# DISENTANGLED TOKEN-LEVEL CREDIT ASSIGNMENT FOR EFFICIENT REASONING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large reasoning models (LRMs) rely on explicit multi-step reasoning to substantially improve complex problem solving but are also prone to overthinking, which consumes computation with no gain in accuracy and can even derail otherwise correct trajectories. Existing approaches primarily mitigate overthinking through length penalties. However, such sequence-level signals penalize an entire long response uniformly, which suppresses useful reasoning along with redundant computation. To address this limitation, we propose Per-Objective Asymmetric Credit Estimation (PACE), which disentangles correctness from efficiency and assigns their learning signals separately at the token level, charging each generation decision for the computation it induces while reinforcing correct reasoning. Specifically, PACE employs intra-prompt correctness credit to characterize relative solution quality among responses to the same prompt, while inter-prompt rate credit apportions success gains and remaining-compute penalties to individual tokens. We further propose an asymmetric correctness budget that bounds negative efficiency credit on correct trajectories and retains the full remaining-compute penalty on erroneous trajectories. Theoretically, we establish that the inter-prompt rate credit constitutes a token-level cross-fitted estimator of the pooled solve-throughput gradient, yielding a principled decomposition of the global accuracy-compute objective into token-wise success gains and remaining-compute penalties. Extensive experiments across ten reasoning benchmarks and two model scales demonstrate that PACE reduces generation length up to 64% while improving average accuracy.

## 1 INTRODUCTION

With model capacity held fixed, allocating additional test-time compute to an individual problem substantially improves performance on complex reasoning tasks (OpenAI, 2024; Guo et al., 2025; Snell et al., 2025). Large reasoning models (LRMs) have recently extended this paradigm to agentic reasoning, tackling long-horizon, multi-stage tasks through sustained interaction with external tools and coding environments (DeepSeek-AI et al., 2026; Kimi Team et al., 2026). Such models adopt a slow-thinking paradigm, allocating computation to iterative analysis, solution exploration, and result verification before producing a final answer (Li et al., 2025), thereby establishing test-time computation as a new scaling dimension for language model capabilities.

Expanding test-time computation also induces pervasive *overthinking* in LRMs: even after finding a valid solution, models continue to verify, backtrack, and explore, increasing latency and diverting otherwise correct trajectories away from the solution (Chen et al., 2024; Sui et al., 2025). A key driver is outcome-based reinforcement learning, which rewards only final-answer correctness and ignores generation cost, allowing models to increase their probability of success by extending their search (Shao et al., 2024; Yu et al., 2025). Existing approaches pursue reasoning efficiency along two main lines. Latent reasoning moves intermediate steps from discrete text into continuous representations to reduce autoregressive decoding (Hao et al., 2024; Zhu et al., 2025; Tan et al., 2025; Jiang et al., 2026; Lu et al., 2026), but remains less accurate than explicit reasoning on complex tasks. Post-training methods retain explicit reasoning and shape the outcome reward with response length, either through length penalties (Luo et al., 2025; Aggarwal & Welleck, 2025; Arora & Zanette, 2025; Yuan et al., 2025) or through reasoning budgets adapted to problem difficulty and to the lengths of successful trajectories (Shen et al., 2025; Liu et al., 2025a; Yi et al., 2025; Wu et al.,



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

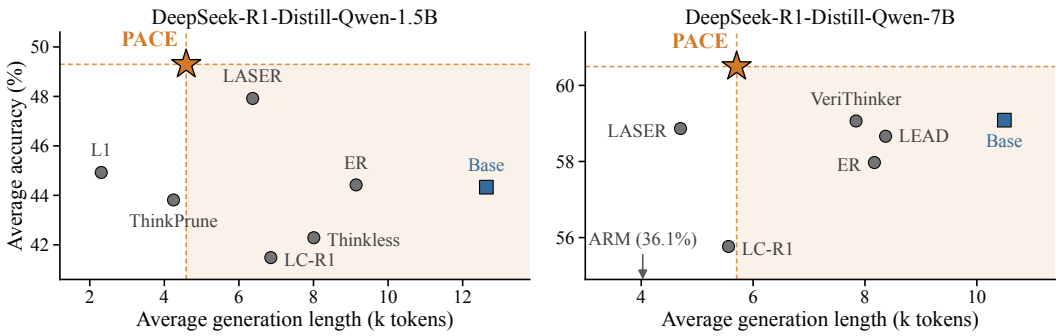


Figure 1: **PACE improves accuracy and efficiency together.** Average accuracy versus average generation length over the ten benchmarks for DeepSeek-R1-Distill-Qwen-1.5B (left) and 7B (right). PACE is the most accurate method at both scales. The shaded region contains the methods that are both longer and less accurate.

2025b). Consequently, length-based post-training has become the mainstream approach to efficient reasoning, as it preserves the expressive capacity of explicit reasoning while improving generation efficiency (Sui et al., 2025).

However, *length-based reward shaping suffers from a fundamental credit-assignment issue.* First, these methods compress generation efficiency into a single sequence-level length signal. When a correct trajectory is penalized for its length, the negative credit is broadcast to all of its tokens, suppressing necessary reasoning steps together with redundant computation. Consistent with this, all four length-based baselines evaluated at the 7B scale are less accurate than the base model on competition mathematics (Table 2). Second, within-prompt relative length pressure continues to penalize longer trajectories whenever sampled responses differ in length, even when all of them are already concise. Third, correct-only length shaping assigns efficiency credit exclusively to successful trajectories, excluding long incorrect trajectories from efficiency optimization. These failures dominate token consumption: on competition benchmarks, the incorrect responses of DS-1.5B are far longer than its correct ones and account for more than 80% of the generated tokens on each benchmark (Section 3.4). Shortening successful trajectories alone therefore cannot control total generation cost. Effective mitigation of overthinking requires token-level attribution of task correctness and subsequent computation cost to the corresponding generation decisions.

To this end, we propose Per-Objective Asymmetric Credit Estimation (PACE), which jointly optimizes reasoning correctness and generation efficiency from the perspective of token-level credit assignment. PACE disentangles the two objectives into complementary learning signals. An intra-prompt correctness credit characterizes solution quality through relative outcomes among trajectories sampled for the same prompt, providing a stable correctness anchor. An inter-prompt rate credit leverages cross-prompt success rates and compute usage to attribute success gains and the subsequent computation induced by each token to the corresponding generation decisions, thereby directly optimizing solve-throughput, defined as the number of correct solutions produced per generated token. Since the relative weight of success and computation is estimated from the other prompts in each batch, it adapts to the policy instead of being fixed in advance. PACE further introduces an asymmetric correctness budget that bounds the negative rate credit of correct trajectories but applies the full penalty to incorrect ones. This asymmetry brings failures into efficiency optimization without suppressing effective reasoning. Theoretically, we establish that the inter-prompt rate credit constitutes an unbiased token-level cross-fitted estimator of the solve-throughput gradient up to a positive scale, which provides a principled foundation for decomposing success gains and computation costs at the token level.

Experiments across ten reasoning benchmarks and two model scales show that PACE improves the accuracy–efficiency frontier (Figure 1), reducing generation length by 45–64% while achieving the highest average accuracy among all compared methods. Further analysis at the 1.5B scale shows that incorrect responses shrink more than correct ones on all ten benchmarks, and that PACE remains more accurate than the base model under every matched thinking budget.

# 2 METHOD

In this section, we present Per-Objective Asymmetric Credit Estimation (PACE), which decomposes the correctness–efficiency trade-off into two credits with distinct scopes. An intra-prompt correctness advantage preserves relative solution quality among responses to the same prompt, while an inter-prompt rate credit assigns success gains and remaining-compute costs to individual token decisions. PACE then applies an asymmetric correctness budget that bounds negative rate credit on correct trajectories while retaining the full remaining-compute cost of incorrect trajectories.

## 2.1 BUDGETED SOLVE-THROUGHPUT OBJECTIVE

Consider a rollout batch containing $M$ prompts $\{q_g\}_{g=1}^{M}$ sampled from the training distribution. For each prompt, the policy $\pi_\theta$ independently samples $G$ responses. We denote the $i$-th response to prompt $q_g$ by $o_{g,i} = (o_{g,i,1}, \ldots, o_{g,i,T_{g,i}})$, where $i \in \{1, \ldots, G\}$, $B$ denotes the generation budget, and $T_{g,i} = |o_{g,i}| \in \{1, \ldots, B\}$ is the response length. A verifier assigns each response a binary correctness reward $r_{g,i} \in \{0, 1\}$. We define its normalized generation cost as $\ell_{g,i} = T_{g,i}/B \in (0, 1]$.

With the expected correctness reward $\mu_r(\theta) = \mathbb{E}_{\pi_\theta}[r]$ and expected normalized generation cost $\mu_\ell(\theta) = \mathbb{E}_{\pi_\theta}[\ell]$ under $\pi_\theta$, we define *solve-throughput* under generation budget $B$ as

$$\eta_B(\theta) = \frac{\mu_r(\theta)}{\mu_\ell(\theta)}. \tag{1}$$

This ratio measures the number of correct solutions produced per unit of generated tokens, and provides a common exchange rate between success and compute that adapts to the policy.

## 2.2 INTER-PROMPT TOKEN-LEVEL RATE CREDIT

The solve-throughput gradient is $\nabla_\theta \eta_B = (\mu_\ell \nabla_\theta \mu_r - \mu_r \nabla_\theta \mu_\ell)/\mu_\ell^2$. Let $\boldsymbol{\psi}_{g,i,t} = \nabla_\theta \log \pi_\theta(o_{g,i,t} \mid q_g, o_{g,i,<t})$ denote the token-level log-policy gradient. The policy-gradient identities for expected correctness and generation cost are

$$\nabla_\theta \mu_r = \mathbb{E}\left[ r_{g,i} \sum_{t=1}^{T_{g,i}} \boldsymbol{\psi}_{g,i,t} \right], \qquad \nabla_\theta \mu_\ell = \mathbb{E}\left[ \sum_{t=1}^{T_{g,i}} \frac{T_{g,i} - t + 1}{B} \boldsymbol{\psi}_{g,i,t} \right]. \tag{2}$$

Define the normalized remaining compute from token $t$ onward as $\kappa_{g,i,t} = (T_{g,i} - t + 1)/B$, so that $\nabla_\theta \mu_\ell = \mathbb{E}\big[\sum_t \kappa_{g,i,t} \boldsymbol{\psi}_{g,i,t}\big]$. Substituting these identities into the solve-throughput gradient yields

$$\mu_\ell^2 \nabla_\theta \eta_B = \mathbb{E}\left[ \sum_{t=1}^{T_{g,i}} (\mu_\ell r_{g,i} - \mu_r \kappa_{g,i,t})\, \boldsymbol{\psi}_{g,i,t} \right]. \tag{3}$$

The coefficient $\mu_\ell r_{g,i} - \mu_r \kappa_{g,i,t}$ assigns each token its contribution to the solve-throughput gradient.

Estimating $\mu_r$ and $\mu_\ell$ from group $g$'s own responses correlates these estimates with the log-policy gradients they weight since both depend on the same sampled outcomes. This dependence persists under leave-one-response-out estimation, as the remaining responses are conditioned on the same prompt $q_g$ and therefore share its difficulty. PACE therefore applies leave-one-prompt-out cross-fitting, estimating $\mu_r$ and $\mu_\ell$ from all other prompt groups:

$$\hat{\mu}_{r,-g} = \frac{1}{(M-1)G} \sum_{h \neq g} \sum_{j=1}^{G} r_{h,j}, \qquad \hat{\mu}_{\ell,-g} = \frac{1}{(M-1)G} \sum_{h \neq g} \sum_{j=1}^{G} \ell_{h,j}. \tag{4}$$

Independent prompt and rollout sampling makes $\hat{\mu}_{r,-g}$ and $\hat{\mu}_{\ell,-g}$ independent of the responses and token gradients associated with $q_g$, while providing unbiased estimates of $\mu_r$ and $\mu_\ell$. Substituting these estimates into the expected-gradient coefficient yields the inter-prompt token-level rate credit

$$\hat{A}^{\text{rate}}_{g,i,t} = r_{g,i}\hat{\mu}_{\ell,-g} - \hat{\mu}_{r,-g}\kappa_{g,i,t}. \tag{5}$$

The success term $r_{g,i}\hat{\mu}_{\ell,-g}$ rewards correct responses at the generation-cost scale of other prompts, whereas the compute term $\hat{\mu}_{r,-g}\kappa_{g,i,t}$ charges each action for its downstream generation at the cross-prompt success rate. The resulting credit assigns larger costs to earlier decisions that initiate long continuations. As shown in Figure 2, the rate credit charges each token only for the computation it induces, unlike a sequence-level length penalty that charges every token for the total response length. Equivalently, $\hat{A}^{\text{rate}}_{g,i,t} = \hat{\mu}_{\ell,-g}\big(r_{g,i} - \hat{\eta}_{-g}\kappa_{g,i,t}\big)$ with $\hat{\eta}_{-g} = \hat{\mu}_{r,-g}/\hat{\mu}_{\ell,-g}$: a correctness reward minus a remaining-compute (reward-to-go) cost priced at the solve-throughput of other prompts. This price is estimated from the batch and tracks the policy as responses become shorter and more accurate.

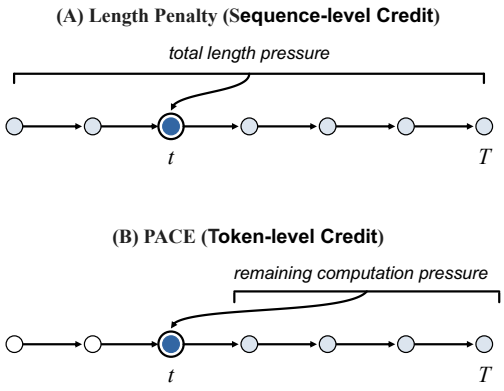


Figure 2: **Length penalty *vs.* PACE.** (A) A length penalty decays the reward with a hand-designed function of the total response length and gives every token the same decayed reward. (B) PACE derives its credit from the solve-throughput gradient: each token is charged for the computation from it onward, priced at the success rate of the other prompts.

**Proposition 1 (Cross-fitted solve-throughput gradient).** Under the grouped on-policy sampling process above,

$$\mathbb{E}\left[\sum_{g=1}^{M}\sum_{i=1}^{G}\sum_{t=1}^{T_{g,i}} \hat{A}^{\text{rate}}_{g,i,t}\boldsymbol{\psi}_{g,i,t}\right] = MG\mu_\ell^2\nabla_\theta\eta_B. \tag{6}$$

The proof (Appendix B) relies only on the independence and unbiasedness of the leave-one-prompt-out estimates. The inter-prompt rate credit thus constitutes an unbiased token-level cross-fitted estimator of $MG\mu_\ell^2\nabla_\theta\eta_B$. For every incorrect response, $\hat{A}^{\text{rate}}_{g,i,t} = -\hat{\mu}_{r,-g}\kappa_{g,i,t} \le 0$: whenever $\hat{\mu}_{r,-g} > 0$, every action followed by additional generation receives negative rate credit.

**Intra-Prompt Correctness Advantage.** For correctness, we follow GRPO's use of responses to the same prompt as a group-relative baseline (Shao et al., 2024). Let $\bar{r}_g = G^{-1}\sum_{j=1}^{G} r_{g,j}$ denote the mean reward for prompt $q_g$. We define $\hat{A}^{\text{acc}}_{g,i} = r_{g,i} - \bar{r}_g$. This response-level advantage is shared across tokens and serves as the correctness anchor in the composition below.

### 2.3 Asymmetric Correctness Budget

The solve-throughput objective trades success probability against generation cost at the global rate $\mu_r/\mu_\ell$. Under this global exchange rate, the remaining-compute term can assign negative rate credit to early tokens in a long correct response, including tokens that support its successful reasoning path. PACE applies an *asymmetric correctness budget* across outcomes: it bounds negative rate credit on correct responses and preserves the full rate credit on incorrect responses.

Given a correctness-budget coefficient $\gamma \in [0, 1]$, PACE defines the budgeted rate credit as

$$\widetilde{A}^{\text{rate}}_{g,i,t} = \begin{cases} \max\Big(\hat{A}^{\text{rate}}_{g,i,t}, -\gamma\hat{A}^{\text{acc}}_{g,i}\Big), & r_{g,i} = 1, \\ \hat{A}^{\text{rate}}_{g,i,t}, & r_{g,i} = 0. \end{cases} \tag{7}$$

The final PACE advantage is $\hat{A}^{\text{PACE}}_{g,i,t} = \hat{A}^{\text{acc}}_{g,i} + \widetilde{A}^{\text{rate}}_{g,i,t}$. For every correct response, $\hat{A}^{\text{acc}}_{g,i} = 1 - \bar{r}_g \ge 0$. The asymmetric budget therefore guarantees

$$\hat{A}^{\mathrm{PACE}}_{g,i,t} \geq (1-\gamma)\hat{A}^{\mathrm{acc}}_{g,i} \geq 0. \tag{8}$$

The parameter $\gamma$ specifies the fraction of the intra-prompt correctness credit that negative rate credit can consume. Positive rate credit remains unchanged, while incorrect responses receive both the group-relative correctness penalty and the full remaining-compute penalty. Proposition 1 characterizes the rate credit before this bound; the budget raises the credit of a correct response only on tokens whose combined advantage would otherwise fall below $(1-\gamma)\hat{A}^{\mathrm{acc}}_{g,i}$. The correctness budget ensures that every token in a correct response receives an advantage of at least $(1-\gamma)(1-\bar{r}_g)$, providing a stronger positive learning signal for rare successful responses to difficult prompts.

The combined token-level advantage defines the PACE policy gradient direction as

$$\boldsymbol{g}_{\mathrm{PACE}}(\theta) = \mathbb{E}\left[\frac{1}{B}\sum_{t=1}^{T_{g,i}} \hat{A}^{\mathrm{PACE}}_{g,i,t} \nabla_\theta \log \pi_\theta\left(o_{g,i,t} \mid q_g, o_{g,i,<t}\right)\right]. \tag{9}$$

# 3 Experiments

## 3.1 Experimental Settings

**Datasets.** We train PACE on DAPO-Math-17k (Yu et al., 2025), which contains 17,909 mathematical problems with reference answers used to compute correctness rewards during post-training. We compare PACE with baseline methods on 10 benchmarks spanning three categories: general mathematical reasoning (GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024)); competition and olympiad reasoning (AIME 2024, AIME 2025, AMC 2023, BRUMO 2025, CMIMC 2025, HMMT 2025 (Balunović et al., 2025), and OlympiadBench (He et al., 2024)); scientific reasoning (GPQA-Diamond (Rein et al., 2024)).

**Models and Baselines.** We use DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025) as base reasoning models, abbreviated as DS-1.5B and DS-7B, respectively. We compare PACE with the base models and 9 efficient-reasoning methods. EfficientReason (ER) (Arora & Zanette, 2025) incorporates a continuous length penalty into the outcome reward; LC-R1 (Cheng et al., 2025) combines length and compression rewards to suppress redundant verification; and LASER (Liu et al., 2025a) uses a step-function reward controlled by a target response length. LEAD (Zhang & Zuo, 2025) combines length regularization, penalties for incorrect answers, and difficulty-aware advantage reweighting. L1 (Aggarwal & Welleck, 2025) learns to satisfy user-specified token budgets, while ThinkPrune (Hou et al., 2025) progressively tightens generation limits during reinforcement learning. ARM (Wu et al., 2025a) learns to select among direct answers, short reasoning, code, and long reasoning; Thinkless (Fang et al., 2025) learns to choose between short and long reasoning; and VeriThinker (Chen et al., 2025) reduces redundant self-reflection through auxiliary training on solution verification. EfficientReason, LC-R1, and LASER are evaluated at both scales; ARM, LEAD, and VeriThinker at 7B; and L1, ThinkPrune, and Thinkless at 1.5B. For all baselines, we evaluate their publicly released checkpoints under the same evaluation protocol.

**Implementation Details.** We implement PACE using TRL, with vLLM for rollout generation. Training uses LoRA (Hu et al., 2022) of rank 256 applied to all attention and feed-forward projection layers, optimized with AdamW at a constant learning rate of $10^{-5}$. Each rollout batch contains $M = 64$ prompts, and for each prompt we sample $G = 16$ responses at temperature 1.0 with a generation budget of $B = 16{,}384$ tokens. For both models, the correctness budget is set to $\gamma = 0.9$ and the reference-policy KL coefficient to 0.01. Policy updates use GSPO-token importance ratios (Zheng et al., 2025) for 1,000 optimizer steps. All training experiments are conducted on 4 NVIDIA H100 GPUs; Appendix D lists the complete hyperparameters. For evaluation, all methods use temperature 0.6, top-$p$ 0.95, a maximum generation length of 32,768 tokens, and the same benchmark-specific answer extraction and verification procedures. We sample 16 responses per problem on AIME 2024, AIME 2025, AMC 2023, BRUMO 2025, CMIMC 2025, and HMMT 2025; eight on GPQA-Diamond; and four on the remaining benchmarks. We report average accuracy across sampled responses and mean completion length.

Table 1: Accuracy (%) and generation length (tokens) on mathematical and scientific reasoning benchmarks; bold and underlined accuracies indicate the best and second-best results.

| Method | GSM8K | | MATH-500 | | AMC 2023 | | OlympiadBench | | GPQA-Diamond | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ |
| **DeepSeek-R1-Distill-Qwen-1.5B** | | | | | | | | | | | | |
| Base Model | 2488 | 86.09 | 5694 | 86.20 | 10193 | 71.25 | 11645 | 54.12 | 10510 | 32.58 | 8106 | 66.05 |
| ER | 935 | 84.51 | 3102 | 84.00 | 5617 | 72.19 | 8084 | 52.89 | 7498 | 32.83 | 5047 | 65.28 |
| LC-R1 | 950 | 83.64 | 2602 | 82.15 | 4315 | 69.53 | 5634 | 49.81 | 7781 | 31.06 | 4256 | 63.24 |
| LASER | 1803 | 86.20 | 3282 | **87.95** | 5037 | <u>77.03</u> | 5885 | <u>59.61</u> | 5987 | **37.18** | 4399 | <u>69.59</u> |
| L1 | 1328 | **88.12** | 1777 | 84.00 | 2174 | 74.53 | 2181 | 55.86 | 2105 | 36.68 | 1913 | 67.84 |
| ThinkPrune | 845 | 86.07 | 2068 | 84.60 | 2934 | 71.09 | 3752 | 54.12 | 4021 | 33.71 | 2724 | 65.92 |
| Thinkless | 409 | 82.54 | 2618 | 81.00 | 5803 | 69.06 | 6623 | 51.97 | 11286 | 30.56 | 5348 | 63.03 |
| **PACE** | 822 | <u>87.07</u> | 2130 | <u>87.90</u> | 3632 | **80.62** | 4257 | **60.20** | 4123 | <u>36.81</u> | 2993 | **70.52** |
| **DeepSeek-R1-Distill-Qwen-7B** | | | | | | | | | | | | |
| Base Model | 1664 | 92.29 | 4120 | **94.05** | 6453 | 90.47 | 8826 | 69.14 | 8933 | 49.49 | 5999 | 79.09 |
| ER | 913 | 92.15 | 2967 | 93.20 | 4874 | 88.12 | 6991 | 68.29 | 6938 | 49.62 | 4537 | 78.28 |
| LC-R1 | 423 | 88.48 | 1586 | 91.00 | 3184 | 86.88 | 4248 | 64.99 | 5397 | 48.11 | 2968 | 75.89 |
| LASER | 961 | **93.23** | 1919 | 93.00 | 2900 | <u>90.62</u> | 3739 | <u>69.47</u> | 4287 | 48.55 | 2761 | 78.98 |
| ARM | 381 | 89.76 | 1498 | 75.25 | 2774 | 56.72 | 3241 | 43.95 | 2623 | 36.30 | 2103 | 60.40 |
| LEAD | 1344 | 92.59 | 3320 | 93.60 | 5163 | 90.31 | 7042 | **69.55** | 7275 | 47.35 | 4829 | 78.68 |
| VeriThinker | 1078 | <u>93.21</u> | 2703 | 93.90 | 4401 | <u>90.62</u> | 6654 | 69.03 | 5863 | <u>50.76</u> | 4140 | <u>79.51</u> |
| **PACE** | 883 | 92.93 | 2060 | <u>94.00</u> | 3359 | **90.78** | 4599 | 69.36 | 5471 | **50.95** | 3274 | **79.60** |

Table 2: Accuracy (%) and generation length (tokens) on competition mathematics benchmarks; bold and underlined accuracies indicate the best and second-best results.

| Method | AIME 2024 | | AIME 2025 | | BRUMO 2025 | | CMIMC 2025 | | HMMT 2025 | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ | Len. ↓ | Acc.(%) ↑ |
| **DeepSeek-R1-Distill-Qwen-1.5B** | | | | | | | | | | | | |
| Base Model | 17291 | 28.33 | 16355 | <u>25.00</u> | 15689 | 33.54 | 18501 | 13.91 | 17989 | 12.29 | 17165 | 22.61 |
| ER | 13121 | 29.58 | 12940 | 24.79 | 11025 | 36.67 | 14800 | 14.06 | 14308 | 12.71 | 13239 | 23.56 |
| LC-R1 | 8733 | 23.12 | 8549 | 20.21 | 8914 | 27.92 | 10085 | 14.06 | 10993 | 13.33 | 9455 | 19.73 |
| LASER | 8493 | <u>35.00</u> | 8312 | 24.17 | 7727 | <u>40.83</u> | 8571 | 16.41 | 8589 | <u>14.79</u> | 8339 | <u>26.24</u> |
| L1 | 2854 | 27.71 | 2557 | 24.38 | 2637 | 30.21 | 2721 | <u>17.34</u> | 2755 | 10.42 | 2705 | 22.01 |
| ThinkPrune | 5668 | 28.96 | 5262 | 21.67 | 5113 | 31.87 | 6102 | 15.62 | 6699 | 10.42 | 5769 | 21.71 |
| Thinkless | 10377 | 25.83 | 10429 | 23.96 | 9524 | 31.87 | 11453 | 13.59 | 11567 | 12.50 | 10670 | 21.55 |
| **PACE** | 6545 | **36.67** | 6175 | **28.33** | 5827 | **41.25** | 6606 | **18.28** | 5746 | **15.83** | 6180 | **28.07** |
| **DeepSeek-R1-Distill-Qwen-7B** | | | | | | | | | | | | |
| Base Model | 13667 | <u>54.37</u> | 14395 | <u>38.96</u> | 13169 | 51.04 | 16917 | 26.88 | 16772 | <u>24.17</u> | 14984 | <u>39.08</u> |
| ER | 10742 | 51.88 | 11780 | 36.25 | 9596 | 51.25 | 13603 | 25.00 | 13283 | 23.96 | 11800 | 37.67 |
| LC-R1 | 7751 | 48.12 | 8091 | 34.58 | 6698 | 50.83 | 9019 | 24.06 | 9212 | 20.62 | 8154 | 35.65 |
| LASER | 6264 | 53.12 | 6843 | <u>38.96</u> | 5450 | <u>53.33</u> | 7238 | 27.34 | 7419 | 21.04 | 6643 | 38.76 |
| ARM | 6999 | 16.04 | 7094 | 10.62 | 5142 | 22.50 | 5094 | 7.19 | 5425 | 2.29 | 5951 | 11.73 |
| LEAD | 11515 | 50.00 | 11404 | 38.54 | 9875 | 52.71 | 13554 | 27.19 | 13191 | **24.79** | 11908 | 38.65 |
| VeriThinker | 9916 | **55.83** | 10889 | 37.92 | 9620 | 49.79 | 13396 | <u>27.50</u> | 13885 | 22.08 | 11541 | 38.62 |
| **PACE** | 7354 | 53.75 | 8215 | **41.25** | 7003 | **56.04** | 9279 | **31.72** | 8834 | <u>24.17</u> | 8137 | **41.39** |

### 3.2 Main Results

**Overall Performance.** PACE achieves the highest average accuracy among the compared methods across both benchmark groups and model scales, while reducing generation length by 45.4–64.0% relative to the distilled base models. On the mathematical and scientific benchmarks in Table 1, PACE attains average accuracies of 70.52% for DS-1.5B and 79.60% for DS-7B, improving over the base models by 4.47 and 0.51 while reducing response length by 63.1% and 45.4%, respectively. On competition mathematics (Table 2), the corresponding accuracy gains reach 5.46 and 2.31, accompanied by length reductions of 64.0% and 45.7%. These improvements hold across both model scales, with larger accuracy gains on competition problems.

**Comparison with Efficient-Reasoning Baselines.** PACE improves both accuracy and efficiency over the strong baseline LASER at 1.5B: average accuracy increases by 0.93 with 32.0% fewer tokens in Table 1, and by 1.83 with 25.9% fewer tokens in Table 2. At 7B, PACE edges out Veri-Thinker's accuracy on the mathematical and scientific reasoning benchmarks while using 20.9% fewer tokens; on competition mathematics benchmarks, it gains 2.77 with 29.5% fewer tokens. More aggressive compression yields a different trade-off: L1 and ThinkPrune generate shorter responses at 1.5B, but their average competition accuracies trail PACE by 6.06 and 6.36. LASER also produces shorter responses at 7B, while PACE achieves higher average accuracy in both groups.

**Performance on Competition Mathematics.** PACE achieves the highest accuracy on all five competition benchmarks at 1.5B and on AIME 2025, BRUMO 2025, and CMIMC 2025 at 7B. On DS-7B, AIME 2025 accuracy increases from 38.96% to 41.25%, while mean response length falls from 14,395 to 8,215 tokens. On CMIMC 2025, accuracy rises from 26.88% to 31.72%, with length decreasing from 16,917 to 9,279 tokens. Substantial generation savings thus extend to demanding mathematical problems while preserving the reasoning needed for higher accuracy.

**Performance on Out-of-Distribution Scientific Reasoning.** PACE also improves performance on GPQA-Diamond following post-training on mathematical problems. Accuracy increases by 4.23 for DS-1.5B and 1.46 for DS-7B, while mean response length decreases by 60.8% and 38.8%, respectively. PACE's accuracy-efficiency improvements thus transfer from mathematical training to scientific reasoning.

### 3.3 Ablation Study

**Setup.** Each variant removes one component of PACE on DS-1.5B and keeps all other training and evaluation settings.

- *correct-only* sets $\widetilde{A}^{\mathrm{rate}}_{g,i,t} = 0$ on incorrect responses, restricting efficiency credit to successful trajectories as correct-only length shaping does. These responses still receive the correctness advantage and still enter the cross-prompt moments.
- *w/o cross-fitting* replaces $\hat{\mu}_{r,-g}$ and $\hat{\mu}_{\ell,-g}$ with batch moments over all $MG$ responses, including those of $q_g$.
- *w/o budget* removes the lower bound $-\gamma \hat{A}^{\mathrm{acc}}_{g,i}$ on the rate credit of correct responses.

Table 3: **Each component pays off where problems are hard.** Ablation on DS-1.5B, where each variant removes one component of PACE. We report generation length (tokens) and accuracy (%); averages weight benchmarks equally. PACE attains the highest average accuracy and the lowest average length, and the differences concentrate on the competition benchmarks. Bold marks the highest accuracy in each row; lengths are not marked, as the shortest variant is not the best.

| | **PACE** | | **correct-only** | | **w/o cross-fitting** | | **w/o budget** | |
|---|---|---|---|---|---|---|---|---|
| **Benchmark** | Len.↓ | Acc.↑ | Len.↓ | Acc.↑ | Len.↓ | Acc.↑ | Len.↓ | Acc.↑ |
| GSM8K | 822 | 87.07 | 902 | 86.32 | 808 | **87.11** | 823 | 86.81 |
| MATH-500 | 2130 | **87.90** | 2534 | 87.40 | 2122 | 87.60 | 2034 | 87.55 |
| AMC 2023 | 3632 | 80.62 | 4560 | 80.00 | 3681 | 80.94 | 3611 | **81.72** |
| OlympiadBench | 4257 | **60.20** | 5704 | 57.86 | 4321 | 60.13 | 4039 | 58.86 |
| GPQA-Diamond | 4123 | 36.81 | 5732 | 36.93 | 4295 | 37.31 | 4141 | **37.69** |
| AIME 2024 | 6545 | 36.67 | 9342 | **37.92** | 7075 | 34.58 | 6918 | 36.88 |
| AIME 2025 | 6175 | 28.33 | 8680 | 26.88 | 6097 | **28.96** | 6024 | 28.12 |
| BRUMO 2025 | 5827 | 41.25 | 8097 | 38.54 | 6049 | 40.42 | 6231 | **42.29** |
| CMIMC 2025 | 6606 | **18.28** | 10016 | 13.75 | 6913 | 15.00 | 6599 | 14.84 |
| HMMT 2025 | 5746 | **15.83** | 8980 | 13.96 | 6255 | **15.83** | 6090 | 15.42 |
| Average | 4586 | **49.30** | 6455 | 47.96 | 4762 | 48.79 | 4651 | 49.02 |

**Charging failures drives the compression.** PACE attains both the highest average accuracy and the lowest average length in Table 3, and every removal loses on both axes on average. Removing the rate credit on incorrect responses is the most costly: responses grow longer on all ten benchmarks,

by 40.7% on average, and accuracy drops by 1.34. The increase ranges from 9.7% on GSM8K to 56.3% on HMMT 2025 and tracks the failure rate of the benchmark, with a Spearman correlation of 0.98, which identifies the failures as the channel. Once their rate credit is removed, the credit of an incorrect response no longer depends on its length: every token receives $-\bar{r}_g$ whether the response ends early or exhausts the budget, so the more often a benchmark is failed, the more generation escapes the objective.

**Cross-fitting and the correctness budget matter where problems are hard.** On the benchmarks of Table 1 both variants match PACE, at $+0.10$ and $+0.01$ points, and both lose on the competition benchmarks of Table 2, by 1.11 and 0.56, while generating 4.8% and 3.1% more tokens: the two components are inert on problems the policy already solves and decisive on the ones it does not. Without cross-fitting, a prompt whose responses fail and run long lowers the compute price $\hat{\mu}_r$ applied to its own tokens. The perturbation is only $1/M$ of the batch, but it is correlated with the very gradients it weights, which is the dependence that Proposition 1 removes; accuracy falls by 2.09 on AIME 2024 and 3.28 on CMIMC 2025. Without the budget, the remaining-compute term pushes the advantage of early tokens in long correct responses below zero (Appendix C.1), and accuracy falls by 3.44 on CMIMC 2025.

### 3.4 Analysis

To understand how PACE spends fewer tokens while solving more problems, we analyze the responses of DS-1.5B and PACE-1.5B, i.e., PACE applied to DS-1.5B, on the ten benchmarks. The rate credit makes two predictions that benchmark averages cannot separate: the compute charge should fall hardest on the responses that consume computation without producing a solution, and the reasoning that survives the charge should still be sufficient to solve problems. We test the first by comparing the length distributions of correct and incorrect responses, and the second by giving both models the same thinking budget at inference, which removes response length itself as an explanation of the gains.

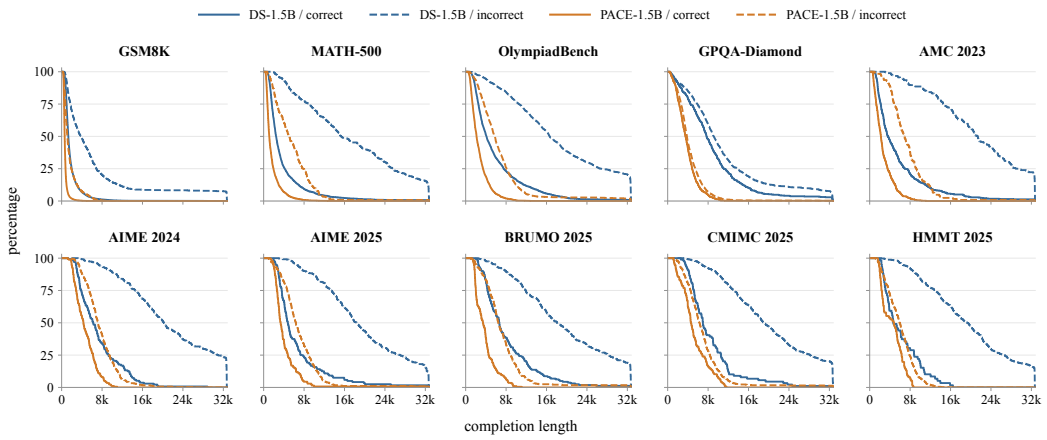


Figure 3: **PACE collapses the failure tail.** Fraction of responses whose completion length exceeds each threshold, shown separately for correct (solid) and incorrect (dashed) responses. The drop at the right edge marks responses truncated at the 32,768-token generation limit.

**PACE collapses the failure tail.** Figure 3 traces the fraction of responses that exceed each length, separately for correct and incorrect ones. The long tail of DS-1.5B consists mainly of incorrect responses: on AIME 2024, 23.5% of them exceed 32k tokens, and on every competition benchmark of Table 2, incorrect responses consume more than 80% of all generated tokens. Under PACE, fewer than 2% of incorrect responses exceed 32k tokens on every benchmark, the 95th-percentile response length on the competition benchmarks drops from 32.6–32.7k to 10.4–11.8k tokens, and on these benchmarks PACE-1.5B spends about as many tokens on a failed attempt as DS-1.5B spends on a successful one. The compression follows the asymmetry of the credit design. Incorrect responses shrink more than correct ones on all ten benchmarks: on AIME 2024, the mean incorrect response shortens by 64% (21.2k $\rightarrow$ 7.6k tokens) and the mean correct response by 36% (7.3k $\rightarrow$ 4.7k), while

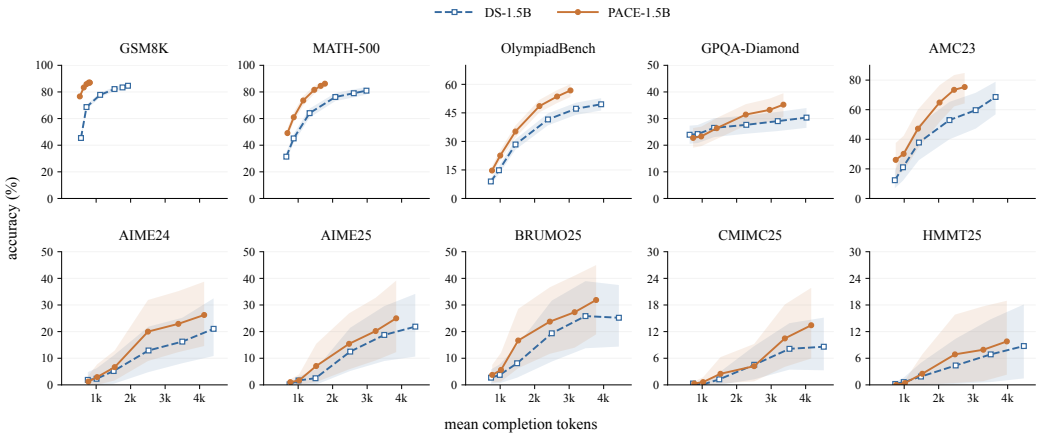


Figure 4: **PACE reaches higher accuracy with fewer tokens under matched thinking budgets.** Accuracy versus mean completion tokens under thinking budgets of 256, 512, 1024, 2048, 3072, and 4096 tokens (markers from left to right); token counts include reasoning, answer, and the inserted closing prompt. Shading denotes 95% bootstrap intervals.

accuracy rises from 28.33% to 36.67%. Incorrect trajectories bear the full remaining-compute cost, whereas the correctness budget bounds the cost on correct ones.

**PACE solves more problems with fewer tokens under matched budgets.** Both models now receive the same thinking budget, from 256 to 4096 tokens: a model that reaches the budget has its thinking closed and receives 512 tokens for the answer. Averaged over the ten benchmarks, PACE-1.5B is more accurate than DS-1.5B at every budget, by 4.65–6.83 (Figure 4). With a 3072-token budget, PACE already surpasses DS-1.5B with a 4096-token budget by 2.11 while generating 30.3% fewer tokens, and on eight of the ten benchmarks a smaller budget suffices for PACE to do so. The gap is widest under the tightest budget, where the ten-benchmark average reaches 6.83: with thinking cut off after 256 tokens, PACE reaches 76.6% accuracy on GSM8K against 45.4% for DS-1.5B. PACE is trained with a remaining-compute charge on every token, so it reaches an answer earlier and keeps answering when its thinking is closed for it, which DS-1.5B has never been trained to do. PACE also allocates computation by need. At the 4096-token budget, it reduces token cost by 57.8% on GSM8K and by 6.2% on AIME 2024, where both models use nearly the full budget, and its cost on GSM8K stays at 0.8k tokens from the 2048-token budget onward while that of DS-1.5B continues to grow. PACE learns this allocation without a difficulty-dependent length target.

## 4 Conclusion

Efficient reasoning is commonly treated as a length problem, and the prevailing remedy is to shape the outcome reward with the length of the response. Our results place the difficulty in where that signal is applied: a sequence-level penalty charges every token of a correct response for the whole response, suppressing the reasoning that produced the solution along with the computation that followed it, while the trajectories that consume most of the budget receive no efficiency signal.

PACE follows from this observation. It charges each token for the computation that comes after it and prices that computation at the solve-throughput of the other prompts in the batch. The resulting rate credit is an unbiased token-level cross-fitted estimator of the solve-throughput gradient up to a positive scale, while the correctness budget prevents efficiency pressure from overwhelming the correctness signal on correct trajectories. Across 10 benchmarks and two model scales, the resulting policy is shorter and more accurate than the base models and 9 efficient-reasoning baselines, and its accuracy gains and token savings fall on the same questions rather than trading against each other. Pricing computation, not constraining length, is what lets accuracy and efficiency improve together.

## AI USE STATEMENT

Large language models (LLMs) were used solely as auxiliary tools for coding assistance, grammar correction, and formatting checks. The research idea, experimental design, and writing of this paper were carried out entirely by the authors. No LLM-generated content was included without thorough review and verification by the authors, who take full responsibility for the entire content of this work.

## REPRODUCIBILITY STATEMENT

We provide a complete description of our method, including the training configuration, hyperparameters, evaluation metrics, and evaluation protocol, in the main paper and supplementary material. All benchmark datasets used in this work are publicly available. All source code for data preprocessing, training, evaluation, and analysis will be publicly released upon publication.

## REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Conference on Language Modeling*, 2025.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi:10.52202/085713-2032.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi:10.52202/085713-0679.

Xingyu Chen et al. Do NOT think that much for 2+3=? on the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024. doi:10.48550/arXiv.2412.21187.

Zigeng Chen, Xinyin Ma, Gongfan Fang, Ruonan Yu, and Xinchao Wang. VeriThinker: Learning to verify makes reasoning model efficient. *arXiv preprint arXiv:2505.17941*, 2025. doi:10.48550/arXiv.2505.17941.

Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. Optimizing length compression in large reasoning models. *arXiv preprint arXiv:2506.14755*, 2025. doi:10.48550/arXiv.2506.14755.

Karl Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi:10.48550/arXiv.2110.14168.

Muzhi Dai, Shixuan Liu, and Qingyi Si. Stable reinforcement learning for efficient reasoning. *arXiv preprint arXiv:2505.18086*, 2025a. doi:10.48550/arXiv.2505.18086.

Muzhi Dai, Chenxu Yang, and Qingyi Si. S-GRPO: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025b. doi:10.48550/arXiv.2505.07686.

DeepSeek-AI et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. doi:10.48550/arXiv.2606.19348. URL `https://arxiv.org/abs/2606.19348`.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM learns when to think. *arXiv preprint arXiv:2505.13379*, 2025. doi:10.48550/arXiv.2505.13379.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025. doi:10.48550/arXiv.2504.05185.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi:10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855. Association for Computational Linguistics, July 2025. doi:10.18653/v1/2025.findings-acl.1274.

Shibo Hao et al. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024. doi:10.48550/arXiv.2412.06769.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 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)*, pp. 3828–3850. Association for Computational Linguistics, August 2024. doi:10.18653/v1/2024.acl-long.211.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Bairu Hou et al. ThinkPrune: Pruning long chain-of-thought of LLMs via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025. doi:10.48550/arXiv.2504.01296.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-Reasoner-Zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025. doi:10.48550/arXiv.2503.24290.

Cong Jiang, Xiaofeng Zhang, Fangzhi Zhu, Xiaowei Chen, Junxiong Zhu, and Zheng Zhang. Rethinking LLM reasoning: From explicit trajectories to latent representations. In *International Conference on Learning Representations*, 2026.

Kimi Team et al. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025. doi:10.48550/arXiv.2501.12599.

Kimi Team et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. doi:10.48550/arXiv.2607.24653. URL https://arxiv.org/abs/2607.24653.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Xiao Liang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025. doi:10.48550/arXiv.2502.17419. URL https://arxiv.org/abs/2502.17419.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Wei Liu et al. Learn to reason efficiently with adaptive length-based reward shaping. *arXiv preprint arXiv:2505.15612*, 2025a. doi:10.48550/arXiv.2505.15612.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Conference on Language Modeling*, 2025b.

Jinghui Lu et al. Xiaomi OneVL: One-step latent reasoning and planning with vision-language explanation. *arXiv preprint arXiv:2604.18486*, 2026. doi:10.48550/arXiv.2604.18486.

Haotian Luo et al. O1-Pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025. doi:10.48550/arXiv.2501.12570.

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

OpenAI. Learning to reason with LLMs. OpenAI, September 2024. URL https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2026-08-26.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi:10.48550/arXiv.1707.06347.

Zhihong Shao et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi:10.48550/arXiv.2402.03300.

Yi Shen et al. DAST: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025. doi:10.48550/arXiv.2503.04472.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Yang Sui et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025. doi:10.48550/arXiv.2503.16419.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi:10.52202/085713-0164.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. doi:10.52202/068431-1800.

Siye Wu et al. ARM: Adaptive reasoning model. *arXiv preprint arXiv:2505.20258*, 2025a. doi:10.48550/arXiv.2505.20258.

Xingyu Wu et al. LAPO: Internalizing reasoning efficiency via length-adaptive policy optimization. *arXiv preprint arXiv:2507.15758*, 2025b. doi:10.48550/arXiv.2507.15758.

Violet Xiang et al. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *arXiv preprint arXiv:2506.05256*, 2025. doi:10.48550/arXiv.2506.05256.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025. doi:10.48550/arXiv.2502.18600.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models. In *International Conference on Learning Representations*, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jingyang Yi, Jiazheng Wang, and Sida Li. ShorterBetter: Guiding reasoning models to find optimal inference length for efficient reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi:10.52202/085713-1302.

Qiying Yu et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. doi:10.48550/arXiv.2503.14476.

Danlong Yuan et al. Shorten after you're right: Lazy length penalties for reasoning RL. *arXiv preprint arXiv:2505.12284*, 2025. doi:10.48550/arXiv.2505.12284.

Jixiao Zhang and Chunsheng Zuo. GRPO-LEAD: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. *arXiv preprint arXiv:2504.09696*, 2025. doi:10.48550/arXiv.2504.09696.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025. doi:10.48550/arXiv.2507.18071.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J. Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi:10.52202/085713-2670.

# A Related Work

## A.1 Large Reasoning Models

Chain-of-Thought (CoT) prompting (Wei et al., 2022) improves multi-step reasoning by eliciting intermediate inference steps. Self-Consistency (Wang et al., 2023) aggregates answers over parallel reasoning paths, Tree of Thoughts (Yao et al., 2023) searches over backtrackable reasoning trees, and outcome verifiers (Cobbe et al., 2021) and process supervision (Lightman et al., 2024) score complete solutions or intermediate steps. Snell et al. (2025) further establish test-time computation as a scalable dimension of reasoning capability. Recent large reasoning models (LRMs) internalize long-horizon reasoning into the model policy through reinforcement learning post-training: DeepSeekMath (Shao et al., 2024) introduces Group Relative Policy Optimization (GRPO), while OpenAI o1 (OpenAI, 2024), DeepSeek-R1 (Guo et al., 2025), and Kimi k1.5 (Kimi Team et al., 2025) exhibit self-reflection, verification, and strategy adaptation induced by large-scale reinforcement learning. PPO (Schulman et al., 2017) estimates token-level advantages with a learned value function, whereas RLOO (Ahmadian et al., 2024) and GRPO assign all tokens of a response a single advantage computed from a group of responses to the same prompt. DAPO (Yu et al., 2025), Open-Reasoner-Zero (Hu et al., 2025), and Dr. GRPO (Liu et al., 2025b) further improve RLVR through clipping rules, sampling strategies, training scalability, and length-bias correction.

## A.2 Efficient Reasoning Mechanisms

Existing approaches pursue reasoning efficiency along two main lines: latent reasoning and post-training. In latent reasoning, continuous hidden states, dynamically compressed embeddings, and parallel latent variables map multi-step inference onto shorter computational paths in latent space (Hao et al., 2024; Zhu et al., 2025; Tan et al., 2025; Jiang et al., 2026; Lu et al., 2026), compressing both the reasoning representation and the depth of sequential decoding. For explicit reasoning, fixed or difficulty-adaptive token budgets, concise draft-style intermediate steps, and confidence-guided early exit also regulate generation length at inference time (Muennighoff et al., 2025; Han et al., 2025; Xu et al., 2025; Yang et al., 2026). Among them, post-training has become the mainstream approach, as it preserves the expressive capacity of explicit reasoning and achieves higher accuracy than latent reasoning on complex tasks (Sui et al., 2025).

Post-training methods internalize computation-allocation policies into model parameters. One line of work imposes direct length constraints: O1-Pruner (Luo et al., 2025) constructs a length-harmonizing reward from baseline response lengths, L1 (Aggarwal & Welleck, 2025) jointly optimizes answer correctness and adherence to a user-specified token budget, ThinkPrune (Hou et al., 2025) progressively tightens the generation limit, and S-GRPO (Dai et al., 2025b) learns when to terminate through serial early-exit rollouts; Arora & Zanette (2025) and Fatemi et al. (2025) incorporate continuous length regularization directly into outcome rewards, and LC-R1 (Cheng et al., 2025) combines length and compression rewards. Another line dynamically estimates target lengths: DAST (Shen et al., 2025) allocates token budgets according to problem difficulty, LASER (Liu et al., 2025a) uses a step-function reward with a target length, which its variant LASER-D adapts to training progress and problem difficulty, and ShorterBetter (Yi et al., 2025) derives the target length from the shortest correct trajectory sampled for the same prompt; ALP (Xiang et al., 2025), LAPO (Wu et al., 2025b), and Lazy Length Penalty (Yuan et al., 2025) regulate length pressure using within-prompt success rates, the length distribution of successful trajectories, and training stability, respectively, and LEAD (Zhang & Zuo, 2025) combines length regularization with difficulty-aware advantage reweighting. GRPO-$\lambda$ (Dai et al., 2025a) stabilizes training by disabling length penalties when the within-prompt correctness rate is low. Beyond length-based rewards, ARM (Wu et al., 2025a) and Thinkless (Fang et al., 2025) learn to select a reasoning mode for each problem, and VeriThinker (Chen et al., 2025) reduces redundant self-reflection through auxiliary training on solution verification. Length penalties mitigate overthinking by shortening reasoning trajectories, but their sequence-level rewards can also suppress necessary derivation steps and reduce accuracy on complex problems. PACE requires no target length, no difficulty estimate, and no auxiliary supervision: it prices each token's remaining computation at the success rate of the other prompts in the batch, turning efficiency from a length constraint imposed on the policy into a solve-throughput objective the policy optimizes.

## B Proof of Proposition 1

Independence and unbiasedness of the leave-one-prompt-out estimates give

$$\begin{aligned}\mathbb{E}\left[\hat{\mu}_{\ell,-g}\sum_t r_{g,i}\boldsymbol{\psi}_{g,i,t}\right] &= \mathbb{E}[\hat{\mu}_{\ell,-g}]\,\mathbb{E}\left[\sum_t r_{g,i}\boldsymbol{\psi}_{g,i,t}\right] = \mu_\ell \nabla_\theta \mu_r,\\ \mathbb{E}\left[\hat{\mu}_{r,-g}\sum_t \kappa_{g,i,t}\boldsymbol{\psi}_{g,i,t}\right] &= \mathbb{E}[\hat{\mu}_{r,-g}]\,\mathbb{E}\left[\sum_t \kappa_{g,i,t}\boldsymbol{\psi}_{g,i,t}\right] = \mu_r \nabla_\theta \mu_\ell.\end{aligned} \tag{10}$$

Subtracting the second identity from the first gives

$$\mathbb{E}\left[\sum_t \hat{A}^{\text{rate}}_{g,i,t}\boldsymbol{\psi}_{g,i,t}\right] = \mu_\ell \nabla_\theta \mu_r - \mu_r \nabla_\theta \mu_\ell = \mu_\ell^2 \nabla_\theta \eta_B \tag{11}$$

for each response $(g, i)$. Under i.i.d. prompt sampling and conditionally independent rollouts, this expectation is identical across all $(g, i)$. Summing over all $MG$ responses therefore yields the identity in Proposition 1.

## C Additional Analyses

### C.1 Token-Level Credit Along Responses

To show what the rate credit and the correctness budget assign to an individual decision, we trace the advantage $\hat{A}^{\text{PACE}}_{g,i,t}$ along individual responses under illustrative batch statistics in Figure 5.

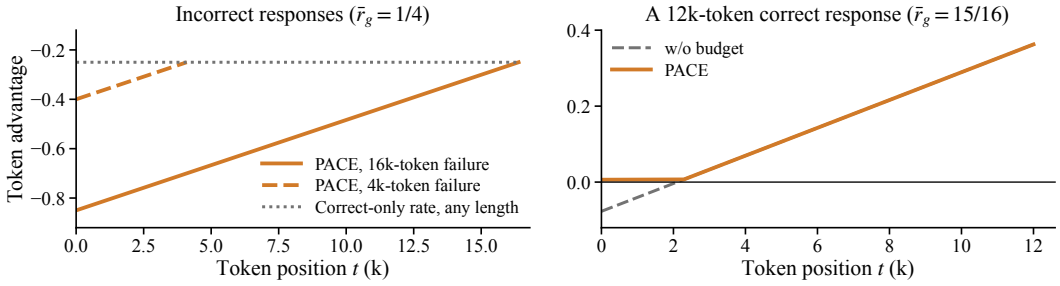


Figure 5: **Token-level PACE advantage along responses.** Illustrative batch statistics $\hat{\mu}_{r,-g} = 0.6$ and $\hat{\mu}_{\ell,-g} = 0.3$, with $\gamma = 0.9$, $G = 16$, and $B = 16{,}384$. Left: incorrect responses to a prompt with $\bar{r}_g = 1/4$. Right: a 12k-token correct response to a prompt with $\bar{r}_g = 15/16$, with and without the correctness budget.

**Each token is charged for the computation it induces.** On incorrect responses, the early decisions of a failure that exhausts the generation budget receive the largest penalty, whereas the correct-only variant of Section 3.3 assigns every failure the same advantage $-\bar{r}_g$ regardless of its length. On a long correct response to a prompt that the policy already solves, the correctness advantage is small: the unbounded rate credit pulls the first 2.3k tokens below $(1-\gamma)\hat{A}^{\text{acc}}_{g,i}$ and turns the first 2.1k negative. The correctness budget raises exactly these tokens and leaves the rest unchanged.

### C.2 Question-Level Accuracy and Length

To check whether the accuracy gains and the length reductions of Tables 1 and 2 occur on the same questions, we pair the two models question by question. For each question $x$, we estimate the accuracy $\hat{p}_x$ and the mean completion length $\bar{T}_x$ of each model from its sampled responses, and define the length ratio $\rho_x = \bar{T}^{\text{PACE}}_x / \bar{T}^{\text{DS}}_x$ and the accuracy difference $\Delta\hat{p}_x = \hat{p}^{\text{PACE}}_x - \hat{p}^{\text{DS}}_x$ over the 2,891 questions of the 10 benchmarks.

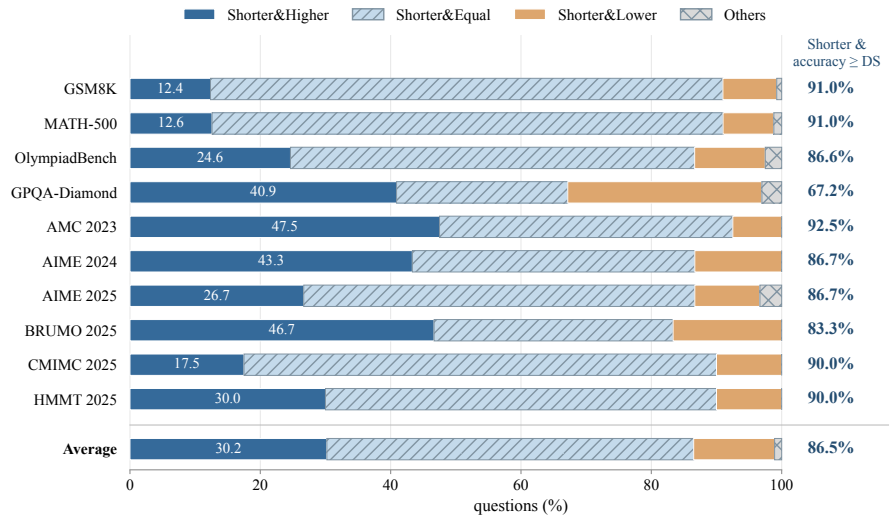


Figure 6: **Accuracy gains and length reductions co-occur on the same questions.** Share of questions by the length ratio $\rho_x$ and the accuracy difference $\Delta\hat{p}_x$; *Others* collects the questions with $\rho_x \geq 1$. The right column gives the share with $\rho_x < 1$ and $\Delta\hat{p}_x \geq 0$.

Table 4: **Question-level statistics of PACE-1.5B relative to DS-1.5B.** Shorter denotes $\rho_x < 1$, and the median length reduction is taken over the questions that are both more accurate and shorter. Brackets give 95% confidence intervals from paired question bootstrap within benchmarks.

| | **Questions with $\Delta\hat{p}_x > 0$** | | | **All questions** | |
|---|---|---|---|---|---|
| **Benchmark** | Count | Shorter (%) | Median length reduction (%) | Shorter with $\Delta\hat{p}_x \geq 0$ (%) | Accuracy difference (pp) |
| GSM8K | 164 | 99.4 | 72.1 | 91.0 [89.4, 92.5] | +0.99 [+0.04, +1.93] |
| MATH-500 | 64 | 98.4 | 68.3 | 91.0 [88.4, 93.4] | +1.70 [+0.25, +3.15] |
| OlympiadBench | 169 | 98.2 | 70.1 | 86.6 [84.1, 89.2] | +6.08 [+4.34, +7.86] |
| GPQA-Diamond | 82 | 98.8 | 62.0 | 67.2 [60.6, 73.7] | +4.23 [+1.07, +7.39] |
| AMC 2023 | 19 | 100.0 | 70.0 | 92.5 [82.5, 100.0] | +9.38 [+4.69, +14.69] |
| AIME 2024 | 13 | 100.0 | 64.0 | 86.7 [73.3, 96.7] | +8.33 [+2.71, +14.58] |
| AIME 2025 | 8 | 100.0 | 56.0 | 86.7 [73.3, 96.7] | +3.33 [−1.25, +8.33] |
| BRUMO 2025 | 14 | 100.0 | 61.9 | 83.3 [70.0, 96.7] | +7.71 [+2.29, +13.54] |
| CMIMC 2025 | 7 | 100.0 | 63.1 | 90.0 [80.0, 97.5] | +4.38 [+0.31, +9.22] |
| HMMT 2025 | 9 | 100.0 | 65.7 | 90.0 [76.7, 100.0] | +3.54 [+0.42, +7.29] |
| Average | – | 99.5 | 65.3 | 86.5 [83.5, 89.2] | +4.97 [+3.71, +6.26] |

**Accuracy gains cost no additional computation.** PACE-1.5B is shorter on 98.9% of all 2,891 questions, and on every question of AMC 2023, AIME 2024, BRUMO 2025, CMIMC 2025, and HMMT 2025 (Figure 6). Of the 549 questions on which it is more accurate, 543 are also shorter, again 98.9%, and their median length reduction ranges from 56.0% on AIME 2025 to 72.1% on GSM8K (Table 4). The length reductions of Tables 1 and 2 thus hold question by question, and the questions PACE solves more often are exactly those it solves with fewer tokens.

**Shorter without losing accuracy on 86.5% of questions.** Averaged over benchmarks, 86.5% of questions (95% CI [83.5%, 89.2%]) satisfy $\rho_x < 1$ and $\Delta\hat{p}_x \geq 0$, and 30.2% improve strictly in both. Ties dominate where both models already answer most questions correctly, covering 78.6% of GSM8K and 78.4% of MATH-500. GPQA-Diamond has the fewest ties (26.3%), so its shortened questions split into higher (40.9%) and lower (29.8%) accuracy, with a net gain of 4.23 points.

Accuracy improves on all 10 benchmarks, and its 95% confidence interval excludes zero on nine of them.

### C.3 Generation After the First Correct Boxed Answer

Early-exit methods target the tokens generated after a correct answer has appeared (Dai et al., 2025b; Yang et al., 2026). We therefore measure how much of the saving of PACE comes from this segment: in each response we locate the first correct boxed answer and count the tokens that follow it, separately for the remaining thinking and the output section.

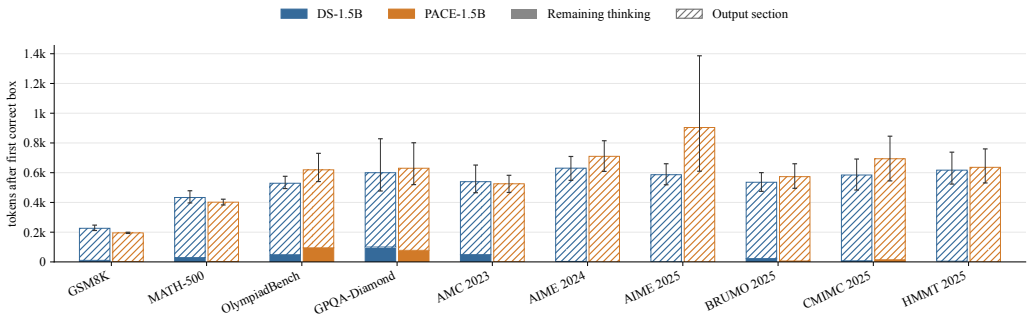


Figure 7: Mean number of tokens generated after the first verifier-accepted boxed answer, with DS-1.5B (left) and PACE-1.5B (right) in each pair. Solid segments denote remaining thinking and hatched segments the output section. Means are taken over the 8,783 DS-1.5B and 9,419 PACE-1.5B responses, out of 14,756 per model on 2,891 questions, that contain at least one correct boxed answer. Error bars show 95% confidence intervals from 10,000 paired-question bootstrap resamples within benchmarks.

**The saving lies before the answer, not after it.** Both models close their thinking almost immediately once a correct answer appears: the generation that follows it stays below 1k tokens on every benchmark, and remaining thinking accounts for less than one fifth of it (Figure 7). PACE shortens this remaining thinking on 8 benchmarks, from 29.0 to 2.6 tokens on MATH-500 and from 49.0 to 1.4 tokens on AMC 2023, while the output section that presents the final solution grows on 8 benchmarks, by amounts that lie within their bootstrap intervals. The direction is what matters: PACE removes post-answer thinking and leaves the solution it presents intact. Removing the entire post-answer segment would shorten a DS-1.5B response by less than 0.7k tokens, whereas PACE shortens the mean response by 1.7k tokens on GSM8K and 12.2k tokens on HMMT 2025. The saving therefore comes from the reasoning that leads to an answer and from the failures that never reach one, which is where the remaining-compute term concentrates its pressure (Figure 5).

## D IMPLEMENTATION DETAILS

Table 5 lists the training hyperparameters of PACE. The advantages are used without further normalization. The policy loss applies the clipped GSPO-token surrogate (Zheng et al., 2025) to these token-level advantages, adds a per-token $k_3$ estimate of the KL divergence to the reference policy, and divides the sum of token losses by the number of responses times $B$ (Liu et al., 2025b).

Table 5: **Training hyperparameters of PACE.**

| **Hyperparameter** | **Value** |
|---|---|
| Base models | DeepSeek-R1-Distill-Qwen-1.5B and 7B |
| Training data | DAPO-Math-17k |
| Prompts per rollout batch $M$ | 64 |
| Responses per prompt $G$ | 16 |
| Generation budget $B$ | 16,384 tokens |
| Rollout temperature | 1.0 |
| Correctness budget $\gamma$ | 0.9 |
| Importance ratio | GSPO-token |
| Clipping range $(\epsilon_{\text{low}}, \epsilon_{\text{high}})$ | (0.2, 0.28) |
| KL penalty | per-token $k_3$ estimate in the loss, coefficient 0.01 |
| Loss normalization | summed token losses divided by (number of responses $\times B$) |
| Optimizer | AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.1 |
| Learning rate | $10^{-5}$, constant schedule, no warmup |
| LoRA | rank 256, $\alpha = 512$, dropout 0, attention and feed-forward projections |
| Updates per rollout batch | 8 (128 responses per update) |
| Optimizer steps | 1,000 |