Stepwise Penalization for Length-Efficient Chain-of-Thought Reasoning

Xintong Li[1]*, **Sha Li**[2], **Rongmei Lin**[2], **Hongye Jin**[2], **Linwei Li**[2], **Hejie Cui**[2], **Sarah Zhang**[3]
Chia-Yuan Chang[2], **Kewei Cheng**[2], **Yuwei Zhang**[1], **Qingyu Yin**[3], **Jingbo Shang**[1], **Bing Yin**[2]
[1]UC San Diego [2]Amazon [3]Meta
{xil240,yuz163,jshang}@ucsd.edu
{slliz,linrongm,hongyjin}@amazon.com

Abstract

Large reasoning models improve with more test-time computation, but often overthink, producing unnecessarily long chains-of-thought that raise cost without improving accuracy. Prior reinforcement learning approaches typically rely on a single outcome reward with trajectory-level length penalties, which cannot distinguish essential from redundant reasoning steps and therefore yield blunt compression. Although recent work incorporates step-level signals, such as offline pruning, supervised data construction, or verifier-based intermediate rewards, reasoning length is rarely treated as an explicit step-level optimization objective during RL. We propose **S**tep-**w**ise **A**daptive **P**enalization (SWAP), a fine-grained framework that allocates length reduction across steps based on intrinsic contribution. We estimate step importance from the model's on-policy log-probability improvement toward the correct answer, then treat excess length as a penalty mass redistributed to penalize low-importance steps more heavily while preserving high-importance reasoning. We optimize with a unified outcome–process advantage within group-relative policy optimization. Experiments demonstrate that SWAP reduces reasoning length by **56.3%** on average while improving accuracy by **4.3%** relative to the base model.

1 Introduction

Large language models (LLMs) have achieved strong performance in complex reasoning tasks through Chain-of-Thought (CoT) prompting, which encourages the model to decompose a problem into intermediate reasoning steps [1, 10]. While this enables models to navigate complex logical structures, it often leads to overthinking: models generate excessively long reasoning chains dominated by redundant or low-value steps without improving final accuracy [32]. Such unnecessary verbosity substantially increases inference cost and latency, and can even degrade performance by introducing opportunities for hallucination in later stages of generation [18, 20].

To mitigate overthinking, recent work has increasingly turned to post-training strategies, including supervised preference learning for shorter responses [27], distillation from long to short CoT models [17], and reinforcement learning (RL) with explicit length constraints [16]. Among these, RL-based approaches offer a particularly flexible mechanism to trade off accuracy and efficiency. Most existing methods, however, operate at the trajectory level, for example, by imposing global length penalties, fixed token budgets, or difficulty-dependent reasoning modes [2, 23]. Although effective in reducing the overall length of the reasoning, these coarse-grained strategies suffer from a fundamental limitation by treating all the reasoning steps as equally valuable.

In practice, reasoning trajectories are highly heterogeneous at the step level [4]. Some steps introduce critical information that decisively pivots the solution toward correctness, while others correspond to

* Work done during internship at Amazon.

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

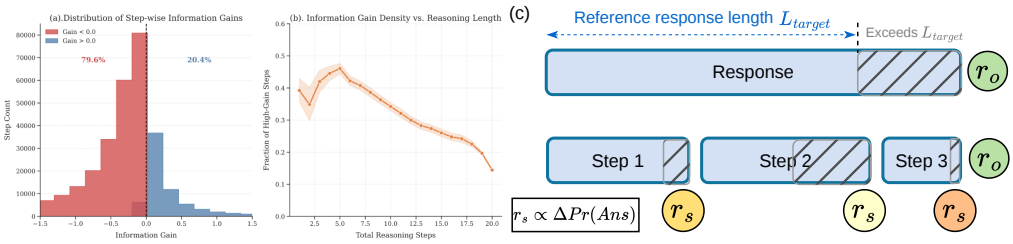

Figure 1: **Step-level redundancy in reasoning.** We segment the model's responses into steps by token length and use the change in the probability of generating the ground truth answer to measure each step's progress. (a) Distribution of step-wise information gain: most steps contribute little or no progress toward the correct answer, and high-gain steps are rare (4% with gain > 0.5). (b) Longer reasoning trajectories contain a lower fraction of high-gain steps. (c) In SWAP, in addition to the outcome reward r_o, each step also gets assigned a progress reward r_s. When the response is correct but exceeds the reference length, the penalty (dashed grey area) gets distributed to each step based on its progress (darker shades represent more progress).

redundant verification, reformulation, or low-utility exploration. As a result, trajectory-level penalties often induce indiscriminate compression, risking the removal of essential reasoning while preserving irrelevant text [39]. Figure 1 empirically demonstrates this imbalance. As shown in (a), the majority of reasoning steps yield negligible or no information gain toward the final answer, with only a small fraction contributing substantial progress. Moreover, (b) shows that longer reasoning trajectories contain a markedly lower density of high-impact steps, indicating that redundancy accumulates as reasoning length increases. Together, these observations indicate that overthinking is not simply a function of how much reasoning is performed, but of where redundant computation accumulates within a trajectory.

These findings highlight two principles that are largely underexplored in existing reward design. First, length regularization should be aware of the difficulty. Harder problems naturally require longer reasoning and should not be penalized identically to easier ones [36, 31]. Second, and more critically, length control should be conducted at the step level, rather than applied uniformly across an entire trajectory within a single trajectory, different reasoning steps contribute unequally to correctness and should therefore be treated asymmetrically [30]. While recent work has begun to incorporate step-level information, such as offline pruning, supervised data construction, or verifier-based intermediate rewards, most approaches either use step signals outside of RL or rely on auxiliary models to assess step correctness [40, 12, 35]. As a result, reasoning length itself is rarely treated as an explicit step-level optimization objective during RL, and existing methods lack a principled mechanism for deciding which steps to shorten and by how much when a trajectory becomes overly long.

In this work, we propose **S**tep-**w**ise **A**daptive **P**enalization (SWAP), a fine-grained RL framework that allocates length reduction across reasoning steps based on their intrinsic contribution to solving the problem. Rather than relying on external reward models or heuristic estimators, SWAP derives step importance directly from the model's own behavior. Specifically, we quantify the information gain of each reasoning step by measuring the improvement in the log-probability assigned to the correct answer after that step. Steps that induce large gains are interpreted as carrying substantial reasoning mass and should be preserved, while steps with minimal or no gain are treated as low-impact and preferentially penalized.

To preserve correctness while enforcing efficiency, we implement this algorithm via a stepwise weighted length penalty mechanism. For each prompt, we define a reference length that is based on the difficulty of the correct responses and compute a global penalty when a trajectory exceeds this target. This penalty mass is then redistributed across steps according to their importance, so that low-utility steps absorb most of the penalty while high-impact steps are protected from aggressive compression. This design enables adaptive, fine-grained length control without requiring pre-specified token budgets or external supervision. To ensure correctness while enforcing efficiency, we integrate the stepwise signal with trajectory-level outcome rewards through a unified outcome–process advantage under Group Relative Policy Optimization (GRPO) [10]. The outcome advantage promotes globally correct solutions, while a backward-propagated process advantage assigns credit to tokens based on the cumulative contribution of future reasoning steps. Together, this unified formulation decomposes

learning into complementary roles: outcome rewards gate correctness, while progress-aware step rewards shape where and how reasoning is compressed.

Experiments across six reasoning benchmarks spanning mathematics, science, and coding demonstrate that SWAP achieves a substantially better accuracy–efficiency trade-off than prior methods. On DeepSeek-Distill-Qwen-1.5B, SWAP reduces the average reasoning length by **56.3%** while improving average accuracy from 42.2 to **46.5**, outperforming all baselines across the six-benchmark average. On the 7B model, SWAP reduces token usage by **50.0%** on average while achieving the best overall accuracy, with particularly strong gains on challenging reasoning benchmarks such as AIME24, AIME25, and GPQA. These results show that significant overthinking persists across mathematical, scientific, and coding tasks, and that stepwise optimization provides a principled and effective path toward efficient reasoning.

2 Related Work

Efficient reasoning in LLMs. Modern reasoning LLMs often suffer from overthinking, producing unnecessarily long CoT that increase latency and inference cost without improving accuracy [8, 32]. Recent work on efficient reasoning spans both supervised fine-tuning and preference-based post-training. For instance, O1-Pruner [21] applies length-harmonized fine-tuning to prune redundant reasoning while preserving performance, while AdaptThink [41] introduces hybrid strategies that dynamically switch between "thinking" and "no-thinking" modes based on problem difficulty. SABER [42] further enables user-controllable, token-budgeted reasoning at inference time. Complementary analyses highlight that inefficiency is not only a matter of how much reasoning is generated, but also where redundancy accumulates. LC-R1 [6] explicitly targets such redundant segments while preserving a sufficient core of reasoning steps. In parallel, recent studies on reasoning distillation and trace selection show that data quality and selection criteria strongly shape distilled reasoning behavior [17, 5], underscoring the importance of fine-grained control over reasoning structure.

RL for reducing overthinking. A prominent line of work leverages RL to trade off accuracy and efficiency through explicit length constraints. Early approaches such as L1 [2] demonstrate that RL can enforce strong length control under fixed token budgets. Subsequent methods introduce progressively tighter constraints or iterative pruning strategies to compress long CoT [14], adaptive reward shaping and target-length policies [19], and accuracy–length control rules or hierarchical budgeting mechanisms [16, 23]. While effective at reducing overall reasoning length, most of these approaches apply penalties uniformly across the trajectory, leading to indiscriminate compression. More recent step-aware RL methods address this limitation by incorporating verifiable intermediate feedback [34]. VSRM assigns rule-based verifiable rewards to encourage effective steps and penalize ineffective ones [40], while SmartThinker combines supervised fine-tuning with step-wise RL to guide reasoning refinement [12]. These advances highlight the importance of step-level credit assignment for suppressing overthinking.

3 Framework

We introduce a progress-aware stepwise reward framework that redistributes trajectory-level length penalties across reasoning steps based on their marginal contribution to reaching the correct answer, and optimizes this signal jointly with outcome correctness under GRPO. Our approach consists of three stages: estimating intrinsic step importance from on-policy log-probability improvements, allocating length penalties to individual steps according to their importance, and integrating the resulting stepwise signal with the outcome reward through a unified outcome–process advantage estimator. The complete algorithm is summarized in Algorithm 1 in the appendix.

3.1 Step Reward Measurement

To assign step rewards, we define a robust step segmentation scheme and introduce an intrinsic measure of step importance based on answer log-probability improvement.

Step segmentation by token budget. Unlike prior work that relies on heuristic sentence boundaries or explicit reasoning transition words (e.g., *wait*, *however*, *therefore*) [3, 33, 37], we define reasoning steps using a simple and robust token-based segmentation scheme.

Given a generated response, we extract the reasoning portion (e.g., content before `</think>`) and partition it into at most K steps $\{s_k\}_{k=1}^{K}$ by grouping approximately a fixed token budget M per step. If the response would exceed K steps, we adapt M upward to ensure a bounded number of steps. We additionally align step boundaries to the next new line to produce cleaner, sentence-level segments. Under this construction, each step corresponds to a contiguous span of tokens and forms a semantically coherent chunk of reasoning.

Step reward via answer log-probability gain. We define step importance intrinsically using the model's own confidence toward the correct answer. Intuitively, a reasoning step is important if it substantially increases the likelihood of the final answer.

Let x denote the prompt, $a^\star$ the task-specific target answer, and $\{s_1, \ldots, s_K\}$ the segmented reasoning steps. For each prefix consisting of the first k steps, we construct the conditional context $x_k = (x, s_1, \ldots, s_k)$, and compute the average per-token log-probability of the correct answer,

$$\ell_k = \frac{1}{|a^\star|} \log p_\theta(a^\star \mid x_k)\,.$$

We also compute a baseline score ℓ_0 using the prompt without any reasoning steps. We then define a progress-aware step reward as monotone incremental information gain

$$\Delta_k = \max\Big(0,\ \ell_k - \max_{j<k} \ell_j\Big), \qquad k = 1, \ldots, K. \tag{1}$$

This construction attributes positive reward only when a step improves the model's confidence beyond all previous steps, preventing later redundant steps from accumulating spurious credit. Steps that do not introduce new information receive zero reward. Each Δ_k is assigned to the end of the corresponding step in the token sequence, yielding sparse, step-aligned reward markers that are compatible with on-policy reinforcement learning.

3.2 Step-Weighted Length Penalty Redistribution

To discourage overlong reasoning while preserving essential computation, we introduce a step-weighted length penalty that selectively penalizes low-importance reasoning steps when a trajectory exceeds an adaptive length target.

Under GRPO, responses are generated in groups corresponding to the same prompt. For each group, we define a target length L_{target} as the median length of *correct* responses within the group, which serves as a difficulty-aware reference. If a group contains no correct rollouts, we set L_{target} to the maximum rollout length. Given a response of length L, if $L \le L_{\text{target}}$, no length penalty is applied. Otherwise, we compute a total penalty mass

$$P = \lambda \cdot \frac{L - L_{\text{target}}}{L_{\text{target}}}, \tag{2}$$

where λ controls the overall strength of length regularization. Rather than applying this penalty uniformly across tokens, we redistribute it according to step importance. Let $g_k = \ell_k - \ell_{k-1}$ denote the local log-probability gain associated with step k. We assign each step a penalty weight $w_k \propto \exp(-g_k/\tau)$, where τ is a temperature parameter. This construction assigns larger weights to steps with smaller gains, while protecting high-impact steps from aggressive penalization. The weights are normalized such that $\sum_{k=1}^{K} w_k = 1$, and each step k receives a penalty of $-P \cdot w_k$.

Finally, we combine intrinsic progress reward and length penalty into a single step-level reward,

$$r_k = \Delta_k - P \cdot w_k, \tag{3}$$

where Δ_k is the monotone information gain defined in Eq. 1. This step-weighted redistribution mechanism ensures that excess length is preferentially removed from low-utility reasoning segments, enabling fine-grained and adaptive reasoning compression without indiscriminately shortening the entire trajectory.

3.3 Unified Outcome–Process Advantage

We integrate stepwise rewards into GRPO by constructing a unified token-level advantage that combines standard group-relative outcome supervision with a backward-propagated process signal derived from step rewards. This formulation preserves the stability of outcome-based GRPO while enabling fine-grained efficiency optimization at the reasoning-step level.

Group-relative outcome advantage. For each query x, we sample a group of N responses $\{y_i\}_{i=1}^{N}$. Each response receives a sparse outcome reward $r_i^{\text{out}} \in \mathbb{R}$, defined as positive if the final answer is correct and zero otherwise. Following the standard GRPO, outcome rewards are normalized within the group using mean $\mu_{\text{out}}(x)$ and standard deviation $\sigma_{\text{out}}(x)$, resulting in the outcome advantage $A_i^{\text{out}} = \frac{r_i^{\text{out}} - \mu_{\text{out}}(x)}{\sigma_{\text{out}}(x) + \varepsilon}$.

Backward-propagated process advantage. Each response y_i is segmented into K_i reasoning steps, with step rewards $\{r_{i,k}\}_{k=1}^{K_i}$ defined in Eq. (3) and placed at the end token of each step. To stabilize training, step rewards are normalized within each group using statistics computed over *correct* trajectories only,

$$\tilde{r}_{i,k} = \frac{r_{i,k} - \mu_{\text{step}}(x)}{\sigma_{\text{step}}(x) + \varepsilon}, \tag{4}$$

where $\mu_{\text{step}}(x)$ and $\sigma_{\text{step}}(x)$ denote the mean and standard deviation of step rewards across correct responses for the prompt x.

Let $e_{i,k}$ denote the token index corresponding to the end of step k in response y_i. We first construct a token-level process advantage by accumulating rewards from future reasoning steps, $A_{i,t}^{\text{proc}} = \sum_{k:\, t \le e_{i,k}} \tilde{r}_{i,k}$. We then normalize this signal by its average absolute magnitude over the trajectory,

$$\hat{A}_{i,t}^{\text{proc}} = \frac{A_{i,t}^{\text{proc}}}{s_i^{\text{proc}} + \varepsilon}, \qquad s_i^{\text{proc}} = \frac{1}{T_i^{+}} \sum_{t'} \left| A_{i,t'}^{\text{proc}} \right|. \tag{5}$$

This reverse cumulative construction assigns each token credit proportional to the cumulative contribution of future reasoning steps, allowing step-level efficiency signals to influence earlier generation decisions. The scale term s_i^{proc} stabilizes the process advantage across rollouts with different lengths, where T_i^{+} denotes the number of tokens in the trajectory.

Unified advantage. Finally, we combine outcome and process advantages into a single token-level signal,

$$A_{i,t} = \beta\, A_i^{\text{out}} \;+\; \theta\, \mathbb{I}\big[r_i^{\text{out}} > 0\big] \cdot \hat{A}_{i,t}^{\text{proc}}, \tag{6}$$

where $\beta, \theta > 0$ control the trade-off between global correctness and local efficiency. Importantly, the process term is gated by correctness, ensuring that step-level signals influence optimization only for correct trajectories and preventing noisy step rewards from corrupting learning on incorrect samples.

Let $\pi_{\theta_{\text{old}}}$ denote the behavior policy used to generate rollouts. We optimize the standard clipped GRPO objective using the unified advantage:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{x,i,t}\Big[\min\big(r_{i,t}(\theta)A_{i,t},\, \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)A_{i,t}\big)\Big] \tag{7}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}$ is the token-wise importance ratio.

Algorithm 1 summarizes the overall SWAP training procedure, including step importance estimation, penalty redistribution, and unified advantage construction under GRPO.

4 Experiments

4.1 Experimental Setup

Models and Training Data. We evaluate SWAP on three widely used reasoning models, DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B [7], and Qwen3-14B [38]. These models

Method	MATH-500		AMC23		AIME24		AIME25		GPQA		LiveCodeBench		Avg.	
	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓
DeepSeek-Distill-Qwen-1.5B														
Original	83.4	4777	70.0	7722	26.7	12571	**23.3**	12247	33.2	9474	17.3	13139	42.3	9988
ThinkPrune	83.1	2332	72.5	3791	24.8	7469	20.4	7024	36.2	6219	16.0	8027	42.2	5810
LC-R1	81.9	2249	68.3	3607	23.3	7061	18.8	6647	34.4	5837	14.8	9271	40.3	5778
AdaptThink	83.9	1863	70.0	3628	31.3	7236	22.3	7195	34.5	7808	18.1	10611	43.4	6390
JET	85.1	3275	75.9	5455	31.3	9899	22.7	9427	35.7	7566	18.8	9125	44.9	7457
Laser-D	85.2	2589	74.2	4227	30.8	7929	22.1	7387	35.0	6524	16.6	8555	44.0	6201
Laser-DE	85.1	2821	75.2	4503	31.0	8330	21.3	7954	35.7	6769	18.8	8836	44.5	6535
SwAP (ours)	**86.5**	1597	**78.3**	2645	**34.0**	4915	**23.3**	4627	**36.9**	4610	**19.8**	7808	**46.5**	4367
DeepSeek-Distill-Qwen-7B														
Original	93.0	3743	84.7	5794	52.3	10579	37.5	11076	48.0	8153	31.5	31055	57.8	11733
L1-Exact	91.9	2186	81.1	2589	43.5	4026	28.8	4042	48.1	3672	28.6	12617	53.7	4855
L1-Max	92.0	2134	79.5	2594	44.6	4139	31.5	4093	48.3	3662	29.6	12763	54.3	4897
LC-R1	89.9	1568	81.6	2922	49.6	6972	36.3	7657	48.1	5927	27.9	21331	55.6	7729
AdaptThink	91.8	1825	80.0	3835	52.9	8600	37.5	9548	49.3	7384	**32.6**	27468	57.4	9776
Laser-D	93.5	1961	**87.0**	2925	55.6	6108	37.9	6629	48.2	5675	30.7	22093	58.8	7565
Laser-DE	**93.6**	1898	85.0	2901	53.5	5968	37.9	6161	48.6	5287	32.4	20984	58.5	7199
SwAP (ours)	**93.6**	1573	85.6	2685	**56.5**	4708	**40.8**	5361	**49.8**	3978	30.9	16929	**59.5**	5872

Table 1: Performance and reasoning length across math, scientific, and coding benchmarks. Avg. denotes the average pass@1 across all datasets. Three shortest-length methods are shaded green.

demonstrate strong reasoning capability but frequently exhibit overthinking behavior, producing unnecessarily long chains-of-thought. For training, we use the DeepScaleR-Preview dataset [22], which contains 40K high-quality mathematical reasoning problems collected from AIME, AMC, Omni-Math [9], and STILL [26]. This dataset provides diverse difficulty levels and supports studying both reasoning accuracy and efficiency.

Baselines. We compare SwAP against a diverse set of state-of-the-art methods for efficient reasoning, covering three representative categories. Length-penalty-based RL methods include ThinkPrune [14], which progressively enforces stricter token limits during RL to prune redundant reasoning; LC-R1 [6], which combines GRPO with global length and compression rewards to remove invalid post-solution reasoning; JET [11], which uses trajectory truncation and quality-controlled length rewards to train models to stop unnecessary reasoning; and two variants of the LASER [19] framework, LASER-D and LASER-DE, which apply dynamic and difficulty-aware length reward shaping, with LASER-DE additionally encouraging exploration on incorrect trajectories. Adaptive thinking-mode methods include AdaptThink [41], which trains models to dynamically select between thinking and no-thinking modes based on problem difficulty. Length-control methods include L1-Exact and L1-Max [2], which optimize reasoning accuracy under prompt-specified exact or maximum length constraints via length-controlled policy optimization.

Evaluation Benchmarks. We evaluate all methods on six reasoning benchmarks spanning mathematical, scientific, and coding domains: MATH-500 [13], AMC23 [25], AIME24, AIME25 [24], GPQA [28], and LiveCodeBench [15]. Evaluation is performed with temperature $T = 0.6$, top-$p = 0.95$, and a maximum generation length of 32,768 tokens. For each benchmark, we generate 16 samples per query and report both pass@1 accuracy and the average response length in tokens, capturing the trade-off between correctness and reasoning efficiency.

Implementation Details. We implement SwAP using the VeRL framework [29]. During training, we use a batch size of 128 and generate 16 rollouts per prompt with a maximum response length of 12,000 tokens and sampling temperature 0.6. Training is conducted for two epochs for both model sizes, resulting in a total of 623 steps.

For step segmentation, we allocate approximately 350 tokens per step and cap the maximum number of reasoning steps at 25. The target-answer instantiation is provided in Appendix A.2. We set the length penalty coefficient $\lambda = 1$ to apply the full penalty mass when responses exceed the target length. The outcome advantage weight β is set to 1, while the process advantage weight θ is set to 0.3 to avoid overly aggressive early length reduction. Training of the 1.5B model is conducted on a single node with $8\times$H100 GPUs, whereas the 7B model is trained on a single node with $8\times$H200 GPUs.

Reward	Penalty	MATH-500		AMC23		AIME24		AIME25		GPQA		LiveCodeBench		Avg.	
		Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓
Base model (no RL)		83.4	4777	70.0	7722	26.7	12571	23.3	12247	33.2	9474	17.3	13139	42.3	9988
Outcome	Trajectory-level	79.5	1534	70.6	2953	29.8	6287	21.9	6048	35.6	5274	18.1	8780	42.6	5146
Step	–	79.9	1526	67.8	2915	27.1	6019	20.8	5872	34.8	5410	14.9	9488	40.9	5205
Outcome + Step	–	85.7	2594	77.2	4169	32.5	8027	24.4	7602	35.6	6172	18.9	11112	45.7	6613
Outcome + Step	Static (per-step)	83.7	1276	72.8	1899	25.2	3264	20.6	2702	35.8	2961	18.3	5477	42.7	2930
Outcome + Step	Uniform	86.4	2064	77.8	3469	32.9	6568	**25.4**	6163	36.4	5913	18.7	8817	46.3	5499
Outcome + Step	Adaptive (**Ours**)	**86.5**	1597	**78.3**	2645	**34.0**	4915	23.3	4627	**36.9**	4610	**19.8**	7808	**46.5**	4367

Table 2: Performance and reasoning length of different model variants. All results are based on the DeepSeek-Distill-Qwen-1.5B model. Avg. denotes the average pass@1 across all datasets.

4.2 Main results

Table 1 reports results across six reasoning benchmarks spanning mathematics, science, and coding. Across both model scales, SWAP achieves the best average accuracy while substantially reducing reasoning length, demonstrating a strong accuracy–efficiency trade-off.

Relative to the base models, SWAP removes a large fraction of redundant reasoning without sacrificing correctness. On DeepSeek-Distill-Qwen-1.5B, SWAP reduces the average response length by 56.3% while improving average accuracy from 42.2 to 46.5, achieving the best performance on all six benchmarks. On DeepSeek-Distill-Qwen-7B, SWAP reduces token usage by 50.0% while improving average accuracy from 57.8 to 59.5, with particularly strong gains on challenging benchmarks such as AIME24, AIME25, and GPQA. These results suggest that substantial overthinking persists across mathematical, scientific, and coding tasks, even in larger reasoning models.

Compared with efficient reasoning baselines, SWAP provides a more favorable balance between accuracy and length. Prior methods such as ThinkPrune, LC-R1, and L1-Exact/Max achieve notable compression but often incur accuracy drops, especially on difficult benchmarks. LASER-D and LASER-DE are the most competitive baselines, but SWAP achieves higher accuracy with shorter average reasoning length on both model scales. These results show that progress-aware stepwise optimization can more selectively reduce redundant reasoning while preserving task-critical computation.

4.3 Reward Components and Penalty Strategies

We ablate the reward and penalty design of SWAP on DeepSeek-Distill-Qwen-1.5B. As shown in Table 2, we compare variants that isolate the outcome reward, the step-level reward, and different ways of imposing length penalties.

Effect of reward components. Using only the outcome reward with a trajectory-level penalty substantially shortens responses, reducing the average length from 9,988 to 5,146 tokens. However, its accuracy improves only marginally over the base model, suggesting that global compression alone cannot reliably distinguish redundant reasoning from task-critical steps. The step-only variant performs worse, reducing average accuracy from 42.2 to 40.9, indicating that intrinsic progress signals alone are insufficient without outcome-level grounding. In contrast, combining outcome and step rewards without an explicit penalty improves average accuracy to 45.7, but still produces relatively long responses. This confirms that outcome and step signals are complementary, while an explicit length penalty is necessary for efficient reasoning.

Effect of penalty strategies. We next compare different penalty designs under the unified outcome-plus-step reward. A static per-step penalty achieves the shortest responses but substantially hurts accuracy, especially on difficult benchmarks such as AIME24 and AIME25. Uniformly distributing the penalty across steps improves accuracy to 46.3, but remains less efficient, with an average length of 5,499 tokens. Our adaptive penalty achieves the best average accuracy of 46.5 while further reducing the average length to 4,367 tokens. Compared with the uniform penalty, SWAP uses 20.6% fewer tokens while slightly improving accuracy. These results show that dynamically assigning larger penalties to less important steps better removes redundant reasoning while preserving the steps needed for correctness.

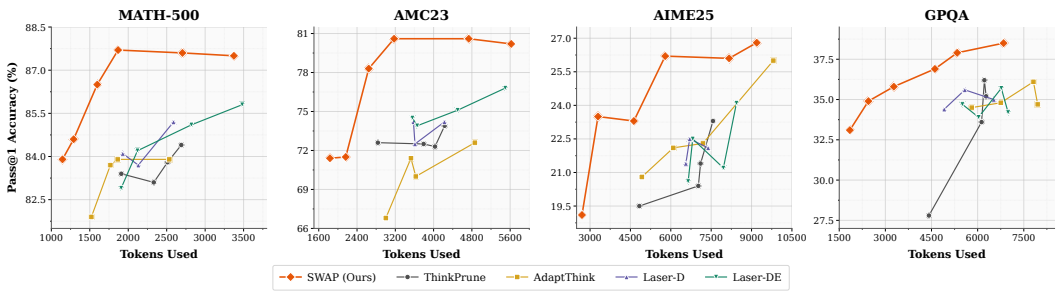

Figure 2: Comparison of Pass@1 accuracy vs. token usage across math benchmarks. SwAP establishes the Pareto frontier for best performance under every token budget.

Method	MATH-500		AMC23		AIME24		AIME25		GPQA		Avg.	
	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓
Original	96.7	4723	89.8	7267	68.1	11480	60.8	12800	63.9	5354	75.9	8325
GRPO	96.8	3820	91.1	5632	**73.3**	9995	62.1	11174	63.5	4252	77.4	6975
Laser-D	96.7	2591	87.2	4116	69.6	8265	58.3	9106	62.1	3233	74.8	5462
Laser-DE	97.0	2432	90.3	3750	71.3	6204	61.9	8359	63.3	2908	76.8	4730
SwAP (ours)	**97.3**	1972	**91.7**	3144	72.7	6625	**62.3**	7583	**64.0**	2298	**77.6**	**4324**

Table 3: Performance and reasoning length comparison on Qwen3-14B. Avg. denotes average accuracy across benchmarks. We retrain LASER-D and LASER-DE on the 14B model.

5 Analysis

5.1 Performance under Different Budgets

Figure 2 compares Pass@1 accuracy and token usage across mathematical and scientific reasoning benchmarks. Overall, SwAP consistently achieves higher accuracy than prior length-control methods under comparable or smaller token budgets.

On MATH-500 and AMC23, SwAP maintains strong performance even under aggressive compression, while baselines such as ThinkPrune, AdaptThink, and LASER variants typically require longer responses to approach similar accuracy. This suggests that SwAP can more effectively remove low-utility reasoning while preserving task-relevant computation. On the more challenging AIME25 benchmark, SwAP continues to dominate the trade-off frontier, achieving higher accuracy across different token budgets and remaining robust even when the average response length is substantially reduced. The same trend extends beyond mathematical reasoning to GPQA, where SwAP achieves the best accuracy while using fewer tokens than most competing methods. These results demonstrate that progress-aware stepwise optimization improves the accuracy–efficiency trade-off not only on math tasks but also on scientific reasoning benchmarks.

5.2 Scaling to Larger Models

We further evaluate SwAP on Qwen3-14B to test its scalability. As shown in Table 3, SwAP achieves the best average accuracy while using the fewest tokens on average. Compared with the original model, it reduces the average response length from 8,324.8 to 4,324.4 tokens while improving accuracy from 75.88 to 77.60. It also outperforms GRPO and LASER, a strong length-reduction baseline, in terms of average accuracy and response length. These results suggest that stepwise optimization remains effective for larger reasoning models.

5.3 Validation of Step Importance

Our step-importance score Δ_k measures a step's positive gain in correct-answer log-probability beyond the best preceding prefix. To assess whether this score reflects a step's contribution to final correctness, we analyze 570 correct trajectories from MATH-500, AMC23, and AIME24. We first test how deleting a step affects accuracy by generating new continuations with and without that step. Comparing steps at similar positions and with similar token lengths, Table 4 shows that removing high-Δ_k steps reduces accuracy by 11.6 percentage points, compared with 5.5 for random steps and

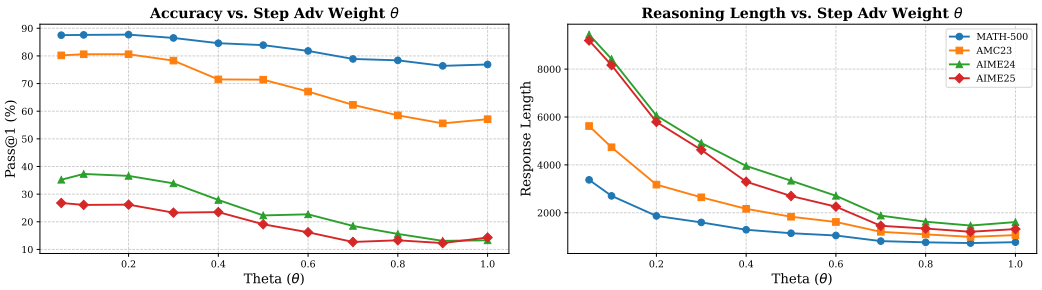

Figure 3: Impact of step advantage weight θ on model performance and response length. A moderate value of $\theta \in [0.2, 0.4]$ achieves the best balance. Hard datasets such as AIME 2024/2025 are more sensitive to a large θ value.

Deleted step	Acc. drop	95% CI
High-Δ_k	**11.6**	[7.5, 15.9]
Random	5.5	[2.6, 8.5]
Low-Δ_k	1.3	[0.0, 3.0]

Table 4: Accuracy drop (percentage points) after deleting a late-position step and resampling the continuation and final answer. Steps are matched by position and token length; intervals indicate 95% confidence.

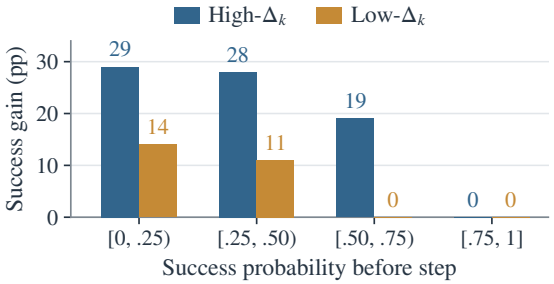

Figure 4: Change in final-answer success after including a step (percentage points), estimated with 16 continuations per prefix.

only 1.3 for low-Δ_k steps. This larger loss indicates that high-scoring steps contain reasoning that is more important to preserve. We then test whether including a step improves the probability of reaching the correct answer, using 16 sampled continuations from each prefix before and after the step. Figure 4 shows that high-Δ_k steps improve success by 29 percentage points, versus 14 for low-Δ_k steps, when initial success is below 25%. The corresponding gains are 28 versus 11 points for initial success of 25–50%, and 19 versus 0 for 50–75%. These consistently larger improvements show that higher scores also identify steps that help the model complete the solution correctly. Together, both analyses support using Δ_k to guide selective compression: preserving steps that contribute to correctness while concentrating compression on less useful reasoning.

5.4 Effect of Step Advantage Coefficient θ

We analyze the effect of the step advantage weight θ in Figure 3. Larger θ strengthens step-level penalization, leading to shorter responses across all benchmarks. When θ is moderate, around 0.2–0.4, SwAP substantially reduces reasoning length while largely preserving accuracy. However, further increasing θ causes noticeable accuracy degradation, especially on harder benchmarks such as AIME24 and AIME25, where excessive penalization may remove reasoning steps necessary for solving difficult problems. For large values of θ, response length begins to saturate, while accuracy remains low, indicating diminishing returns from stronger compression. These results suggest that step-level process advantages should be moderately weighted to suppress redundant reasoning without eliminating essential intermediate computation.

5.5 Effect of Step Segmentation

We study how different step segmentation strategies affect the quality of step-level supervision. As shown in Table 5, overly fine-grained segmentation, such as sentence-level splitting, leads to shorter responses but noticeably lower accuracy, suggesting that fragmented steps make importance estimation less reliable. Newline-based segmentation also reduces length aggressively, but underperforms token chunks on both MATH-500 and AIME25. Discourse-marker segmentation performs competitively, but produces longer responses than our token-chunk strategy.

Step size	MATH-500		AIME25	
	pass@1↑	Len↓	pass@1↑	Len↓
Newline	84.4	1450	22.5	4307
Sentence	79.3	1319	21.3	4128
Marker	85.7	1788	**23.3**	5306
TokenChunk	**86.5**	1597	**23.3**	4627

Table 5: Effect of step segmentation strategies on accuracy and reasoning length.

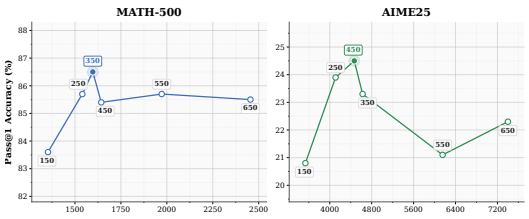

Figure 5: Effect of token-based step size on Pass@1 accuracy.

We further vary the token chunk size in Figure 5. Very small chunks can over-penalize local low-gain segments, while overly large chunks reduce the resolution of step-level rewards. A moderate chunk size, around 350 tokens, achieves the best overall performance, motivating our segmentation choice.

5.6 Additional Analysis

We provide additional analysis in Appendix A. Figure 6 shows that SWAP achieves both shorter responses and higher validation accuracy compared to alternative reward designs. Figure 7 examines the effect of step segmentation granularity, where overly small steps lead to fragmented importance estimation and overly large steps reduce reward resolution; a moderate step size performs best. Figure 8 further shows a clear shift toward positive step-wise information gain after training, indicating that low-utility reasoning steps are suppressed while high-impact transitions are preserved.

6 Conclusion

We presented SWAP, a step-wise length penalization framework that addresses overthinking in large reasoning models through fine-grained reinforcement learning. By measuring each reasoning step's intrinsic contribution via on-policy information gain and redistributing length penalties accordingly, SWAP enables selective compression that preserves critical reasoning while eliminating redundancy. Integrated with a unified outcome–process advantage under GRPO, our approach balances correctness and response length without relying on external reward models or heuristic budgets. Extensive experiments demonstrate that step-level optimization consistently improves performance across mathematical, scientific, and coding benchmarks. Our findings suggest that overthinking is fundamentally a step-level phenomenon, and that stepwise credit assignment is a principled direction for improving reasoning efficiency in future large-scale models.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

[3] Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houliston, et al. Reasoning language models: A blueprint. *arXiv preprint arXiv:2501.11223*, 2025.

[4] Paul C Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2025.

[5] Xinghao Chen, Zhijing Sun, Guo Wenjin, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, et al. Unveiling the key factors for distilling chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15094–15119, 2025.

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

[7] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

[8] Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*, 2025.

[9] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

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

[11] Jinyi Han, Ying Huang, Ying Liao, Zishang Jiang, Xikun Lu, Haiquan Zhao, Xinyi Wang, Guanghao Zhou, Sihang Jiang, Jiaqing Liang, et al. Your models have thought enough: Training large reasoning models to stop overthinking. *arXiv preprint arXiv:2509.23392*, 2025.

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

[13] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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

[15] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[16] Ruosen Li, Ziming Luo, Quan Zhang, Ruochen Li, Ben Zhou, Ali Payani, and Xinya Du. Aalc: Large language model efficient reasoning via adaptive accuracy-length control. *arXiv preprint arXiv:2506.20160*, 2025.

[17] Yang Li, Youssef Emad, Karthik Padthe, Jack Lanchantin, Weizhe Yuan, Thao Nguyen, Jason Weston, Shang-Wen Li, Dong Wang, Ilia Kulikov, et al. Naturalthoughts: Selecting and distilling reasoning traces for general reasoning tasks. *arXiv preprint arXiv:2507.01921*, 2025.

[18] Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. Compressing chain-of-thought in llms via step entropy. *arXiv preprint arXiv:2508.03346*, 2025.

[19] Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. *arXiv preprint arXiv:2505.15612*, 2025.

[20] Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17907–17918, 2025.

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

[22] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.

[23] Shangke Lyu, Linjuan Wu, Yuchen Yan, Xingyu Wu, Hao Li, Yongliang Shen, Peisheng Jiang, Weiming Lu, Jun Xiao, and Yueting Zhuang. Hierarchical budget policy optimization for adaptive reasoning. *arXiv preprint arXiv:2507.15844*, 2025.

[24] MAA. American invitational mathematics examination (AIME). Mathematics Competition Series, n.d.. URL `https://maa.org/math-competitions/aime`.

[25] MAA. American mathematics competitions (AMC 10/12). Mathematics Competition Series, n.d.. URL `https://maa.org/math-competitions/amc`.

[26] Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*, 2024.

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

[28] 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. *arXiv preprint arXiv:2311.12022*, 2023.

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

[30] Feifan Song, Shaohang Wei, Bofei Gao, Yejie Wang, Wen Luo, Wei Li, Linli Yao, Weimin Xiong, Liang Chen, Tianyu Liu, et al. Mitigating overthinking through reasoning shaping. *arXiv preprint arXiv:2510.09535*, 2025.

[31] Mingyang Song and Mao Zheng. Walk before you run! concise llm reasoning via reinforcement learning. *arXiv preprint arXiv:2505.21178*, 2025.

[32] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

[33] Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. *arXiv preprint arXiv:2310.05707*, 2023.

[34] Zhen Wang, Zhifeng Gao, and Guolin Ke. Masked-and-reordered self-supervision for reinforcement learning from verifiable rewards. *arXiv preprint arXiv:2511.17473*, 2025.

[35] Junda Wu, Yuxin Xiong, Xintong Li, Zhengmian Hu, Tong Yu, Rui Wang, Xiang Chen, Jingbo Shang, and Julian McAuley. Ctrls: Chain-of-thought reasoning via latent state-transition. *arXiv preprint arXiv:2507.08182*, 2025.

[36] Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *arXiv preprint arXiv:2506.05256*, 2025.

[37] Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23336–23351, 2025.

[38] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[39] Sheldon Yu, Yuxin Xiong, Junda Wu, Xintong Li, Tong Yu, Xiang Chen, Ritwik Sinha, Jingbo Shang, and Julian McAuley. Explainable chain-of-thought reasoning: An empirical analysis on state-aware reasoning dynamics. *arXiv preprint arXiv:2509.00190*, 2025.

[40] Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Guojun Yin, and Wei Lin. Promoting efficient reasoning with verifiable stepwise reward, 2025. *URL https://arxiv.org/abs/2508.10293*, 2025.

[41] Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*, 2025.

[42] Kai Zhao, Yanjun Zhao, Jiaming Song, Shien He, Lusheng Zhang, Qiang Zhang, and Tianjiao Li. Saber: Switchable and balanced training for efficient llm reasoning. *arXiv preprint arXiv:2508.10026*, 2025.

A Appendix

A.1 Full Algorithm

Algorithm 1 SWAP

Require: Policy π_θ, dataset $\mathcal{D}$, group size N, weights (β, θ)

for $t = 1, \ldots, T$ **do**
  Sample prompts $x \sim \mathcal{D}$ and rollouts $\{y_i\}_{i=1}^{N} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$
  for each rollout y_i **do**
    Segment into steps $\{s_{i,k}\}_{k=1}^{K_i}$ via token budget M
    Calculate step information gain $\Delta_{i,k}$ **(Eq. 1)**
    Penalty redistribution: compute penalty mass P_i and weight $w_{i,k} \propto \exp(-g_{i,k}/\tau)$
    Step reward $r_{i,k} = \Delta_{i,k} - P_i w_{i,k}$ **(Eq. 3)**
  end for
  Outcome advantage $A_i^{\text{out}} = \dfrac{r_i^{\text{out}} - \mu_{\text{out}}(x)}{\sigma_{\text{out}}(x) + \varepsilon}$
  Normalize step rewards on correct rollouts: $\tilde{r}_{i,k}$ **(Eq. 4)**
  Process advantage $A_{i,t}^{\text{proc}}$ **(Eq. 5)**
  Unified advantage $A_{i,t} = \beta A_i^{\text{out}} + \theta\, \mathbb{I}[r_i^{\text{out}} > 0] A_{i,t}^{\text{proc}}$
  Update θ with clipped GRPO objective using $A_{i,t}$ **(Eq. 7)**
end for

We present the full training procedure of our stepwise adaptive penalization framework in Algorithm 1. The algorithm summarizes the three key components described in the main text: (i) step importance estimation via on-policy log-probability gains, (ii) step-weighted redistribution of trajectory-level length penalties, and (iii) unified outcome–process advantage computation under GRPO.

A.2 Target Answer Instantiation for Progress Estimation

Our framework requires a task-specific target answer $a^\star$ to compute step-level progress, which we instantiate in a straightforward manner across datasets. For mathematical reasoning tasks, we directly use the normalized ground-truth final answer. For multiple-choice QA, we use the correct option and compute the average log-probability over its canonical forms when applicable. For coding tasks, where a single canonical answer may not exist, we instead use key decision points or verifiable solution components (e.g., final outputs, function signatures, or test-passing behavior) as the target signal for progress estimation. This unified design ensures that the progress signal is well-defined, easy to implement, and aligned with the evaluation protocol across all tasks.

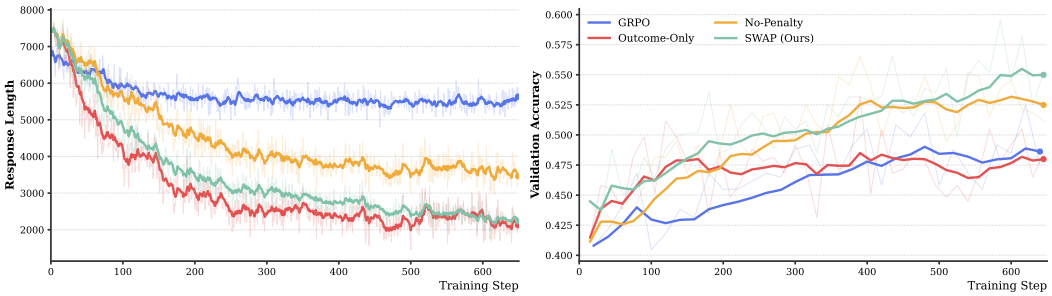

Figure 6: Training dynamics of different model ablations. SWAP achieves the best validation performance, with a final average response length similar to Outcome-Only (length penalty assigned at the trajectory level).

A.3 Training Dynamics

To study how different reward and penalty designs influence optimization, we analyze the training dynamics of response length and validation accuracy shown in Figure 6. We compare standard GRPO, Outcome-Only (outcome reward with a global length penalty), No-Penalty (combined outcome and step rewards without length penalties), and SWAP.

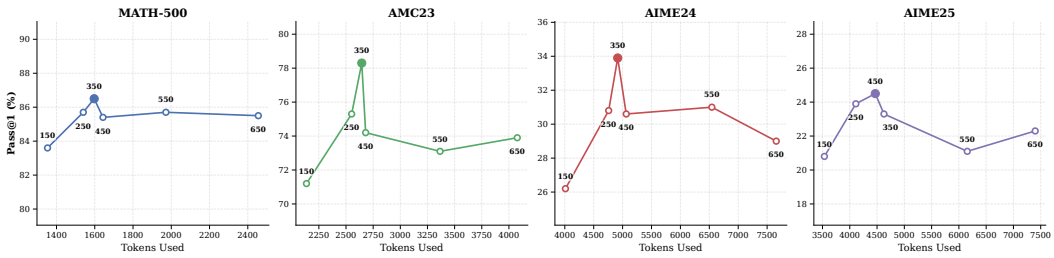

Figure 7: Comparison of Pass@1 accuracy vs. step length across four math benchmarks.

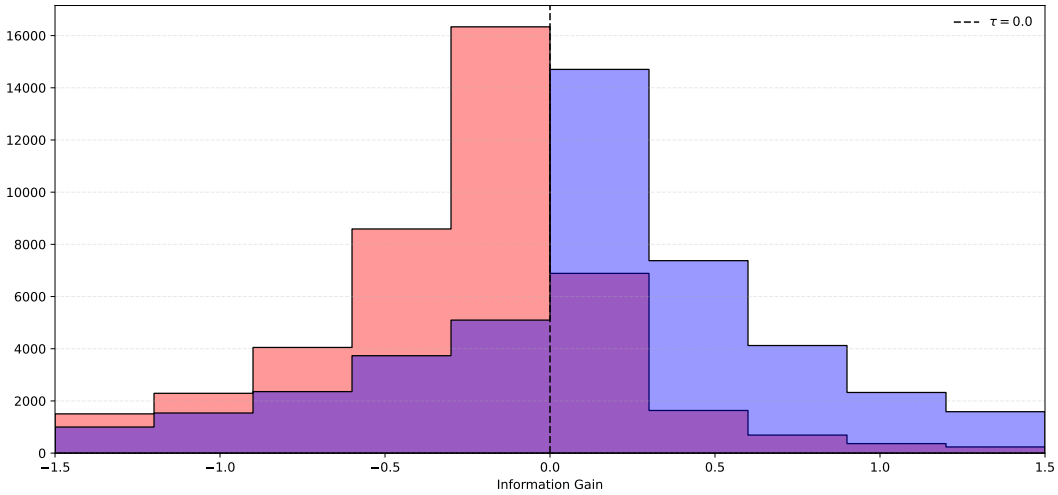

Figure 8: Distribution of step-wise information gain for the original model (red) and SWAP (blue), computed from 50k randomly sampled reasoning steps. After training, SWAP exhibits a clear shift toward positive information gain, indicating that low-importance reasoning steps are suppressed while informative steps are preserved.

Standard GRPO reduces response length slowly and converges to long reasoning traces (around 5,500 tokens), with relatively low validation accuracy. Adding a global length penalty accelerates early length reduction; however, its accuracy remains noticeably lower, indicating that trajectory-level penalties alone encourage indiscriminate compression without improving reasoning quality. In contrast, SWAP steadily reduces response length throughout training while simultaneously achieving the highest validation accuracy. Compared to No-Penalty and GRPO, our model converges to substantially shorter responses with better accuracy, and compared to Outcome-Only, it achieves similar or higher accuracy with significantly fewer tokens. These dynamics demonstrate that combining outcome grounding with progress-aware, step-weighted penalties enables both efficient compression and stable performance improvement during training.

A.4 Effect of Step Size on Step Importance Estimation

We analyze the impact of step size M in token-based step segmentation, which determines the granularity of step importance estimation. Results are shown in Figure 7. When the step size is too small, the response is split into many short steps, leading to excessive penalization of low-gain segments. For example, with $M = 150$, accuracy on MATH-500 drops to 83.6% despite a short average length of 1,353 tokens, compared to 86.5% at $M = 350$. Similar degradation is observed on AMC23 and AIME24, indicating that overly fine-grained steps distort step importance estimation. Conversely, overly large step sizes reduce the resolution of step rewards and weaken length control. On AIME24, increasing M from 350 to 650 increases reasoning length from 4,915 to 7,658 tokens while accuracy decreases from 34.0% to 29.0%. Intermediate chunk sizes provide the best accuracy–efficiency trade-offs: $M = 350$ performs best on MATH-500, AMC23, and AIME24, while AIME25 favors $M = 450$. We therefore use $M = 350$ as the default across datasets.

A.5 Distribution of Step-wise Information Gain

Figure 8 compares the distribution of step-wise information gain between the original model and SWAP, computed from 50k randomly sampled reasoning steps. After training with stepwise reward penalization, the distribution shifts noticeably toward positive information gain. In contrast to the original model, where a large fraction of steps exhibit near-zero or negative gain, SWAP concentrates mass on steps that contribute meaningful progress toward the final answer. This indicates that SWAP effectively suppresses low-utility or redundant reasoning steps, preserving primarily high-impact reasoning transitions.

B Limitations

SWAP relies on a task-specific target answer $a^\star$ to estimate step-level progress, which is most natural for tasks with well-defined and verifiable correctness signals, such as mathematical reasoning and multiple-choice QA. While we also evaluate transfer to coding tasks using verifiable solution components, fully open-ended settings without clear target answers remain outside the scope of this work. Extending the framework to such scenarios would require alternative progress signals (e.g., verifier-based or learned reward models). Nevertheless, SWAP already demonstrates consistent improvements across a diverse set of reasoning tasks, indicating that step-wise penalty allocation is broadly effective in practical settings with well-defined evaluation criteria.

C Broader Impacts

This work aims to improve the efficiency of large reasoning models by reducing unnecessary chain-of-thought length while preserving or improving task accuracy. A positive societal impact is that more efficient reasoning can lower inference cost, latency, and energy consumption, making advanced reasoning models more accessible to researchers and practitioners with limited computational resources. However, these efficiency gains may also have negative societal impacts. Shorter reasoning traces may make model behavior harder to inspect if important intermediate steps are compressed away. Therefore, SWAP should be carefully evaluated before deployment, especially in high-stakes domains such as education, healthcare, finance, and legal decision-making.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the proposed method, its novelty, and the empirical results, and accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses limitations in Appendix B, including the scope of the evaluated models and benchmarks, assumptions behind target-answer-based progress estimation, and the computational cost of step-level reward computation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not include formal theoretical results (e.g., theorems or proofs), focusing instead on empirical analysis.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provide the full algorithm in Appendix A.1, the reward and penalty definitions in Section 3, and the training data, model backbones, evaluation settings, hyperparameters, and hardware details in the Implementation Details of Section 4.1. These details provide the necessary information to reproduce the main experimental results.

Guidelines:

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

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

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

   Answer: [No]

   Justification: The paper uses publicly available datasets and open model families, and provides method, training, evaluation, and hyperparameter details in Sections 3 and 4.1 and Appendix A. However, we do not provide an anonymized code repository or one-click reproduction scripts at the time of submission.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the models, training data, evaluation benchmarks, decoding settings, rollout configuration, key hyperparameters, and hardware setup in Section 4.1, with additional algorithmic and target-answer details in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper reports pass@1 accuracy and average response length across multiple benchmarks in Tables 1–3 and provides distributional analysis in Figure 7, but does not include error bars, confidence intervals, or statistical significance tests for the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Section 4.1 reports the GPU resources used for training, including 8×H100 GPUs for the 1.5B model and 8×H200 GPUs for the 7B model.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research uses publicly available benchmarks, open model families, and standard evaluation protocols, and does not involve human subjects, private user data, or sensitive personal information.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: Appendix C discusses both positive impacts, such as reduced inference cost and improved accessibility of reasoning models, and potential negative impacts, including broader deployment risks and the need for careful evaluation in high-stakes applications.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release a new high-risk pretrained model, image generator, or scraped dataset. The work uses existing public datasets and open model families, and therefore no additional safeguards for responsible release are required.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: Datasets and model families are attributed via citations and standard benchmark references. Use follows the publicly documented releases associated with those works.

Guidelines:

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

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new datasets, code packages, or model checkpoints as new assets at the time of submission.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method has no LLM component; LLMs were used only for editing (e.g., grammar, spelling, word choice) and drafting sections of the paper.

Guidelines:

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