Concise Reasoning Through the Lens of Lagrangian Optimization

Chengqian Gao[1], **Haonan Li**[1], **Taylor W. Killian**[2], **Jianshu She**[1], **Renxi Wang**[1], **Liqun Ma**[1], **Zhoujun Cheng**[3], **Shibo Hao**[3], **Zhiqiang Xu**[1*]

[1]MBZUAI [2]Brigham Young University [3]UCSD

Abstract

Concise reasoning in large language models (LLMs) seeks to generate only essential steps needed to arrive at a final answer, thereby alleviating issues of "overthinking". Most proposed approaches reduce conciseness to a hand-tuned coefficient or threshold that must be re-tuned across domains and model scales. We address this brittleness by treating concise reasoning as a constrained problem that minimizes length subject to an accuracy floor, and develop Performance-Aware Length Update (PALU), an algorithm guided by a Lagrangian view of this problem. PALU maintains a per-prompt length budget and updates it using two data-driven signals: the prompt's pass rate, which determines whether to compress, and the dispersion of its correct responses, which determines how much. On DeepSeek-R1-Distill-Qwen-1.5B, PALU reduces generation length by 64% while improving accuracy from 44.2% to 51.6% across six benchmarks, matching GRPO accuracy under a 2.5× tighter length budget. The same hyperparameters transfer across math, logic and STEM, and across 1.5B to 14B models, suggesting that constrained optimization remains a productive design scaffold for training reasoning LLMs even where its classical guarantees no longer apply.

1 Introduction

Overthinking, the tendency of reasoning-trained large language models (LLMs) to produce verbose chains of self-reflection, backtracking, and redundant validation, inflates inference costs without improving answer quality [1, 2]. This has motivated a growing literature on concise reasoning: training LLMs to produce only the steps necessary to reach a correct answer [3–5].

Existing methods share a common form: they combine length and reward into a single weighted objective. Reward-shaping methods subtract a hand-crafted length penalty [3, 6, 5]; budget-based methods truncate trajectories at heuristic thresholds [4, 7]; multi-stage pipelines train short-CoT supervision and length-aware modes in sequence [8]. All require coefficients or thresholds that must be re-tuned across domains and model scales, and all treat conciseness as a soft preference traded against accuracy. The recurring symptom is brittleness: a length penalty that works on math degrades on STEM; a budget that suits a 1.5B model can be suboptimal for a 14B one.

We argue this brittleness is structural, not incidental. The methods above, whether they shape rewards with a length penalty or impose a globally scheduled token budget, decide in advance how to pressure length, and that decision does not respond to whether the model is currently meeting its accuracy target. The natural formulation is constrained: minimize length subject to an accuracy floor, with the pressure on length tied directly to whether the floor is currently met. Lagrangian duality is the standard tool for such problems. Its formal guarantees do not transfer to LLM training, where rewards

*Corresponding author: Zhiqiang Xu (zhiqiang.xu@mbzuai.ac.ae).

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

Table 1: Length control mechanisms in RL-based concise reasoning. We compare whether each method assigns prompt-specific budgets, enforces budgets during rollout, and avoids additional costs.

Method	Length Control	Per-Prompt Budget	Pre-Rollout Constraint	No Extra Rollout Cost
Overlong Punishment [15]	length penalty	✗	✗	✓
L1 [16]	length + difficulty	✗	✗	✓
ShorterBetter [17]	target-length penalty	✗	✗	✓
ThinkPrune [4]	global budget schedule	✗	✓	✓
GFPO [18]	advantage masking	✗	✗	✗
PALU (ours)	**per-prompt rollout budget**	✓	✓	✓

are discrete and policies non-convex, but the view itself remains useful: it indicates what an algorithm should track and how the tracked quantities should interact.

We instantiate this view as Performance-Aware Length Update (PALU). The Lagrangian optimization prescribes three updates: a constraint residual $C - R$ that signals whether to compress, a dual variable λ that scales this signal, and a budget gradient $\nabla_L R$ that sets the compression step size. Together they form a feedback loop: *read performance, decide whether to act, set how far to move*. Each component is impractical in LLM training. PALU therefore replaces them with tractable counterparts. The residual is estimated from the prompt's pass rate on prior-epoch rollouts, avoiding extra rollout cost. The dual variable, which cannot be tuned over the few epochs available, is collapsed into a binary regime: the budget is shortened when the pass rate exceeds the accuracy floor, and restored to its maximum otherwise. The non-differentiable budget gradient is replaced by a quantile gap of correct-response lengths, which sets how much the budget is shortened. The result is a per-prompt length budget that adapts during training, with no rollout cost beyond standard reasoning training.

On DeepSeek-R1-Distill-Qwen-1.5B, PALU reduces generation length by 64% while improving accuracy from 44.2% to 51.6%, matching GRPO (51.4%) under a 2.5× tighter length budget. Throughout training, PALU keeps shrinking responses even after length and accuracy start to trade off, because compression decisions are per-prompt rather than population-level. Across math, logic, and STEM, and across 1.5B to 14B models, a single hyperparameter setting transfers, precisely where prior methods report the brittleness we identify. Together, these results suggest the value of the Lagrangian view here lies not in its classical guarantees, but in its prescription of what an algorithm should track.

2 Related Work

Existing solutions for overthinking fall into three paradigms: ***(i)*** training-free methods that operate at inference time [9–12]; ***(ii)*** SFT- and DPO-based methods that fine-tune toward shorter reasoning [13, 14, 5]; and ***(iii)*** RL-based methods, to which our approach belongs.

Reward-based approaches add length-aware penalties to the RL reward. [5, 3, 19, 6, 20] subtract a penalty proportional to response length; others [3, 21, 22] also condition on question difficulty; a further line discounts the reward by the deviation between generated and target length [23, 17, 24]. Aggregating such heterogeneous components into a single reward can distort the length signal once group normalization is applied [25]. More fundamentally, the penalty coefficient fixes an exchange rate between length and accuracy that must be re-tuned whenever this trade-off shifts: across data domains, model scales, or training stages.

Length-budgeting methods impose hard limits on the generation length budget. ThinkPrune [4] and ALP [7] progressively reduce a global budget over training, but the schedule is set in advance and does not respond to model performance. GFPO [18] filters trajectories after generation by length or difficulty; the resulting per-prompt selection is post-hoc rather than predictive. Our method differs from both: PALU allocates per-prompt budgets before rollout and updates each budget based on the prompt's own pass rate during training.

The constrained-optimization view underlying PALU has a long history in safe reinforcement learning and constrained MDPs [26]. Classical Lagrangian guarantees do not transfer to neural policy optimization: dual ascent is unstable and is typically replaced by adaptive controllers such as PID Lagrangian [27], and exact constraint satisfaction is relaxed to in-expectation versions [28]. PALU

follows this tradition, treating the Lagrangian as a structural decomposition rather than an exact optimization recipe, and replacing each intractable component with a tractable surrogate.

3 Proposed Method: PALU

3.1 Formulation and Intuition

Let L denote the per-question length budget, r a rule-based reward over responses, $\pi_{\boldsymbol{\theta}}$ the reasoning model, and $C \in [0, 1]$ a target accuracy floor. Let

$$R(\boldsymbol{\theta}, L, q) = \mathbb{E}_{\boldsymbol{o} \sim \pi_{\boldsymbol{\theta}}(\cdot|q,L)}[r(\boldsymbol{o}, a)]$$

denote the expected reward for question q under budget L. We formulate concise reasoning as

$$\min_{\boldsymbol{\theta},\, L>0} \; L \quad \text{s.t.} \quad R(\boldsymbol{\theta}, L, q) \;\geq\; C. \tag{1}$$

Why constrained rather than scalarized. Prior methods commit to a target response length, either explicitly through a budget cap or a reference length, or implicitly through a length-penalty coefficient whose optimum favors a particular length. The target length has no setting-independent meaning: it depends on the task, the model scale, and the stage of training, and is typically fixed by trial and error. The constrained formulation removes this commitment. The user specifies only an accuracy floor $C \in [0, 1]$, whose meaning is the same across settings. The length itself becomes an outcome of optimization rather than an input: the smallest L at which the constraint is satisfied.

Lagrangian view. Problem (1) admits a standard *Lagrangian* relaxation:

$$\min_{\boldsymbol{\theta},\, L>0} \; \max_{\lambda \geq 0} \; \mathcal{L}(\boldsymbol{\theta}, L, \lambda) \;=\; L \;+\; \lambda\big(C - R(\boldsymbol{\theta}, L, q)\big), \tag{2}$$

where the outer minimization keeps the original objective of shortening L, while the inner maximization over $\lambda \geq 0$ converts constraint violation $C - R$ into a penalty: whenever the accuracy floor is not met, the inner problem inflates the cost of L until $\boldsymbol{\theta}$ and L are pushed back into the feasible region. The corresponding gradient-based updates are

$$\lambda \;\leftarrow\; \lambda + \eta_\lambda\big(C - R(\boldsymbol{\theta}, L, q)\big), \tag{3}$$

$$\boldsymbol{\theta} \;\leftarrow\; \boldsymbol{\theta} + \eta_\theta \cdot \lambda \cdot \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}, L, q), \tag{4}$$

$$L \;\leftarrow\; L - \eta_L\big(1 - \lambda \cdot \nabla_L R(\boldsymbol{\theta}, L, q)\big). \tag{5}$$

Each of the three updates is impractical in LLM training. Two reasons: the classical guarantees that motivate this view in convex settings do not transfer to neural policy optimization (Section 2), and each update faces an obstacle specific to the LLM setting. We address them in turn below.

3.2 Practical Algorithm

Each of the three updates is replaced by a tractable surrogate motivated by its specific obstacle in the LLM setting. We address them in the order they appear in Eqs. (3–5): the residual estimate, the dual variable, and the budget sensitivity.

Off-policy estimation of the residual (Eq. (3)) A fresh on-policy estimate of $R(\boldsymbol{\theta}, L, q)$ would require repeatedly loading the latest checkpoint and generating additional rollouts. Since rollout collection already accounts for roughly half of GRPO training time [29], doing so would nearly double the cost. We instead approximate R using the evaluation from the previous training epoch:

$$R(\boldsymbol{\theta}, L, q) \approx \mathbb{E}_{o \sim \pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot \,|\, q, L_{\text{old}})}\big[r(o, a)\big]. \tag{6}$$

Remark 3.1 (Off-policy estimate properties). The estimator in Eq. (6) reflects the policy from the previous epoch rather than the current one. This is by design: it lets us reuse rollouts that were already collected for the policy update, at no additional cost. The resulting estimate is systematically conservative: the policy has improved since those rollouts were drawn, so the true pass rate is at least as high as what Eq. (6) reports. The gap between the two also shrinks over training as per-epoch policy change diminishes (Fig. 4). Conservatism is therefore largest early in training, when budget caution is most useful, and fades as the policy stabilizes.

Algorithm 1 PALU (instantiated with GRPO)

Input: initial model $\pi_{\boldsymbol{\theta}}$, dataset $\mathcal{D}$, bound $L_{\max}$, performance threshold C

for epoch in range(N) **do**
  for each mini-batch $\mathcal{D}_b \subset \mathcal{D}$ **do**
    if first epoch **then**
      Initialize the per sample budget: $L = L_{\max}$
    else
      Reuse the last round pass rate via Eq. (6)
      Update L for each $q \in \mathcal{D}_b$ using rule Eq. (7)
    end if
    Collect responses $\boldsymbol{o}$ with parameter $\boldsymbol{\theta}$ and per sample budget L
    Update $\boldsymbol{\theta}$ with GRPO as per Eq. (12)
  end for
end for
Output: concise reasoning model $\pi_{\boldsymbol{\theta}}$

Regime-based controller for the dual variable Continuous gradient ascent on λ is impractical in our setting. The update $\lambda \leftarrow \lambda + \eta_\lambda(C - R)$ moves λ at most by η_λ per step, but λ has to reach a magnitude comparable to that of L (in the thousands of tokens) before the term $\lambda \cdot \nabla_L R$ in Eq. (5) meaningfully drives the budget. With LLM post-training typically spanning 5–20 epochs, λ does not have time to reach this regime under any reasonable choice of η_λ. We therefore replace the continuous λ update with a two-regime rule:

$$\text{Regime} = \begin{cases} L \leftarrow L - \alpha_\tau^q & \text{if } R(\boldsymbol{\theta}, L, q) \geq C \\ L \leftarrow L_{\max} & \text{otherwise} \end{cases}, \tag{7}$$

where $\alpha_\tau^{(q)} \geq 0$ is the per-prompt step size we will explain later. The first regime handles the case where the constraint is currently met: the budget shrinks. The second handles the violation case: the budget resets to $L_{\max}$, returning to a regime where the policy can recover performance before any further compression is attempted.

Quantile-driven step size (Eq. (5)) To set the per-question reduction step $\alpha_\tau^{(q)}$, we use Eq. (5) as an interpretive guide. The term $\nabla_L R(\boldsymbol{\theta}, L, q)$ captures the sensitivity of performance to the length budget. Because R is a non-differentiable, rule-based reward, we quantify this sensitivity via the difference between two nearby operating points in the distribution of correct response lengths. Let

$$Q_\tau^{(q)} := \text{Quantile}_\tau \left(\left\{ \text{len}(o_i) \right\}_{i=1}^G \;\middle|\; o \sim \pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot|q, L_{\text{old}});\; r(o, a) = 1 \right),$$

and define the *quantile gap*

$$\alpha_\tau^{(q)} := Q_{1.0}^{(q)} - Q_{1.0-\tau}^{(q)}. \tag{8}$$

We argue that $\alpha_\tau^{(q)}$ plays a role analogous to $\nabla_L R$. To see this, recall that the role of $\nabla_L R$ in Eq. (5) is to scale the budget update by how costly compression is: a large $\nabla_L R$ means small reductions in L already hurt accuracy substantially, warranting a small step; a small $\nabla_L R$ means the policy tolerates substantial compression, warranting a larger step. The quantile gap $\alpha_\tau^{(q)}$ captures the same signal from the empirical distribution of correct-response lengths. When correct responses cluster tightly in length, the quantile gap is small: even a modest reduction in L would risk truncating many of them, so accuracy is highly sensitive ($\nabla_L R$ would be large), and the surrogate prescribes a small step. When correct responses spread over a long tail, the quantile gap is large: a wide range of lengths already produce correct answers, so accuracy is robust to compression ($\nabla_L R$ would be small), and the surrogate permits a larger step. PALU therefore adopts $\alpha_\tau^{(q)}$ directly as the step size:

$$L \;\leftarrow\; L - \alpha_\tau^{(q)} \quad \text{when } R \geq C. \tag{9}$$

For readers interested in the derivation, the qualitative correspondence above admits a quantitative form when L is close to $Q_{1.0}^{(q)}$, typical immediately after a reset to $L_{\max}$. In this regime, reducing L

Table 2: Performance and conciseness comparison of different concise reasoning methods with DEEPSEEK-R1-DISTILL-QWEN-1.5B as the base model. **P@1**: average pass@1 accuracy (%); **Tok**: average response length in tokens. **AE Score**: accuracy-efficiency score for balancing length reduction and accuracy preservation [30]. Methods marked with '-' on AIME 2025 did not report results on this benchmark; their macro average is computed over the remaining tasks.

Model & Methods	MATH 500		AIME 2024		AIME 2025		AMC 2023		Olympiad		Minerva-Math		Macro Average		AE Score ↑
	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	
R1-DISTILL-QWEN-1.5B	82.1	5534	28.5	16590	22.3	16381	62.7	10615	43.5	11587	26.0	7076	44.2	11297	0.0
SFT- & DPO-Based															
Kimi 1.5 SFT [5]	68.5	6761	22.0	17400	-	-	60.4	9323	39.4	10036	23.6	2804	*42.7*	*9865*	*-0.499*
Kimi 1.5 DPO [5]	83.3	4464	31.7	13389	-	-	63.0	8678	44.5	9604	26.9	6070	*49.9*	*8441*	*0.289*
TokenSkip [31]	64.1	1120	6.8	2231	-	-	37.3	1401	25.8	2061	20.7	1674	*30.9*	*1697*	*-1.173*
RL-Based															
CosFn [6]	75.6	2735	27.5	12492	-	-	61.1	6970	42.9	8307	27.1	3485	*46.8*	*6798*	*0.249*
DIET [25]	83.0	3061	31.8	10578	-	-	65.4	6425	43.7	6917	26.9	3505	*50.2*	*6097*	*0.547*
Kimi 1.5 RL [5]	66.3	1552	18.8	9109	-	-	44.7	3808	28.5	4774	16.7	1009	*35.0*	*4050*	*-0.871*
O1-Pruner [30]	79.1	2531	25.0	8961	-	-	62.5	5010	39.0	5242	23.7	2400	*45.9*	*4829*	*0.193*
GRPO [29]	84.4	5437	38.3	15050	28.2	14257	82.5	8615	50.8	10586	24.6	8683	51.4 +16.4%	10438 −7.6%	0.563
AutoThink-Stage1 [8]	82.1	2473	33.5	12716	22.9	10028	66.0	5440	45.6	7328	27.0	5372	46.2 +4.53%	7226 −36.0%	0.552
AutoThink-Stage2 [8]	85.2	3702	31.8	12117	25.6	11557	66.6	7415	46.4	8030	27.2	5481	47.1 +6.68%	8050 −28.7%	0.521
AutoThink-Stage3 [8]	85.1	1897	41.9	9033	25.4	8968	71.9	4696	49.0	5005	30.5	3834	50.6 +14.6%	5563 −50.7%	1.077
ALP [3]	80.5	1435	37.9	8084	26.2	8556	76.5	3513	47.6	4670	24.5	2197	48.9 +10.6%	4742 −58.0%	0.958
L1-Max [16]	83.5	3337	21.7	4093	25.1	3792	66.3	3350	45.6	2698	25.2	2595	44.6 +0.80%	3311 −70.7%	0.551
ShorterBetter [17]	62.9	626	22.9	4617	21.0	4454	65.0	2311	34.8	2674	19.8	827	37.7 −14.6%	2585 −77.1%	0.030
ThinkPrune-4k [4]	83.0	2745	29.5	8557	22.6	7952	71.7	4241	45.2	5505	26.5	3341	46.4 +5.05%	5390 −52.3%	0.659
PALU (ours)	85.3	1502	40.0	7132	28.8	6869	81.8	3174	49.5	3958	24.2	1922	**51.6** +16.8%	**4093** −63.8%	**1.185**

by $\alpha_\tau^{(q)}$ truncates the longest τ-fraction of correct responses, so a finite-difference estimate gives

$$\nabla_L R(\boldsymbol{\theta}, L, q) \;\sim\; \frac{R(\boldsymbol{\theta}, L, q) - R(\boldsymbol{\theta}, L - \alpha_\tau^{(q)}, q)}{\alpha_\tau^{(q)}} \;\sim\; \frac{\tau}{\alpha_\tau^{(q)}}. \tag{10}$$

Substituting into the budget update of Eq. (5) recovers Eq. (9), with the rate η_L absorbed into the choice of τ.

Remark 3.2 (Why reset when $R < C$). The derivation above relies on having enough correct rollouts to estimate $Q_{1.0}^{(q)}$ and $Q_{1.0-\tau}^{(q)}$. When $R \ll C$, correct rollouts become too scarce for these quantiles to be reliably computed. In this regime, what the algorithm needs is to *enlarge* L to recover feasibility, but estimating an upward step faces a symmetric problem: information about the distribution beyond the current L is unobservable. Moreover, gradual recovery is not the right goal in the first place. Once feasibility has been lost, the priority is to restore it immediately, not step toward it. PALU's reset rule, $L \leftarrow L_{\max}$, follows directly from these two observations.

Summary. At training time, PALU follows a simple loop: *read performance, decide whether to act, set how far to move*. The *read* step uses the previous epoch's rollouts to estimate each prompt's pass rate. The *decide* step compares this estimate against the accuracy floor C: if satisfied, the budget shrinks; otherwise it resets to $L_{\max}$. The *move* step shrinks the budget by the quantile gap $\alpha_\tau^{(q)}$, a per-prompt estimate of how much budget can safely be removed. Algorithm 1 gives the pseudocode, instantiated with GRPO [29]. At inference time, only a fixed maximum-length budget is imposed; the conciseness behavior is realized entirely through the trained policy weights.

4 Experiment

The constrained formulation in Section 3.2 makes three empirical predictions that distinguish PALU from length-target-based approaches. *First*, that PALU achieves the best length-accuracy trade-off among existing length-compression methods on standard math benchmarks (Section 4.1). *Second*, that per-prompt budgets sustain compression even after population-level length and accuracy enter

genuine trade-off (Section 4.2). *Third*, that an accuracy floor, unlike a length target, transfers across model families, domains, and scales without re-tuning (Section 4.3).

4.1 Comparison with Existing Solutions

Setup. We finetune DEEPSEEK-R1-DISTILL-QWEN-1.5B [32] on 12k math problems from DEEPSCALER [23]. We use this base model because most baselines fine-tune and evaluate on it, which lets us compare against their released checkpoints and reported numbers rather than re-implementing each method. We train for 20 epochs with 8 rollouts per prompt, step size $\alpha_{0.5}$ ($\tau = 0.5$ in Eq. (9)), and threshold $C = 0.8$. The initial generation budget is $16k$ tokens; the evaluation budget is fixed at $32k$. Following [25], we report Pass@1 (averaged over 32 rollouts on AIME, 10 elsewhere) and average response length on six math benchmarks: MATH-500 [33], AIME 24, AIME 25, AMC 2023, MINERVA, and OLYMPIADBENCH [34]. We additionally report the Accuracy-Efficiency (AE) Score [30] as a composite metric. Our PALU implementation is based on VERL [35]; full hyperparameters are listed in Appendix B.

Baseline reproduction status. We reproduce GRPO under our exact setup, differing from PALU only in that the budget is held fixed at $16k$. Remaining baselines are either re-evaluated from their released checkpoints or taken from their publications, mostly using our base model and training data. Kimi 1.5 DPO and RL are industrial systems with proprietary models and data, included as contextual reference. See Appendix B.3 for the full provenance table.

PALU dominates the length-accuracy Pareto frontier. Across six tasks (Table 2), PALU reduces macro-average length from 11,297 to 4,093 tokens (-63.8%) while *improving* accuracy from 44.2 to 51.6 ($+16.8\%$), achieving the highest AE score (1.185) among all methods. The comparison reveals a consistent trade-off pattern in prior work: reward-shaping methods such as ShorterBetter [17] achieve aggressive compression (-77%) but sacrifice accuracy (-14.6%); staged-budget methods such as ThinkPrune-4k [4] and AutoThink-Stage3 [8] preserve accuracy but compress less aggressively. PALU is the only method that improves substantially on both axes simultaneously, validating the constrained formulation: by treating length as an outcome of optimization rather than a target hyperparameter, PALU avoids the implicit length commitments that bottleneck prior approaches.

4.2 PALU's Training Dynamics

PALU sustains compression even when trade-offs become genuine. Fig. 1 tracks accuracy and length on AIME 24 across training. Two regimes emerge. In the **green regime** (early training, $\rho < 0$), length and accuracy move in opposite directions—the easy redundancy that any reasonable method should catch. The interesting behavior is in the **red regime** (after roughly 25% of training, $\rho > 0$), where compressing further now genuinely costs accuracy. Length-penalty methods are known to either stall in this regime or trade accuracy for further compression [16, 17]. PALU does neither: in this regime alone, average length drops from 10.2k to 8.5k tokens (a further 16% reduction), while accuracy continues to drift upward (37.4% vs 35.0% at the regime boundary).

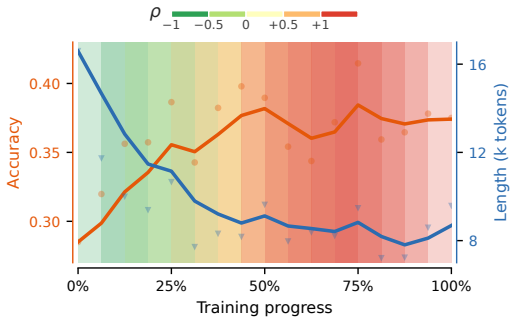


Figure 1: Accuracy-length evolution during PALU training. Background color: rolling Spearman ρ.

This happens because PALU's compression decisions are made *per prompt*, not against a population-level target. Even when the eval-set correlation turns positive, individual prompts still independently satisfy $R(\boldsymbol{\theta}, L, q) \geq C$ and continue to be compressed at their own pace, governed by the quantile gap $\alpha_\tau^{(q)}$, tight on prompts whose correct responses cluster in length, aggressive on prompts whose correct responses spread out. Population-level trade-off and per-prompt compression opportunity are decoupled. This decoupling is the structural advantage of the constrained formulation: it does not commit to a global length target, so compression continues wherever the per-prompt constraint still admits it.

4.3 Generalization across Models, Domains, and Scales

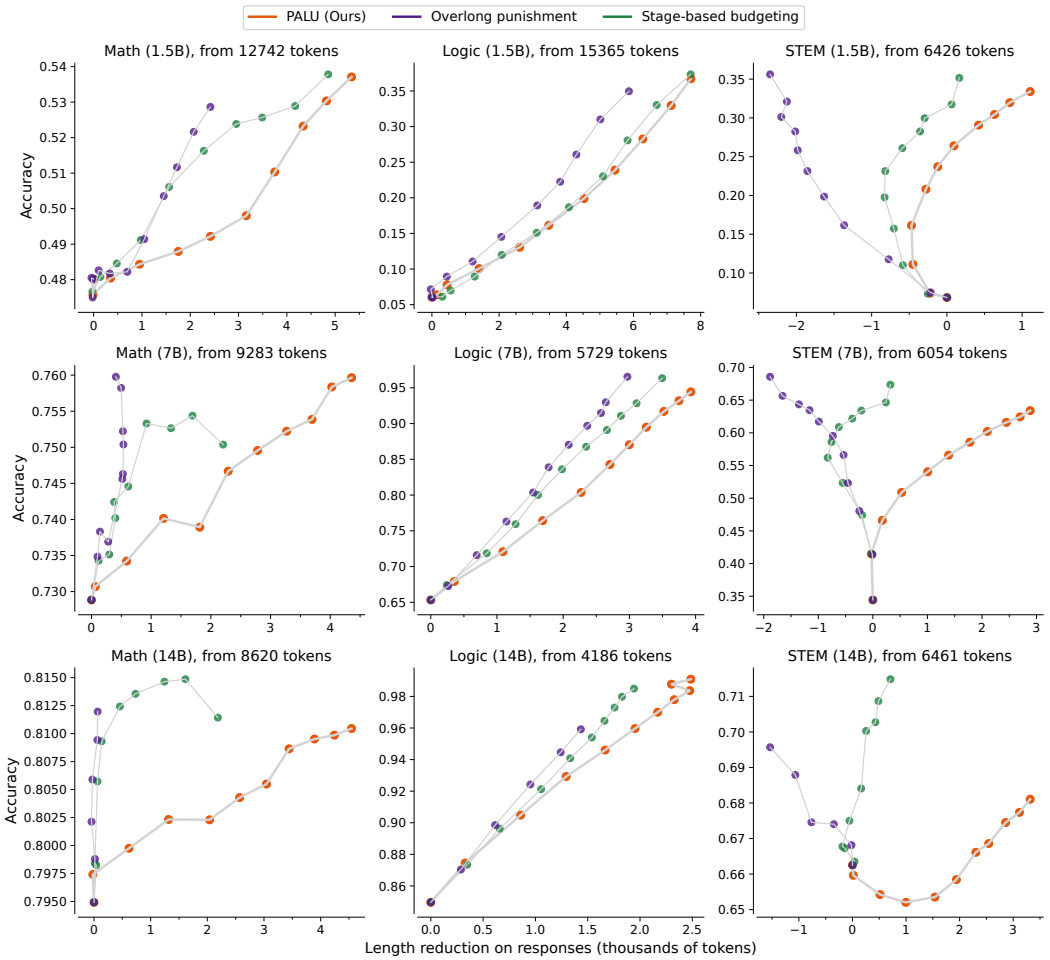


Figure 2: Conciseness-performance evolution of DEEPSEEK-R1-DISTILL-QWEN at 1.5B, 7B, and 14B trained on a mixed math/logic/STEM dataset. Each panel: x-axis is length reduction (k tokens), y-axis is accuracy. Subtitles show initial response length per setting. Curves are smoothed with exponential moving average.

Unlike a length target, which must be recalibrated as base-model length shifts across model families, domains, and scales, PALU's accuracy floor C has the same meaning in every setting. We test whether a single $C = 0.8$ transfers without re-tuning, first to a second model family, then across a 3×3 grid of data domains and model scales.

Across model families. We replicate the comparison on PHI-4-MINI-REASONING [36], training with an 8k context against two representative baselines: ThinkPrune-4k [4] (budget-based) and overlong punishment [15] (reward-based). PALU outperforms both on all three macro-level metrics (Table 3)—higher accuracy at shorter length, with a positive AE score where both baselines are negative. Per-task results (Table 7) show two distinct gains: a large improvement on OLYMPIADBENCH (+7.6 over ThinkPrune-4k) and consistent preservation of AIME accuracy under compression—a regime where both baselines degrade noticeably.

Table 3: PALU on PHI-4-MINI-REASONING (macro-avg over 6 math benchmarks). Full per-task results in Table 7.

Method	P@1	Tok	AE
Base	50.8	8265	0.00
Overlong punish.	48.6	4539	−0.15
ThinkPrune-4k	51.6	4583	−0.01
PALU (ours)	**55.7**	**4283**	**0.27**

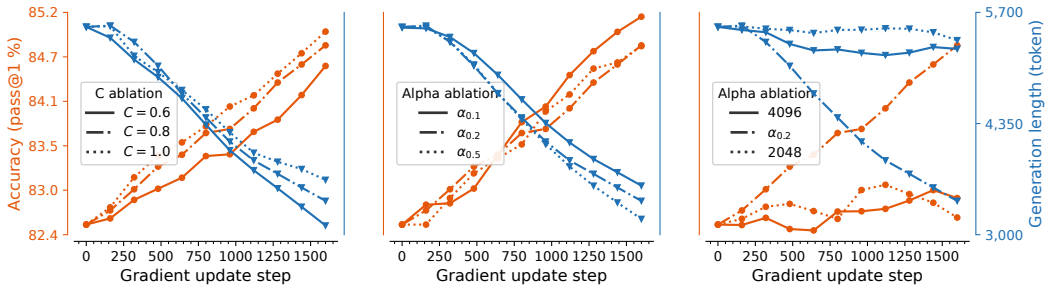


Figure 3: Ablation studies on MATH-500. **Left:** Threshold C at fixed $\alpha_{0.2}$. **Middle:** Step size α_τ at fixed $C = 0.8$. **Right:** Replacing the quantile-driven $\alpha_\tau^{(q)}$ with fixed step sizes $\Delta L \in \{2048, 4096\}$. Orange (left axis): accuracy; blue (right axis): generation length.

Across domains and scales. We extend the comparison to a mixed training set spanning math, logic, and STEM [37], evaluated on DEEPSEEK-R1-DISTILL-QWEN at 1.5B, 7B, and 14B (Fig. 2). Initial response length varies wildly across the resulting nine settings, from 4.2k (Logic, 14B) to 15.4k tokens (Logic, 1.5B). ThinkPrune and overlong punishment both rely on an 8k target, which provides compression pressure only where the base model already exceeds it. In the four cells where it does not—all three STEM rows ($\sim$6.4k tokens) and Logic at 14B (4.2k)—baselines stall near zero compression or, in STEM specifically, *drift to longer responses* (Fig. 2, column 3). On Math 1.5B, where the base model emits 12.7k tokens, baselines compress comparably to PALU—the regime in which a fixed target is correctly calibrated. PALU uses identical hyperparameters in all nine settings: its constraint is on accuracy ($R \geq C$) rather than on length, so there is no length target requiring calibration, and it compresses wherever compression is feasible. Baselines reduce length in 4 of 9 settings; PALU reduces length in 9 of 9. This is the structural payoff of replacing fixed length targets with an accuracy floor: identical hyperparameters transfer across model families, domains, and scales.

4.4 Ablation and Computational Overhead Analysis

PALU has two hyperparameters, a step size α_τ and a threshold C, that govern the regime rule:

$$L = \begin{cases} L - \alpha_\tau^{(q)} & \text{if } R \geq C \\ L_{\max} & \text{otherwise} \end{cases}.$$

We ablate each, then test the design of $\alpha_\tau^{(q)}$ against fixed-step alternatives.

Performance threshold in PALU (Figure 3, left) The threshold C serves as a constraint boundary between samples that should receive more or less generation budget and thus balances accuracy and conciseness. We validate this intuition by sweeping $C \in \{0.6, 0.8, 1.0\}$, with a fixed step size $\alpha_{\tau=0.2}$, and reporting results on the MATH-500 task in the left panel of Figure 3. A high threshold, *e.g.*, $C = 1.0$, drives the optimization toward performance recovery, preserving accuracy at the cost of longer outputs, while a lower threshold, $C = 0.6$, yields the most compact generations with a modest drop in accuracy. The intermediate setting, $C = 0.8$, achieves a balance between the two objectives.

Step size in PALU (Figure 3, middle) α_τ^q measures the gap between the longest correct response and the $(1 - \tau)$-quantile length. To examine its sensitivity and guide practitioners, we sweep the update step size in $\{\alpha_{0.1}, \alpha_{0.2}, \alpha_{0.5}\}$, with a fixed performance threshold $C = 0.8$. We evaluate the training checkpoints on the MATH 500 task and report the accuracy and generation length. As shown in Figure 3, middle, a larger step ($\alpha_{0.5}$) accelerates length reduction but slightly compromises accuracy, whereas smaller steps stabilize performance but provide weaker pressure for conciseness.

Distribution-based vs heuristic step size (Figure 3, right). PALU's step size $\alpha_\tau^{(q)}$ is read from the empirical distribution of correct-response lengths (Eq. 10), serving as a surrogate for the unavailable $\nabla_L R$ term in the Lagrangian formulation. We compare this against purely heuristic alternatives: fixed values $\Delta L \in \{2048, 4096\}$ chosen without reference to model or prompt state. Keeping the regime rule (Eq. 7) and threshold $C = 0.8$ otherwise unchanged, we evaluate all three variants on MATH-500 (Fig. 3). Both heuristic step sizes underperform $\alpha_{0.2}$ in their interaction with the

regime rule, yielding unstable trade-offs between compression and reset. The quantile-driven $\alpha_\tau^{(q)}$, by contrast, descends smoothly while maintaining accuracy—empirical evidence that grounding the step size in correct-response statistics is more reliable than fixed heuristics, even without a rigorous derivation.

Off-policy fidelity of PALU's pass rate estimate. Section 3.2 argued that the off-policy estimate $\hat{R}_{\text{off}}(t) = R(t-1)$ tends to underestimate the true pass rate, with bias shrinking as training stabilizes. Using the training run of Table 2, Fig. 4 verifies this empirically by comparing epoch-mean pass rates across consecutive epochs. The estimate is an empirical lower bound in every epoch (average bias $+1.2$ pp, max 3.2 pp at epoch 2) and shrinks below 1 pp by epoch 8. PALU's caution is therefore largest in early training, when budget collapse would be most harmful, and disappears once the policy has stabilized.

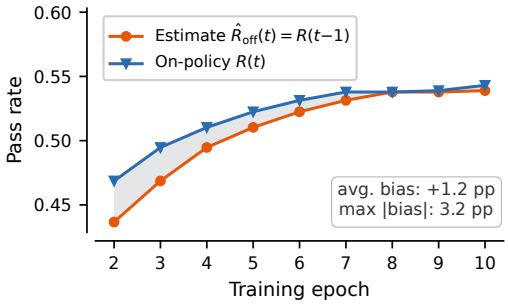


Figure 4: Off-policy fidelity: bias shrinks as the policy stabilizes.

Computational overhead analysis On the memory side, PALU introduces three arrays and one scalar to store: *(i)* pass rates from the previous epoch, *(ii)* length quantiles of correct responses from the previous epoch, *(iii)* length budgets for each prompt, and *(iv)* a global performance threshold. In total, this requires approximately 144 KB memory for training on a dataset of 12k prompts. On the computation side, PALU assigns each prompt a generation length budget that progressively encourages concise reasoning traces compared to the GRPO baseline. Moreover, since the length budget is assigned prior to rollout generation, prompts can be sorted by their budgets to form more balanced batches. Together, these two factors reduce idle time during inference, thereby improving overall efficiency by approximately 9.2% compared to the GRPO baseline.

5 Limitation and Conclusion

Limitation PALU assumes that overthinking LLMs have a broad response length distribution. While we empirically verified this in a large scale experiment, we acknowledge an extreme case where the model always generates responses of identical length. In such a scenario, the quantile gap $\alpha_\tau^{(q)}$ collapses to zero and PALU's compression stalls: the regime rule continues to evaluate $R \geq C$ but produces no budget movement. Another limitation is that we do not claim PALU's concise reasoning behavior will generalize to out-of-distribution domains. We view such generalization as stemming primarily from the diversity of training data and the RL component, rather than from PALU itself.

Conclusion Existing concise-reasoning methods combine length and reward into a single weighted objective and commit, in advance, to a target length. The resulting brittleness across domains and model scales is structural, not incidental, and the natural formulation is constrained: minimize length subject to an accuracy floor, with the pressure on length tied to whether the floor is currently met. PALU instantiates this view through three tractable surrogates: an off-policy residual estimate, a binary regime controller, and a quantile-based step size, yielding a per-prompt length budget that adapts during training. On DeepSeek-R1-Distill-Qwen-1.5B, PALU reduces response length by 64% while improving accuracy from 44.2% to 51.6%, matching a GRPO baseline under a 2.5× tighter length budget. A single hyperparameter setting transfers across math, logic, and STEM, and across 1.5B to 14B models, precisely the regimes where prior methods report the brittleness we identify. The Lagrangian view does not carry its classical guarantees here, but it provides a useful scaffold: it prescribes what an algorithm should track and how the tracked quantities should interact. The empirical robustness of PALU across settings suggests that the scaffold is sound.

References

[1] Chen, X., J. Xu, T. Liang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

[2] Fatemi, M., B. Rafiee, M. Tang, et al. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

[3] Xiang, V., C. Blagden, R. Rafailov, et al. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *arXiv preprint arXiv:2506.05256*, 2025.

[4] Hou, B., Y. Zhang, J. Ji, et al. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

[5] Team, K., A. Du, B. Gao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[6] Yeo, E., Y. Tong, M. Niu, et al. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

[7] Hammoud, H. A. A. K., K. Alhamoud, A. Hammoud, et al. Train long, think short: Curriculum learning for efficient reasoning. *arXiv preprint arXiv:2508.08940*, 2025.

[8] Tu, S., J. Lin, Q. Zhang, et al. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage rl. *arXiv preprint arXiv:2505.10832*, 2025.

[9] Xu, S., W. Xie, L. Zhao, et al. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

[10] She, J., Z. Li, Z. Huang, et al. Hawkeye: Efficient reasoning with model collaboration. *arXiv preprint arXiv:2504.00424*, 2025.

[11] Muennighoff, N., Z. Yang, W. Shi, et al. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[12] Hao, S., S. Sukhbaatar, D. Su, et al. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

[13] Munkhbat, T., N. Ho, S. H. Kim, et al. Self-training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*, 2025.

[14] Su, D., H. Zhu, Y. Xu, et al. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

[15] Yu, Q., Z. Zhang, R. Zhu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[16] Aggarwal, P., S. Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[17] Yi, J., J. Wang, S. Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *arXiv preprint arXiv:2504.21370*, 2025.

[18] Shrivastava, V., A. Awadallah, V. Balachandran, et al. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, 2025.

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

[20] Song, M., M. Zheng. Walk before you run! concise llm reasoning via reinforcement learning. *arXiv preprint arXiv:2505.21178*, 2025.

[21] Shen, Y., J. Zhang, J. Huang, et al. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.

[22] Li, Z., Q. Dong, J. Ma, et al. Selfbudgeter: Adaptive token allocation for efficient llm reasoning. *arXiv preprint arXiv:2505.11274*, 2025.

[23] Luo, M., S. Tan, J. Wong, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.

[24] Team, P. I., S. Jaghouar, J. Mattern, et al. Intellect-2: A reasoning model trained through globally decentralized reinforcement learning. *arXiv preprint arXiv:2505.07291*, 2025.

[25] Chen, W., J. Yuan, T. Jin, et al. The overthinker's diet: Cutting token calories with difficulty-aware training. *arXiv preprint arXiv:2505.19217*, 2025.

[26] Altman, E. *Constrained Markov decision processes*. Routledge, 2021.

[27] Stooke, A., J. Achiam, P. Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International conference on machine learning*, pages 9133–9143. PMLR, 2020.

[28] Achiam, J., D. Held, A. Tamar, et al. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. Pmlr, 2017.

[29] Shao, Z., P. Wang, Q. Zhu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[30] Luo, H., L. Shen, H. He, et al. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

[31] Xia, H., C. T. Leong, W. Wang, et al. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

[32] Guo, D., D. Yang, H. Zhang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[33] Hendrycks, D., C. Burns, S. Kadavath, et al. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[34] He, C., R. Luo, Y. Bai, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

[35] Sheng, G., C. Zhang, Z. Ye, et al. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[36] Xu, H., B. Peng, H. Awadalla, et al. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *arXiv preprint arXiv:2504.21233*, 2025.

[37] Cheng, Z., S. Hao, T. Liu, et al. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. *arXiv preprint arXiv:2506.14965*, 2025.

A Background: GRPO

Group Relative Policy Optimization (GRPO [29]) replaces PPO's value model with a group-relative reward normalization. Given a question-answer pair $(q, a) \sim \mathcal{D}$, G rollouts $\{o_i\}_{i=1}^{G}$ are sampled and their advantages computed as

$$\hat{A}_i(o_i, a) = \frac{r(o_i, a) - \text{mean}(\{r(o_i, a)\}_{i=1}^{G})}{\text{std}(\{r(o_i, a)\}_{i=1}^{G})}. \tag{11}$$

The training objective uses PPO's clipped surrogate (we omit the KL term following 15):

$$J_{\text{GRPO}}(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{(q,a)\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot|q,L)} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min \left\{ r_{i,t}(\boldsymbol{\theta})\, \hat{A}_i,\ \text{clip}(r_{i,t}(\boldsymbol{\theta}), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\, \hat{A}_i \right\} \right], \tag{12}$$

where $r_{i,t}(\boldsymbol{\theta}) = \pi_{\boldsymbol{\theta}}(o_{i,t} \mid q, o_{i,<t}) / \pi_{\boldsymbol{\theta}_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$ is the per-token probability ratio. The length budget L enters only through the sampling loop, which is forcibly terminated once a rollout reaches L tokens; L does not appear in the gradient computation.

B Experimental Details

B.1 Training

Training recipe We integrate our PALU strategy to the VeRL implementation of GRPO and finetune DEEPSEEK-R1-DISTILL-QWEN-1.5B, 7B, and 14B models using the following recipe:

Table 4: Training recipe for finetuning DEEPSEEK-R1-DISTILL-QWEN at 1.5B, 7B, and 14B. Parameters marked with * vary across experiments; non-starred parameters are shared across all runs.

Parameter	Value
Learning rate	$1e-6$
Rollout batch size (prompts)	512
Gradient update batch size (prompts)	32
KL-divergence coefficient	0.0
Max response length	16k
Loss aggregation mode	token-loss
Clip ratio low	0.2
Clip ratio high	0.28
Number of rollouts per sample	8
*Length update step size (Table 2, Table 3)	$\alpha_{0.5}$
*Length update step size (Figure 2)	$\alpha_{0.2}$
*Performance threshold C	0.8

Training datasets For training, we employ two types of datasets:

- 12k mathematics question-answer pairs for the run in **Table 2, Table 3, Figure 4.2, and Figure 4** (benchmarking comparison and its in-depth analysis). This dataset is a slice from the GURU's DEEPSCALER partition. We train DEEPSEEK-R1-DISTILL-QWEN-1.5B for 20 epochs on it.
- 5k multi-domain questions for comparisons in **Figure 2** and **Figure 3**. We randomly select *(i)* 2k math samples from DEEPSCALER, *(ii)* 2k STEM samples from the STEM-web partition, and *(iii)* 1k logic questions from the logic ordering puzzle partition of the GURU collection. We train DEEPSEEK-R1-DISTILL-QWEN at 1.5B, 7B, and 14B for 10 epochs for the multi-domain comparison and ablation study.

Compute resources We conduct our experiments on H200 GPUs clusters. Results in Table 2 are from DEEPSEEK-R1-DISTILL-QWEN-1.5B trained on 12k DEEPSCALER questions, which takes 2 nodes (16 GPUs) for 1100 GPU hours. Results in Figure 2 are from DEEPSEEK-R1-DISTILL-QWEN-1.5B, 7B, and 14B models trained on 5k multi-domain questions, which takes 2 nodes, 4 nodes and 8 nodes for roughly 300, 700, and 2300 GPU hours.

B.2 Evaluation

Decoding and rollouts We follow the decoding protocol from [25] as listed in Table 5. For the rollout numbers, we collect 32 responses for the small dataset (AIME24, AIME25) and 10 responses for others.

Table 5: Decoding parameters used for evaluation rollouts.

Parameter	Value
Temperature	0.6
Top_p	0.95
Top_k	-
Max response length	32768

Accuracy-Efficiency (AE) score We adopt the AE Score from [30], which combines length reduction and accuracy change against a base model into a single number:

$$\text{AE Score} = \begin{cases} \varphi \cdot \Delta\text{Length} + \eta \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} \geq 0 \\ \varphi \cdot \Delta\text{Length} - \theta \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} < 0 \end{cases},$$

where $\Delta\text{Length} = (\text{Length}_{\text{base}} - \text{Length}_{\text{model}})/\text{Length}_{\text{base}}$ and $\Delta\text{Acc} = (\text{Acc}_{\text{model}} - \text{Acc}_{\text{base}})/\text{Acc}_{\text{base}}$. We follow [30]'s hyperparameter setting: $\varphi = 1$, $\eta = 3$, $\theta = 5$, where the asymmetric weighting ($\theta > \eta$) penalizes accuracy drops more than it rewards accuracy gains. Per-task AE scores are reported in Table 8.

B.3 Baselines

Table 6: Baseline provenance. *Reproduced*: trained by us under the same setup as PALU. *Checkpoint*: evaluated by us using the authors' released checkpoint. *Reported*: numbers taken directly from the original publication.

Baseline	Source
GRPO	Reproduced
AutoThink (Stage 1, 2, 3)	Checkpoint
ALP	Checkpoint
L1-Max	Checkpoint
ShorterBetter	Checkpoint
ThinkPrune-4k	Checkpoint
CosFN	Reported
DIET	Reported
O1-Pruner	Reported
TokenSkip	Reported
Kimi k1.5 SFT	Reported (industrial; proprietary base & data)
Kimi k1.5 DPO	Reported (industrial; proprietary base & data)
Kimi k1.5 RL	Reported (industrial; proprietary base & data)

We compare against two families of methods. ***(i) SFT/DPO-based methods***: Kimi k1.5 SFT, Kimi k1.5 DPO [5], and TokenSkip [31]. ***(ii) RL-based methods***, which we further split by

length-control mechanism. *Reward-shaping*: GRPO [29] (fixed-budget reference), CosFN [6], Kimi k1.5 RL [5], DIET [25], ShorterBetter [17], and Overlong Punishment [15]. *Length budgeting*: L1-Max [16], ALP [7], and ThinkPrune [4], which progressively shrinks the rollout budget. *Multi-stage pipelines*: AutoThink [8].

Provenance. Table 6 lists how each baseline's numbers were obtained. All non-industrial baselines use DEEPSEEK-R1-DISTILL-QWEN-1.5B as the base model and DEEPSCALER as training data, matching PALU's setup; differences with PALU are only in the length-control mechanism. Numbers for AIME 2025 are reported only for baselines we re-evaluated, since the original publications predate this benchmark.

B.4 Detailed Results

Table 7: Performance and conciseness comparison on PHI-4-MINI-REASONING.

Model & Methods	MATH 500		AIME 2024		AIME 2025		AMC 2023		Olympiad		Minerva-Math		Macro Average		AE Score ↑
	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	P@1	Tok	
PHI-4-MINI-REASONING	89.9	3997	45.6	13354	31.7	13733	84.5	7451	19.7	5164	33.5	5894	50.8	8265	0.0
Overlong punishment [15]	88.1	2795	28.5	3824	22.3	4019	79.7	5524	43.5	6065	30.0	4954	48.6	4539	-0.15
ThinkPrune-4k [4]	88.9	2819	35.5	6163	29.2	5941	82.5	4764	42.2	5135	31.3	2677	51.6	4583	-0.01
PALU (Ours)	89.3	1929	45.3	7423	30.7	7152	84.6	3241	49.8	3900	34.3	2056	55.7	4283	0.27

Table 8: Detailed Accuracy-Efficiency (AE) score comparison (DEEPSEEK-R1-DISTILL-QWEN-1.5B).

Methods/Model	MATH 500	AIME24	AIME 2025	AMC23	Olympiad	MinervaMath	Macro Average ↑
R1-Distill-Qwen-1.5B	0.000	0.000	0.000	0.000	0.000	0.000	0.00
Kimi 1.5 SFT	-1.050	-1.189	-	-0.062	-0.337	0.142	*-0.499*
Kimi 1.5 DPO	0.237	0.530	-	0.197	0.240	0.246	*0.290*
TokenSkip	-0.299	-2.941	-	-1.158	-1.212	-0.256	*-1.173*
CosFn	0.110	0.072	-	0.216	0.214	0.634	*0.249*
DIET	0.480	0.710	-	0.524	0.417	0.609	*0.548*
Kimi 1.5 RL	-0.243	-1.251	-	-0.794	-1.136	-0.931	*-0.871*
O1-Pruner	0.360	-0.154	-	0.512	0.030	0.219	*0.193*
GRPO	0.102	1.124	0.923	1.136	0.590	-0.496	0.563
AutoThink-Stage1	0.553	0.760	0.470	0.645	0.512	0.356	0.552
AutoThink-Stage2	0.444	0.617	0.740	0.488	0.507	0.364	0.521
AutoThink-Stage3	0.767	**1.866**	0.870	0.998	0.947	**0.977**	1.077
ALP	0.643	1.502	1.000	1.329	0.880	0.401	0.958
L1-Max	0.448	-0.440	1.150	0.857	0.912	0.479	0.551
ShorterBetter	-0.282	-0.261	0.440	0.892	-0.231	-0.309	0.030
ThinkPrune	0.536	0.589	0.550	1.031	0.642	0.585	0.659
PALU (ours)	**0.846**	1.781	1.460	**1.615**	**1.072**	0.382	**1.185**

This subsection provides per-task breakdowns supporting two macro-average claims in the main paper. Section 4.3 reports a single AE score against PHI-4-MINI-REASONING baselines and a single 1.185 AE score for PALU on R1-DISTILL-QWEN-1.5B; here we expand both into per-task numbers, which lets us check whether PALU's gains are concentrated on a few benchmarks or distributed across the suite.

On PHI-4-MINI-REASONING. Table 7 expands the macro-average comparison from Section 4.3. Two patterns are visible. First, PALU's accuracy improvement is broadly distributed: PALU outperforms both baselines on every task except MATH 500, where the three methods cluster within 1 point. Second, PALU's compression is most pronounced on the longest-response tasks—AIME 2024, AIME 2025, and MINERVA-MATH—where overlong punishment compresses aggressively but loses double-digit accuracy. PALU compresses comparably without the accuracy collapse.

On R1-DISTILL-QWEN-1.5B. Table 8 expands PALU's headline AE score of 1.185 into the six tasks that compose it. PALU achieves the highest AE score on four of the six tasks (MATH

500, AMC 23, OLYMPIAD, AIME 2025) and the second highest on a fifth (AIME 2024, where AutoThink-Stage 3 leads). Methods that are competitive on individual tasks tend to lose ground on others—ALP wins on AMC 23 but trails on MATH 500; AutoThink-Stage 3 wins on AIME 2024 but trails on AMC 23—whereas PALU is the only method whose performance does not collapse on any single task.

C Additional Empirical Evidence

This section provides additional figures supporting three points made in the main paper: the broad-length-distribution assumption underlying PALU's design (Section 3.2), the setting-dependence of base-model response length that motivates the constrained formulation (Section 4.3), and PALU's training behavior beyond the single benchmark shown in the main paper (Section 4.2).

C.1 Length Distribution Across Models and Questions

PALU's quantile gap $\alpha_\tau^{(q)}$ measures how much the budget can shrink before truncating correct rollouts. The gap is informative only if correct responses themselves span a wide range of lengths on a given question: if they were tightly concentrated, $\alpha_\tau^{(q)}$ would collapse to zero and PALU's compression would stall (Section 5).

Figure 5 examines this premise across three open-source reasoning families. We highlight two observations:

- **Within-question spread.** On a given question, the longest correct responses from a single model are typically two to three times longer than the shortest. This is the empirical premise behind a non-trivial $\alpha_\tau^{(q)}$.
- **Cross-family differences.** Response lengths also differ systematically across model families: QWEN3 models are consistently longer than DEEPSEEK-R1-DISTILL-QWEN at comparable scales, and DEEPSEEK-R1-0528 produces the longest responses overall. This foreshadows the setting-dependence examined in the next subsection.

C.2 Setting-Dependence of Response Length

A central claim in Section 4.3 is that any fixed length target is mis-calibrated for some settings, because base-model response length itself varies systematically with both the task and the model. The cross-family pattern observed in the previous subsection points to this directly; here we make it concrete along two axes:

- **Across task domains** (Figure 6a). The same base model (DEEPSEEK-R1-DISTILL-QWEN-1.5B) produces median responses near 16k tokens on AIME 24 but under 4k on STEM questions—roughly a 4× gap. Any single budget that meaningfully compresses one of these distributions either truncates correct trajectories on the other or applies negligible pressure.
- **Across model scales** (Figure 6b). On the same task (AIME 2024), median response length drops from roughly 17k tokens at 1.5B to 8k at 32B. A budget calibrated to compress the 1.5B model would sit above the 32B model's median, applying no real pressure.

The two axes are not independent of each other: a 32B model on STEM and a 1.5B model on AIME 24 sit at opposite extremes of the length distribution, but the constrained formulation does not need to model the interaction. By specifying an accuracy floor rather than a length target, PALU side-steps the calibration problem entirely.

C.3 Training Dynamics Beyond a Single Benchmark

Figure 1 in the main paper tracks PALU's training on AIME 24. Figure 7 extends this to five evaluation tasks. In every task, the green-regime to red-regime transition described in Section 4.2 is visible: early training shows length and accuracy moving in opposite directions (negative ρ, green background), and later training shifts to a regime where compression and accuracy are positively

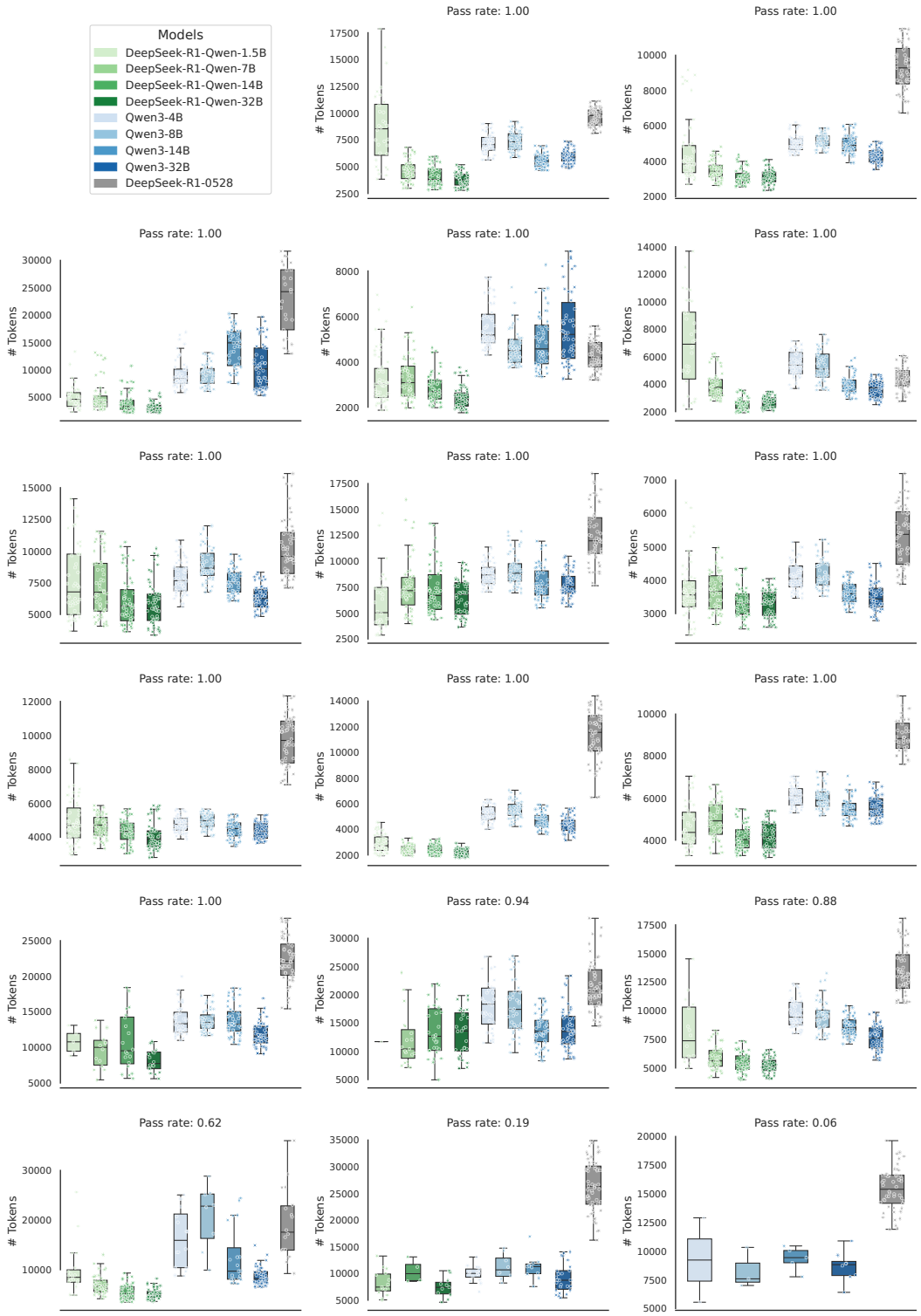
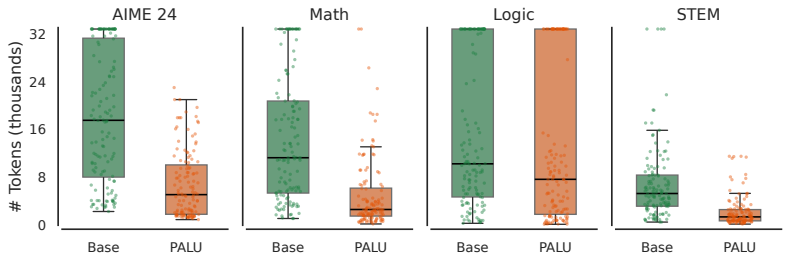
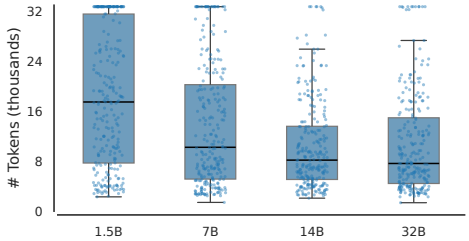


Figure 5: **Correct responses span a wide range of lengths.** Token-length distributions of correct responses from DEEPSEEK-R1-DISTILL-QWEN, QWEN3, and DEEPSEEK-R1-0528 on 18 randomly selected questions from the GURU dataset. Box plots show the interquartile range (25th–75th percentiles).



(a) **Across question domains.** The base model produces noticeably shorter responses on STEM questions and much longer ones on AIME 24.



(b) **Across model scales.** A single budget that compresses the 1.5B model meaningfully would barely apply pressure on the 32B model.

Figure 6: **Base-model response length is setting-dependent.** Top: DEEPSEEK-R1-DISTILL-QWEN-1.5B (Base) vs. the same model finetuned with PALU on the 12k DEEPSCALER subset, evaluated on four task domains. Bottom: DEEPSEEK-R1-DISTILL-QWEN at four scales, evaluated on AIME 2024. Scatter points show raw lengths; box plots show the interquartile range.

correlated (positive ρ, red background). PALU continues to reduce length in the red regime across all five tasks, with no task showing accuracy collapse.

Figure 8 examines training-time behavior from two complementary angles:

- **Across the full training run.** The on-policy pass rate climbs steadily from $\sim$0.43 to $\sim$0.55 (left panel, blue) despite the budget pressure indicated by the orange shading. This is the empirical basis for the off-policy estimator in Section 3.2: per-epoch policy change is gradual enough that the previous epoch's pass rate remains a useful proxy for the current one.
- **Within the final epoch.** There are 5 of 24 batches trigger compression; the remaining 19 hold at $L_{\max}$ (right panel, orange). The 5 compressed batches share a characteristic profile: their pass rates cluster around 0.5 immediately after compression. This is exactly the behavior $\alpha_{0.5}$ predicts—reducing L by the median quantile gap is designed to truncate the longest $\sim$50% of correct rollouts (Section 3.2, Eq. 10), which lands the post-compression pass rate near 0.5. The mechanism we built and the dynamics we observe coincide.

A subtler point follows. PALU shrinks the budget on a small fraction of batches throughout training, yet the final policy produces 64% shorter responses on math benchmarks (Table 2). The two are consistent: each compression event teaches the policy to reach the answer in fewer tokens, and the shortened behavior persists even when the budget returns to $L_{\max}$. Methods that apply length pressure on every step of every batch—reward-shaping penalties, fixed shrinking schedules—both hurt accuracy and, paradoxically, achieve less compression. PALU's sparse, targeted intervention is doing more with less.

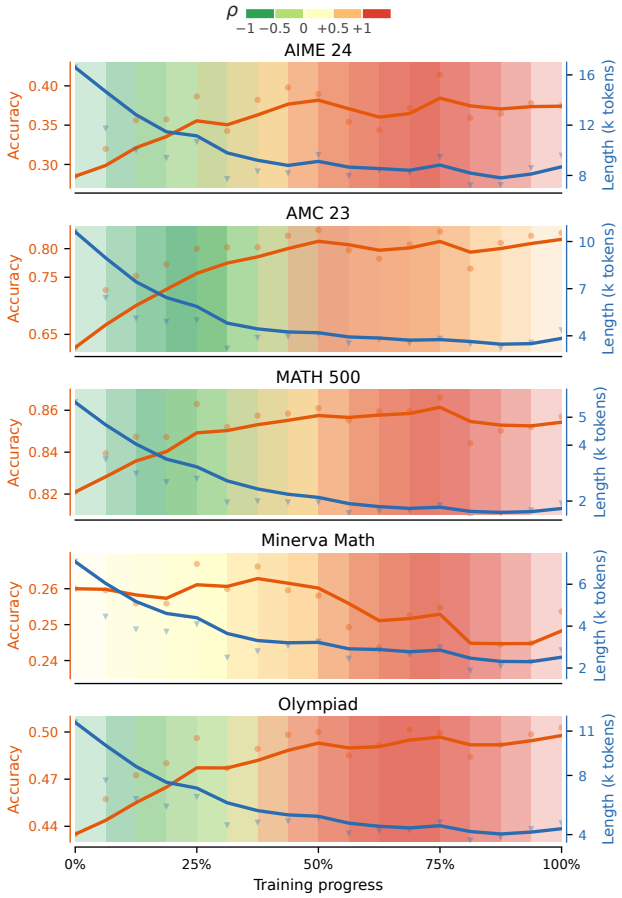


Figure 7: **PALU reduces both easy and hard redundancy across tasks.** Performance-conciseness evolution during PALU training on five evaluation tasks. *Orange:* rollout accuracy. *Blue:* generation length (k tokens). *Background:* rolling Spearman ρ between length and accuracy—green for negative ρ (easy redundancy, early training), red for positive ρ (genuine trade-off, later training); the colorbar at the top shows the mapping. PALU continues to reduce length in the red regime across all five tasks.

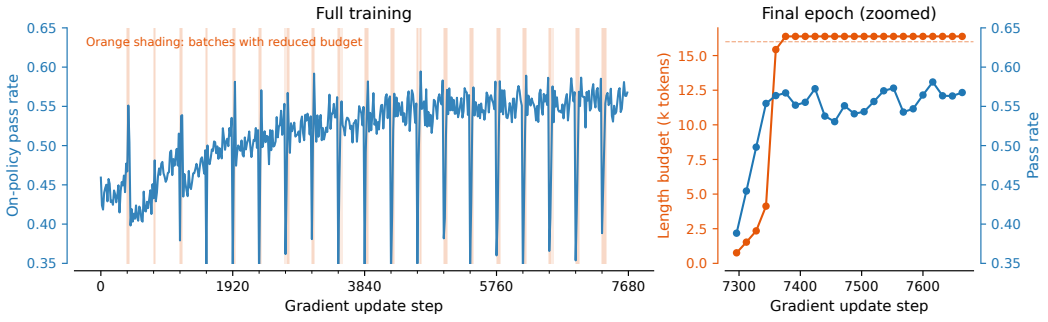


Figure 8: **Length budget and on-policy pass rate during training.** DEEPSEEK-R1-DISTILL-QWEN-1.5B on the 12k-sample dataset. *Left:* on-policy pass rate over the full training run (blue), with orange background shading marking batches where PALU reduced the budget below $L_{\max} = 16$k. The pass rate climbs steadily from ∼0.43 to ∼0.55 despite the budget pressure. *Right:* zoom on the final epoch's 24 batches, showing budget (orange) and pass rate (blue) on the same x-axis. Only 5 of 24 batches see their budget shrunk; the rest hold at $L_{\max}$, indicating that compression has converged for most prompts.

D Generation Examples

Question 1: Which one is greater, 8.11 or 8.9?

DeepSeek-R1-Distill-Qwen-1.5B:

> *First, I need to compare the whole number parts of both numbers. Both 8.11 and 8.9 have the same whole number part, which is 8. Since the whole numbers are equal, I'll move on to comparing the decimal parts. Looking at 8.11, the decimal part is 0.11, while the decimal part of 8.9 is 0.9. To make the comparison easier, I can rewrite 0.9 as 0.90. Now, comparing 0.11 and 0.90, it's clear that 0.90 is larger. Therefore, 8.9 is greater than 8.11.* Solution:
>
> To determine which number is larger between 8.11 and 8.9, follow these steps:
>
> 1. Compare the Whole Numbers: - Both numbers have the same whole number part: 8.
>
> 2. Compare the Decimal Parts: - Convert 8.9 to have two decimal places: 8.90. - Now, compare 0.11 (from 8.11) and 0.90 (from 8.90).
>
> 3. Determine Which Decimal is Larger: - 0.90 is greater than 0.11.
>
> 4. Conclusion: - Since $0.90 > 0.11$, 8.90 is greater than 8.11.
>
> Therefore, 8.9 is larger than 8.11.
>
> $\boxed{8.9}$

PALU:

> *To determine which number is larger between 8.11 and 8.9, I'll compare their decimal places. Both numbers have the same whole number part, which is 8. Next, I'll look at the tenths place. In 8.11, the tenths digit is 1, while in 8.9, it's 9. Since 9 is greater than 1, 8.9 is larger than 8.11.*
>
> To determine which number is larger between 8.11 and 8.9:
>
> 1. Compare the Whole Number Part: - Both numbers have the same whole number part, which is 8.
>
> 2. Compare the Tenths Place: - 8.11 has a tenths digit of 1. - 8.9 has a tenths digit of 9. Since 9 is greater than 1, 8.9 is larger than 8.11.
>
> $\boxed{8.9}$

Question 2: What is the first prime number after two hundred?

DeepSeek-R1-Distill-Qwen-1.5B:

To determine the first prime number after 200, I'll start by checking each number sequentially to see if it's prime. I'll begin with 201 and check if it's divisible by any prime numbers less than its square root. The square root of 201 is approximately 14.17, so I only need to check divisibility by primes up to 13. Next, I'll check 202. Since it's even, it's divisible by 2, so it's not prime. Then, 203: I'll check if it's divisible by 7, 11, and 13. 203 divided by 7 is 29, so it's not prime. After that, 204 is even, so it's divisible by 2. 205 ends with a 5, so it's divisible by 5. 206 is even, so it's divisible by 2. 207: Adding the digits gives 9, which is divisible by 3, so 207 is not prime. 208 is even, so it's divisible by 2. 209: Checking divisibility by 11, since 209 divided by 11 is 19, it's not prime. Finally, 210 is even, so it's divisible by 2. 211: Checking divisibility by primes up to 14, it doesn't divide evenly by 2, 3, 5, 7, 11, or 13. Therefore, 211 is the next prime number after 200.

To find the first prime number after 200, follow these steps:

1. Understand What a Prime Number Is: - A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

2. Start Checking Numbers After 200: - Begin with the number 201 and check each subsequent number to see if it's prime.

3. Check Divisibility: - 201: - 201 / 3 = 67 - Since 201 is divisible by 3, it's not prime. - 202: - It's even, so divisible by 2. - Not prime. - 203: - Check divisibility by primes up to sqrt203 = 14.25. - 203 / 7 = 29, so it's not prime. - 204: - It's even. - Not prime. - 205: - Ends with 5, divisible by 5. - Not prime. - 206: - It's even. - Not prime. - 207: - Sum of digits is 9, divisible by 3. - Not prime. - 208: - It's even. - Not prime. - 209: - Check divisibility by primes up to sqrt209 = 14.45. - 209 / 11 = 19, so it's not prime. - 210: - It's even. - Not prime. - 211: - Check divisibility by primes up to sqrt211 = 14.52. - 211 is not divisible by 2, 3, 5, 7, 11, or 13. - Conclusion: 211 is a prime number.

4. Final Answer: The first prime number after 200 is: $\boxed{211}$

PALU:

To determine the first prime number after two hundred, I'll start checking each number sequentially from 201 onwards.201 is divisible by 3, so it's not prime.202 is even, thus not prime.203 is divisible by 7, so it's not prime.204 is even, not prime.205 ends with a 5, so it's divisible by 5, not prime.206 is even, not prime.207 is divisible by 3, not prime.208 is even, not prime.209 is divisible by 11, so it's not prime.210 is even, not prime.211 is a prime number because it isn't divisible by any number other than 1 and itself. Therefore, the first prime number after two hundred is 211.

The first prime number after 200 is $\boxed{211}$.

Question 3 (the first AIME 2025 problem): Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.

DeepSeek-R1-Distill-Qwen-1.5B:

Okay, so I have this problem where I need to find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b. Hmm, let me think about how to approach this. First, I know that numbers in base b can be converted to decimal (base 10) by expanding each digit multiplied by powers of b . So, 17_b should be equal to $1 * b + 7 * 1$, which is $b + 7$. Similarly, 97_b would be $9 * b + 7x1$, so that's $9b + 7$. The problem says that 17_b divides 97_b . In decimal terms, this means that b + 7 is a divisor of 9b + 7 . So, mathematically, we can write this as: $\frac{9b+7}{b+7}$ is an integer Let me denote this fraction as k, where k is an integer. So, $k = \frac{9b+7}{b+7}$ To find k, I can perform polynomial division or rearrange the equation. Let me try rearranging it: Multiply both sides by $b + 7$: $9b + 7 = k(b + 7)$ Expanding the right side: $9b + 7 = kb + 7k$ Now, let's bring all terms to one side: $9b + 7 - kb - 7k = 0$ Factor out b from the terms that contain it: $b(9-k)+(7-7k) = 0$ Let me solve for b : $b(9-k) = 7k-7$ So, $b = \frac{7k-7}{9-k}$ Simplify numerator: $7(k - 1)$ So, $b = \frac{7(k-1)}{9-k}$. Since b must be an integer greater than 9, the right side must evaluate to an integer. Also, $b > 9$, so we need to find integer values of k such that b is an integer greater than 9. Let me denote m = k - 1 , so k = m + 1 . Substitute back into the equation: $b = \frac{7m}{9-(m+1)} = \frac{7m}{8-m}$ So now, $b = \frac{7m}{8-m}$. Since b must be a positive integer greater than 9, both numerator and denominator must be integers, and $8 - m$ must divide $7m$. So, $8 - m$ divides $7m$. Let me write this as: $8 - m \mid 7m$ Which implies that $8 - m$ divides $7m$. Since $8 - m$ is a divisor, let me denote $d = 8 - m$. Then, d divides $7m$, which is $7(8 - d)$. So, $d \mid 7(8 - d)$ Which is: $d \mid 56 - 7d$. Therefore, $d \mid 56 - 7d + 7d$ (since $d \mid 7d$, so adding it doesn't change divisibility). Simplifying: $d \mid 56$ So, d is a positive integer divisor of 56. Since $d = 8-m$, and m is an integer (because k is an integer, so $m = k - 1$ is also an integer), we can find possible values of d.Let me list all positive divisors of 56:1, 2, 4, 7, 8, 14, 28, 56.So, $d \in \{1, 2, 4, 7, 8, 14, 28, 56\}$.But remember that $d = 8 - m$, and $m = k - 1$. Also, since k is positive (as it's a divisor in $9b + 7 = k(b + 7)$), m must be positive or zero, but since $d = 8 - m$, and d is positive, we have $8 - m > 0$, so $m < 8$. Therefore, m can be 0, 1, 2, 3, 4, 5, 6, or 7.So, let's find corresponding d for each... (exceed 32k)

PALU:

*Okay, so I need to find the sum of all integer bases $b > 9$ where 17 base b divides 97 base b. Hmm, let me break this down.First, let's convert both numbers from base b to decimal.17 base b is equal to $1 * b + 7$. Similarly, 97 base b is $9 * b + 7$.So we need 17_b divides 97_b. Translating to decimal:$1 * b + 7$ divides $9 * b + 7$.So the condition is $(b + 7)$ divides $(9b + 7)$.So let's express this as $(9b + 7)$ divided by $(b + 7)$ must be an integer.Let me compute the division:Divide $9b+7$ by $b+7$.Using polynomial division:Divide $9b + 7$ by $b + 7$.So 9b divided by b is 9. Multiply $(b + 7)$ by 9: $9b + 63$.Subtract this from $9b + 7$: $(9b + 7) - (9b + 63) = 7 - 63 = -56$.So the division gives 9 with a remainder of -56.Therefore, $(9b+7) = 9*(b+7)-56$.So for $(b+7)$ to divide $9b+7$, $(b+7)$ must divide the remainder, which is -56.But since divisibility is concerned with absolute value, so $(b+7)$ divides 56.So $b + 7$ is a positive divisor of 56.Given that $b > 9$, so $b + 7 > 16$.So $b + 7$ is a divisor of 56 greater than 16.Let's list the divisors of 56:$1, 2, 4, 7, 8, 14, 28, 56$.Which of these are greater than 16? 28 and 56.So possible $b + 7 = 28$ or 56.Therefore $b = 21$ or 49.So the bases are 21 and 49.Sum is $21 + 49 = 70$.Hence the answer is 70.**Final Answer**The sum of all such bases is* $\boxed{70}$.

To find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b, we convert these numbers from base b to decimal: $17_b = 1 \cdot b + 7 = b + 7$-$97_b = 9 \cdot b + 7$. We need $b + 7$ to divide $9b + 7$. Using polynomial division, we find that the remainder when $9b + 7$ is divided by $b + 7$ is -56. Thus, $b + 7$ must be a divisor of 56.Given $b > 9$, we find the divisors of 56 greater than 16: 28 and 56. Therefore, $b + 7 = 28$ or $b + 7 = 56$, giving $b = 21$ or $b = 49$.The sum of these bases is $21 + 49 = 70$.Thus, the sum of all such bases is $\boxed{70}$.

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 introduction state the constrained formulation, the three-surrogate instantiation, and the empirical claims of compression-with-accuracy and cross-setting transfer; all are supported by Sections 4.1–4.3.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 5 discusses three limitations: the broad-length-distribution assumption, the lack of cross-domain generalization claims, and the conditions under which the off-policy residual estimate degrades.

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 does not contain formal theoretical results; the Lagrangian view is used as a structural scaffold, not a convergence claim.

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: The full training recipe, including hyperparameters and evaluation protocol, is provided in Appendix B.

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: [No]

   Justification: Code and trained models will be released after the review period. All datasets and base models used in the paper are publicly available and cited at first use.

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: Training details, hyperparameters, and evaluation protocols are described in Section 4.1 and Appendix B.

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: [No]

   Justification: Following common practice in reasoning LLM evaluation, we report Pass@1 averaged over multiple rollouts (32 on AIME, 10 elsewhere) without per-run error bars; the multi-rollout averaging reduces sampling variance.

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: Compute resources used for training and evaluation are described in Appendix B.

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 research conforms with the NeurIPS Code of Ethics.

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The work improves inference efficiency of reasoning LLMs and does not introduce capabilities with direct dual-use concerns beyond those of the underlying base models.

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: [N/A]

    Justification: The paper does not release new pretrained models or scraped datasets that pose elevated misuse risk.

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 base models, datasets, and frameworks (VeRL) are cited at first use; we use them under their original licenses without modification.

13. **New assets**

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

    Answer: [No]

    Justification: No new datasets, models, or code are released with the current submission; we plan to release code and trained checkpoints after acceptance.

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 human-subject research.

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 human subjects.

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 were used only for writing and editing assistance; no LLM is part of the core methodology.