Shorten After You're Right: Lazy Length Penalties for Reasoning RL

Anonymous ACL submission

Abstract

Long-reasoning models achieve strong accuracy on complex reasoning tasks, but their extended reasoning trajectories incur substantial memory and latency costs. Several existing shortening methods rely on additional supervision or multi-stage post-training, which primarily reduces inference length and does not reduce the rollout tokens during on-policy reinforcement learning (RL). We instead target *on-policy* response shortening, aiming to improve both inference efficiency and RL training throughput. However, because on-policy RL couples optimization with exploration, naively penalizing length can destabilize training and suppress exploration. To impose length pressure safely, we propose a *lazy length penalty* integrated into the rule-based RL pipeline: it activates only on correct trajectories, only after training accuracy enters a stably improving regime, and only when responses exceed a *tolerance band* beyond the minimal correct length. Across four settings, our method significantly reduces response length without extra training stages while maintaining or improving performance. In a logic reasoning setting, we achieve a 40% reduction in step-averaged response length alongside a 14-point gain in performance. For math problems, we reduce step-averaged response length by 33% while preserving performance.

1 Introduction

Long reasoning models (LRMs) trained with large-scale, rule-based on-policy reinforcement learning (RL) have achieved strong performance on complex reasoning tasks, often exhibiting self-reflection and self-correction (Chen et al., 2025a; DeepSeek-AI et al., 2025; OpenAI et al., 2024). A recurring empirical trend in these pipelines is that reasoning trajectories grow longer as training progresses, which can correlate with improved accuracy (DeepSeek-AI et al., 2025). However, this length growth is costly: longer outputs increase inference latency and KV-cache memory, and longer rollouts directly reduce RL training throughput, sometimes making large-scale on-policy RL impractical.

Importantly, longer reasoning is not always better. Models can overthink, repeat steps, or drift into redundant verification, and unnecessary length may even hurt performance (Chen et al., 2025b; Team et al., 2025; Yang et al., 2025; Sui et al., 2025). These observations motivate methods that shorten reasoning trajectories while preserving correctness.

Most existing shortening approaches rely on extra supervision, distillation, or off-policy/post-training stages (Xia et al., 2025; Kang et al., 2024; Ma et al., 2025b; Munkhbat et al., 2025; Yu et al., 2024; Liu et al., 2024; Cui et al., 2025; Luo et al., 2025a; Shen et al., 2025). Such methods can reduce inference length, but they do not reduce the rollout tokens already spent during the main on-policy RL stage. A natural alternative is to directly penalize length in the on-policy reward (e.g., Kimi (Team et al., 2025)). Yet in our reproduction, applying length shaping early causes a failure mode: trajectories collapse to overly short outputs, exploration is suppressed, and training becomes unstable, leading to suboptimal performance. Similar degradation is reported elsewhere (Arora and Zanette, 2025; Hou et al., 2025). This suggests that on-policy length control must respect the coupling between exploration and optimization.

We take the view that **length is an auxiliary property of a reasoning trajectory**: success is defined by correctness, while brevity is a preference *among successful trajectories*. Hence, length regularization in on-policy RL should be *lazy*: apply it **where** it is safe (only on correct trajectories), **when** learning is stable, and **what** is truly redundant (only excess length beyond a tolerance band).

Guided by this principle, we propose Short-RL, a lazy length penalty integrated into rule-based on-policy RL. It consists of three gates: RIGHT-

GATE applies shaping only to correct trajectories, SLACKBAND penalizes only excess length beyond a tolerated minimum, and STABLESWITCH activates shaping only once batch accuracy is sufficiently stable. Across four settings, Short-RL significantly shortens trajectories without extra training stages while maintaining or improving performance. On Logic-RL, we reduce step-averaged response length by 40% while improving performance by 14 points; on three math RL pipelines, we reduce step-averaged length by up to 33% while preserving performance.

2 Related Work

We build on (i) long-reasoning models trained with large-scale, rule-based on-policy RL, and (ii) methods for shortening long-form reasoning. Our focus is the intersection: *shortening trajectories during on-policy RL* without extra post-training stages.

2.1 Rule-Based RL for Long Reasoning

Recent LRMs improve performance by generating explicit multi-step trajectories and are often trained with large-scale, rule-based on-policy RL, which can enhance reasoning without human preference labels (DeepSeek-AI et al., 2025; OpenAI et al., 2024). A consistent empirical observation is that response length tends to increase during training, raising inference latency and KV-cache usage, and more critically, increasing rollout token cost during RL (DeepSeek-AI et al., 2025). This trend appears across domains, including logic reasoning (e.g., Logic-RL) (Xie et al., 2025, 2024) and mathematical reasoning pipelines (Zeng et al., 2025; Luo et al., 2025b; Hu et al., 2025; Yu et al., 2025), motivating efficiency improvements that operate *within* the on-policy training loop.

2.2 Shortening Long-Form Reasoning

A broad literature seeks to reduce the cost of long-form reasoning by shortening trajectories or reducing generated tokens.

Supervised and off-policy shortening. Many methods rely on supervised fine-tuning or distillation to compress chains-of-thought or skip steps (Xia et al., 2025; Kang et al., 2024; Ma et al., 2025b; Munkhbat et al., 2025; Yu et al., 2024; Liu et al., 2024; Cui et al., 2025), or use off-policy RL / extra post-training stages to prune and revise traces (Luo et al., 2025a; Shen et al., 2025). While effective, these approaches typically require additional data or training phases and are not directly aligned with the *in-process* on-policy RL pipelines used to train LRMs.

Inference-time control. Prompting and decoding-time strategies (e.g., concise prompting, token budgets, early stopping) reduce inference cost by limiting or guiding generation (Han et al., 2025; Renze and Guven, 2024; Xu et al., 2025; Ma et al., 2025a). Related work on routing and dynamic computation allocates reasoning effort based on difficulty or intent (Anthropic, 2025; Aytes et al., 2025; Chuang et al., 2025; Ong et al., 2025; Pu et al., 2025; Aggarwal and Welleck, 2025; She et al., 2025; Wu et al., 2025). These methods primarily affect inference and do not reduce the rollout tokens already consumed during RL training.

On-policy length rewards. Closest to our setting are length-based rewards integrated into RL. Kimi (Team et al., 2025) proposes a direct length reward but applies it in a *post-RL* stage, noting that using it early can harm training. Efficient (Arora and Zanette, 2025) scales rewards based on response length and observes accuracy–length tradeoffs, while ThinkPrune (Hou et al., 2025) penalizes correct responses beyond a limit and similarly highlights the brevity–accuracy tension.

Our position. In contrast to post-RL or inference-only approaches, we target *training-time* efficiency in on-policy RL. We treat length as an auxiliary trajectory property and introduce a *lazy* length penalty that (i) applies only on correct trajectories, (ii) penalizes only excess length beyond a tolerance band, and (iii) activates only after training becomes stable. This design aims to shorten rollouts *during* RL training while avoiding the exploration and stability failures of naive always-on length shaping.

3 Methodology

3.1 Notation and Setup

Let x be the prompt and y^* the reference answer. Each rollout produces a full output $\hat{u}_i$, parsed into a reasoning trace z_i and final answer y_i. We define correctness $c_i := \mathbb{I}[y_i = y^*]$ and length $l_i := l(\hat{u}_i)$ (token count). Let acc be the batch correctness rate over rollout samples and $\text{acc}_{\max}$ its running maximum.

3.2 Background: Length-Aware Rewards in Rule-Based On-Policy RL

A standard way to encourage shorter trajectories is to add a length term to the original task reward:

$$R(x, \hat{u}) = R_{\text{task}}(x, \hat{u}) + \alpha \cdot R_{\text{len}}(x, \hat{u}), \quad (1)$$

where R_{task} is the original rule-based reward (e.g., correctness/format), R_{len} is a shaping term based on output length, and α controls its strength. Figure 1 illustrates representative designs (Kimi, Efficient, ThinkPrune).

Among them, Kimi (Team et al., 2025) is a common choice and serves as our case study. For each prompt x, sample k outputs and compute $l_{\min} = \min_i l_i$, $l_{\max} = \max_i l_i$. If $l_{\max} = l_{\min}$, set the length reward to 0. Otherwise,

$$R_{\text{len}}(i) = \begin{cases} \lambda_i, & c_i = 1, \\ \min(0, \lambda_i), & c_i = 0, \end{cases} \quad (2)$$

where $\lambda_i = 0.5 - \frac{l_i - l_{\min}}{l_{\max} - l_{\min}}$.

3.3 Why Naive On-Policy Length Shaping Fails

Length is an *auxiliary* trajectory attribute: correctness defines success, while shorter outputs are preferred *among correct trajectories*. In on-policy RL, optimization and data collection are coupled, so always-on length shaping can reshape the sampled trajectory distribution and interfere with exploration.

Empirical early collapse. Using Kimi-style shaping from the start of Logic-RL training (Xie et al., 2025) causes reward-hacking: trajectories rapidly collapse to very short outputs and accuracy becomes unstable or degrades (Figure 2).

Two conflicts. (i) **Exploration suppression:** Eq. 2 computes $l_{\min}, l_{\max}$ over *all* samples and can further penalize incorrect rollouts, biasing sampling toward short (often uninformative) trajectories and reducing reasoning-path diversity (Figure 3). (ii) **Training instability:** applying length pressure throughout training can compete with competence acquisition, since longer traces may be necessary to discover correct strategies at certain stages.

These observations motivate a simple principle: *auxiliary-trajectory regularization should be lazy*—it should specify **where** it applies (only correct trajectories), **what** it penalizes (only excess length), and **when** it activates (only after learning is stable).

3.4 Short-RL: Lazy Length Penalties

We implement this principle via three gates:

- **RIGHTGATE (Where).** Apply length shaping only to correct trajectories. For each prompt x, compute $l_{\min}$ and $l_{\max}$ using only correct samples $\mathcal{C}(x) = \{j \mid c_j = 1\}$, and set the length term to 0 for incorrect samples.
- **SLACKBAND (What).** Penalize only *excess* length beyond a tolerance τ_l. If $l_i \le l_{\min} + \tau_l$, we apply a constant baseline (no preference among these correct trajectories); only samples exceeding the band receive decreasing reward.
- **STABLESWITCH (When).** Activate length shaping only when training is stable: enable the length term only if acc $\ge$ acc$_{\max} - \tau_{\text{acc}}$.

3.5 Unified Reward

Putting the gates together, the length shaping term for sample i is

$$R_{\text{len}}(i) = \begin{cases} \beta_i, & \text{if } c_i = 1,\ \text{acc} \ge \text{acc}_{\max} - \tau_{\text{acc}}, \\ 0, & \text{otherwise}, \end{cases}$$
$$\beta_i = \begin{cases} \lambda_i, & \text{if } l_i > l_{\min} + \tau_l, \\ 0.5, & \text{otherwise}, \end{cases}$$
$$\lambda_i = 0.5 - \frac{l_i - l_{\min}}{l_{\max} - l_{\min}}, \quad (3)$$

with $\mathcal{C}(x) = \{j \mid c_j = 1\}$ and

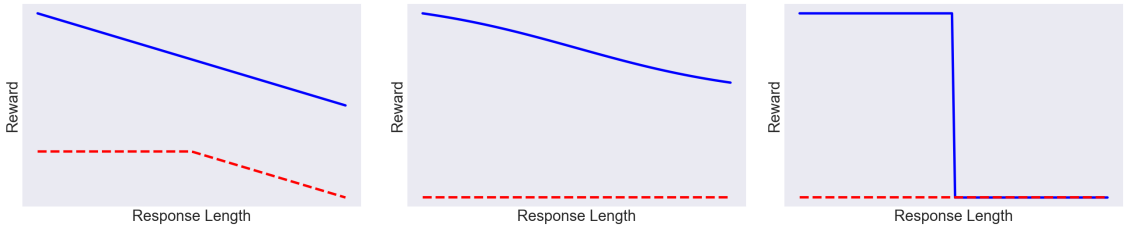
$$l_{\min} = \min_{j \in \mathcal{C}(x)} l_j, \qquad l_{\max} = \max_{j \in \mathcal{C}(x)} l_j.$$

If $|\mathcal{C}(x)| = 0$, we set $R_{\text{len}}(i) = 0$ for all i. If $l_{\max} = l_{\min}$, the "excess-length" condition is never satisfied, so λ_i is unused and correct samples default to the constant baseline.

Finally, we plug R_{len} into Eq. 1. The tolerance τ_l controls how much length is ignored as acceptable slack, while τ_{acc} controls how early and how frequently length shaping activates.

4 Experiments

In this section, we evaluate whether Short-RL can *shorten reasoning trajectories during on-policy RL* while maintaining (or improving) task performance. Since the dominant cost of on-policy RL



(a) Kimi (b) Efficient (c) ThinkPrune

Figure 1: Reward values as a function of response length. Blue lines indicate rewards for correct responses and red lines represent rewards for incorrect responses.

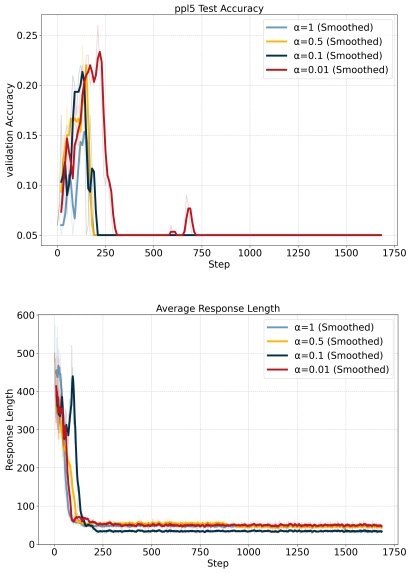


Figure 2: Test accuracy (left) and average response length (right) across different α.

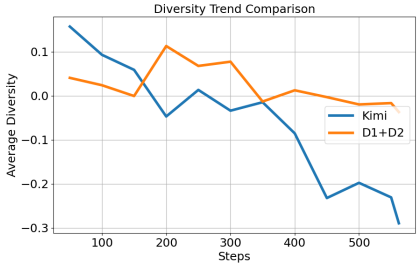


Figure 3: Diversity metric on ppl5 during training.

scales with rollout tokens, we emphasize both *accuracy* and *efficiency*. We report two complementary length metrics: **Training (step-avg)** as a proxy for training-time token cost of a response, and **Inference (final)** as a proxy for test-time decoding cost for one question.

4.1 Experimental Settings

We first describe the evaluation domains, training setups, metrics, and baselines used throughout the experiments.

Evaluation domains and training setups. We evaluate on two domains: logic reasoning and mathematical reasoning. The logic domain follows Logic-RL (Xie et al., 2025). The math domain includes three representative on-policy RL pipelines: DeepScaleR (Luo et al., 2025b), SimpleRL-Reason (Zeng et al., 2025), and Open-Reasoner-Zero (Hu et al., 2025). Across all experiments, we use the same model architecture and training framework as the corresponding original projects in the VeRL framework (Sheng et al., 2024). For the three math settings, we adopt a DeepSeek-R1-style prompt template and include the format reward used by the original pipelines. Full prompts and hyperparameters are in Appendix Training Details.

Training-cost and inference-cost metrics. On-policy RL cost scales with the number of generated tokens in rollouts and overlong responses require significant memory cost and computation time. To make efficiency explicit, we report two length metrics: **(i) Training (step-avg)**: response length averaged over training steps (proxy for training-time token cost and throughput); **(ii) Inference (final)**: response length measured at the final checkpoint (proxy for test-time decoding cost). Unless stated otherwise, accuracy is always evaluated at the final checkpoint.

4.1.1 Logic Reasoning

We begin with logic reasoning, where long rollouts naturally emerge and the training-time efficiency benefit of on-policy shortening is most visible. We use the same datasets and evaluation protocol as Logic-RL (Xie et al., 2025), initializing from Qwen2.5-7B (Qwen et al., 2025). For Short-RL, we set $\tau_l = 200$, $\tau_{\text{acc}} = 0.05$, and $\alpha = 1$

(details in Appendix Training Details). We report in-domain accuracy on 2- to 8-person tasks, and out-of-domain generalization on AMC and AIME following the Logic-RL protocol. We also report Training (step-avg) and Inference (final) lengths.

4.1.2 Math Reasoning

We next evaluate on mathematical reasoning, which provides three different RL pipelines and tests whether the same lazy length penalty generalizes across implementations. We evaluate on DeepScaleR (Luo et al., 2025b), Open-Reasoner-Zero (Hu et al., 2025), and SimpleRL-Reason (Zeng et al., 2025), reusing nearly all hyperparameters from each original implementation. Short-RL hyperparameters for each setting are listed in Appendix Table 3. We evaluate on five benchmarks: AIME2024 (invitational mathematics examination, 2024), AMC23 (AI-MO, 2025), MATH-500 (Lightman et al., 2023), Minerva Math (Hendrycks et al., 2021), and Olympiad Bench (He et al., 2024). We again report Training (step-avg) and Inference (final) lengths.

4.1.3 Baselines

Finally, we describe the baselines used to isolate the effects of on-policy length shaping and to compare against prior length-control rewards. We compare against:

- **Standard**: on-policy RL with the original task reward R_{task} (no length shaping).
- **Kimi (post)**: the two-stage procedure used by Kimi (Team et al., 2025): first run Standard RL, then apply the length reward in a later stage. **Importantly**, this post-RL setup can shorten *inference* trajectories, but it does *not* reduce the rollout tokens already consumed in stage-1 RL. Therefore, in our tables we report the **Training (step-avg)** length of the first (Standard) stage for Kimi (post), as this reflects the actual training-time cost.
- **Efficient**: length-aware scaling reward from (Arora and Zanette, 2025). We tune the scaling coefficient over $\{0.02, 0.05, 0.08, 0.10\}$ per setting: Logic-RL (0.05), DeepScaleR (0.10), and Open-Reasoner-Zero / SimpleRL-Reason (0.02). Note this coefficient differs from our α.
- **ThinkPrune**: cosine length reward from (Hou et al., 2025). We select a length limit that yields a comparable **Inference (final)** length to Short-RL: 1700 (Logic-RL), 2500 (DeepScaleR), and 1500 (Open-Reasoner-Zero / SimpleRL-Reason).

Method	In Domain								Out of Domain		Avg. Response Length	
	ppl2	ppl3	ppl4	ppl5	ppl6	ppl7	ppl8	Avg	AMC	AIME	Training (step-avg)	Inference (final)
Standard	82	87	88	81	76	69	70	79	39.76	7.77	1477	2632
Kimi (post)	84	88	89	84	79	74	76	82	39.89	8.13	1477	763
Efficient	76	81	79	77	62	48	51	68	37.35	7.77	772	843
ThinkPrune	80	84	86	82	70	66	64	76	38.47	7.35	832	793
Short-RL	**97**	**97**	**99**	**95**	**92**	**83**	**87**	**93**	**42.17**	**8.74**	889	535

Table 1: Logic-RL evaluation at the final checkpoint. "Training (step-avg)" is the response length averaged over training steps (proxy for training-time token cost); "Inference (final)" is the response length at the final checkpoint (proxy for test-time decoding cost).

4.2 Main Results

We now present the main quantitative results. We start with logic reasoning, highlighting both accuracy and training-time token savings, and then move to math reasoning to evaluate generality across three pipelines.

4.2.1 Logic Reasoning: Accuracy Improves While Cost Drops

Table 1 shows that Short-RL consistently improves accuracy while significantly reducing training-time rollout tokens. Compared with Standard RL, Short-RL reduces **Training (step-avg)** length by 40% (1477→889) while improving average in-domain accuracy by 14 points (79→93). Inference efficiency also improves substantially (2632→535).

A key distinction from post-RL length control is training efficiency: Kimi (post) reduces **Inference (final)** length, but it inherits the **Training (step-avg)** length of the initial Standard RL stage, and therefore does not reduce training-time rollout cost. In contrast, Short-RL applies a lazy length penalty *on-policy* after the policy reaches stable competence, shortening trajectories during RL itself and directly reducing training cost—analogous in spirit to trajectory truncation in standard RL, but achieved through a learned, stability-aware mechanism rather than a hard cap.

In addition, Figure 4 shows that our penalty is activated *on-policy* during RL: once accuracy becomes stable, the length control rate rises and the average rollout length decreases, explaining the reduction in Training (step-avg) tokens without relying on a separate post-RL stage.

Method	Math Benchmarks						Avg. Response Length	
	AIME2024	AMC23	MATH500	Minerva Math	Olympiad Bench	Avg	Training (step-avg)	Inference (final)
DeepScaleR								
Standard	26.67	59.04	**81.40**	**26.10**	42.65	47.17	2523	3072
Kimi (post)	23.33	**61.45**	81.00	25.37	**42.79**	46.79	2523	1678
Efficient	20.00	49.40	57.8	16.54	33.73	35.49	1517	1537
ThinkPrune	26.67	56.63	78.40	25.74	41.31	45.75	1589	1621
Short-RL	**30.00**	**60.24**	80.60	**26.47**	**42.65**	**47.99**	1692	1700
Open-Reasoner-Zero								
Standard	16.67	**50.60**	**78.80**	30.88	38.04	43.00	746	840
Kimi (post)	**20.00**	49.40	77.40	**31.25**	**38.63**	**43.34**	746	621
Efficient	13.33	46.99	66.40	26.47	35.96	37.83	578	655
ThinkPrune	13.33	48.19	76.80	27.57	37.15	40.61	677	682
Short-RL	16.67	**50.60**	78.60	30.52	38.19	42.92	660	670
SimpleRL-Reason								
Standard	13.33	48.19	77.00	**32.72**	**39.97**	42.24	703	791
Kimi (post)	16.67	48.19	77.40	31.99	39.67	42.78	703	601
Efficient	6.67	38.55	64.8	22.06	28.68	32.15	492	532
ThinkPrune	10.00	46.99	69.40	31.62	37.30	39.06	613	598
Short-RL	**20.00**	**49.40**	**78.20**	**32.72**	39.23	**43.91**	554	620

Table 2: Math reasoning evaluation. "Training (step-avg)" proxies training-time rollout token cost; "Inference (final)" is the final-checkpoint response length.

4.2.2 Math Reasoning: Cost Savings Without Sacrificing Accuracy

We next turn to math reasoning to test robustness across three different RL pipelines and evaluation suites. Table 2 shows that Short-RL yields consistent reductions in **Training (step-avg)** length while preserving (and occasionally improving) average accuracy. Relative to Standard RL, Short-RL reduces Training (step-avg) length by 33%, 11%, and 21% on DeepScaleR, Open-Reasoner-Zero, and SimpleRL-Reason, respectively. Efficient and ThinkPrune typically shorten outputs more aggressively but exhibit clearer accuracy–length tradeoffs. Similar to logic reasoning, Kimi (post) can reduce **Inference (final)** length but does not reduce training-time rollout cost because its first-stage RL is unchanged.

We further inspect training dynamics in Figure 4: after competence stabilizes, the length control rate increases and the rollout length decreases, confirming that the savings in Training (step-avg) length come from *on-policy* shortening rather than post-hoc compression.

4.3 Training Dynamics: When Does the Lazy Length Penalty Activate?

The main results above emphasize that Short-RL reduces **Training (step-avg)** length—and therefore training-time rollout token cost—which is fundamentally different from post-RL shortening (e.g., Kimi (post)) that cannot reduce tokens already spent in the initial RL stage. We now make this difference explicit by analyzing *when* the lazy length penalty is activated during training. This analysis directly reflects the behavior of StableSwitch (stability-gated activation) and SlackBand (penalize only excess length): early training should prioritize exploration and competence acquisition, while shortening should take effect only after the policy becomes reliably correct.

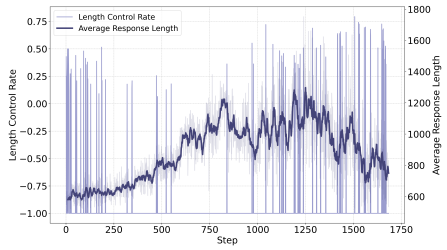
Length control rate. To quantify how frequently length shaping is applied, we introduce a batch-wise metric called *length control rate* γ_l. For each batch, let N be the number of *correct* responses (i.e., number of samples with $c_i = 1$). Among them, let R be the number of correct responses that receive a strict length penalty, i.e., those with $R_{\text{len}}(i) < 0.5$. Since $R_{\text{len}}(i) < 0.5$ occurs exactly when a correct trajectory exceeds the tolerance band (SlackBand), γ_l measures the fraction of correct trajectories that are actively shortened. We define:

$$\gamma_l = \begin{cases} -1, & \text{if acc} < \text{acc}_{\max} - \tau_{\text{acc}}, \\ 0, & \text{if } N = 0, \\ \frac{R}{N}, & \text{otherwise.} \end{cases} \tag{4}$$

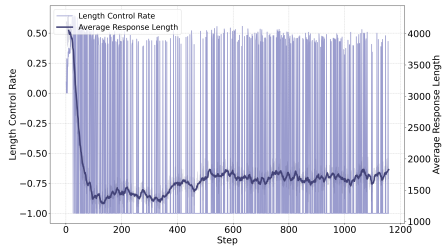
Here, $\gamma_l = -1$ indicates that StableSwitch disables the length penalty due to unstable accuracy; $\gamma_l \in [0, 1]$ indicates that the penalty is enabled and quantifies its active rate on correct rollouts.

From stable accuracy to shorter rollouts. Figure 4 tracks γ_l and the average response length during training. We observe a consistent pattern: in early training, γ_l is frequently -1, meaning that the length penalty is intentionally *inactive* while the model is still learning to solve the task. As training progresses and batch accuracy stabilizes, StableSwitch enables length shaping ($\gamma_l \geq 0$), and a non-trivial fraction of correct trajectories are penalized for exceeding the tolerance band ($\gamma_l > 0$). This coincides with a clear reduction in average rollout length, explaining why Short-RL reduces **Training (step-avg)** length and thus training cost.

Why on-policy matters. This behavior contrasts with post-RL shortening: because Short-RL applies the penalty *during* the main RL stage (but only after the policy is "right" and stable), it reduces rollout tokens for subsequent updates, aligning with the practical role of truncating overlong trajectories in standard RL. In DeepScaleR, we observe a higher length control rate, indicating that more correct trajectories exceed the tolerance band once competence stabilizes, which leads to stronger training-time savings. Curves for SimpleRL-Reason and Open-Reasoner-Zero are provided in Appendix Track the Length Reward.



(a) Logic-RL



(b) DeepScaleR

Figure 4: Tracking the lazy length penalty during training. We plot the length control rate γ_l and the average response length over training steps. $\gamma_l = -1$ indicates that StableSwitch disables length shaping; $\gamma_l \in [0, 1]$ measures the fraction of correct trajectories that are penalized for exceeding the tolerance band.

5 Ablation Study

This section validates two questions: **(i)** which components are necessary for safe on-policy shortening, and **(ii)** how sensitive Short-RL is to the two key tolerances that control SlackBand and StableSwitch. Unless stated otherwise, all ablations are conducted on the Logic-RL training setup and evaluated on the `ppl5` validation protocol used by Logic-RL. We fix the length-reward weight to $\alpha = 1$ throughout to isolate the effects of the gating mechanisms.

5.1 Component Ablation: Which Gates Matter?

Recall that Short-RL implements a *lazy length penalty* via three gates: RightGate (correctness gating), SlackBand (tolerance band), and StableSwitch (stability-triggered activation). We ablate these components by progressively adding them on top of Standard RL:

- **D1 (RightGate)**: apply length shaping only on correct trajectories.
- **D1+D2 (RightGate+SlackBand)**: additionally penalize only excess length beyond the tolerance band.
- **D1+D3 (RightGate+StableSwitch)**: additionally activate length shaping only when accuracy is stable.
- **Short-RL**: full method with all three gates.

For reference, we also include **Standard** (no length shaping) and **Kimi** (direct length reward without lazy gating).

Figure 5 illustrates the training curves for response length and validation accuracy. The results align with our design principle that auxiliary-trajectory regularization should be lazy. **First,** Standard RL produces increasingly long trajectories (Figure 5a), reflecting the well-known tendency of LRMs to lengthen reasoning during on-policy training. **Second,** the direct Kimi reward collapses trajectories to very short outputs early, a signature of reward hacking that harms exploration and yields unstable accuracy (Figure 5b).

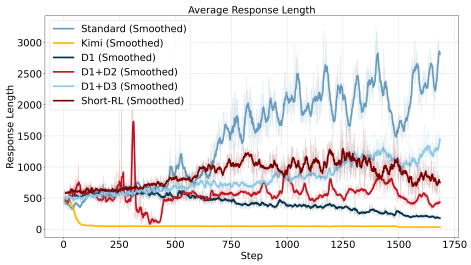
In contrast, the proposed gates yield progressively safer shortening. **D1** already prevents penalizing incorrect exploratory rollouts and therefore avoids the most aggressive collapse. Adding **SlackBand** (**D1+D2**) further stabilizes behavior by avoiding over-optimization once the model reaches an acceptable concise region. Adding **StableSwitch** (**D1+D3**) improves robustness by turning off length pressure when accuracy dips, preventing length optimization from competing with competence acquisition. Finally, **Short-RL** combines both safeguards and achieves the best overall trade-off: it shortens trajectories without collapse and attains the highest validation accuracy.

Remark on evaluation protocol. Following Logic-RL practice, the `ppl5` set here is used as a validation set for tracking curves and may differ from the final evaluation split used in Table 1. This does not affect the qualitative conclusions of the ablation.

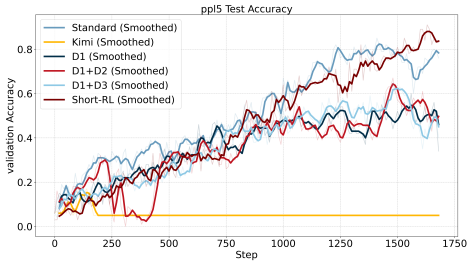
5.2 Sensitivity: Length Tolerance vs. Accuracy Tolerance

We next study sensitivity to the two hyperparameters that implement the "What" and "When" aspects of laziness: the length tolerance τ_l (for SlackBand) and the accuracy tolerance τ_{acc} (for StableSwitch). Figure 6 summarizes the resulting average accuracy and step-averaged response length (both averaged over the ppl tasks).

Varying the length tolerance τ_l (SlackBand). We vary $\tau_l \in \{0, 100, 200, 300\}$ while fixing

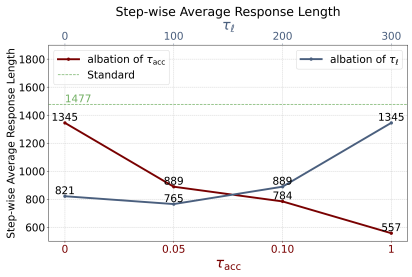


(a) Response Length

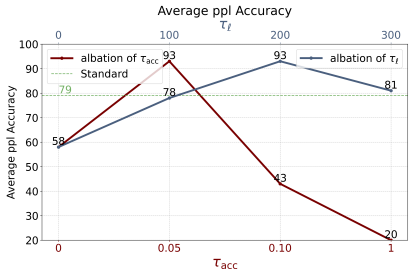


(b) Accuracy

Figure 5: Component ablation on Logic-RL. Standard RL yields long trajectories; direct Kimi shaping collapses length early. Adding lazy gates progressively stabilizes shortening, and Short-RL achieves the best accuracy–length trade-off.



(a)



(b)

Figure 6: Sensitivity to length tolerance τ_l (upper x-axis) and accuracy tolerance τ_{acc} (lower x-axis). Smaller τ_l or larger τ_{acc} increases shortening pressure, which can harm accuracy if overly aggressive.

$\tau_{\text{acc}} = 0.05$. When $\tau_l = 0$, the method degenerates toward a strict linear preference for the shortest correct trajectory, which yields the shortest responses but degrades performance—consistent with our claim that overly aggressive length pressure can suppress useful reasoning steps. For $\tau_l \in \{100, 200, 300\}$, performance remains stable while response length increases moderately with τ_l. Overall, the method is not overly sensitive to τ_l, and $\tau_l \approx 200$ provides a strong balance in this setting.

Varying the accuracy tolerance τ_{acc} (StableSwitch). We vary $\tau_{\text{acc}} \in \{0, 0.05, 0.10, 1.0\}$ while fixing $\tau_l = 200$. This parameter directly controls how early and how frequently length shaping is activated. Larger τ_{acc} makes activation denser (closer to always-on), which yields shorter trajectories but can degrade accuracy by reintroducing the early-training conflict between length optimization and competence acquisition. In contrast, $\tau_{\text{acc}} \approx 0.05$ provides a stable and effective operating point, enabling shortening primarily after training becomes sufficiently stable.

6 Conclusion

We study how to shorten reasoning trajectories in long reasoning models during *on-policy* rule-based RL, where rollout tokens dominate both training cost and learning dynamics. Our key perspective is that response length is an *auxiliary* trajectory property: correctness defines success, while brevity is a preference among successful trajectories. This motivates Short-RL, a *lazy length penalty* that shortens *after the model is right*—it applies length shaping only on correct trajectories (RightGate), only penalizes excess length beyond a tolerance band (SlackBand), and activates only when training accuracy is stable (StableSwitch).

Across logic reasoning and three math RL pipelines, Short-RL consistently reduces **Training (step-avg)** response length (a proxy for training-time rollout token cost) while maintaining or improving accuracy. Compared with post-RL shortening (e.g., Kimi (post)), our on-policy design reduces not only inference length but also the token cost incurred *during* RL training.

7 Limitations

Scope of applicability. Short-RL targets settings where the model produces an explicit multi-step reasoning trajectory and correctness can be reli-

ably assessed with a rule-based signal (e.g., math and logic). In such tasks, excessive length often reflects redundancy, making trajectory shortening both meaningful and measurable. For open-ended generation tasks (e.g., creative writing or dialogue), there may be no clear notion of "correct" answers or minimal-length successful trajectories; in these cases, imposing a length preference can be misaligned with the objective or reduce desirable stylistic variation.

Dependence on reward reliability. Our lazy gates rely on a reasonably accurate correctness signal (c_i) to determine which trajectories are eligible for length shaping. If the rule-based reward is noisy or can be exploited, the system may apply length pressure to trajectories that are not truly correct, potentially biasing learning. While this concern is shared by many rule-based RL pipelines, it becomes more salient when auxiliary trajectory shaping is introduced.

References

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

AI-MO. 2025. Aimo validation amc. 2025, February 2025.

Anthropic. 2025. Anthropic. claude 3.7 sonnet. Accessed on March 10, 2025.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *ArXiv*, abs/2502.04463.

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *Preprint*, arXiv:2503.05179.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *Preprint*, arXiv:2503.09567.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

Yu-Neng Chuang, Helen Zhou, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, and Xia Hu. 2025. Learning to route llms with confidence tokens. *Preprint*, arXiv:2410.13284.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, Suhang Wang, Yue Xing, Jiliang Tang, and Qi He. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *Preprint*, arXiv:2502.13260.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. *Preprint*, arXiv:2412.18547.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. 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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3828–3850. Association for Computational Linguistics.

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

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Preprint*, arXiv:2504.01296.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, and Heung-Yeung Shum Xiangyu Zhang. 2025. Open-reasoner-zero: An open source approach to scaling reinforcement learning on the base model. https://github.com/Open-Reasoner-Zero/Open-Reasoner-Zero.

MAA. American invitational mathematics examination. 2024. In american invitational mathematics examination. 2024, February 2024.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2024. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *Preprint*, arXiv:2412.11664.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? *Preprint*, arXiv:2411.01855.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *Preprint*, arXiv:2501.12570.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. Reasoning models can be effective without thinking. *Preprint*, arXiv:2504.09858.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. Cot-valve: Length-compressible chain-of-thought tuning. *Preprint*, arXiv:2502.09601.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-training elicits concise reasoning in large language models. *Preprint*, arXiv:2502.20122.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal,

Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. 2025. Thoughtterminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models. *Preprint*, arXiv:2504.13367.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, page 476–483. IEEE.

Jianshu She, Zhuohao Li, Zhemin Huang, Qi Li, Peiran Xu, Haonan Li, and Qirong Ho. 2025. Hawkeye:efficient reasoning with model collaboration. *Preprint*, arXiv:2504.00424.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *Preprint*, arXiv:2503.04472.

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

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Preprint*, arXiv:2503.16419.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *Preprint*, arXiv:2501.12599.

Han Wu, Yuxuan Yao, Shuqi Liu, Zehua Liu, Xiaojin Fu, Xiongwei Han, Xing Li, Hui-Ling Zhen, Tao Zhong, and Mingxuan Yuan. 2025. Unlocking efficient long-to-short llm reasoning with model merging. *Preprint*, arXiv:2503.20641.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *Preprint*, arXiv:2502.12067.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning. *ArXiv*, abs/2410.23123.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *Preprint*, arXiv:2502.14768.

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

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. 2025. Dynamic early exit in reasoning models. *Preprint*, arXiv:2504.15895.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *Preprint*, arXiv:2407.06023.

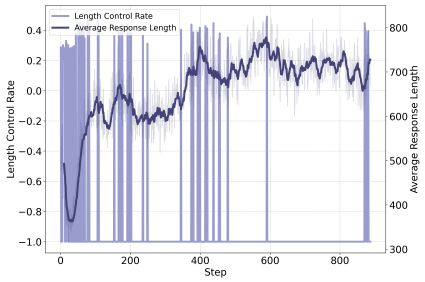
Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong

Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

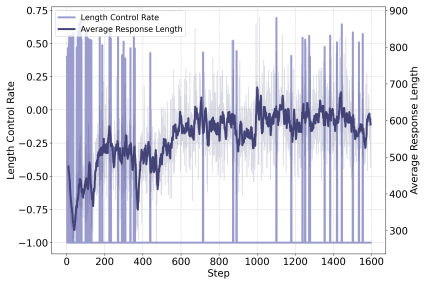
Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *Preprint*, arXiv:2503.18892.

A Appendix

A.1 Additional Experiments



(a) Open-Reasoner-Zero



(b) SimpleRL-Reason

Figure 7: Visualization of the length control rate during training.

A.1.1 Track the Length Reward

We also track the metric defined in Section Training Dynamics: When Does the Lazy Length Penalty Activate? in Figure 7 (Open-Reasoner-Zero and SimpleRL-Reason).

A.2 Training Details

Our experiments were conducted using a compute node equipped with 8 NVIDIA H100 GPUs. The CUDA version we use is 12.3.

```
<|im_start|>system\nYou are a helpful assistant. The
assistant first thinks about the reasoning process in
the mind and then provides the user with the answer. The
reasoning process and answer are enclosed within <think>
</think> and<answer> </answer> tags, respectively, i.e.,
<think> reasoning process here </think><answer> answer
here </answer>.  Now the user asks you to solve a
logical reasoning problem. After thinking, when you
finally reach a conclusion, clearly state the identity
of each character within <answer> </answer> tags. i.e.,
<answer> (1) Zoey is a knight\n(2) ...
</answer>.\n<|im_end|>\n<|im_start|>user\n{quiz}\n<|im_e
nd|>\n<|im_start|>assistant\n<think>
```

(a)

```
The user asks a question, and the Assistant solves
it.The assistant first thinks about the reasoning
process in the mind and then provides the user with the
final answer. The reasoning process and answer are
enclosed within <think> </think> and <answer> </answer>
tags, respectively, i.e., <think> reasoning process here
</think><answer> answer here </answer>.
\n\nUser:{question}\nAssistant: <think>
```

(b)

Figure 8: The prompt template for Logic-RL and Math-RL.

A.2.1 Logic-RL Training and Evaluation Details

The training and evaluation prompt template (Figure 8a) used in Logic-RL remains the same as in the original GitHub project. The training hyperparameters are listed in Table 3. During evaluation, we directly use the code from Logic-RL, which applies a temperature of 1.0 and top_p=1.0 for logic tasks, and a temperature of 0.8 with top_p=0.95 for math tasks.

A.2.2 Training and Evaluation Details for Math

The training and evaluation prompt template for three math settings is shown in Figure 8b. The training hyperparameters are listed in Table 3. During evaluation, we directly use the code from DeepScaleR, which employs a temperature of 1.0.

A.2.3 Reward Details

In all the math experiments, the standard task reward employs a format and outcome-based scheme:

$$R_{\text{task}} = \begin{cases} 3, & \text{format correct and answer correct} \\ -0.5, & \text{format correct and answer wrong} \\ -3, & \text{format wrong.} \end{cases} \tag{5}$$

In Logic-RL experiments, we directly use their original standard reward design.

Setting	**Logic-RL**	**DeepScaleR**	**Open-Reasoner-Zero**	**SimpleRL-Reason**
learning rate	1e-6	1e-6	5e-7	5e-7
batch size	8	128	64	16
ppo_mini_batch_size	32	64	256	64
ppo_micro_batch_size	8	32	64	2
rollout_n	8	8	8	8
temperature	0.7	0.6	1.0	1.0
kl_loss_coef	0.001	0.001	0.001	0.0001
epochs	3	3	1	3
max_response_length	4096	8192	4096	8192
algorithm	reinforce++	grpo	grpo	grpo
τ_l	200	100	100	50
τ_{acc}	0.05	0.05	0.02	0.05
α	1	1	1	1
Model	Qwen2.5-7B	DeepSeek Distill Qwen-1.5B	Qwen2.5-7B	Qwen2.5-7B

Table 3: Training details.