

# LLM-ER: Large Language Model Efficient Reasoning via Balanced Accuracy-Length Control

Anonymous ACL submission

## Abstract

Recent advances in large language models (LLMs) have demonstrated impressive reasoning capabilities, often achieved through prolonged and computationally intensive inference-time deliberation. However, these extended reasoning sequences can lead to redundancy and inefficiency, a phenomenon known as overthinking. This paper introduces a lightweight reward mechanism to a recent reinforcement learning framework to promote efficient reasoning in LLMs by balancing accuracy with brevity. Our approach combines a length-aware reward mechanism with dynamically scheduled accuracy thresholds to mitigate verbosity without sacrificing correctness. Empirical results across six math reasoning benchmarks show that the method significantly reduces output length (over 50%) while preserving or even improving accuracy and semantic quality. Comprehensive reasoning behavior analyses further reveal that the method reduces redundant reasoning strategies. Moreover, our method can refine the structure of LLM inference texts, promoting concise and high-quality reasoning processes.

## 1 Introduction

Recent advances in large language models (LLMs), such as OpenAI o1 (OpenAI, 2024) and DeepSeek-R1 (Guo et al., 2025), have significantly improved performance on complex reasoning tasks by scaling computational effort at inference time. This enables models to produce exploratory reasoning chains resembling internal deliberation, facilitating self-assessment and correction (OpenAI, 2024; Guo et al., 2025; Team, 2024).

Reinforcement learning (RL) has played a central role in enabling such capabilities. During RL training, models exhibit distinct reasoning phases characterized by longer, more intricate outputs and emergent strategies such as self-verification and decomposition (Gandhi et al., 2025). Empirical

findings suggest a strong correlation between reasoning depth and accuracy gains (Zeng et al., 2025; Guo et al., 2025).

Despite their benefits, extended reasoning sequences often introduce inefficiencies due to unnecessary repetition and elaboration—a phenomenon known as overthinking (Chen et al., 2024). This results in increased latency and computational cost with only marginal gains in accuracy. We further analyze this redundancy in Appendix B, comparing several mainstream models—including ChatGPT-4o, Qwen3-235B-A22B, DeepSeek, DeepSeek-R1, and Gemini-2.0-Flash—on 50 representative problems from the Math Dataset.

To mitigate this, recent works explore reward design strategies that balance correctness and brevity (Arora and Zanette, 2025; Aggarwal and Welleck, 2025; Shen et al., 2025; Luo et al., 2025; Yeo et al., 2025), but optimal formulations remain elusive.

This work presents two main contributions. We first introduce a lightweight reward mechanism that can be incorporated in a recent reinforcement learning framework, the Group Relative Policy Optimization (GRPO) (Shao et al., 2024), with a length-aware reward function that significantly shortens model reasoning without degrading accuracy. Secondly, we provide an extensive evaluation across standard benchmarks to assess the trade-offs between response length and reasoning quality, demonstrating the effectiveness and generality of our approach. Our findings reveal:

1. our method can significantly reduce inference length while maintaining accuracy and semantic quality;
2. There is a balanced point in response length where accuracy remains or even increases before it and drops sharply afterward.
3. We find that “subgoal setting” and “verification” behavior contribute primarily to the re-

sponse verbosity; And our method mitigates overthinking by decreasing four reasoning behaviors (see Section 5.2 for details).

4. Fine-tuned models originating from the same base tend to converge toward similarly concise reasoning lengths, suggesting that our method effectively guides models toward minimal yet robust reasoning paths (see Section 5.3 for details)

## 2 Related Work

### 2.1 Scaling Up Inference-Time Compute

Recent advancements in large language models (LLMs), notably represented by OpenAI’s o1 (OpenAI, 2024), highlight the significant benefits of allocating additional computational resources during inference to enhance model reasoning capabilities (Snell et al., 2024). This strategy complements traditional approaches focused on training-time scaling, which emphasize expanding datasets, increasing model parameters, and employing substantial training resources.

Inference-time methods typically broaden the exploration space for candidate solutions, thereby improving the quality of generated outputs. Techniques in this category include structured search methods such as Tree-of-Thoughts (ToT)(Yao et al., 2023), Monte Carlo Tree Search (MCTS)(Tian et al., 2024), and Stream-of-Thought (SoT)(Gandhi et al., 2024); iterative refinement processes like self-refine mechanisms(Madaan et al., 2023; Ferraz et al., 2024); and sampling-based aggregation strategies (Wang et al., 2022; Brown et al., 2024; Team et al., 2025). Additionally, some methods integrate external verification modules to evaluate and rank candidate outputs, further improving final decision quality (Cobbe et al., 2021; Lightman et al., 2023; Li et al., 2024).

### 2.2 Efficient Reasoning

Although increasing inference-time computational resources can improve the reasoning performance of LLMs, it often leads to overly verbose reasoning chains, resulting in higher computational costs and longer inference times (Sui et al., 2025). This “overthinking” phenomenon has motivated research into methods that encourage more concise and efficient reasoning.

To address this, several strategies have been proposed. Model-based approaches, such as reinforcement learning with length-aware rewards, train

models to balance correctness with brevity (see Section 2.3). Supervised fine-tuning on variable-length Chain-of-Thought (CoT) data, as used in Cot-valve (Ma et al., 2025) and TokenSkip (Xia et al., 2025), similarly promotes compact reasoning. Prompt-based methods like Token-Budget (Han et al., 2024) and Chain of Drafts (Xu et al., 2025a) guide models to be concise by explicitly constraining response length. Output-oriented techniques, including Coconut (Hao et al., 2024) and Softcot (Xu et al., 2025b), reduce token overhead by encoding reasoning steps in latent form. Finally, dynamic reasoning frameworks (Liao et al., 2025; Ding et al., 2025) adapt reasoning depth based on real-time feedback to optimize efficiency.

### 2.3 RL with Length Reward Design

Existing works leverage traditional RL optimization techniques (typically policy optimization (PPO) (Schulman et al., 2017)) combined with explicit length-based reward to control the length of Chain-of-Thought (CoT) reasoning. (Arora and Zanette, 2025) introduced a reward schema that prioritizes shorter correct answers, applying traditional policy gradient methods to encourage concise reasoning steps. (Yeo et al., 2025) incorporated a Cosine Reward based on a Dirichlet function and an “exceed length penalty” to manage CoT length growth, thereby enhancing performance stability. Due to the performance impact of CoT length, Kimi k1.5 (Team et al., 2025) integrates a length penalty within its policy optimization framework to improve long CoT activations and facilitate effective model merging. Similarly, (Aggarwal and Welleck, 2025) introduced Length Controlled Policy Optimization (LCPO), a RL method that optimizes for accuracy and adherence to user-specified length constraints, training reasoning language models to produce outputs satisfying a length constraint given in prompts. (Luo et al., 2025) proposed the O1-Pruner, which introduces a Length-Harmonizing Reward combined with a PPO-style loss to optimize reasoning LLMs by effectively shortening CoT length without compromising accuracy. Furthermore, (Shen et al., 2025) developed DAST, a method that fine-tunes reasoning LLMs using a constructed length preference dataset based on a self-defined token-length budget measurement. This measurement is defined as a linear combination of the average length of correct responses and the maximum allowed length.

### 3 Methods

Current language models’ outputs include much redundant content. However, a lightweight and effective length control mechanism is lacking to prevent overthinking. Some prior works either hurt models’ performance while reducing the inference length or do not reduce inference length as much as we do (Aggarwal and Welleck, 2025; Shen et al., 2025; Arora and Zanette, 2025). In this section, we introduce a lightweight, length-controlled reward designed to prevent overthinking without compromising model performance. Unlike most prior works, which only use the length of predictions and reference texts to control inference length, we incorporate validation accuracy directly into the reward function, effectively preserving performance while preventing overthinking during training. We integrate the novel reward into the GRPO training, and our experiment results demonstrate the effectiveness of our reward on multiple models.

#### 3.1 Length Reward

To maintain model performance, it is crucial to incorporate accuracy into the design of the reward function. Specifically, the length reward is activated only when the validation accuracy meets a particular condition; otherwise, the model is encouraged to focus on improving accuracy before optimizing length. The following equations define the length reward for a prediction at the iteration  $i$ :

$$r_{\text{acc}} = \frac{A_{\text{val}}}{A_{\text{target}}}$$

$$r_{\text{len}} = \min(1, \frac{L_{\text{pred}}}{L_{\text{max}}}) \in [0, 1]$$

$$R_{\text{len}} = 1 - \min(r_{\text{acc}}^\beta, r_{\text{len}}) \in [0, 1]$$

where  $A_{\text{val}}$  and  $A_{\text{target}}$  denote the validation accuracy and the dynamically scheduled target accuracy at iteration  $i$ ;  $L_{\text{pred}}$  is the length of the model’s predicted output, and  $L_{\text{max}}$  is the preset maximum allowed length;  $r_{\text{acc}}$  and  $r_{\text{len}}$  represent the normalized accuracy and length, respectively.  $R_{\text{len}}$  is the length reward. A value close to 1 implies that the reward is inactive (no penalty), either because the output is sufficiently short or the model has not yet achieved enough accuracy to trigger length constraints. The hyperparameter  $\beta$  controls the sensitivity of the reward to accuracy. A larger  $\beta$  delays the activation of the length penalty until higher accuracy is achieved, while a smaller  $\beta$  allows earlier enforcement of length constraints. Instead of applying a

hard threshold on accuracy, these formulations use a smooth transition to modulate the length reward, ensuring continuous control that adapts dynamically as the model improves.

#### 3.2 Dynamic Attention to Accuracy and Final Reward

In addition to incorporating the length reward, we adaptively modulate the influence of the raw reward, defined as  $\mathbb{I}(y_{\text{pred}}, y_{\text{gold}})$ , which prioritizes correctness but overlooks text length. To balance performance and brevity, we introduce Accuracy Attention, a mechanism that dynamically adjusts the weight of the raw reward based on the model’s current accuracy. The accuracy attention  $Att_{\text{acc}}$  is computed as:

$$Att_{\text{acc}} = \gamma + (1 - \gamma)(1 - r_{\text{acc}})$$

where  $\gamma$  is the minimum attention to accuracy. As accuracy improves, less emphasis is placed on the raw reward, allowing the length reward to play a greater role.

The final reward is the weighted combination of the raw reward  $R_{\text{raw}}$  and the length reward  $R_{\text{len}}$ .

$$R = Att_{\text{acc}} \cdot R_{\text{raw}} + \alpha \cdot R_{\text{len}}$$

where  $\alpha$  controls the impact of the length reward. Notably, the reward function serves two goals: (1) to guide the model toward accurate predictions when performance is suboptimal, and (2) to promote concise reasoning when accuracy is sufficiently high.

#### 3.3 Dynamic Schedule

The parameter  $\beta$  in the length reward determines when the penalty starts to take effect, but it relies on the target accuracy  $A_{\text{target}}$ . A well-calibrated  $A_{\text{target}}$  is essential to reduce overthinking while preserving model performance. A naive approach is to manually set  $A_{\text{target}}$  and apply the length reward only after the model surpasses a fixed accuracy or training step threshold. However, this method is impractical, as neither the optimal accuracy nor the convergence point is known beforehand. To address this, we propose two dynamic scheduling strategies that adaptively set  $A_{\text{target}}$  during training.

**Exponential Moving Average (EMA).** This method updates the target accuracy by smoothing

it toward the best validation accuracy seen so far:

$$A_{\text{val}}^{\max} = \max_{0 < i \leq t} A_{\text{val},i}$$

$$A_{\text{target},i} = \max \left( \epsilon \cdot A_{\text{target},i-1} + (1 - \epsilon) \cdot A_{\text{val}}^{\max}, A_{\text{val}}^{\max} \right)$$

where  $t$  is the current step, not the number of validation steps, and the  $\epsilon$  controls the inertia of the target. When validation accuracy exceeds the previous target, the target is directly updated to prevent lag.

**Potential Scheduling (PS).** This strategy models the target accuracy as the current best validation accuracy plus a decaying potential:

$$A_{\text{val}}^{\max} = \max_{0 < i \leq t} A_{\text{val},i}$$

$$P_i = \begin{cases} 1 - A_{\text{val},0}, & i = 0 \\ \min(1 - A_{\text{val}}^{\max}, \epsilon \cdot P_{i-1}), & i > 0 \end{cases}$$

$$A_{\text{target},i} = A_{\text{val}}^{\max} + P_i$$

where  $t$  is the same as the above method, and  $\epsilon$  controls how fast the potential decays, smaller values reduce  $P_i$  more aggressively over time. The min function ensures  $A_{\text{target},i} \leq 1$  even when accuracy improves rapidly. This method ensures that  $A_{\text{target}}$  always stays above the best validation accuracy while gradually reducing the potential gap.

## 4 Experiments

### 4.1 Settings

**Models and Datasets** We conduct our training experiments on two commonly used and well-known models: Qwen2.5-Math-7B and DeepSeek-R1-Distill-Qwen-7B, which have shown excellent reasoning ability on various math tasks. Both models are trained on the GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) datasets. Both training set has about 7.5k questions. During our training, we split both training sets in 95%:5% ratio to create validation sets for the training process. We evaluate our models on different reasoning datasets: GSM8k, MATH, AIME24, AMC24, CNMO24, and GPQA.

We fine-tune models using the GRPO methods. The default parameters during the training process are as follows. The values of  $\alpha$  and  $\beta$  are  $10^{-6}$  and 128, respectively. Both values of  $\gamma$  and  $\epsilon$  are 0.9. We choose the first dynamic scheduling method as the default training method. The length limitation of both prompts and generations is 1k tokens,

which is long enough for most of our questions and answers in training datasets. Models are fine-tuned for about 300 steps to reach convergence on both inference length and accuracy. All experiments are conducted on the AWS EC2 platform using 8 NVIDIA H100 GPUs.

**Baselines** To evaluate the superiority of our method, we compare our results against three baselines. (1) Original models: are published models without any further fine-tuning. (2) Original models-RS: are fine-tuned original models by raw scores (0/1 scores). If an answer extracted from the inference text is correct, the raw score is 1. Otherwise, it is 0.

**Metrics** We apply accuracy, the length of inference texts, and CCA (Nayab et al., 2024) as our evaluation metrics. We calculate the average results of 3 runs for all metrics.

### 4.2 Results

**Length Analysis** Table 1 shows the performance of various methods on different models. Notably, results demonstrate that our length reward framework can dramatically prevent overthinking while improving models’ reasoning ability compared to the original model. On the GSM8k and MATH datasets, our framework only needs less than a quarter and a half of the original length to generate correct answers, respectively. Moreover, applying the raw score (0/1 score) during training also causes the model outputs to gradually decrease in most cases. This is mainly because post-training can slightly simplify outputs by removing redundant content. Detailed analysis of the change of inference texts will be discussed in section 5. Furthermore, the training process on the GSM8k dataset is unstable as shown in Figure 2. It is mainly because the dataset is simple, and post-training by raw score as the reward cannot effectively simplify it.

**Evaluation on Datasets** According to recent research, longer LLM outputs, in many contexts, allow the model to explore a broader range of possible solutions and reasoning paths, which is crucial for solving complex problems that require multi-step reasoning (Jiang et al., 2024; Wu et al., 2025). Thus, the harder a question is, the longer it takes LLMs to generate an answer. However, a model with overthinking issues may not be able to evaluate the complexity of questions since it will generate redundant reasoning steps for simple questions.



Models	GSM8k			MATH		
	Accuracy↑	Tokens↓	CCA↑	Accuracy↑	Tokens↓	CCA↑
Qwen2.5-Math	93.21	439.04	26.4	62.27	577.94	24.3
Qwen2.5-Math-RS	97.17	193.73	35.8	73.01	395.31	28.7
Qwen2.5-Math-LR	<b>97.59</b>	<b>97.01</b>	<b>38.9</b>	<b>73.87</b>	<b>288.24</b>	<b>31.6</b>
DS-Distill-Qwen	94.72	455.61	30.7	70.33	1006.94	16.7
DS-Distill-Qwen-RS	97.06	497.88	28.4	<b>82.87</b>	557.12	25.6
DS-Distill-Qwen-LR	<b>97.72</b>	<b>100.58</b>	<b>40.1</b>	82.77	<b>270.51</b>	<b>33.1</b>

Table 1: Performance of models on GSM8k and MATH datasets. Each model is trained and evaluated on the same dataset, and then assessed by three metrics. “RS” indicates that the base model is fine-tuned by raw score. “LR” denotes that length reward is applied to the base model during the training process. “Qwen2.5-Math” and “DS-Distill-Qwen” are the abbreviations of “Qwen2.5-Math-7B” and “DeepSeek-R1-Distill-Qwen-7B”

Models	AIME24		AMC24		CNMO24		GPQA	
	Accuracy↑	Tokens↓	Accuracy↑	Tokens↓	Accuracy↑	Tokens↓	Accuracy↑	Tokens↓
Qwen2.5-Math-LR	26.7	421.51	23.9	395.1	22.2	546.67	30.8	568.01
DS-Distill-Qwen-LR	53.3	2113.08	52.2	1130.42	66.7	2114.31	46.1	2403.78

Table 2: Evaluate models on out-of-distribution math datasets.

Since our models can reliably prevent overthinking while maintaining their performance, we can use them as evaluators to test the complexity of questions in those datasets. Table 2 shows our evaluation results. Due to the complexity of out-of-domain reasoning datasets, the inference length increases dramatically. It demonstrates that “GSM8k” is the simplest dataset, while “GPQA” contains the most complex questions. The “Math” dataset is more complicated than “GSM8k” but simpler than the other three datasets. AMC24 is a special dataset since neither model performs well on it, nor generates long inference texts.

### 4.3 Ablation Study

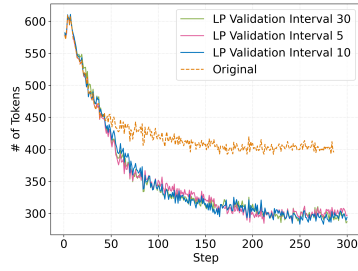
**Hyper Parameters  $\alpha$ ,  $\beta$ ,  $\gamma$ , and  $\epsilon$**  We run all the following experiments following the second dynamic schedule method on Qwen2.5-Math-7B model training on the MATH dataset. The baseline settings are the same as the “Length Penalty” setting in the main result section, which is  $\alpha = 1e^{-6}$ ,  $\beta = 128$ ,  $\gamma = 0.9$ , and  $\epsilon = 0.9$ .

For  $\alpha$ , we have tried the following values:  $10^{-5}$ ,  $5 \times 10^{-6}$ ,  $2 \times 10^{-6}$ ,  $10^{-6}$ , and  $10^{-7}$ . Results are in the Figures 1b and 1e. According to Figures 1b and 1e, the best value of  $\alpha$  is  $10^{-6}$ , which reduces the model overthinking while maintaining the high accuracy. A low  $\alpha$  value, such as  $10^{-7}$ ,

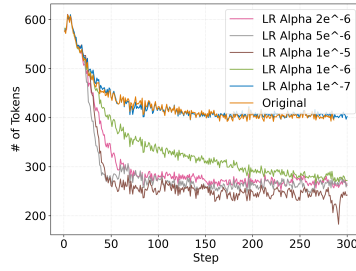
does not help the model prevent overthinking. A high  $\alpha$  value, such as  $10^{-5}$ , can help the model mitigate overthinking but hurt its performance as its accuracy decreases from 73% to 71%, but the length only reduces by around 30 tokens. Moreover, looking through the lines of  $10^{-5}$ , it fluctuates more severely than others. At the end of its training process, both the output length and accuracy drop dramatically and then rise back. It demonstrates the auto-adaptability of our framework. When the accuracy drops abnormally, our dynamic schedule framework helps the model focus on improving accuracy instead of continuing to prevent overthinking.

**Dynamic Schedule** We compare our dynamic scheduling methods under the same hyperparameter settings as shown in the above section. Figure 1c and 1f show that the two methods are similar in effectiveness. However, the first dynamic scheduling method converges faster and more stably at the end of training.

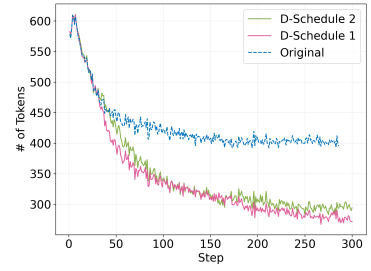
**Validation Interval** Since validation accuracy is an essential part of formulas and controls two parts of the framework, the frequency of obtaining the value may be vital to the training process. We have tried three interval values: 5, 10, and 30. Results are in Figure 1a and 1d. Experiments show that



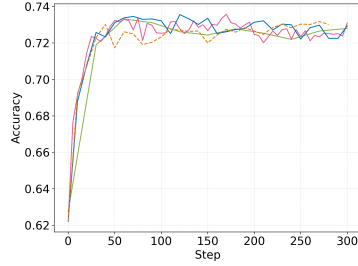
(a) The output length at different val step.



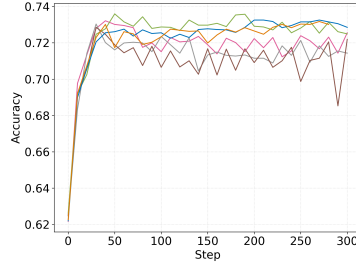
(b) The output length of different alpha value.



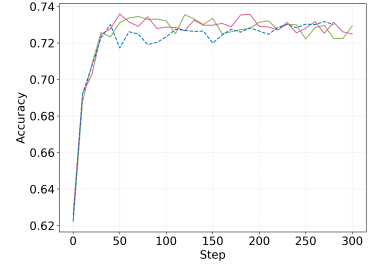
(c) The output length of different dynamic schedule methods.



(d) The output accuracy at different val step.



(e) The output accuracy of different alpha value.



(f) The output accuracy of different dynamic schedule methods.

Figure 1: Ablation analysis of token usage and accuracy under different training configurations. Top row (a–c): Impact on output length (token count) when varying validation step interval (a), length reward weight  $\alpha$  (b), and different dynamic schedule strategy (c). Bottom row (d–f): Corresponding effects on validation accuracy for the same settings.

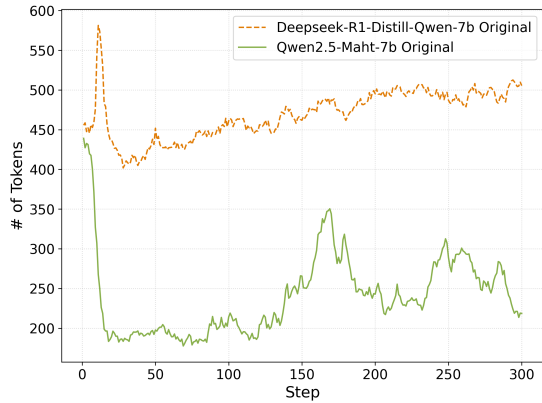


Figure 2: # of inference tokens of models during training with raw scores.

there is almost no difference in choosing different validation intervals. With a small validation interval, the reduction of generation length is smoother than with others. However, it does not affect the final results.

## 5 Analysis

### 5.1 Semantic Quality Analysis

In this section, in addition to standard metrics such as accuracy and response length, we evaluate the semantic quality of truncated responses, providing a

complementary dimension for assessing efficiency in reasoning tasks. Specifically, we present the win rate of our proposed method compared to other baseline in Table 1. This evaluation includes both manual assessments and automated comparisons using GPT-4o (Hurst et al., 2024). During these pairwise comparisons, each model response earns 1 point for a win, 0.5 points for a tie, and 0 points for a loss. The win rate is thus calculated as the proportion of total points earned by our method relative to the baseline method across all comparisons. Given a problem instance  $x$  with corresponding solutions  $y_1$  and  $y_2$ , the evaluation criteria are as follows:

- If  $y_1$  is correct and  $y_2$  incorrect,  $y_1$  is declared the winner;
- If  $y_2$  is correct and  $y_1$  incorrect,  $y_2$  wins;
- If both  $y_1$  and  $y_2$  are incorrect, neither receives points;
- If both solutions are correct, their semantic quality is evaluated by human annotators or GPT-4o to determine the superior response.

The evaluation prompt template, detailed in Appendix C, is specifically designed to mitigate length

Our methods	Opponent	GSM8K		MATH	
		Human	GPT-4o	Human	GPT-4o
Qwen2.5-Math-LR	Qwen2.5-Math	64.4%	55.9%	66.8%	57.2%
	Qwen2.5-Math-RS	60.5%	53.4%	58.3%	53.6%
DS-Distill-Qwen-LR	DS-Distill-Qwen	61.3%	52.8%	67.5%	54.6%
	DS-Distill-Qwen-RS	57.8%	52.5%	61.6%	54.1%

Table 3: Win rates of our proposed method against various baselines settings on GSM8k and MATH benchmark. Evaluations include human annotators and GPT-4o.

bias during semantic quality assessment. To avoid positional bias, the order of  $y_1$  and  $y_2$  is randomized. This evaluation approach is motivated by the fact that, in reasoning tasks, only correct solutions are semantically meaningful; incorrect solutions lack value regardless of their conciseness or perceived faithfulness.

The summarized results from this analysis are presented in Table 3. Our approach consistently achieves a win rate above 50% against all baselines. Moreover, we observe that human evaluations yield a higher win rate compared to GPT-4o assessments, likely due to GPT-4o’s known preference bias towards lengthier responses (Singhal et al., 2023). Despite a slight reduction in raw accuracy scores compared to baseline methods shown in Table 1, our method demonstrates superior semantic quality. These findings validate our method’s ability to generate concise yet semantically richer solutions.

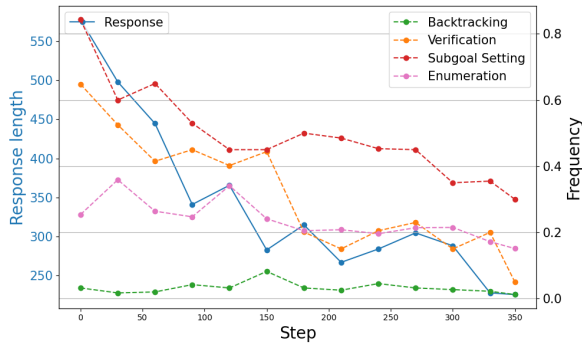


Figure 3: Response length and frequency of reasoning behaviors identified in Qwen2.5-Math-LR on MATH benchmark.

## 5.2 Reasoning Behavior Analysis

To better understand the changes of model’s reasoning patterns throughout the training process, following (Zeng et al., 2025), we adopt the cognitive behavior framework proposed by (Gandhi et al., 2025), leveraging GPT-4o (Hurst et al., 2024) to identify distinct reasoning behaviors: “Backtracking”, “Verification”, “Subgoal Setting”, and

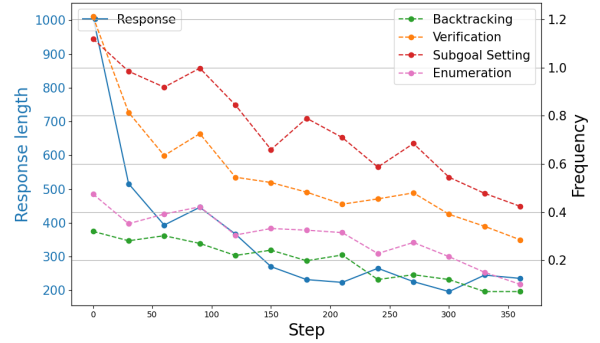


Figure 4: Response length and frequency of reasoning behaviors identified in DS-Distill-Qwen-LR on MATH benchmark.

“Enumeration”. This approach provides greater robustness compared to traditional keyword-based methods (Xie et al., 2025; Yeo et al., 2025), effectively capturing nuanced reasoning patterns even without explicit textual indicators. Definitions and evaluation prompts for these behaviors are provided in Appendix A.

Specifically, following the experiment setting in Table 1, we conduct this reasoning behavior analysis on two models, Qwen2.5-Math-LR and DS-Distill-Qwen-LR on the MATH benchmark. Figures 3 and 4 show the correlation between response length and the frequency of identified reasoning behaviors across different training stages. Our analysis reveals that “Subgoal Setting” and “Verification” are predominant, significantly contributing to reasoning verbosity. As training progresses, we observe a concurrent reduction in the frequency of all identified reasoning behaviors, aligning closely with decreased output lengths. These observations suggest that our strategy for improving model efficiency primarily functions by effectively minimizing redundant reasoning behaviors.

## 5.3 Reasoning Behavior Variations Across Different Models

To understand the impact of length penalties on inference behavior, we compare the final original

Dataset / Question	Final Original Output (Step 270 Checkpoint)	Final Length-Penalized Output (Step 350 Checkpoint, Dynamic Length Penalty, $\alpha = 1e - 6$ , 2nd dynamic schedule method)
<b>Math:</b> <i>What is the sum of the first 10 positive even numbers?</i>	$S = 10/2 * (2 + 20) = 5 * 22 = 110$ Final answer: <span style="border: 1px solid black; padding: 0 2px;">110</span>	<i>First 10 even numbers: 2 to 20. Sum = <math>10/2 \times (2 + 20) = 5 \times 22 = 110</math>.</i>
<b>GSM8K:</b> <i>Kylar wants to buy 16 glasses. One costs \$5, every second costs 60%. What's the total cost?</i>	8 full-price: $8 \times \$5 = \$40$ 8 discounted: $8 \times \$3 = \$24$ Total = \$64. Final answer: <span style="border: 1px solid black; padding: 0 2px;">64</span>	<i>8 full-price: <math>8*5 = 40</math>. 8 discount: <math>8*3 = 24</math>. Total = 64.</i>

Table 4: Comparison between final original model (Step 270 Checkpoint) and length-penalized model (Step 350 Checkpoint, Dynamic Length Penalty,  $\alpha = 1e - 6$ ). The length-penalized outputs remove commentary and structure, preserving only core computations.

model (Step 270 checkpoint) with a variant trained using a dynamic length penalty, a learning rate of  $1e-6$ , and a dynamic scheduling strategy (Step 350 checkpoint). An example is provided in Table 4. While both models yield correct answers, the length-penalized model significantly compresses explanations and eliminates scaffolding, presenting solutions in a more direct and streamlined manner.

This behavior is consistent across both the MATH and GSM8k datasets. In MATH, full derivations and formulaic explanations are replaced by concise inline calculations. In GSM8k, narrative setups are omitted, and responses are reduced to compact arithmetic chains. Structural elements such as framing phrases, code-style reasoning, and boxed outputs are removed entirely.

These shorter, higher-confidence completions often result in improved exact match accuracy. We attribute these gains to the model’s preference for more direct solution paths under length constraints, which suppresses narrative redundancy and promotes structurally compressed, low-variance reasoning.

**Takeaways** Our qualitative analysis of early- vs. mid-training stages and base vs. distilled models (Appendix E) highlights three key insights. First, the effects of length penalties emerge early in training (as soon as step 60), reducing narrative scaffolding while preserving accuracy (Table 5). Second, interpretability is the first aspect to degrade under compression: models begin by eliminating symbolic verification and explanatory framing, even before modifying core logical steps. Third, distil-

lation improves reasoning not merely by shortening outputs, but by restructuring them—correcting conceptual errors, removing redundancy, and yielding more compact and reliable inference (Table 6). These observations suggest that the benefits of compression arise not from truncation alone, but from structural refinement in reasoning behavior.

## 6 Conclusion

This work addresses the inefficiency of overthinking in LLM reasoning by introducing a lightweight reinforcement learning mechanism centered on a length-aware reward. Our method dynamically balances brevity and accuracy by activating length penalties only when performance is sufficient, leading to significant reductions in inference length without sacrificing correctness. Extensive evaluations across multiple math reasoning benchmarks demonstrate that our approach outperforms existing methods in both efficiency and semantic quality. Behavioral analyses reveal that improvements stem not just from response truncation, but from structural refinement—reducing redundant reasoning behaviors like subgoal setting and verification. Moreover, our method generalizes across models and datasets, suggesting its potential as a practical framework for training more efficient and interpretable LLMs.

Future work may explore extending this reward design to other domains and tasks, as well as integrating behavior-level feedback for more fine-grained control over reasoning styles.



**Limitation** Our approach is mainly tested on math benchmarks and its generalization to tasks needing richer language, such as commonsense reasoning or dialog, is unverified. The method’s effectiveness is sensitive to hyperparameter choices, and does not dynamically adjust reasoning depth based on task complexity, which may limit adaptability. Additionally, evaluation partly depends on GPT-4o, which may introduce bias despite mitigation efforts. These limitations highlight the need for future work on adaptive reward schemes, broader domain coverage, and user-controllable verbosity.

## References

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

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

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, et al. 2025. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*.

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2024. Llm self-correction with decrim: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. *arXiv preprint arXiv:2410.06458*.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. 2024. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*.

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

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

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. *arXiv preprint arXiv:2103.03874*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, et al. 2024. Technical report: Enhancing llm reasoning with reward-guided tree search. *arXiv preprint arXiv:2411.11694*.

Ruosen Li, Ziming Luo, and Xinya Du. 2024. [Fg-prm: Fine-grained hallucination detection and mitigation in language model mathematical reasoning](#).

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.

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

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

678	Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler	Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le,	731
679	Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon,	Ed Chi, Sharan Narang, Aakanksha Chowdhery, and	732
680	Nouha Dziri, Shrimai Prabhumoye, Yiming Yang,	Denny Zhou. 2022. Self-consistency improves chain	733
681	et al. 2023. Self-refine: Iterative refinement with	of thought reasoning in language models. <i>arXiv</i>	734
682	self-feedback. <i>Advances in Neural Information Pro-</i>	<i>preprint arXiv:2203.11171</i> .	735
683	<i>cessing Systems</i> , 36:46534–46594.		
684	Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea	Yuhao Wu, Yushi Bai, Zhiqing Hu, Shangqing Tu,	736
685	Saracino, Giorgio Buttazzo, Nicolamaria Manes, and	Ming Shan Hee, Juanzi Li, and Roy Ka-Wei Lee.	737
686	Fabrizio Giacometti. 2024. Concise thoughts: Impact	2025. Shifting long-context llms research from input	738
687	of output length on llm reasoning and cost. <i>arXiv</i>	to output. <i>arXiv preprint arXiv:2503.04723</i> .	739
688	<i>preprint arXiv:2407.19825</i> .		
689	OpenAI. 2024. <a href="#">Learning to reason with llms</a> . Accessed:	Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie	740
690	2025-04-30.	Wang, and Wenjie Li. 2025. Tokenskip: Control-	741
		lable chain-of-thought compression in llms. <i>arXiv</i>	742
		<i>preprint arXiv:2502.12067</i> .	743
691	John Schulman, Filip Wolski, Prafulla Dhariwal,	Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo,	744
692	Alec Radford, and Oleg Klimov. 2017. Proxi-	Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhi-	745
693	mal policy optimization algorithms. <i>arXiv preprint</i>	rong Wu, and Chong Luo. 2025. Logic-rl: Un-	746
694	<i>arXiv:1707.06347</i> .	leashing llm reasoning with rule-based reinforcement	747
		learning. <i>arXiv preprint arXiv:2502.14768</i> .	748
695	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu,	Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng	749
696	Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan	He. 2025a. Chain of draft: Thinking faster by writing	750
697	Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath:	less. <i>arXiv preprint arXiv:2502.18600</i> .	751
698	Pushing the limits of mathematical reasoning in open		
699	language models. <i>arXiv preprint arXiv:2402.03300</i> .	Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan	752
		Miao. 2025b. Softcot: Soft chain-of-thought	753
700	Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wen-	for efficient reasoning with llms. <i>arXiv preprint</i>	754
701	jing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and	<i>arXiv:2502.12134</i> .	755
702	Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-		
703	thinking for large reasoning models. <i>arXiv preprint</i>	Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran,	756
704	<i>arXiv:2503.04472</i> .	Tom Griffiths, Yuan Cao, and Karthik Narasimhan.	757
705	Prasann Singhal, Tanya Goyal, Jiacheng Xu, and	2023. Tree of thoughts: Deliberate problem solving	758
706	Greg Durrett. 2023. A long way to go: Investi-	with large language models. <i>Advances in neural</i>	759
707	gating length correlations in rlhf. <i>arXiv preprint</i>	<i>information processing systems</i> , 36:11809–11822.	760
708	<i>arXiv:2310.03716</i> .		
709	Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Ku-	Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neu-	761
710	mar. 2024. Scaling llm test-time compute optimally	big, and Xiang Yue. 2025. Demystifying long	762
711	can be more effective than scaling model parameters.	chain-of-thought reasoning in llms. <i>arXiv preprint</i>	763
712	<i>arXiv preprint arXiv:2408.03314</i> .	<i>arXiv:2502.03373</i> .	764
713	Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu	Weihaio Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Ke-	765
714	Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu,	qing He, Zejun Ma, and Junxian He. 2025. Simplertl-	766
715	Andrew Wen, Hanjie Chen, Xia Hu, et al. 2025.	zoo: Investigating and taming zero reinforcement	767
716	Stop overthinking: A survey on efficient reason-	learning for open base models in the wild. <i>arXiv</i>	768
717	ing for large language models. <i>arXiv preprint</i>	<i>preprint arXiv:2503.18892</i> .	769
718	<i>arXiv:2503.16419</i> .		
719	Kimi Team, Angang Du, Bofei Gao, Bowei Xing,		
720	Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun		
721	Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025.		
722	Kimi k1.5: Scaling reinforcement learning with llms.		
723	<i>arXiv preprint arXiv:2501.12599</i> .		
724	Qwen Team. 2024. <a href="#">Qwq-32b-preview</a> . Accessed: 2025-		
725	04-30.		
726	Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian		
727	Yu, Lei Han, Haitao Mi, and Dong Yu. 2024. Toward		
728	self-improvement of llms via imagination, search-		
729	ing, and criticizing. <i>Advances in Neural Information</i>		
730	<i>Processing Systems</i> , 37:52723–52748.		

## A Reasoning Behavior Analysis

(Gandhi et al., 2024) identify four core cognitive behaviors displayed by language models:

- (1) **Verification**: systematically checking intermediate results for correctness;
- (2) **Backtracking**: detecting errors mid-generation and explicitly revising prior steps;
- (3) **Subgoal setting**: decomposing a complex problem into smaller, tractable sub-tasks;
- (4) **Enumeration**: exhaustively considering multiple cases or possibilities when searching for a solution.

We substitute “Enumeration” for the original “Backward Chaining,” which is less pertinent to mathematical-reasoning tasks.

### Prompt Template for Identifying and Analyzing Reasoning Behaviors

Below is a chain-of-reasoning generated by a Language Model when attempting to solve a math problem. Evaluate this chain-of-reasoning to determine whether it demonstrates beneficial problem-solving behaviors that deviate from typical linear, monotonic reasoning patterns commonly observed in language models.

```
<start_of_reasoning>
{input}
<end_of_reasoning>
```

Specifically, actively identify and emphasize beneficial behaviors such as:

- **(1) Backtracking**: Explicitly revising approaches upon identifying errors or dead ends (e.g., *"This approach won't work because..."*).
- **(2) Verification**: Systematically checking intermediate results or reasoning steps (e.g., *"Let's verify this result by..."*).
- **(3) Subgoal Setting**: Breaking down complex problems into smaller, manageable steps (e.g., *"To solve this, we first need to..."*).
- **(4) Enumeration**: Solving problems by exhaustively considering multiple cases or possibilities.

Additionally, remain attentive to and encourage the identification of other beneficial behaviors not explicitly listed here, such as creative analogies, abstraction to simpler cases, or insightful generalizations.

#### Important:

Clearly specify each beneficial behavior you identify.

Provide explicit examples from the reasoning chain.

If no beneficial behaviors are observed, explicitly return an empty list.

Provide your evaluation clearly, formatted as follows:

```
{
    "behavior": "",
    "example": ""
}
```

## B Preliminary Analysis on Redundant Contents

To better understand the **sources and roles of redundancy** in language model outputs, we analyze the reasoning behaviors of five models—**ChatGPT-4o**, **Qwen3-235B-A22B**, **DeepSeek**, **DeepSeek-R1**, and **Gemini-2.0-Flash**—on 50 representative problems from the **Math Dataset**. We categorize their reasoning strategies using four behavioral types introduced by [Gandhi et al. \(2024\)](#): **Subgoal Setting**, **Verification**, **Backtracking**, and **Enumeration**.

**Subgoal Setting** emerges as the most prevalent behavior across models. All models—except Gemini—systematically break problems into smaller intermediate steps (e.g., calculating individual item costs before comparing totals). While this decomposition supports interpretability and correctness, it often introduces structural redundancy. Repetitive cues such as “Next, subtract 5 from the result” or “We solve for  $B$ ” restate information already evident from accompanying equations. This pattern is especially prominent in ChatGPT-4o and DeepSeek, which tend to be pedagogical. In contrast, Qwen3-235B-A22B adopts a more concise presentation, omitting such narrative scaffolding while still producing accurate solutions.

**Verification** is commonly observed in DeepSeek and DeepSeek-R1, where models re-calculate or re-assert intermediate values even after computing the final answer. For example, solving  $99 = B - 5$  after already establishing  $B = 104$ . Although this may signal confidence or thoroughness, it contributes limited functional value and increases output length. ChatGPT-4o occasionally includes such verification steps, while Qwen consistently avoids them, preferring direct and economical conclusions. This distinction highlights a trade-off between perceived diligence and actual informativeness.

**Backtracking** appears only weakly in DeepSeek-R1, which occasionally revisits earlier logic after reaching a conclusion. However, no model demonstrates genuine error-driven backtracking with correction, suggesting limited adaptive reasoning in these deterministic tasks.

**Enumeration** is absent across all outputs, which aligns with the single-solution structure of the problems analyzed.

We distinguish between two types of redundancy:

- *Functionally useful redundancy*, such as goal decomposition and verification, which can aid user comprehension.
- *Stylistic redundancy*, including informal fillers and verbose narration, which inflate output length without improving clarity.

Gemini’s responses contain the most stylistic redundancy, frequently using conversational fillers like “Let’s add those up!” that contribute little to reasoning transparency. Qwen, in contrast, is the most efficient: it minimizes both types of redundancy while maintaining correctness and completeness.

Overall, all five models exhibit redundancy through restatements, over-explanation, or narratively padded transitions. These findings support the case for *selective pruning and reformatting*—removing unnecessary scaffolding and repetition—to streamline outputs without compromising accuracy or interpretability.

An annotated example illustrating these behaviors is provided below.

### B.1 Problem Prompt

#### Word Problem

Gloria is shoe shopping when she comes across a pair of boots that fit her shoe budget. However, she has to choose between the boots and two pairs of high heels that together cost five dollars less than the boots. If one pair of heels costs \$33 and the other costs twice as much, how many dollars are the boots? Let’s think step by step and output the final answer within `\boxed{ }`.



## Qwen Output

We are given:

- Gloria can either buy a **pair of boots** or **two pairs of high heels**. - The **total cost of the two pairs of heels is \$5 less than the cost of the boots**. - One pair of heels costs \$33, and the other costs twice as much.

**Step 1:** Total cost of heels:

$$33 + (2 \times 33) = 33 + 66 = 99$$

**Step 2:** Cost of boots:

$$99 + 5 = \boxed{104}$$

## Qwen Output

We are given:

- Gloria can either buy a **pair of boots** or **two pairs of high heels**.
- The **total cost of the two pairs of heels is \$5 less than the cost of the boots**.
- One pair of heels costs \$33, and the other costs **twice as much**.

**Step 1: Define variables**

Let:

- $B$  = cost of the **boots**
- The first pair of heels = \$33
- The second pair of heels =  $2 \times 33 = \$66$

So, total cost of both pairs of heels =

$$33 + 66 = 99$$

We're told this is \$5 less than the boots, so:

$$B = 99 + 5 = 104$$

**Final Answer:**

The boots cost **\$104**.

ChatGPT-4o Output

ChatGPT Output

Let's solve the problem **step by step**.

**Step 1: Understand the problem**

Gloria finds a pair of **boots** that cost a certain amount.  
She also considers buying **two pairs of high heels** instead.

- The **first pair** of heels costs \$33.
- The **second pair** of heels costs **twice as much**:

$$2 \times 33 = 66 \text{ dollars}$$

- The **total cost of the two heels** is:

$$33 + 66 = 99 \text{ dollars}$$

- The **boots cost \$5 more** than the two pairs of heels:

$$\text{Boots} = 99 + 5 = 104 \text{ dollars}$$

Final Answer:

104

Gemini-2.0 Output

Gemini Output

Let's break this down step by step:

- 1. Cost of the more expensive heels:** One pair of heels costs \$33, and the other costs twice as much. So, the more expensive heels cost  $\$33 \times 2 = 66$ .
- 2. Total cost of the two pairs of heels:** To find the combined cost of both pairs of heels, we add their individual prices:  $\$33 + 66 = 99$ .
- 3. Cost of the boots:** The problem states that the two pairs of high heels together cost five dollars less than the boots. This means the boots cost \$5 more than the combined cost of the heels. So, the boots cost  $\$99 + 5 = 104$ .

Therefore, the boots cost **104** dollars.

## DeepSeek Output

Let's determine the cost of the boots step by step.

**1. Identify the Costs of the High Heels:**

- **First pair of heels:** \$33
- **Second pair of heels:** Twice as much as the first pair.

$$2 \times \$33 = \$66$$

**2. Calculate the Total Cost of Both Pairs of Heels:**

$$\$33 \text{ (first pair)} + \$66 \text{ (second pair)} = \$99$$

**3. Determine the Cost of the Boots:**

- The problem states that the two pairs of heels together cost **\$5 less** than the boots.
- Let  $B$  be the cost of the boots.

$$\$99 = B - \$5$$

$$B = \$99 + \$5 = \$104$$

**Final Answer:**

104
-----

## C Win Rate Analysis

Three criteria helps GPT-4o keep a focus on instruction following and semantic quality while avoiding length bias:

### Response Semantic Quality Comparison Template

Given the following instruction and two candidate responses:

Instruction: {}

Response 1: {}

Response 2: {}

Please evaluate which response better follows the instruction based on the following criteria:

1. Adherence to the instruction.
2. Overall semantic quality.
3. Conciseness: the response should **avoid** redundancy, unnecessary verbosity, or inclusion of irrelevant information.

Based on your evaluation, respond in the format:

*'Response 1 is better than Response 2', 'Response 2 is better than Response 1', or 'Response 1 is equal to Response 2'.*

## D More Ablation Study

For  $\beta$ , we have tried the following values: 8, 32, and 128. Results are in the figures ?? and ?. The  $\beta$  controls when the length reward intervenes in the training process. With a low  $\beta$  value, such as 8, the length reward starts being effective early as the validation accuracy reaches around 45% of the target accuracy. According to the ??, early engagement in the training process can reduce the generation length more, but the model's performance also decreases from 73% to 71%. With a high  $\beta$  value, such as 128, the length reward does not affect the training process until the validation accuracy reaches 95% of the target accuracy. The validation accuracy process also shows the model's performance is even better than the "Original" training process. In our framework, our goal is to maintain the model's performance while preventing overthinking. Thus, a high value is preferred in this case.

## E More Qualitative Analysis

### E.1 When Does the Length Penalty Start Working? A Step 30 vs Step 60 Analysis

To investigate when the length penalty begins to meaningfully affect model behavior, we compare inference outputs at step 30 and step 60 under Dynamic Length Penalty settings (1e-6 learning rate, dynamic schedule) on both the Math and GSM8K datasets. These checkpoints are chosen to align with the transition period of the d-schedule mechanism, which gradually increases the weight of the length penalty during post-training. By step 60, we already observe clear signs of compression: outputs are more concise, pedagogical scaffolding is reduced, and verbose explanation structures are stripped away. Table 5 illustrates this behavioral shift with representative examples.

At step 30, the model still retains a "teacher-like" reasoning style. For example, in response to a math problem asking for the sum of the first 10 positive even numbers, the model describes the arithmetic sequence, defines variables explicitly, applies the formula  $S = \frac{n}{2}(a + l)$ , and concludes with a boxed answer. In contrast, the step 60 output removes the setup entirely and provides the final arithmetic chain directly: *"Sum = 10/2 × (2 + 20) = 110. #### 110"*. A similar transition is observed in the GSM8K example: at step 30, the model walks through glass pricing logic with Python-like code and a `print()` statement for validation; by step 60, this is reduced to three compact sentences with no code, commentary, or explanation.

These early changes follow a consistent pattern. Narrative framing—phrases like "Let's solve this by..." or "Now compute..." — are dropped first. Embedded symbolic or code-based verification disappears



Dataset / Question	Step 30 Checkpoint Output	Step 60 Checkpoint Output
<b>Math:</b> <i>What is the sum of the first 10 positive even numbers?</i>	Detailed explanation: defines sequence, uses formula $S = \frac{n}{2}(a + l)$ , explains each term, then computes: $S = 10/2 * (2 + 20) = 5 * 22 = \boxed{110}$ Includes full derivation and justification.	Condensed version: <i>"First 10 even numbers: 2 to 20. Sum = <math>10/2 \times (2 + 20) = 5 \times 22 = 110</math>. #### 110"</i> No sequence explanation or setup.
<b>GSM8K:</b> <i>Kylar wants to buy 16 glasses. One costs \$5, every second costs 60%.</i>	Full arithmetic + Python code: <pre>cost_first = 5, cost_second = 5 * 0.6 total_cost = (cost_first + cost_second) * 8 print(total_cost) # Output: 64.0</pre> Explains logic of pairing and validation.	Shortened to only arithmetic steps: <i>"8 cheaper glasses cost <math>8 * \\$3 = \\$24</math>. 8 full-price glasses cost <math>8 * \\$5 = \\$40</math>. Total = 64. #### 64"</i> All narrative and code removed.

Table 5: Comparison between early (Step 30) and mid-stage (Step 60) model outputs. Length penalty begins to affect verbosity and explanation format while preserving core reasoning.

shortly after. While the arithmetic logic is preserved, the accompanying verbal scaffolding is eliminated. This compression begins after the d-schedule’s early burn-in phase (typically around step 20–40), as the model starts receiving stronger training signals to minimize token count while retaining correctness. By step 60, the pressure from the length penalty is sufficiently strong to shape model behavior measurably.

In summary, the model begins transitioning to a compressed reasoning style as early as step 60. The resulting outputs are more efficient and better aligned with inference-time brevity goals, but less transparent in their reasoning. This supports the hypothesis that the d-schedule enforces compression gradually, with tangible effects emerging soon after step 30. While answer correctness remains intact, the interpretability of the solution path is the first to be sacrificed.

## E.2 Shorter Does Better: How Distillation Improves Reasoning Quality

Despite a marginal difference of just  $\sim 20$  tokens in final output length, the distilled model Deepseek-R1-Distill-Qwen-7B outperforms the base Qwen2.5-Math-7B by 9% in accuracy (81% vs. 72%). This improvement is not due to additional information but rather better reasoning compression. The distilled model eliminates hedging expressions (e.g., “Let’s try to...”), avoids redundant commentary, and reorganizes computation steps into a more structured and compact form. It also prioritizes early calculation of key values and reuses them directly, reducing cognitive redundancy.

These qualitative differences—though subtle—lead to more robust and readable inference chains. As shown in Table 6, the distilled model demonstrates clearer mathematical reasoning and corrects conceptual errors present in the base model. This illustrates that performance gains under length-penalty and distillation arise from structural refinement, not mere truncation.

Dataset / Question	Base Model Output (Qwen2.5-Math-7B)	Distilled Model Output (Deepseek-R1-Distill-Qwen-7B)
<i>What is the positive difference between 120% of 30 and 130% of 20?</i>	Incorrect due to misinterpreting percentages: “120% of 30 is $0.12 \times 30 = 3.6$ . 130% of 20 is $0.13 \times 20 = 2.6$ . The positive difference is $3.6 - 2.6 = 1$ . <span style="border: 1px solid black; padding: 0 2px;">1</span> ”	Correct and concise: “120% of 30 is $1.2 \times 30 = 36$ . 130% of 20 is $1.3 \times 20 = 26$ . The difference is $36 - 26 = $ <span style="border: 1px solid black; padding: 0 2px;">10</span> .” Accurate numeric logic and no redundant commentary.

Table 6: Comparison of reasoning behavior between the base and distilled models. The base model misinterprets percentage values due to incorrect decimal logic and lacks structural clarity. The distilled model applies correct numeric conversions, removes redundant phrasing, and presents a concise, high-confidence reasoning chain—illustrating how compression improves both accuracy and readability.