean, median, and minimum coincide. With two, the conventional median equals the mean. The aggregator comparison in Table 3 therefore reflects differences primarily in larger correct sets. Sparse-sample target uncertainty remains a limitation discussed in Appendix A.

A separate control sets $r_\ell = 0$ for every rollout when $|\mathcal{C}_q| = 0$. It obtains $52.06/3719/0.594$, compared with $52.84/3551/0.659$ for the corresponding full-fallback run. These runs are separate from the main comparison. Substantial accuracy and length gains remain after removing the fallback, with a 0.78-point accuracy difference between the two runs.

F.10 Observed target movement and the training cap

Table 17 uses the submitted PSI training log on MATH Level 3-5. Target means and medians are computed over solvable prompt groups and then averaged across updates in each window. Accuracy and cap-hit rate use all rollouts. Both the mean and median target decrease as training-batch accuracy rises.

Table 17: Submitted 1.5B/4K PSI trajectory: window averages. Target statistics exclude zero-correct groups.

Steps	Mean L_q^*	Median L_q^*	Batch accuracy	Cap-hit rate
1-10	2628	2619	0.427	0.471
50-59	1868	1681	0.680	0.154
451-460	1830	1677	0.780	0.050

The cap bounds the available rollout lengths, while correctness filtering selects the trajectories used to re-estimate the target. Longer incomplete or incorrect rollouts are excluded from this correct set. Selection and re-estimation can therefore contribute to downward target movement as training progresses. The symmetric reward encourages lengths near the current target, with re-estimation allowing that preferred length to change.

Table 2's correctness-only static LEAD reaches $50.60/3141/0.42$, compared with dynamic LEAD's $53.36/3714/0.68$. Compression is therefore already present without the efficiency channel, while the full method preserves more reasoning tokens and achieves higher accuracy and AES. The 8K results further show that LEAD retains compression relative to the base under a relaxed cap. Together, these experiments characterize the accuracy and length trade-off under different reward settings and training budgets.

F.11 Behavioral-floor diagnostic

We analyze per-prompt correct-rollout lengths from the *neutral-fallback PSI control*, which uses LEAD's solved-prompt target and sets the zero-correct efficiency reward to zero. Over its final 100 updates (362 to 461), 8,877 of 12,800 prompt groups have $|\mathcal{C}_q| \geq 4$. Among these eligible groups, the median within-correct length CV is 0.163, and the shortest observed correct rollout is a median 22.4% shorter than its group mean. Only 3.85% have a gap of at most 10%.

To measure how often correct lengths cluster tightly at a high value, we count groups with mean correct length at least b and within-correct population CV at most τ. Table 18 reports sensitivity to both thresholds. At $b = 2000$ and $\tau = 0.10$, 223 of 8,877 eligible groups meet the criterion (2.51%, or 1.74% of all groups). Tight observed clusters are uncommon at these thresholds. These measurements describe sampled trajectories, while the minimum sufficient reasoning length remains unknown.

Table 18: High-and-tight proxy frequency among eligible groups ($|\mathcal{C}_q| \geq 4$) in the neutral-fallback PSI control, final 100 updates. The thresholds b and τ define the observed clustering criterion.

Mean correct length b	$\tau = 0.05$	$\tau = 0.10$	$\tau = 0.15$	$\tau = 0.20$
1500	0.35%	4.66%	15.51%	26.79%
2000	0.27%	2.51%	7.25%	12.40%
2500	0.20%	1.17%	3.18%	4.88%

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state two contributions: (i) PSI-driven dynamic reward weighting with decoupled group normalization, and (ii) per-problem online length target L_q^* with a symmetric efficiency reward. Both are formalized in Section 4 and validated empirically in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated *Limitations and Broader Impact* section in Appendix A discusses four scope-defining aspects of LEAD: (i) it targets RL settings with reliably verifiable correctness and group sampling, with extensions to open-ended generation requiring task-specific reward models, (ii) the bootstrapping design intentionally prioritizes correctness before compression on unsolved prompts, (iii) the single-scalar per-problem target fits unimodal-length reasoning tasks but extending to distributional or multi-target length models is a future direction, (iv) LEAD is a training-time mechanism rather than a deployment-time budget controller, and combining it with test-time budget allocation is left as future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper presents an algorithmic/empirical contribution. It does not contain formal theorems requiring proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5.1 specifies the dataset (MATH level 3-5), base models (DeepSeek-R1-Distill-Qwen-1.5B/7B), all training hyperparameters (learning rate, batch sizes, KL coefficient, rollout count, epochs), and the LEAD-specific hyperparameters ($\alpha, \beta, \lambda_{\min}, B_{\max}$). Algorithm 1 in Appendix C gives the full advantage-computation procedure. The evaluation protocol follows Sober Reasoning settings reported in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is publicly available at `https://github.com/CrazyMint/LEAD`, including the LEAD implementation, top-level training launchers, ablation drivers in `ablations/`, and MATH preparation scripts in `scripts/data/`. The README documents installation, training, evaluation settings, and released checkpoints. The MATH dataset is publicly available under its original license. Appendix F.1 specifies the evaluation protocol, and Appendix F.5 describes the additional LoRA study.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 and Table 4 specify LEAD's optimizer, batch sizes, rollout count, and controller settings. Appendix F.1 records method-specific configurations and evaluation conventions. Train/test splits and per-benchmark n for avg@n are explicit. Appendix F.5 provides the LoRA setup and per-benchmark results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 6 reports means and sample standard deviations over three data-order-seed runs. Appendix F.2 provides per-run values, across-run Student-t confidence intervals, and paired question-bootstrap accuracy intervals, with the resampling procedure and sources of variation specified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Table 4 and Appendix F.1 describe the training configurations. A 1.5B/4K run takes approximately 30 hours on four L40S GPUs. The Qwen3 study uses four L40S GPUs for LEAD and two H200 GPUs for GRPO, taking approximately 114 and 46 hours respectively. Appendix F.5 records the matched global training settings and shared evaluation environment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The work uses publicly released benchmarks and openly distributed pretrained model weights under their original licenses, does not involve human subjects, and does not collect or release any new data containing personal information.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Appendix A discusses lower inference cost, latency, and energy use, along with reduced inspectability of shorter responses and the potential to make harmful uses cheaper. We recommend configurable verbosity in transparency-sensitive applications and retaining the safeguards of the underlying base model.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [No]

Justification: We release training code and mathematics-focused fine-tuned checkpoints without additional access restrictions or dedicated model-level safeguards. No new scraped or generated dataset is released. Appendix A discusses misuse risks and recommends retaining base-model safeguards in deployment.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third-party assets are cited with their original publications: the verl distributed RL framework (Apache 2.0), the MATH dataset [42], the DeepSeek-R1-Distill model family [9], and the math grading utilities from `deepscaler`. The released code retains the upstream Apache 2.0 LICENSE inherited from verl and is released under the same license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public repository at `https://github.com/CrazyMint/LEAD` provides a README covering requirements, installation, training, evaluation settings, results, and licensing. It links released fine-tuned checkpoints, including `https://huggingface.co/Kotom1/math_lead_4k_deepseek-r1-1.5b`. The manuscript documents training configurations, evaluation details, and the additional analyses.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the subject of the experiments (the policy being trained), not a non-standard methodological component: pretrained DeepSeek-R1-Distill and Qwen3 checkpoints are fine-tuned with the proposed RL recipe. No LLM is used as an agent, oracle, judge, or pipeline step in the methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.