

SABER: Switchable and Balanced Training for Efficient LLM Reasoning

Kai Zhao^{1*}, Yanjun Zhao^{1*}, Jiaming Song¹, Shien He¹, Lusheng Zhang¹,
Qiang Zhang^{1,2}, Tianjiao Li^{1†}

¹Bilibili Inc.

²Xi'an Jiaotong University

{zhaokai02, zhaoyanjun01, songjiaming, heshien, zhanglusheng01, zhangqiang, litianjiao01}@bilibili.com

Abstract

Large language models (LLMs) empowered by chain-of-thought reasoning have achieved impressive accuracy on complex tasks but suffer from excessive inference costs and latency when applied uniformly to all problems. We propose SABER (Switchable and Balanced Training for Efficient LLM Reasoning), a reinforcement learning framework that endows LLMs with user-controllable, token-budgeted reasoning. SABER first profiles each training example’s base-model thinking token usage and assigns it to one of the predefined budget tiers. During fine-tuning, the model is guided by system prompts and length-aware rewards to respect its assigned budget. In parallel, we incorporate no-think examples to ensure the model remains reliable even when explicit reasoning is turned off. SABER further supports four discrete inference modes—NoThink, FastThink, CoreThink, and DeepThink, enabling flexible trade-offs between latency and reasoning depth. Extensive evaluations on math reasoning (MATH, GSM8K), code generation (MBPP), and logical reasoning (LiveBench-Reasoning) demonstrate that SABER achieves high accuracy under tight budgets, graceful degradation, and effective cross-scale and cross-domain generalization. In particular, SABER-FastThink cuts reasoning length by **65.4%** and yields a **3.6%** accuracy gain compared with the base model on the MATH benchmark.

Introduction

Recent advances in large language models (LLMs) (Achiam et al. 2023; Bai et al. 2023; Dang et al. 2025) have significantly improved their ability to handle complex reasoning tasks through explicit step-by-step thinking. Approaches such as Chain-of-Thought prompting (Wei et al. 2023) and test-time reasoning expansion (Snell et al. 2024) allow models to break down problems into intermediate steps before producing final answers. These strategies have proven effective across domains. However, they also introduce new challenges: reasoning traces tend to become excessively long, leading to inflated inference costs and latency. More critically, such reasoning behaviors are often applied uniformly across all problems, regardless of task complexity or user preferences.

*These authors contributed equally.

†Corresponding author.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

This mismatch between reasoning depth and task requirement has led to what is increasingly recognized as the *overthinking problem*, where LLMs generate unnecessarily elaborate reasoning even for trivial inputs (Aggarwal and Welleck 2025; Han et al. 2025). For instance, instead of directly answering “What is 1 + 1?”, some models may explore multiple addition strategies, include irrelevant justifications, and consume tens of times more tokens than needed. This not only slows down response but also increases serving costs, limiting deployment efficiency. Although previous works (Aggarwal and Welleck 2025; Li et al. 2025b; Kimi 2025) have attempted to shorten outputs using instruction tuning, response length control, or reward shaping, these methods typically enforce rigid constraints or rely on task-agnostic heuristics. They lack the ability to dynamically adjust reasoning length based on problem difficulty, or to give users explicit control over the reasoning process.

To tackle these limitations, we present **SABER** — **Switchable And Balanced Training for Efficient LLM Reasoning**, a reinforcement learning framework that enables language models to reason under explicitly specified modes. Instead of applying a uniform constraint, SABER first analyzes the base model’s output to estimate the reasoning effort required for each sample, then assigns a tiered token budget that reflects task difficulty. During RL training, the model is guided to respect its target budget using system prompts and length-aware reward shaping. This formulation encourages efficient reasoning on simple inputs while preserving long-form reasoning capability for harder examples.

In particular, SABER defines four discrete reasoning modes—*NoThink*, *FastThink*, *CoreThink*, and *DeepThink*, which allow explicit control over reasoning granularity at inference time. One key feature of SABER is its unified treatment of both thinking-enabled and thinking-disabled modes. While prior works assume reasoning is always active, real-world applications often require immediate responses without thinking process. We explicitly incorporate a curated set of thinking-disabled examples to preserve performance under this setting. As a result, SABER can gracefully support both thoughtful and direct response styles within a single model, reducing the performance gap between both thinking and no-thinking modes. Additionally, these modes are not only interpretable and user-controllable, but also generalize well across domains. Unlike previous works (Li et al.

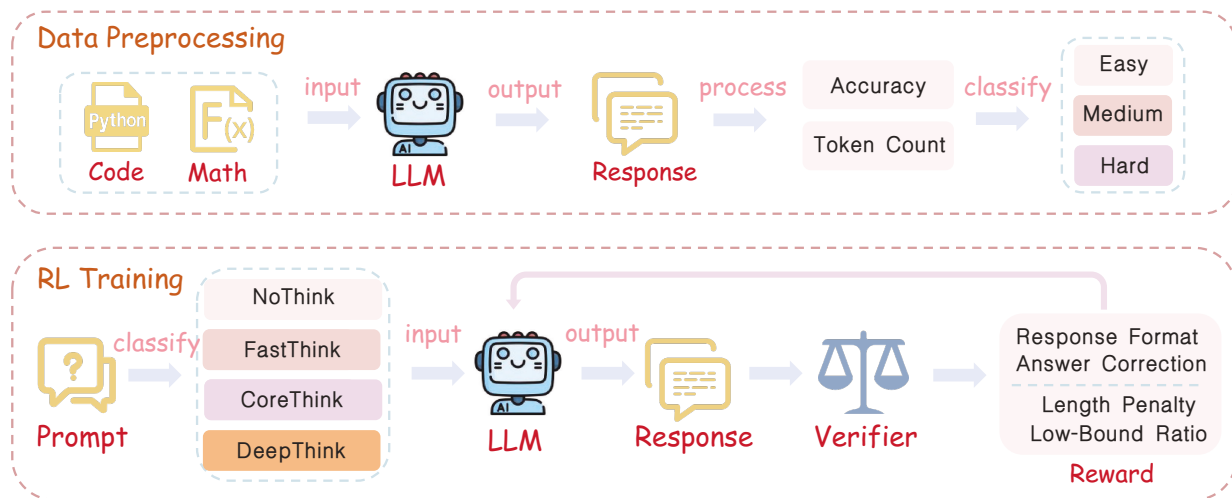


Figure 1: Overview of the SABER framework. The upper part illustrates the data preprocessing pipeline, where reasoning budget is estimated via base model inference and used to categorize training samples by difficulty (Easy / Medium / Hard). The lower part shows the RL training process, where mode-specific prompts guide the model to produce responses, which are then evaluated with multi-part rewards based on format correctness, answer accuracy, and length alignment.

2025b; Huang et al. 2025a) that focus solely on the math reasoning task, SABER trains in both math reasoning and code generation tasks. We further show that SABER can scale to larger models and that the learned reasoning behaviors transfer effectively to the unseen logical reasoning task, highlighting the strong cross-scale and cross-domain generalization of SABER.

Furthermore, the training process of SABER is highly efficient and stable. While many prior RL-based efficient reasoning methods rely on supervised fine-tuning (SFT) as a necessary warm start, SABER can be directly optimized via reinforcement learning from a distilled base model. Its structure-aware design, curriculum token budget, and multi-part reward formulation allow it to converge quickly without pre-training overhead. The main contributions of this work are summarized as follows:

- We propose a unified framework SABER that enables **efficient** and **stable** optimization of long-thinking language models, achieving both high efficiency and training stability through budget-based data grouping, curriculum-style degradation, and reward constraints.
- SABER supports four reasoning modes — NoThink, FastThink, CoreThink, and DeepThink, allowing users to explicitly control the model’s reasoning depth. This unified and switchable design accommodates diverse usage scenarios, from minimal-latency response to complex reasoning.
- Comprehensive experiments on mathematical reasoning, code generation, and logical reasoning benchmarks demonstrate that SABER maintains strong performance under tight token budgets while enabling graceful degradation and broad generalization across tasks.

Related Works

Overthinking in Large Reasoning Models. Several studies (Sui et al. 2025; Feng et al. 2025; Hou et al. 2025; Lu et al. 2025; Zhuang, Wang, and Sun 2025; Lee, Che, and Peng 2025; Zhao et al. 2023a, 2025; Ren et al. 2025) explore compressing chains-of-thought while preserving the benefits of reinforcement learning (RL) fine-tuning. Early work (Wu et al. 2025a) demonstrates that combining a “long-thinker” policy with a concise “answer-only” policy results in a model that maintains accuracy but generates fewer tokens. Mixed distillation (Chenglin et al. 2024) further refines this by distilling both detailed and brief rationales from a powerful teacher. Kimi Long2Short (Kimi 2025) combines these approaches, achieving significant speed-ups while maintaining performance on mathematical tasks. L1 (Aggarwal and Welleck 2025) extends the concept by conditioning on a token budget during training, allowing the model to adjust its reasoning length based on accuracy and latency requirements.

Thinking Budget in Large Reasoning Models. Other works incorporate the thinking budget directly into the optimization objective (Shen et al. 2025; Wu et al. 2025b; Li et al. 2025a; Jiang et al. 2025; Fan et al. 2025; Yuan et al. 2024; Zhao et al. 2023b). SelfBudgeter (Li et al. 2025b) introduces a special token prompting the model to predict its budget, with RL rewards balancing correctness and budget constraints. Token-Budget-Aware LLM Reasoning (TALE) (Han et al. 2025) uses supervisor-provided budgets to encourage early stopping when the answer is clear. AdaCtrl (Huang et al. 2025a) goes further with a two-stage approach, which assesses difficulty and adapts the budget accordingly, enabling faster reasoning. Adaptive Length Penalty (ALP) (Xiang et al. 2025) dynamically adjusts penalties, offering more tokens only when necessary.

Method

In this section, we describe our three-stage approach for SABER framework that supports different user-controlled reasoning modes. We first present how we extract and categorize base model’s thinking token statistics into discrete budget levels. Next, we detail our sample grouping and stability control mechanisms. We then introduce the injection of no-think samples to enable direct-answer mode. Finally, we formalize our reinforcement learning objective, including reward components that jointly enforce format, answer correctness, length alignment, and anti-hacking constraints. Figure 1 summarizes the framework of SABER.

Thinking Collection and Budget Categorization

The design of thinking budget is the heart of SABER. If every example’s target budget were set to the same value, problems requiring few thinking tokens would never incur a length penalty and thus would fail to learn how to switch reasoning modes, while problems requiring many thinking tokens would be continuously penalized and quickly collapse in performance. To address this, SABER applies a per-example budget calibration. First, we run the base model over the entire training set and record the number of tokens generated between `<think>` and `</think>`. Based on the observed distribution and empirical judgments, we partition examples into three difficulty tiers—128 (easy), 4,096 (medium) and 16,384 (hard), and assign each example a corresponding target budget as follows:

- If an example’s thinking token count is less than 128, we still set its target budget to 128.
- If it lies between 128 and 4,096, we set its target budget to 128.
- If it lies between 4,096 and 16,384, we set its target budget to 4,096.
- If it exceeds 16,384, we impose no upper bound, allowing the model full freedom of reasoning on these challenging tasks.

We then prepend each prompt with a system message, as shown in Figure 2, where XXX is replaced by the example’s target budget and DeepThink mode has no target budget.

This tiered downgrade strategy offers two key advantages: (1) it ensures that more training examples incur a length penalty from the outset, thereby efficiently teaching the model to switch between reasoning modes; (2) it respects the inherent variation in reasoning lengths across examples, enabling smooth transitions between adjacent modes and balancing training efficiency with stability.

Stabilizing Mode Transition with Penalties

Applying length penalties uniformly to all training examples from the beginning of fine-tuning can introduce two key issues. First, frequent and abrupt switching between reasoning modes may destabilize training. Second, the model may become overly biased toward generating shorter reasoning traces, leading to underthinking and potential performance degradation. To mitigate these risks, we introduce two complementary strategies that enable smoother and more stable mode transitions.

DeepThink

```
System message: [original sys]
Question: [original prompt]
Answer: <think> ... </think> [response]
```

CoreThink / FastThink

```
System message: [original sys] Your reasoning process
between <think> and </think> should be STRICTLY
UNDER XXX tokens.
Question: [original prompt]
Answer: <think> ... </think> [response]
```

NoThink

```
System message: [original sys] Respond directly without
internal reasoning:\n <think></think>\n\n [response here]
Question: [original prompt]
Answer: [response]
```

Figure 2: Templates for different modes of SABER.

Accuracy-Based Partitioning of Training Data. We evaluate the base model’s ability to answer each training sample correctly. Among the examples that base model fails to solve (approximately 40% of the corpus), we apply two treatments:

- Half are retained at their original budget level, thereby reducing their exposure to length penalties.
- The other half are assigned no target budget at all, allowing unrestricted reasoning during training.

Only the remaining 60% examples the base model can answer correctly are subject to the downgrade procedure. This partitioning scheme ensures that length constraints are applied more conservatively at the early stage of training, promoting stable learning of reasoning-mode switching.

Lower-Bound Ratio Constraint to Prevent Reward Hacking. To avoid degenerate behavior where the model aggressively shortens its reasoning just to minimize penalties, we introduce an additional lower-bound constraint on the generated reasoning length. Specifically, we require that the generated think-token counts t_{gen} remains within a certain proportion of the base model’s counts t_{base} . Formally, we enforce the following constraint:

$$0.2 \cdot t_{\text{base}} \leq t_{\text{gen}} \leq 1.2 \cdot t_{\text{base}}.$$

This range ensures that the model maintains sufficient reasoning content while respecting the target budget, effectively discouraging reward hacking through excessive brevity.

User-Controlled No-Think Mode

In real-world applications, users may sometimes prefer to receive direct answers without any intermediate reasoning (Xiang et al. 2025; Chen et al. 2024). However, directly disabling the reasoning component in a long-thought model without any targeted adaptation often leads to a severe drop in performance. This highlights the necessity of explicitly training the model to handle such no-think scenarios.

To this end, we augment the training corpus with a subset of specially constructed no-think examples. We find that

even a modest proportion of such data substantially improves the model’s compatibility with no-think mode, mitigating performance degradation when reasoning is intentionally skipped. Each no-think example is constructed by manually appending a minimal reasoning block to the input, as shown in Figure 2. This format explicitly instructs the model to bypass the `<think>` span and generate a response immediately. By learning from these examples, the model acquires the ability to gracefully handle user requests for direct answers, offering greater flexibility across use cases.

Direct RL Optimization Without SFT Warmup

Unlike many prior approaches (Huang et al. 2025b,a; Li et al. 2025b; Ma et al. 2025) that require supervised fine-tuning (SFT) as a warm start before reinforcement learning (RL), our SABER framework introduces operations that are naturally aligned with the model’s behavior and training dynamics. As a result, SABER does not require a separate SFT warmup stage and can be directly trained using reinforcement learning from the outset. This significantly simplifies the training pipeline and reduces computational overhead.

In this work, we adopt the widely used Group Relative Policy Optimization (GRPO) algorithm (Shao et al. 2024) to fine-tune the model with structured reward signals. The GRPO objective is defined as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min \left\{ \frac{\pi_{\theta}(o_{i,t} | q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | q, o_{i,<t})} \hat{A}_{i,t}, \right. \right. \right. \\ \left. \left. \left. \text{clip} \left(\frac{\pi_{\theta}(o_{i,t} | q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | q, o_{i,<t})}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}_{i,t} \right) \right. \right. \\ \left. \left. - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \right]$$

Reward design lies at the core of reinforcement learning optimization. As discussed before, the SABER framework incorporates a composite reward signal composed of four distinct components:

$$r = r_{\text{format}} + r_{\text{answer}} + r_{\text{length}} + r_{\text{ratio}}.$$

- **Format Reward:**

$$r_{\text{format}} = \begin{cases} 0, & \text{if format is correct,} \\ -1, & \text{if format is wrong.} \end{cases}$$

The format reward is designed to ensure that the model’s output is syntactically parsable. Specifically, it enforces a structured format of `<think>...</think>...` to clearly separate the model’s internal reasoning content from the final answer.

- **Answer Reward:**

$$r_{\text{answer}} = \begin{cases} 1, & \text{if answer is correct,} \\ 0, & \text{if answer is wrong.} \end{cases}$$

The answer reward evaluates the correctness of the model’s final response. For mathematical problems, the predicted answer is extracted from the `\boxed{\}` and compared with the ground-truth label. For code generation tasks, the reward is computed by extracting the code block enclosed within triple backticks (````\code{}````), executing it in the secure sandbox environment, and verifying it against predefined test cases.

- **Length Penalty:**

$$r_{\text{length}} = \begin{cases} 0, & \text{if } t_{\text{gen}} \leq t_{\text{budget}}, \\ -0.4, & \text{otherwise.} \end{cases}$$

where t_{gen} is the number of thinking tokens generated by the current policy. This length penalty is set to -0.4 to distinguish it from the format reward.

- **Lower-Bound Ratio Penalty:**

$$r_{\text{ratio}} = \begin{cases} 0, & \text{if } 0.2 \cdot t_{\text{base}} \leq t_{\text{gen}} \leq 1.2 \cdot t_{\text{base}}, \\ -0.4, & \text{otherwise.} \end{cases}$$

where t_{base} denotes the number of thinking tokens generated by the base policy. This constraint prevents the model from abusing short or long generations for reward hacking.

By jointly optimizing these rewards, our method achieves precise alignment to user-selected reasoning modes, enforces smooth transitions across different modes, and maintains high answer quality in long-reasoning, short-reasoning and no-reasoning scenarios.

Experiments

In this section, we present the empirical evaluations of the SABER framework to answer the following research questions (RQs):

RQ1: How does SABER compare with existing strong baselines on math reasoning and code generation tasks?

RQ2: Does SABER generalize to larger models and unseen reasoning domains?

RQ3: How important are the individual design choices in SABER?

RQ4: What qualitative differences emerge among the switchable reasoning modes of SABER?

To address these questions, we begin by outlining the experimental setup, including the benchmarks, baselines, and training data. We then present the main results obtained using a 1.5B model on two core tasks: math reasoning (MATH/GSM8K) and code generation (MBPP). Next, we demonstrate cross-scale and cross-domain generalization by applying the same training recipe to a 7B model and to a logical reasoning benchmark (LiveBench-Reasoning). In addition, we conduct ablation studies by removing each component of SABER individually. Finally, we conduct a behavioral analysis of the FastThink, CoreThink, and DeepThink modes, highlighting how each mode affects reasoning depth and answer accuracy.

Model	MATH500		GSM8K		MATH		MBPP	
	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓
Deepseek-R1-Distill-Qwen-1.5B	78.9	8042	82.1	3471	80.6	7866	38.9	5755
L1-Max (1.5B, Num Tokens=512)	-	-	72.9	569	71.2	1533	-	-
L1-Max (1.5B, Num Tokens=3600)	-	-	74.4	633	77.1	1545	-	-
SelfBudgeter (1.5B, slk, Train_Data=90k)	-	-	81.5	662	74.2	919	-	-
SABER - 1.5B - DeepThink	83.2	5353	85.7	1947	85.2	4748	53.7	3010
SABER - 1.5B - CoreThink	82.1	3294	83.1	930	84.3	3045	49.8	1254
SABER - 1.5B - FastThink	81.4	2899	82.5	778	83.5	2719	45.1	942
Deepseek-R1-Distill-Qwen-1.5B-NoThink	65.1	0	61.1	0	65.5	0	30.7	0
SABER - 1.5B - NoThink	76.3	0	78.1	0	76.9	0	44.7	0

Table 1: Main performance comparison for 1.5B size models on math and code benchmark, showing accuracy (Acc), average response length (Len).

Model	MATH500		GSM8K		MATH		MBPP		LiveBench-R	
	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓
Deepseek-R1-Distill-Qwen-7B	91.5	8221	91.4	3402	92.4	7838	59.9	5021	36.4	5619
SABER - 7B - DeepThink	91.9	6265	92.0	2351	92.9	5763	65.8	2426	38.3	4401
SABER - 7B - CoreThink	90.1	2820	91.7	1319	91.2	2557	63.0	1556	32.1	3039
SABER - 7B - FastThink	86.5	1563	90.3	495	87.2	1478	61.1	969	30.6	2166
Deepseek-R1-Distill-Qwen-7B-NoThink	79.1	0	86.4	0	79.7	0	43.2	0	17.9	0
SABER - 7B - NoThink	85.1	0	89.2	0	85.5	0	58.8	0	26.8	0

Table 2: Evaluation of cross-scale and cross-domain generalization for SABER on math, code and logic reasoning tasks, showing accuracy (Acc), average response length (Len).

Experimental Setups

Benchmark. We evaluate on math reasoning and code generation tasks, including the following: (1) MATH500 (Lightman et al. 2023) is a 500-problem slice of the full MATH benchmark, designed to span seven contest domains while remaining compact for quick evaluation. (2) GSM8K (Cobbe et al. 2021) consists of 8.5K crowd-sourced grade-school word problems, each solvable in a few arithmetic steps. (3) MATH (Hendrycks et al. 2021) scales up to 12.5K AMC/AIME-style competition questions with full step-by-step solutions covering algebra, geometry, combinatorics, and more. (4) MBPP (Austin et al. 2021) contains 974 programming challenges designed to test LLMs’ ability to generate code that solves algorithmic problems. (5) LiveBench-Reasoning, a subset of LiveBench (White et al. 2024), is designed to evaluate the model’s logical reasoning ability and contains 200 complex logic puzzles.

Baseline. We use DeepSeek-R1-Distill-Qwen-1.5B as our base model for the main experiment. This model is specifically distilled from DeepSeek-R1 and achieves new state-of-the-art results among LLMs of similar scale. We choose the 1.5B model size for comparison, as many related works in this domain adopt the same scale, making it a standard reference point for fair evaluation. To assess the effectiveness of SABER, we also compare it against two related methods: (1) L1 (Aggarwal and Welleck 2025), which proposes length-controlled policy optimization to produce outputs that ad-

here to strict length constraints specified in the prompt. (2) SelfBudgeter (Li et al. 2025b), which autonomously predicts the required token budgets for reasoning and effectively adheres to self-imposed constraints. We further validate SABER on a larger 7B model. However, due to the absence of published results from the two baseline methods at this scale, we only compare against the base model, DeepSeek-R1-Distill-Qwen-7B.

Training Data. The training of SABER uses only 2K examples: 1K math and 1K code instances. These data are partitioned using an accuracy-based strategy to promote stable learning of reasoning mode switching. As shown in Figure 1, each instance is first annotated with a reasoning-budget tier during data preprocessing. To incorporate the NoThink mode, we further create a duplicate of each sample annotated with a zero reasoning budget, resulting in complete overlap with the core corpus. This ensures that the model encounters both constrained and direct-answer variants within the same training schedule.

Main Results

To answer RQ1, Table 1 compares baselines with variants of our SABER framework, which incorporates different thinking modes. To capture both effectiveness and computational cost, we report accuracy (Acc.) and the average thinking length produced by the model (Len.). L1 effectively shortens chains of thought by rigorously capping the number

Training Setting	Test Setting	MATH500		GSM8K		MATH	
		Acc \uparrow	Len \downarrow	Acc \uparrow	Len \downarrow	Acc \uparrow	Len \downarrow
All Budget Downgrade	DeepThink	81.7	4902	84.8	1771	84.2	4579
	CoreThink	80.2	3663	82.8	850	82.8	3147
	FastThink	80.3	3042	81.0	674	82.0	2731
	NoThink	72.5	0	74.2	0	75.1	0
Without Budget Downgrade	DeepThink	83.3	5636	85.5	2182	84.9	5261
	CoreThink	81.7	5378	84.1	1763	84.4	4856
	FastThink	81.7	4896	83.3	1627	84.4	4583
	NoThink	75.1	0	74.3	0	75.7	0
Reduce NoThink Ratio	DeepThink	81.7	5072	85.2	1944	84.7	4957
	CoreThink	80.7	3816	82.7	977	83.1	3325
	FastThink	79.9	3424	81.8	847	82.5	3107
	NoThink	72.5	0	73.7	0	74.7	0
Remove NoThink Data	DeepThink	82.1	5317	84.9	1838	84.9	4767
	CoreThink	82.5	3819	82.7	1133	83.3	3545
	FastThink	81.0	3488	81.9	915	82.6	3154
	NoThink	61.4	0	58.2	0	62.3	0
Without Accuracy Filtering	DeepThink	83.7	5774	84.6	2210	85.0	5501
	CoreThink	83.0	4845	82.1	1446	84.2	4361
	FastThink	81.3	4254	80.5	960	83.7	4020
	NoThink	71.4	0	70.2	0	71.4	0

Table 3: Ablation results of SABER’s individual components on math reasoning tasks, showing accuracy (Acc), average response length (Len).

of thinking tokens during training. However, this hard constraint results in a steep drop in accuracy. SelfBudgeter softens the trade-off by pre-estimating suitable budgets, recovering much of the lost performance. Nevertheless, it requires multiple training passes over **30K** examples for 3 epochs, thereby incurring substantial computational cost. This inefficiency stems from the lack of task-aware budget assignment: since all samples are treated equally without token-level degradation, only a small fraction of them receive effective length penalty during training. As a result, the method requires a large number of training instances to achieve convergence, significantly increasing the overall training burden.

In contrast, SABER is trained on only **2K** examples and equips the model with the ability to assess problem difficulty, demonstrating high efficiency in learning to switch reasoning modes. Compared to the base model (DeepSeek-R1-Distill-Qwen-1.5B), SABER-FastThink achieves a **72.7%** reduction in average reasoning length. Notably, this concise reasoning does not impair accuracy, on the contrary, it yields an average **3.0%** improvement across all benchmarks, demonstrating a more favorable trade-off between computational cost and performance. SABER-CoreThink further improves this balance, reducing reasoning length by **67.9%** while boosting accuracy by **4.7%**. SABER-DeepThink delivers the strongest results, cutting reasoning by **41.2%** and enhancing accuracy by **6.8%**. These gains can be attributed to two main factors. First, more

concise reasoning helps avoid unnecessary repetition or distractions that may interfere with the final answer. Second, we observed that SFT-trained long-thinking models tend to exhibit repetitive generation. After reinforcement learning with format constraints, such repetition is significantly reduced, leading to improved benchmark performance.

Cross-scale and Cross-domain Generalization

To answer RQ2, we further apply SABER to the larger DeepSeek-R1-Distill-Qwen-7B model to assess its cross-scale generalization. The results are shown in Table 2. In this setting, SABER-FastThink maintains its efficiency advantage, reducing average reasoning length by a substantial **82.1%**. However, this comes with a **2.5%** decline in accuracy, reflecting a trade-off that becomes more pronounced at scale. Meanwhile, SABER-DeepThink achieves a **33.2%** reduction in reasoning length with only a **1.9%** increase in accuracy. These results suggest that while performance varies with model capacity, our method remains effective and adaptable across different model sizes. Furthermore, although the training data includes only math and code examples, the reasoning mode-switching capability of SABER generalizes well to unseen task types. Table 2 presents the performance of SABER on a logical reasoning benchmark, further demonstrating its strong generalization capabilities.

Question: How many positive whole-number divisors does 196 have?

FastThink

Reasoning Process: Okay, I need to find the number of positive whole-number divisors of 196. **First, I'll factorize 196** into its prime factors. Let's start by dividing 196 by 2: $196 \div 2 = 98$. Then, $98 \div 2 = 49$. Now, 49 is 7 squared, so $49 = 7^2$. **Putting it all together, $196 = 2^2 \times 7^2$.**

...
To find the number of divisors, I use the formula where I add one to each of the exponents in the prime factorization and multiply them. **So, $(2 + 1) \times (2 + 1) = 3 \times 3 = 9$.** Therefore, 196 has 9 positive divisors.

Final Answer: 9 (True)

CoreThink

Reasoning Process: Okay, so I need to figure out how many positive whole-number divisors 196 has. **Hmm, I remember that ... I should start by finding the prime factorization of the number.** Let me try that. First, I know that ... That's still even, so I can divide by 2 again to get 49. **Now, 49 is a number I recognize—it's 7 squared, right? So, 49 equals 7 times 7.** Putting that all together, the prime factors of 196 are 2 squared times 7 squared. **So, $196 = 2^2 \times 7^2$.**

...
Now, the formula for finding the number of divisors is to take the exponents of the prime factors, and then multiply those together. For 7^2 , the exponent is also 2, so adding one gives 3 as well. **Multiplying these together: 3×3 equals 9.** So, 196 has 9 positive whole-number divisors.

Final Answer: 9 (True)

DeepThink

Reasoning Process: Okay, so I need to figure out how many positive whole-number divisors 196 has. **Hmm, ... I remember that ... Let me try to recall ... First, let me start by factoring 196...** Therefore, putting it all together, **the prime factors of 196 are $2 \times 2 \times 7 \times 7$...** **I think that's correct because ...**

...
Now, the formula for finding the number of divisors is to ... So in this case ..., and then we can, so that... Wait, let me make sure ... **So that's $(2+1)(2+1) = 3 \times 3 = 9$.** Yeah, that seems right. **Just to double-check, maybe I can list all the divisors of 196.** Starting from ... Then ... And finally ... Let me count them: 1, 2, 4, 7, 14, 28, 49, 98, 196. **Hmm, so... That must be the correct answer...** So yeah, 196 has 9 positive whole-number divisors.

Final Answer: 9 (True)

Figure 3: Given same problem, SABER supports four thinking modes. Except NoThink mode, the other three modes include the core solution steps, such as factorization and counting (shown in green). Additional reasoning and reflection unique to CoreThink are marked in blue. Further self-checking and justification steps in DeepThink are highlighted in brown.

Ablation Experiments

To answer RQ3, we conduct ablation studies to assess the impact of key components in SABER, with each variant modifying exactly one component from the SABER configuration: (1) All Budget Downgrade: all training samples are downgraded by one reasoning level; (2) Without Budget Downgrade: all samples are trained strictly under their base model's original reasoning budget; (3) Reduce NoThink Ratio: decrease the proportion of NoThink samples in training to 30%; (4) Remove NoThink Data: remove all NoThink-mode data from training; and (5) Without Accuracy Filtering: during preprocessing, do not use model answer correctness to filter samples when assigning difficulty tiers.

As shown in Table 3, removing budget downgrade slows the adaptation to different reasoning modes, while applying aggressive downgrade to all samples leads to unstable training and reduced accuracy. Here, the "Without Budget Downgrade" setting closely resembles the baseline (Li et al. 2025b). Without progressive budget guidance, the model struggles to generalize shorter reasoning modes, resulting in slow mode adaptation and reduced efficiency. Reducing or removing NoThink samples significantly hurts NoThink performance without improving other modes, confirming the necessity of joint training. Finally, removing accuracy-based filtering causes supervision noise and degrades stability. These results underscore that all individual components of SABER are essential for effective and stable learning.

Behaviors of Different Reasoning Modes

To answer RQ4, we present a representative example from the MATH500 dataset to illustrate the differences among

three thinking modes. As shown in Figure 3, all modes follow the core problem-solving steps: decomposing 196 and counting its divisors. These steps lead to the correct final answer, which is highlighted in green in the figure. Specifically, FastThink only includes essential steps, making the reasoning process concise and direct. In contrast, CoreThink introduces some initial reasoning, such as "I remember that ...". It also includes additional intermediate steps and occasional reflection during the decomposition process. Furthermore, DeepThink adopts a more thorough and reflective reasoning process. After reaching a solution, it includes justifications such as "I think that's correct because...", followed by a final double-check and self-verification. This demonstrates a deeper and more comprehensive reasoning pattern.

Conclusions

In this work, we present SABER, a unified and switchable reasoning framework that enables large language models to perform efficient and controllable reasoning across specific modes. By combining structured reward design, discrete reasoning modes, and curriculum-style budget assignment, SABER achieves high training stability and reasoning flexibility without requiring supervised warm start. Our experiments demonstrate that SABER generalizes well across math reasoning, code generation, and logical reasoning tasks, maintaining strong performance under varying computational constraints. We further show that SABER effectively supports both thinking and no-thinking modes within a single model, with minimal performance degradation. These results point towards a promising direction for controllable and cost-efficient reasoning in LLMs.

References

- Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Aggarwal, P.; and Welleck, S. 2025. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *arXiv:2503.04697*.
- Austin, J.; Odena, A.; Nye, M.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C.; Terry, M.; Le, Q.; and Sutton, C. 2021. Program Synthesis with Large Language Models. *arXiv:2108.07732*.
- Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- Chen, X.; Xu, J.; Liang, T.; He, Z.; Pang, J.; Yu, D.; Song, L.; Liu, Q.; Zhou, M.; Zhang, Z.; Wang, R.; Tu, Z.; Mi, H.; and Yu, D. 2024. Do NOT Think That Much for $2+3=?$ On the Overthinking of o1-Like LLMs. *arXiv:2412.21187*.
- Chenglin, L.; Chen, Q.; Li, L.; Wang, C.; Tao, F.; Li, Y.; Chen, Z.; and Zhang, Y. 2024. Mixed Distillation Helps Smaller Language Models Reason Better. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Findings of the Association for Computational Linguistics: EMNLP 2024*. Miami, Florida, USA: Association for Computational Linguistics.
- Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *ArXiv*, abs/2110.14168.
- Dang, S.; Guo, Y.; Zhao, Y.; Ye, H.; Zheng, X.; Dai, G.; and Tsang, I. 2025. FZOO: Fast Zeroth-Order Optimizer for Fine-Tuning Large Language Models towards Adam-Scale Speed. *arXiv:2506.09034*.
- Fan, S.; Han, P.; Shang, S.; Wang, Y.; and Sun, A. 2025. Co-think: Token-efficient reasoning via instruct models guiding reasoning models. *arXiv preprint arXiv:2505.22017*.
- Feng, S.; Fang, G.; Ma, X.; and Wang, X. 2025. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*.
- Han, T.; Wang, Z.; Fang, C.; Zhao, S.; Ma, S.; and Chen, Z. 2025. Token-Budget-Aware LLM Reasoning. *arXiv:2412.18547*.
- Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS*.
- Hou, B.; Zhang, Y.; Ji, J.; Liu, Y.; Qian, K.; Andreas, J.; and Chang, S. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.
- Huang, S.; Wang, H.; Zhong, W.; Su, Z.; Feng, J.; Cao, B.; and Fung, Y. R. 2025a. AdaCtrl: Towards Adaptive and Controllable Reasoning via Difficulty-Aware Budgeting. *arXiv:2505.18822*.
- Huang, Z.; Cheng, T.; Qiu, Z.; Wang, Z.; Xu, Y.; Ponti, E. M.; and Titov, I. 2025b. Blending Supervised and Reinforcement Fine-Tuning with Prefix Sampling. *arXiv:2507.01679*.
- Jiang, L.; Wu, X.; Huang, S.; Dong, Q.; Chi, Z.; Dong, L.; Zhang, X.; Lv, T.; Cui, L.; and Wei, F. 2025. Think only when you need with large hybrid-reasoning models. *arXiv preprint arXiv:2505.14631*.
- Kimi. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *arXiv:2501.12599*.
- Lee, A.; Che, E.; and Peng, T. 2025. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*.
- Li, M.; Zhong, J.; Zhao, S.; Lai, Y.; Zhang, H.; Zhu, W. B.; and Zhang, K. 2025a. Think or not think: A study of explicit thinking in rule-based visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.16188*.
- Li, Z.; Dong, Q.; Ma, J.; Zhang, D.; and Sui, Z. 2025b. SelfBudgeter: Adaptive Token Allocation for Efficient LLM Reasoning. *arXiv:2505.11274*.
- Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*.
- Lu, J.; Yu, H.; Xu, S.; Ran, S.; Tang, G.; Wang, S.; Shan, B.; Fu, T.; Feng, H.; Tang, J.; et al. 2025. Prolonged reasoning is not all you need: Certainty-based adaptive routing for efficient llm/mlm reasoning. *arXiv preprint arXiv:2505.15154*.
- Ma, L.; Liang, H.; Qiang, M.; Tang, L.; Ma, X.; Wong, Z. H.; Niu, J.; Shen, C.; He, R.; Cui, B.; and Zhang, W. 2025. Learning What Reinforcement Learning Can't: Interleaved Online Fine-Tuning for Hardest Questions. *arXiv:2506.07527*.
- Ren, T.; Jiang, J.; Yang, H.; Tian, W.; Zou, M.; Li, G.; Zhang, Z.; Wang, Q.; Qin, S.; Zhao, Y.; Tao, R.; Shao, H.; and Peng, Y. 2025. RiskPO: Risk-based Policy Optimization via Verifiable Reward for LLM Post-Training. *arXiv:2510.00911*.
- Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv:2402.03300*.
- Shen, Y.; Zhang, J.; Huang, J.; Shi, S.; Zhang, W.; Yan, J.; Wang, N.; Wang, K.; Liu, Z.; and Lian, S. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.
- Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. *CoRR*, abs/2408.03314.
- Sui, Y.; Chuang, Y.-N.; Wang, G.; Zhang, J.; Zhang, T.; Yuan, J.; Liu, H.; Wen, A.; Zhong, S.; Chen, H.; and Hu, X. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. *arXiv:2503.16419*.
- Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv:2201.11903*.

White, C.; Dooley, S.; Roberts, M.; Pal, A.; Feuer, B.; Jain, S.; Shwartz-Ziv, R.; Jain, N.; Saifullah, K.; Dey, S.; et al. 2024. LiveBench: A challenging, contamination-limited LLM benchmark. *arXiv preprint arXiv:2406.19314*.

Wu, H.; Yao, Y.; Liu, S.; Liu, Z.; Fu, X.; Han, X.; Li, X.; Zhen, H.-L.; Zhong, T.; and Yuan, M. 2025a. Unlocking Efficient Long-to-Short LLM Reasoning with Model Merging. *arXiv:2503.20641*.

Wu, T.; Xiang, C.; Wang, J. T.; Suh, G. E.; and Mittal, P. 2025b. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*.

Xiang, V.; Blagden, C.; Rafailov, R.; Lile, N.; Truong, S.; Finn, C.; and Haber, N. 2025. Just Enough Thinking: Efficient Reasoning with Adaptive Length Penalties Reinforcement Learning. *arXiv:2506.05256*.

Yuan, Y.; Hao, J.; Ma, Y.; Dong, Z.; Liang, H.; Liu, J.; Feng, Z.; Zhao, K.; and Zheng, Y. 2024. Uni-rlhf: Universal platform and benchmark suite for reinforcement learning with diverse human feedback. *arXiv preprint arXiv:2402.02423*.

Zhao, K.; Hao, J.; Ma, Y.; Liu, J.; Zheng, Y.; and Meng, Z. 2023a. Enoto: Improving offline-to-online reinforcement learning with q-ensembles. *arXiv preprint arXiv:2306.06871*.

Zhao, Y.; Dang, S.; Ye, H.; Dai, G.; Qian, Y.; and Tsang, I. 2025. Second-Order Fine-Tuning without Pain for LLMs: A Hessian Informed Zeroth-Order Optimizer. In *The Thirteenth International Conference on Learning Representations*.

Zhao, Y.; Ma, Z.; Zhou, T.; Ye, M.; Sun, L.; and Qian, Y. 2023b. GCformer: An Efficient Solution for Accurate and Scalable Long-Term Multivariate Time Series Forecasting. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM '23*, 3464–3473. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701245.

Zhuang, R.; Wang, B.; and Sun, S. 2025. Accelerating chain-of-thought reasoning: When goal-gradient importance meets dynamic skipping. *arXiv preprint arXiv:2505.08392*.