ThinkSwitcher: When to Think Hard, When to Think Fast

Guosheng Liang[1], Longguang Zhong[1], Ziyi Yang[1], Xiaojun Quan[1,2]*
[1]School of Computer Science and Engineering, Sun Yat-sen University
[2]Shenzhen Loop Area Institute
{lianggsh3,zhonglg5,yangzy39}@mail2.sysu.edu.cn
quanxj3@mail.sysu.edu.cn

Abstract

Large reasoning models (LRMs) excel at solving complex tasks by leveraging long chain-of-thought (CoT) reasoning. However, this often leads to overthinking on simple tasks, resulting in unnecessary computational overhead. We observe that LRMs inherently possess the capability for efficient short CoT reasoning, which can be reliably elicited through prompt design. To leverage this capability, we propose **ThinkSwitcher**, a framework that enables a single LRM to dynamically switch between short and long CoT modes based on task complexity. ThinkSwitcher introduces a lightweight switching module trained with supervision signals derived from the relative performance of each reasoning mode across tasks. Experiments on multiple reasoning benchmarks show that ThinkSwitcher reduces computational cost by 20–30% while maintaining high accuracy on complex tasks. This demonstrates the effectiveness of ThinkSwitcher as a scalable and efficient solution for unified LRM deployment.

1 Introduction

Large reasoning models (LRMs) (Jaech et al., 2024; Guo et al., 2025; Google, 2025; Anthropic, 2025) have demonstrated impressive capabilities in solving complex tasks. They achieve this through long chain-of-thought (CoT) processes, which involve behaviors such as exploration, self-reflection, and verification (Li et al., 2025; Gandhi et al., 2025; Zeng et al., 2025). However, this strength can become a drawback: LRMs tend to *overthink* (Chen et al., 2024; Sui et al., 2025; Cuadron et al., 2025) even the simplest problems (e.g., *2+3=?*), which unnecessarily invokes elaborate reasoning for tasks that require minimal effort. This inefficiency becomes especially problematic in high-throughput applications (Kumar et al., 2025; Qu et al., 2025).

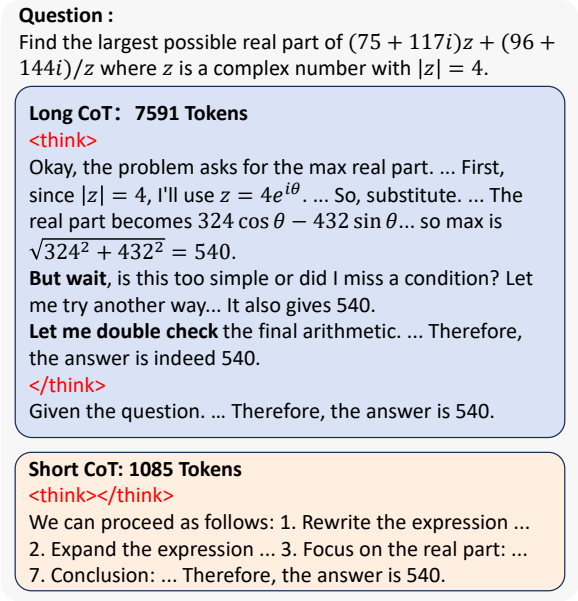


Figure 1: Comparison of long and short CoTs generated using different prompting strategies with Deepseek-R1-Distill-Qwen-7B. While long CoT reasoning often leads to *overthinking* and excessive token consumption due to elaborate reasoning steps, the short CoT can deliver comparable accuracy with substantially fewer tokens.

To mitigate this problem, many systems deploy two separate models: one tailored for complex reasoning and another for simpler tasks. While effective, this dual-model setup incurs additional computational and memory costs. This raises a fundamental question: *Can a single model achieve both robust reasoning capabilities and high efficiency?*

Inspired by the adaptive nature of human cognition—such as the *System 1* and *System 2* framework (Kahneman, 2003, 2011; Hua and Zhang, 2022)—we investigate enabling a single powerful reasoning model to operate in two distinct modes: its native long CoT mode for complex problems and an efficient short CoT mode for simpler tasks. Recent works such as Gemini-2.5-Pro (Google, 2025), Qwen3 (Yang et al., 2025), and Llama-Nemotron (Bercovich et al., 2025) have also explored dual-mode systems. However, these ap-

*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 5185–5201
November 4-9, 2025

proaches often lack public implementation details (Google, 2025) or depend on post-training with curated long and short CoT data (Yang et al., 2025; Bercovich et al., 2025), and typically require manual mode selection based on user inputs. In contrast, we propose a lightweight and adaptive alternative: a switcher module that automatically selects the appropriate reasoning mode based on task complexity, without backbone changes or large-scale training.

Our investigation reveals a key insight, echoed by Ma et al. (2025): advanced LRMs already possess a latent capability for concise and effective short CoT reasoning. Figure 1 illustrates a case where long CoT reasoning results in excessive token consumption, while short CoT achieves an accurate answer with substantially fewer tokens. Notably, we find that the short CoT capability can be reliably activated by appending an empty thinking block (e.g., *<think></think>*) after the user instruction—a phenomenon also observed in Qwen3 (Yang et al., 2025). This simple prompt-based intervention requires no changes to the model itself. Beyond this empirical finding, we offer a theoretical explanation of this latent behavior and its supporting mechanisms, detailed in Appendix C.

Building on these observations, we propose the **ThinkSwitcher** framework, which enables a single LRM to adaptively switch between long and short CoT modes. To support this capability, a lightweight switcher module is employed to predict the reasoning mode likely to yield optimal performance for a given query. The switcher is trained using self-supervised signals derived from the backbone model's own performance when executing both reasoning modes. This eliminates the need for external annotation or extensive post-training.

Our experiments show that ThinkSwitcher noticeably reduces average token usage across various benchmarks while maintaining high accuracy on complex reasoning tasks by retaining long CoT where necessary. For example, on simpler datasets such as GSM8K (Cobbe et al., 2021), it reduces inference tokens by around 30% with a performance loss of less than 1%. On more challenging datasets like AIME (MAA, 2025), ThinkSwitcher achieves token reductions of 38%, with only approximately a 2% decline in performance. Overall, it consistently lowers computational costs by 20–30% across benchmarks while retaining highly competitive accuracy. These results validate our approach to unifying strong reasoning capabilities with efficient resource usage in a single model deployment.

2 Related work

Large Reasoning Models Large reasoning models (LRMs), such as OpenAI-o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025), and QwQ (QwenTeam, 2025), are designed to emulate System-2 reasoning (Li et al., 2025). These LRMs have demonstrated state-of-the-art performance on challenging tasks in mathematics (Cobbe et al., 2021; Hendrycks et al., 2021) and coding (Chen et al., 2021; Codeforces, 2025). These models are typically trained via reinforcement learning (RL) algorithms (Schulman et al., 2017; Shao et al., 2024) to elicit long chain-of-thought (CoT) reasoning. However, their tendency to *overthink* (Chen et al., 2024) even on simple questions results in substantial computational inefficiencies and restricts their practicality in high-throughput scenarios. Several dual-system models have been introduced to mitigate this, including Claude-3.7-Sonnet (Anthropic, 2025), Gemini-2.5-Pro (Google, 2025), Qwen3 (Yang et al., 2025), and Llama-Nemetron (Bercovich et al., 2025). These models offer both long and short CoT modes, allowing users to choose between deep reasoning and quick answers depending on themselves. Among these, approaches from Yang et al. (2025) and Bercovich et al. (2025) implement this dual-mode capability through post-training on carefully curated mixtures of long and short CoT datasets.

Nonetheless, most of these models rely on manual mode selection, and lack the ability to automatically adapt the reasoning depth based on the input query. In contrast, we propose a lightweight mechanism that allows a single model to dynamically switch between short and long CoT modes. This enables both efficiency and strong reasoning without modifying weights or requiring post-training.

Efficient Reasoning Extensive research has focused on reducing inference overhead and improving reasoning efficiency (Sui et al., 2025). Among these methods, Kimi-k1.5 (Team et al., 2025) and O1-pruner (Luo et al., 2025) introduce length-controlled reward functions in reinforcement learning to reduce CoT reasoning length. Alternatively, methods such as DAST (Shen et al., 2025), C3oT (Kang et al., 2024), and TokenSkip (Xia et al., 2025) train LRMs to generate compact CoTs by constructing datasets with varying reasoning lengths and applying post-training techniques like SFT, DPO (Rafailov et al., 2023), or SimPO (Meng et al., 2024). Nevertheless, both strategies

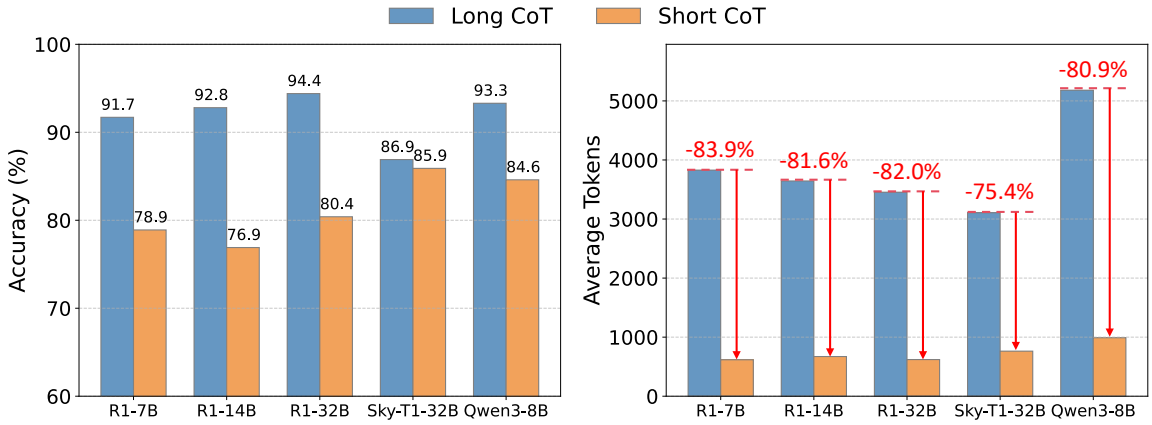


Figure 2: Comparison of long CoT and induced short CoT on the MATH500. "R1" denotes DeepSeek-R1-Distill series. **Left**: Accuracy comparison between long CoT and short CoT. **Right**: Average token usage for each reasoning mode, which demonstrates substantial token reductions with short CoT. Our approach of inducing short CoT consistently achieves substantial token savings while maintaining competitive accuracy across diverse LRMs.

necessitate additional LRM training, which introduces significant computational costs. A distinct direction involves prompt-guided strategies, such as CoD (Xu et al., 2025) and CCoT (Renze and Guven, 2024), which guide models to directly produce concise reasoning without fine-tuning. However, these methods lack the ability to dynamically adapt reasoning depth to question complexity, leading to performance degradation on challenging tasks (Xu et al., 2025). Another line of work on LLM-routing, such as RouteLLM (Ong et al., 2025), employs a router to distribute user questions across specialized LLMs. This approach assigns questions to the most suitable model to reduce average inference costs. While effective, it requires the simultaneous deployment of multiple LLMs. In contrast, our method achieves efficient reasoning with a single LRM, avoiding both significant fine-tuning costs and multi-model deployment complexities. This orthogonal approach dynamically adjusts reasoning depth and strikes an effective balance between deployment overhead and system performance.

3 Observations

Large reasoning models (LRMs) such as DeepSeek-R1 (Guo et al., 2025), Sky-T1-32B (Team, 2025), and Qwen3 (Yang et al., 2025) commonly employ structured generation formats to tackle complex reasoning tasks. A notable characteristic of these models is the use of special tokens, typically `<think>` and `</think>`, which explicitly separate the model's intermediate reasoning process from the final answer. The content after the `</think>` token provides a concise summary of the preceding reasoning and presents the final response.

3.1 Inducing Short CoT in LRMs

Our central observation is that the reasoning behavior of these models can be substantially influenced by manipulating the content placed within the `<think>` and `</think>` delimiters. In particular, we find that minimal or suggestive prompts within the `<think>` block can effectively steer the model toward generating much shorter chains of thought—termed *short CoT*—thereby countering its default tendency toward verbose reasoning.

For instance, prompts such as:

- `<think>`This problem appears straightforward.`</think>` (hinting at low complexity)
- `<think></think>` (no explicit reasoning)

consistently lead to more concise reasoning outputs compared to the model's standard behavior. This finding suggests that the reasoning trajectories of LRMs are highly prompt-sensitive and can be guided in depth and length through carefully designed prompts, even without explicitly instructing the model to generate shorter outputs.

3.2 Performance of Induced Short CoT

To evaluate the utility of prompt-induced short CoT as an efficient reasoning strategy, we conduct a series of experiments across multiple reasoning models. We focus on two key questions:

(1) *To what extent does short CoT reduce computational cost?*

(2) *How much reasoning performance is preserved relative to long CoT?*

As shown in Figure 2, our method consistently yields significantly shorter outputs across all models, including DeepSeek-R1-Distill (7B, 14B, and

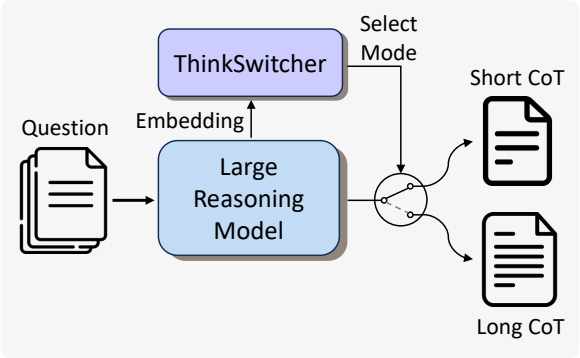


Figure 3: Dynamic mode selection during inference. Given a question embedding from the LRM, ThinkSwitcher dynamically chooses between short and long CoT reasoning based on estimated task difficulty.

32B), Sky-T1-32B, and Qwen3-8B. Additional figures illustrating these trends on other benchmarks are provided in Appendix A. This reduction in output length implies improved computational efficiency, potentially lowering FLOPs, reducing memory usage, and accelerating inference.

While short CoT shows a moderate drop in accuracy on particularly complex problems, it remains competitive across most cases and retains strong problem-solving capability. These results indicate that short CoT can serve as an efficient default reasoning mode, with long CoT invoked only when necessary. Furthermore, theoretical analysis of the mechanisms underlying short CoT induction is provided in Appendix C, offering a deeper explanation that extends the observations made in this section.

4 Methodology

To strike an effective balance between reasoning performance and computational efficiency, we propose **ThinkSwitcher** which dynamically switches between short and long chain-of-thought reasoning modes based on the input. Figure 3 illustrates the overall workflow of this framework.

4.1 Data Construction

Training ThinkSwitcher requires reliable, fine-grained supervision signals that reflect the relative effectiveness of short and long CoT reasoning for each input question. To this end, we adopt a multi-sample evaluation strategy and construct continuous regression targets based on empirical pass rates, rather than relying on unstable single-response outcomes or coarse binary labels. The data construction process is as follows:

Step 1: Prompting. For each query $q \in \mathcal{D}_{\text{train}}$, we construct two prompts corresponding to short CoT (SC) and long CoT (LC) reasoning modes, denoted by $m \in \{\text{SC}, \text{LC}\}$. The specific prompt templates are provided in Appendix B.

Step 2: Sampling. For each query, we generate k responses under both reasoning modes:

$$\mathcal{R}_m(q) = \left\{ r_m^{(i)}(q) \sim \pi(\cdot \mid q) \right\}_{i=1}^{k}. \quad (1)$$

Step 3: Evaluation. Each response is checked for correctness. The empirical pass rate is:

$$\mathcal{P}_m(q) = \frac{1}{k} \sum_{i=1}^{k} \mathbb{I}\left[r_m^{(i)}(q) \text{ is correct} \right]. \quad (2)$$

Step 4: Labeling. The pass rate serves as the regression target for supervising the mode switcher:

$$y_m(q) = \mathcal{P}_m(q). \quad (3)$$

The resulting training data takes the form $(x_q, y_{\text{SC}}(q), y_{\text{LC}}(q))$, where x_q is the query embedding and $y_{\text{SC}}(q), y_{\text{LC}}(q) \in [0, 1]$ denote the empirical pass rates under short and long CoT modes.

4.2 Decision Rule for Switching

ThinkSwitcher is implemented as a lightweight regressor that predicts the expected pass rates for short and long CoTs given a query q. It is trained using the supervision targets $y_{\text{SC}}(q)$ and $y_{\text{LC}}(q)$ defined in Section 4.1. The input is the query embedding x_q extracted from the reasoning model.

At inference time, the switcher takes x_q as input and produces two scalar predictions:

$$[\hat{y}_{\text{SC}}(q),\ \hat{y}_{\text{LC}}(q)] = f_\phi(x_q), \quad (4)$$

which correspond to the estimated pass rates under short and long CoT prompting, respectively. The final decision is made by comparing the difference between them against a tunable threshold τ:

$$m(q) = \begin{cases} \text{LC}, & \text{if } \hat{y}_{\text{LC}}(q) - \hat{y}_{\text{SC}}(q) \geq \tau, \\ \text{SC}, & \text{otherwise.} \end{cases} \quad (5)$$

This means the long CoT pathway is selected when its predicted advantage over short CoT exceeds the threshold τ; otherwise, the short CoT pathway is used, as formalized in Equation (5).

4.3 Margin-Aware Training Objective

We design a margin-aware objective to enhance the switcher's decision quality. A core component of this objective is the mean squared error (MSE),

where $\mathrm{MSE}(\hat{y}, y)$ denotes the squared difference $(\hat{y}-y)^2$ between a prediction $\hat{y}$ and its target y. The initial loss term, $\mathcal{L}_{\text{MSE}}$, applies this to the predicted and target pass rates for both reasoning modes:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2}\left(\mathrm{MSE}(\hat{y}_{\text{SC}}, y_{\text{SC}}) + \mathrm{MSE}(\hat{y}_{\text{LC}}, y_{\text{LC}})\right). \tag{6}$$

However, since the switching decision depends on the predicted margin $\hat{y}_{\text{LC}} - \hat{y}_{\text{SC}}$ (Equation (5)), this objective alone does not directly supervise the decision signal. To address this, we introduce a margin loss ($\mathcal{L}_{\text{margin}}$) that penalizes the error in the predicted difference:

$$\mathcal{L}_{\text{margin}} = \mathrm{MSE}\left(\hat{y}_{\text{LC}} - \hat{y}_{\text{SC}},\ y_{\text{LC}} - y_{\text{SC}}\right). \tag{7}$$

The final training objective is the sum of the standard MSE loss and the margin loss ($\mathcal{L}_{\text{switch}}$):

$$\mathcal{L}_{\text{switch}} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{margin}} \cdot \mathcal{L}_{\text{margin}}, \tag{8}$$

where λ_{margin} is a tunable hyperparameter controlling the weight of margin supervision.

This formulation enhances decision supervision by directly aligning the training objective with the switcher's routing mechanism. In practice, it results in more consistent pathway selection and improved overall performance. The impact of the margin loss $\mathcal{L}_{\text{margin}}$ is further examined in Section 6.3.

5 Experiments

5.1 Experimental Setup

Models We evaluate our ThinkSwitcher framework on three open-source reasoning models from the DeepSeek-R1 series: DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Qwen-14B. These models are distilled versions of DeepSeek-R1 and fine-tuned from Qwen2.5-Math-1.5B, Qwen2.5-Math-7B, and Qwen2.5-14B respectively. All three exhibit strong performance on complex reasoning tasks for their respective scales and serve as representative backbones for our evaluation.

Training Data To train ThinkSwitcher, we construct a comprehensive and representative dataset by aggregating the training sets of several well-established math benchmarks, spanning a wide range of reasoning difficulties. Specifically, our training data combines several sources: we use the training splits from MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021). The data also includes the historical AIME problems from 1983 to 2023 (MAA, 2025; Di Zhang, 2025). Finally, a subset from Omni-MATH (Gao et al., 2025) is incorporated, which consists of approximately 3,900 problems obtained by excluding the Omni-MATH-500 test set used for evaluation. There is no overlap between the data used to train ThinkSwitcher and any of the test sets used in our experiments. Detailed training hyperparameters and implementation specifics are provided in Appendix D.

Evaluation We evaluate ThinkSwitcher on a diverse set of datasets spanning three difficulty levels: (1) **Basic**, represented by GSM8K (Cobbe et al., 2021); (2) **Intermediate**, using MATH-500, a subset of MATH (Hendrycks et al., 2021); and (3) **Competition**, including AIME (MAA, 2025), LiveAoPSBench (Mahdavi et al., 2025), Omni-MATH-500 (Gao et al., 2025), and the math subset of OlympiadBench (He et al., 2024). Detailed descriptions are provided in Appendix E.

Baselines We compare the performance of ThinkSwitcher against four baselines, each representing a distinct reasoning strategy. **SC-Only** applies our prompting method to induce short CoT on all problems. This method prioritizes efficiency at the expense of reasoning performance on more complex tasks. **LC-Only** reflects the default behavior of LRMs, which follows long CoT without intervention and typically achieves higher accuracy on complex tasks. The **Random** switcher selects between short and long CoT prompts for each input based on predefined probabilities; specifically, "Random (x/y)" indicates that the long CoT mode is selected with a probability of x%, and the short CoT mode with a probability of y%. This serves as a stochastic control to assess whether the learned switcher offers improvements over naive selection. Finally, we include a **BERT**-based switcher. For this baseline, we trained a classifier based on an advanced BERT variant, ModernBERT-base (Warner et al., 2024). This classifier, trained on the same data as ThinkSwitcher, learns to select the reasoning mode likely to yield a higher pass rate and exemplifies a conventional routing method.

Efficiency Metrics We measure computational cost as the average number of tokens generated per question. For ThinkSwitcher, this depends on the reasoning path (SC or LC) selected at each question, while for static baselines it reflects the token usage of the fixed prompting strategy.

Method	GSM8K		MATH500		AIME24		AIME25		LiveAoPS		OmniMATH		OlymBench		Avg.	
	Acc.	Tok.	Acc.	Tok.	Acc.	Tok.	Acc.	Tok.	Acc.	Tok.	Acc.	Tok.	Acc.	Tok.	Acc.	Tok.
DeepSeek-R1-Distill-Qwen-1.5B																
SC-Only	71.7	246	68.2	880	12.5	4124	14.2	3808	35.0	2028	28.3	1949	32.7	1805	37.5	2120
LC-Only	85.5	2335	83.2	4907	28.3	12275	29.2	11235	46.5	10195	37.4	10613	43.7	9251	50.5	8687
Random (50/50)	78.7	1281	76.5	2821	18.3	8913	22.5	7187	40.3	6055	33.3	6224	38.5	5663	44.0	5449
Random (25/75)	82.2	1812	80.5	3838	24.6	10517	25.0	10186	42.9	8273	35.1	8265	40.9	7468	47.3	7194
Random (10/90)	84.5	2124	82.4	4465	25.4	11534	25.8	10780	44.7	9560	36.9	9582	42.9	8616	49.0	8094
BERT	80.6	**1424**	80.1	**3743**	18.8	**6872**	25.0	**6438**	40.4	**4534**	31.5	**3995**	38.4	**3766**	45.0	**4396**
ThinkSwitcher	**84.7**	2114	**82.4**	4544	**23.3**	8192	**28.3**	6689	**43.9**	7010	**35.3**	6238	**43.1**	5831	**48.7**	5803
DeepSeek-R1-Distill-Qwen-7B																
SC-Only	87.5	257	78.9	617	20.8	1781	18.3	1548	39.4	1171	35.6	1135	41.2	1044	46.0	1079
LC-Only	93.0	1672	91.7	3828	51.7	10884	39.2	11214	64.3	8592	54.1	9399	58.5	7728	64.6	7617
Random (50/50)	90.4	962	84.9	2244	39.6	6687	23.3	4620	52.8	5137	44.9	4948	50.5	4244	55.2	4120
Random (25/75)	91.7	1340	87.6	2963	45.0	9445	25.8	7733	57.9	6989	49.0	6985	55.2	5973	58.9	5918
Random (10/90)	92.6	1536	90.4	3509	47.5	10274	30.8	9169	61.9	7964	51.7	8396	57.3	7073	61.8	6846
BERT	87.9	**314**	89.5	**2686**	45.4	9985	**39.2**	11214	57.5	**5797**	50.0	7129	55.4	**5024**	60.7	6021
ThinkSwitcher	**92.5**	1389	**91.3**	3495	**48.3**	**7936**	37.5	**6955**	**61.6**	6571	**51.2**	**6345**	**57.0**	5147	**62.8**	**5405**
DeepSeek-R1-Distill-Qwen-14B																
SC-Only	90.7	248	76.9	672	18.3	2420	18.3	1593	43.6	1404	34.5	1386	40.4	1269	46.1	1284
LC-Only	95.3	1512	92.8	3644	61.7	9887	42.5	11081	68.5	7843	59.2	8619	60.0	7097	68.6	7098
Random (50/50)	93.1	894	84.3	2066	42.1	5316	24.2	5866	55.7	4619	47.3	5157	50.2	4068	56.7	3998
Random (25/75)	94.2	1213	88.1	2809	49.2	6764	30.0	6459	62.3	6270	53.7	7018	56.0	5563	61.9	5157
Random (10/90)	95.1	1395	90.9	3262	53.8	9044	40.0	9353	66.8	7304	56.6	7903	58.4	6530	65.9	6399
BERT	91.3	**303**	88.3	**2445**	57.1	9568	**42.5**	11081	63.5	6199	**55.1**	7018	**57.6**	5269	65.0	5983
ThinkSwitcher	**94.3**	1042	**92.7**	3572	**60.4**	**8044**	**42.5**	**10065**	**65.8**	**6018**	54.9	**5828**	**57.6**	**4651**	**66.9**	**5603**

Table 1: Performance of ThinkSwitcher and baseline methods across three model sizes (1.5B, 7B, and 14B) in the **DeepSeek-R1-Distill-Qwen** series, evaluated on diverse math benchmarks. 'Avg." shows macro-averaged accuracy and token counts across the seven benchmarks. **Bold** numbers indicate performance surpassing the BERT baseline

To evaluate the overall trade-off performance, we use the **AUC-AC** to quantify how well ThinkSwitcher maintains accuracy across its spectrum of token efficiencies, and its normalized variant **nAUC-AC** to measure the advantage gained by ThinkSwitcher's adaptive mechanism over a linear interpolation between SC-Only and LC-Only performance points. Detailed definitions of these metrics are provided in Appendix F.

5.2 Overall Results

To assess the effectiveness of ThinkSwitcher, we evaluate its performance across the three previously detailed model scales, analyzing the trade-off between reasoning performance and computational efficiency. For the main results presented in Table 1, decision thresholds (τ) were selected to achieve a favorable balance between accuracy and efficiency; specifically, τ is 0.04 for the 1.5B model, 0.05 for the 7B model, and 0.03 for the 14B model. We identify several key insights from these results.

First, ThinkSwitcher achieves a consistent balance between reasoning accuracy and computational cost. Compared to the default LC-Only strategy, which applies full long-form reasoning for every input, ThinkSwitcher reduces the average number of generated tokens by over 20% while incurring only a slight drop of 1–2% in average accuracy. This supports our core hypothesis that a single model can reason adaptively without relying on costly, always-on deep CoT processing.

In addition, we also compare ThinkSwitcher against a BERT-based switcher, a conventional classification approach widely used for decision routing. While BERT achieves lower token usage on the 1.5B model, it consistently underperforms ThinkSwitcher in both accuracy and efficiency on the larger models. Even on the 1.5B setting, this token saving comes at the cost of a substantial drop in accuracy, limiting its practical utility. In contrast, ThinkSwitcher delivers stronger performance with fewer parameters and consistently achieves lower token consumption across all model scales, making it a more robust and deployable solution.

Finally, we observe that smaller models benefit disproportionately from adaptive switching. For example, on the 1.5B model, ThinkSwitcher reduces token usage by more than 30% while maintaining robust accuracy. This suggests that weaker models are more prone to over-elaboration and thus stand to gain more from selective short-form reasoning.

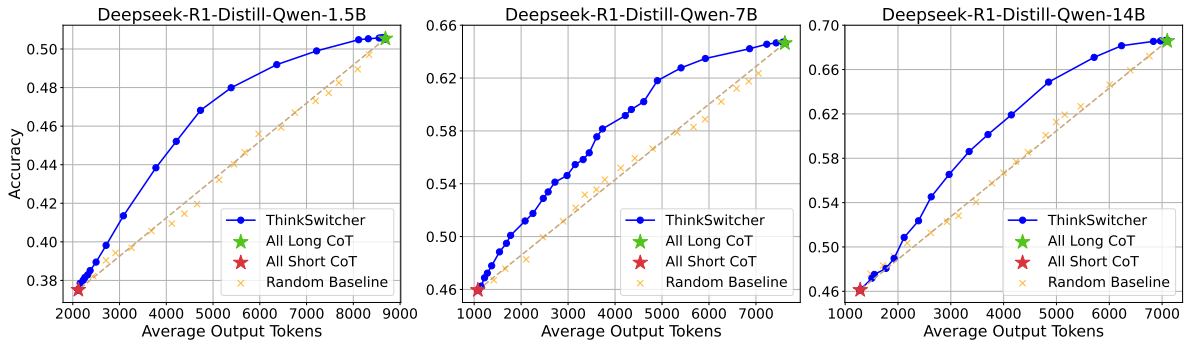


Figure 4: Trade-off between average accuracy and cost (measured by average output tokens) for the three DeepSeek-R1-Distill-Qwen model sizes. Each point on the ThinkSwitcher curves corresponds to a different τ value.

6 Analyses

To gain a deeper understanding of the behavior and effectiveness of ThinkSwitcher, we conduct in-depth analyses. These include evaluating cross domain generalization (Sec 6.1), examining decision dynamics (Sec 6.2), quantifying the impact of different training objectives (Sec 6.3), evaluating key design components (Sec 6.4), and assessing the computational efficiency of the switching mechanism (Sec 6.5). Unless otherwise specified, experiments in this section use the DeepSeek-R1-Distill-Qwen-7B model, selected for its favorable trade-off between efficiency and generality.

6.1 Cross-Domain Generalization

To examine whether ThinkSwitcher generalizes beyond mathematics, we evaluate it *without any retraining* on two additional domains: (i) scientific question answering (GPQA (Rein et al., 2024)) and (ii) code generation (LiveCodeBench (Jain et al., 2025)). The switcher used here is the same model trained solely on mathematics data in Sec 5; thus the evaluation setting constitutes a zero-shot cross-domain transfer scenario.

Method	**GPQA**		**LiveCode**	
	Acc.	Tok.	Acc.	Tok.
Long CoT	48.0	5653	45.4	8050
Short CoT	32.5	598	26.0	665
ThinkSwitcher	45.9	4487	43.5	6635

Table 2: Zero-shot cross-domain performance of ThinkSwitcher (DeepSeek-R1-Distill-Qwen-7B) on GPQA and LiveCodeBench.

As shown in Table 2, ThinkSwitcher preserves most of the Long CoT accuracy while reducing token usage. On GPQA, accuracy decreases 2.1%, accompanied by a 20.6% reduction in tokens. On LiveCodeBench, ThinkSwitcher incurs a 1.9% accuracy drop while reducing tokens by 17.6%. These results mirror the behavior observed in mathematics: the adaptive selection recovers most of the performance loss from naive short reasoning while retaining a appreciable efficiency gain.

The switcher was never exposed to code or QA instances during training, yet it produces useful routing decisions in these domains. This suggests that the learned decision function captures generalizable signals correlated with problem difficulty rather than overfitting to mathematical surface patterns.

6.2 Scaling the Trade-off

In this section, we evaluate the balance between accuracy and cost of ThinkSwitcher across three model scales: DeepSeek-R1-Distill-Qwen-1.5B, 7B, and 14B. For each model, we plot average accuracy against average token cost by systematically sweeping the decision threshold τ. The resulting trade-off curves are presented in Figure 4.

Across all model sizes, ThinkSwitcher consistently outperforms the random baseline, demonstrating its ability to dynamically select appropriate reasoning modes to balance performance and computational cost. Compared to fixed prompting strategies (SC-only and LC-only), it achieves a stronger Pareto frontier by delivering better accuracy–cost trade-offs across operating points.

The accuracy–cost curves reveal a notable trend: smaller models exhibit trade-off curves positioned closer to the top-left corner, indicating a more favorable balance between performance and efficiency. This observation aligns with discussions in Section 5.2, where it was noted that these smaller models (e.g., 1.5B) tend to generate significantly longer responses under LC-only prompting, resulting in higher average token costs. Such behavior suggests that weaker models rely more heavily on extended reasoning to arrive at correct answers and are thus more sensitive to the choice of prompting strategy.

Consequently, ThinkSwitcher achieves greater cost savings in these settings by selectively avoiding unnecessarily long reasoning paths.

These findings demonstrate that ThinkSwitcher provides a practical and scalable solution for cost-aware reasoning across models of varying capacity.

6.3 Effectiveness of the Switcher Loss

As detailed in Section 4.3, our final training objective incorporates a differential loss term, $\mathcal{L}_{\text{margin}}$, which explicitly supervises the predicted performance gap between CoT reasoning pathways. This additional supervision is critical for improving the quality of switcher decisions, as the routing mechanism is defined by Equation (5) and depends directly on the predicted margin $\hat{y}_{\text{LC}} - \hat{y}_{\text{SC}}$.

To examine its impact, we conducted an ablation study by varying the weight λ_{margin} in the combined loss defined in Equation (8). For the metric, we use normalized AUC-AC (nAUC-AC), which reflects the overall cost–accuracy trade-off relative to a linear baseline. The results are shown in Table 3.

Margin Loss Weight (λ_{margin})	**nAUC-AC**
0 (No Margin Loss)	167
1 (Moderate Weight)	**199**
2 (High Weight)	166

Table 3: Effect of margin loss weight (λ_{margin}) on downstream task performance, measured by nAUC-AC.

The results show that incorporating margin supervision with a moderate weight ($\lambda_{\text{margin}} = 1$) yields the best overall performance. This confirms the utility of explicitly aligning the training objective with the decision criterion. However, overly emphasizing the margin term ($\lambda_{\text{margin}} = 2$) degrades performance, suggesting that maintaining a balance between absolute accuracy and margin accuracy is essential for reliable switching.

6.4 Impact of Pass Rate Estimation Quality

The switcher is trained using pass rates estimated over k sampled responses per pathway (SC and LC) for each training instance. To assess how the estimation quality of these supervisory signals affects downstream performance, we vary $k \in \{1, 2, 4, 8, 16\}$ during the training data construction phase and evaluate the resulting switchers within the full ThinkSwitcher framework. Results are shown in Figure 5, measured by nAUC-AC.

We observe a clear upward trend in performance as k increases: nAUC-AC improves from approximately 174 at $k = 1$ to around 199 at $k = 8$,

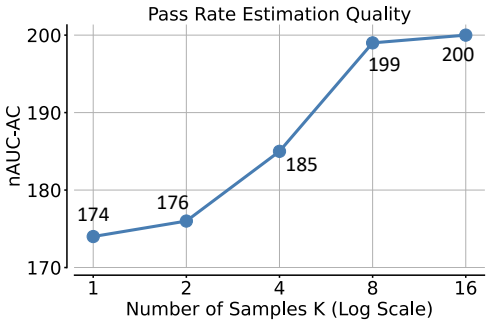


Figure 5: Performance with different values of k used to estimate pass rates in training data construction.

indicating that higher-quality pass rate estimates provide more reliable training signals and lead to more effective switching policies. However, this trend saturates beyond $k = 8$, as increasing to $k = 16$ yields only marginal improvement. This suggests diminishing returns, where further reducing the variance of the pass rate estimates no longer translates to meaningful performance gains. We therefore adopt $k = 8$ in our main experiments, as it provides a favorable trade-off between supervision fidelity and the computational overhead of constructing the training dataset. This setting ensures stable switcher performance without incurring the high cost of excessive sampling.

6.5 Computational Efficiency of Switching

To assess the practical efficiency of the proposed switching mechanism, we analyze its impact on inference-time computational cost across different model scales. This evaluation is crucial to ensure that the efficiency gains brought by adaptive reasoning are not offset by the additional overhead introduced by the switcher module.

Metric	**1.5B**	**7B**	**14B**
Switcher Params	2.89M	4.98M	6.56M
Switcher FLOPs	5.77M	9.97M	13.11M
LLM FLOPs (LC-Only)	26.06T	106.64T	198.74T
LLM FLOPs (TS)	17.41T	75.67T	156.88T

Table 4: Per-query computational costs across model scales. "Switcher Params" and "Switcher FLOPs" denote the size and compute overhead of the switching module. "LLM FLOPs (LC-Only)" refers to decoding cost under long CoT reasoning, while "LLM FLOPs (TS)" reflects the decoding cost with ThinkSwitcher.

As shown in Table 4, the switcher module introduces negligible overhead: its per-query compute cost consistently remains in the range of millions of FLOPs, which is insignificant compared to the trillions required for LLM decoding. Although the number of switcher parameters varies slightly with model scale, this variation is solely attributed to

differences in the hidden state dimensionality of the underlying LRMs, which only affects the input layer of the switcher. The rest of the switcher architecture remains fixed, ensuring the module stays lightweight and suitable for practical deployment.

More importantly, switching enables substantial reductions in overall decoding cost. Compared to the LC-Only baseline, which always applies long-form reasoning, adaptive switching reduces LLM decoding FLOPs by 20–30% across all model sizes. These savings arise from dynamically selecting shorter reasoning paths for simpler queries, thereby improving efficiency without compromising performance. Collectively, these findings validate our switching-based approach as a practical and scalable solution for cost-aware reasoning.

7 Conclusion

While large reasoning models (LRMs) excel at complex tasks, they often default to unnecessarily long chain-of-thought (CoT) reasoning even for simple problems and lead to significant computational overhead. We introduced ThinkSwitcher, a lightweight and adaptive framework that addresses this *overthinking* tendency by dynamically selecting between short and long CoT modes based on task complexity. ThinkSwitcher harnesses the model's latent ability for concise reasoning—elicited via prompt design—and integrates a self-supervised switcher module to automate mode selection without modifying the backbone architecture or requiring additional training. Experiments across diverse benchmarks show that ThinkSwitcher reduces computational cost by 20–30% while preserving strong performance on complex tasks. These findings highlight adaptive reasoning control as a scalable and effective approach to mitigating overthinking in LRMs and supporting efficient unified deployment.

Limitations

While our method demonstrates strong empirical performance, it still has several limitations. First, while the ThinkSwitcher framework demonstrates strong results on mathematical reasoning benchmarks, its applicability to other complex reasoning tasks such as agent has not yet been explored. Second, due to computational constraints, our experiments are limited to models with up to 14B parameters. Nonetheless, the method is inherently scalable, and we expect it to extend effectively to larger or architecturally diverse models, which we leave for future investigation.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62176270) and the Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515012832).

References

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Anthropic. 2025. Claude 3.7 sonnet system card.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, and 1 others. 2025. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 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, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Codeforces. 2025. Codeforces - competitive programming platform. Accessed: 2025-03-18.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Di Zhang. 2025. Aime_1983_2024 (revision 6283828).

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*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang,

Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2025. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*.

Google. 2025. Gemini 2.5 pro.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

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

Wenyue Hua and Yongfeng Zhang. 2022. System 1+ system 2= better world: Neural-symbolic chain of logic reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 601–612.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*.

Daniel Kahneman. 2003. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 93(5):1449–1475.

Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

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

Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. 2025. Overthink: Slowdown attacks on reasoning llms. *arXiv preprint arXiv:2502.02542*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

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

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *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*.

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

MAA. 2025. American invitational mathematics examination - aime.

Sadegh Mahdavi, Muchen Li, Kaiwen Liu, Christos Thrampoulidis, Leonid Sigal, and Renjie Liao. 2025. Leveraging online olympiad-level math problems for llms training and contamination-resistant evaluation. *arXiv preprint arXiv:2501.14275*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.

QwenTeam. 2025. Qwq-32b: Embracing the power of reinforcement learning. urlhttps://qwenlm.github.io/blog/qwq-32b/. Accessed: 6 March 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*.

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

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

NovaSky Team. 2025. Sky-t1: Train your own o1 preview model within $450. https://novasky-ai.github.io/posts/sky-t1.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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

A Induced Short CoT Performs on Challenging Tasks

This section extends the analysis in Section 3.2 by evaluating the performance of induced short CoT reasoning on additional two high-difficulty benchmarks: AIME24 and AIME25.

As shown in Figures 6 and 7, induced short CoT consistently yields substantial reductions in token usage on these high-difficulty AIME benchmarks. Aligning with our observations in the main paper, while short CoT exhibits an accuracy drop on such particularly complex problems, it still retains problem-solving capabilities. These AIME results therefore confirm that the significant efficiency benefits of short CoT induction extend to competition-level tasks. This reinforces its utility as an efficient reasoning mode, particularly when long CoT can be strategically invoked for instances requiring maximum performance.

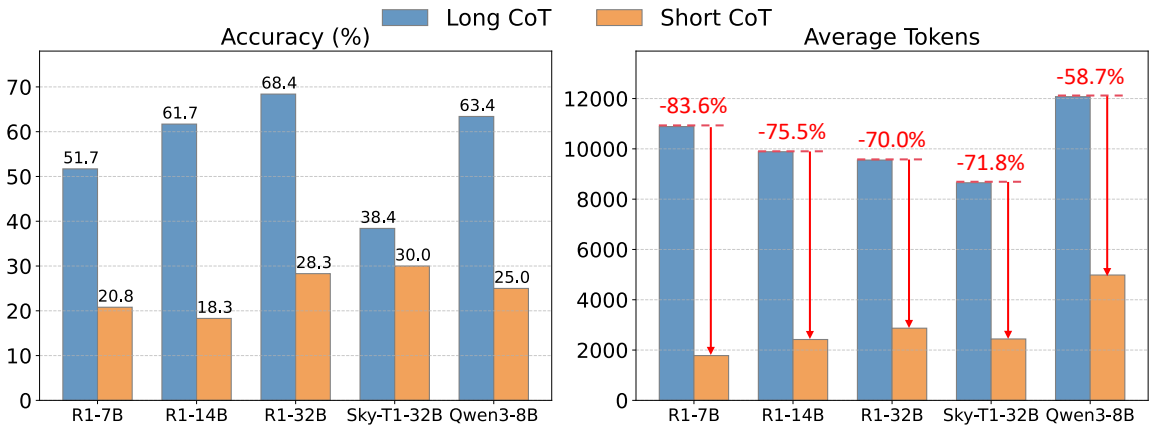


Figure 6: Comparison of long CoT and induced short CoT on the AIME24.

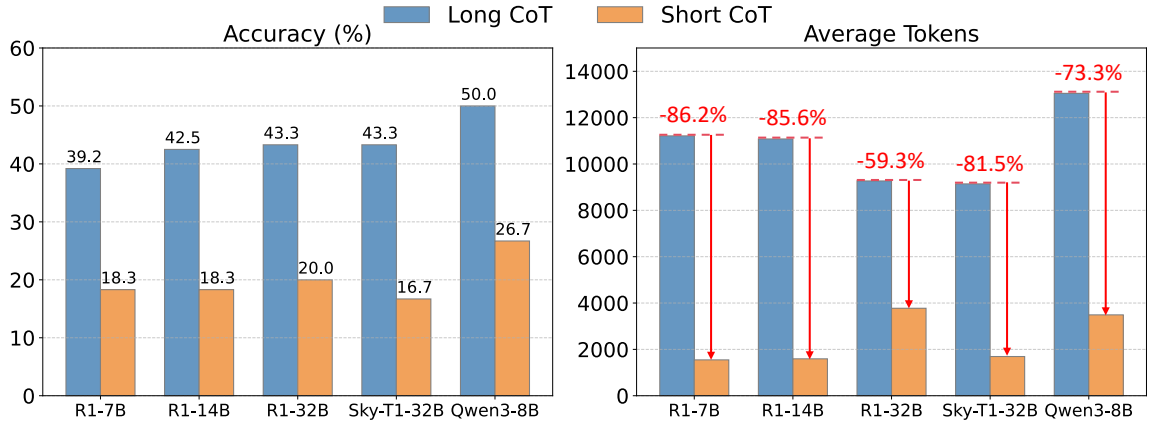


Figure 7: Comparison of long CoT and induced short CoT on the AIME25.

B Templates for Short and Long CoT

This section presents the prompt templates used to induce long and short CoT reasoning modes from LRMs. Each template contains a placeholder `{question}` for inserting the input query and includes instructions for the model to reason step by step, with the final answer enclosed in `\boxed{}`.

Long CoT Template

User: {question}
Please reason step by step, and put your final answer within \boxed{}
Assistant:

Short CoT Template

User: {question}
Please reason step by step, and put your final answer within \boxed{}
Assistant: <think></think>

Figure 8: Long and short CoT prompt template.

As shown in Figure 8, two templates differ only in the inclusion of an empty `<think>` block in the short CoT version, which reliably triggers concise reasoning behavior. As discussed in Section 3.1, this minimal intervention is sufficient to suppress unnecessary elaboration and induce efficient CoT responses without modifying the model itself.

C Mechanism Behind Short CoT Induction

What enables simple prompts to reliably induce short chains of thought in LRMs? We hypothesize that this behavior arises from the suppression or bypassing of the elaborate reasoning processes and stylistic conventions instilled during reasoning-specific fine-tuning. When such patterns are deactivated or inhibited, the model appears to revert to a generation mode that more closely resembles its pre-finetuning behavior, favoring concise and direct outputs over elaborated reasoning.

To investigate this hypothesis, we conducted a semantic similarity analysis across three categories of outputs on the MATH500 benchmark (Lightman

et al., 2024). We used the `all-mpnet-base-v2` model[1] to compute sentence embeddings for the reasoning outputs generated under different prompting conditions, and calculated cosine similarity between the following categories:

1. **ISC (Induced Short CoT):** Responses generated by DeepSeek-R1-Distill-Qwen-7B using our short CoT induction strategy.

2. **LCS (Long CoT Summary):** The final summary portion generated after the `</think>` token in DeepSeek-R1-Distill-Qwen-7B's default long CoT setting, typically containing the conclusion or final answer.

3. **OC (Original CoT):** Natural CoT responses generated by Qwen2.5-Math-7B (the base model for DeepSeek-R1-Distill-Qwen-7B), which is optimized for short CoT reasoning.

Pairwise Comparison	Cosine Similarity
ISC vs. OC	0.926
ISC vs. LCS	0.919
LCS vs. OC	0.916

Table 5: Cosine similarity between response embeddings from ISC, LCS, and OC categories.

The cosine similarity scores are reported in Table 5. Based on these values, we observe the following ranking:

$$\text{Sm(ISC, OC)} > \text{Sm(ISC, LCS)} > \text{Sm(LCS, OC)}$$

This ranking supports our suppression-and-reversion hypothesis. ISC responses exhibit greater semantic similarity to OC outputs from the general-purpose reasoning model than to LCS segments produced by the LRM under its long-form reasoning mode. This suggests that short CoT prompting may suppress fine-tuned long-form reasoning patterns, allowing the model to revert to a more concise reasoning style latent in earlier training. These results highlight the flexible behavioral priors retained by LRMs, which can be selectively activated or suppressed through external prompts.

D Implementation Details

The switcher module is a Multi-Layer Perceptron (MLP) consisting of 5 linear layers with hidden dimensions [1024, 768, 512, 256, 2]. The architecture utilizes Rectified Linear Unit (ReLU) activation functions (Agarap, 2019), Batch Normalization (Ioffe and Szegedy, 2015), and Dropout (Srivastava et al., 2014).

For training the switcher, the AdamW optimizer (Loshchilov and Hutter, 2019) was employed. Hyperparameter optimization was performed using Ray Tune (Liaw et al., 2018). The learning rate was selected through a search over a log-uniform distribution in the range $[1 \times 10^{-5}, 1 \times 10^{-2}]$. The batch size was chosen from the set {16, 32, 64, 128}. The dropout rate was explored over a uniform distribution in the interval [0.0, 0.5]. Training was conducted for a maximum of 50 epochs, employing an early stopping strategy based on validation set performance; the model from the best-performing epoch was retained.

The vLLM library (Kwon et al., 2023) was utilized to accelerate the sampling of responses from the backbone LRM for data construction. Evaluation of downstream task performance, which informed both the generation of switcher training labels and the final assessment of the ThinkSwitcher framework, was conducted using a customized evaluation framework from the Qwen2.5-math GitHub repository [2].

All experiments were conducted on a computing setup equipped with 4x NVIDIA RTX 4090 GPUs.

E Datasets Details

This section describes the evaluation datasets used to assess the performance and generalization of ThinkSwitcher across varying levels of difficulty.

GSM8K (Cobbe et al., 2021) A dataset of approximately 8,500 high-quality and linguistically diverse grade school math word problems. These problems typically require 2 to 8 steps to solve, and their solutions are written by human problem-solvers, primarily testing multi-step reasoning abilities. The problems cover various arithmetic operations and fundamental concepts taught at the elementary school level.

MATH (Hendrycks et al., 2021) Comprises 12,500 challenging mathematics problems, with topics including Algebra, Number Theory, Counting & Probability, and Geometry, representing a wide range of difficulties up to the high school

[1] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[2] https://github.com/QwenLM/Qwen2.5-Math

level. For our evaluation, we utilize a subset of 500 problems, referred to as **MATH-500**.

AIME (MAA, 2025) A significant component of the American Mathematics Competitions (AMC) program, positioned in difficulty between the AMC 10/12 and the USAMO (United States of America Mathematical Olympiad). It typically consists of 15 problems to be solved in 3 hours, with answers being integers from 0 to 999. These problems demand deeper mathematical knowledge and creative problem-solving strategies. The AIME dataset used in this evaluation includes problems from the most recent AIME2024 and AIME2025 to cover the latest competition content and challenge levels.

LiveAoPSBench (Mahdavi et al., 2025) A dataset of real-time, competition-level mathematics problems collected from online Olympiad mathematics communities, such as the Art of Problem Solving (AoPS) forums. These problems are often proposed by community members or originate from recent minor competitions, possessing strong timeliness and challenge. The subset of LiveAoPSBench used in this evaluation specifically refers to problems that appeared between August 2024 and December 2024, reflecting current trends and the high difficulty of Olympiad mathematics.

Omni-MATH (Gao et al., 2025) A comprehensive benchmark of Olympiad-level mathematics problems designed for universal LLM evaluation. It meticulously curates problems from diverse sources such as national/international Olympiads, training materials, and online forums, covering algebra, geometry, number theory, and combinatorics. Omni-MATH emphasizes problem diversity, fine-grained difficulty scaling, and depth, testing sophisticated reasoning and insight on complex, non-standard tasks. For our evaluation, we utilize a subset of 500 problems from this dataset, referred to as **Omni-MATH-500**.

OlympiadBench (math subset) (He et al., 2024) A challenging, multilingual benchmark designed to evaluate advanced scientific problem-solving in language models, encompassing disciplines like mathematics and physics. We utilize its mathematics subset, which comprises Olympiad-level problems sourced from prestigious international and national competitions, often structured as progressive tasks with interlinked sub-questions. These problems, provided with step-by-step solutions, demand deep conceptual understanding and innovative multi-step reasoning, rigorously testing mathematical capabilities at a high difficulty ceiling.

F AUC-AC and nAUC-AC: Metrics for Reasoning Efficiency

To systematically evaluate the trade-off between reasoning accuracy and computational cost, we adopt two metrics: the **Area Under the Accuracy–Cost Curve (AUC-AC)** and its normalized counterpart, **nAUC-AC**. These metrics quantify how effectively a model balances performance and efficiency across varying output lengths.

Let $A_{TS}(t)$ denote the accuracy achieved at an average token cost t. This function is empirically constructed by sweeping the decision threshold τ to obtain a set of operating points $\{(T(\tau_i), A(\tau_i))\}$. Let T_{SC} and T_{LC} denote the average output tokens of the SC-Only and LC-Only baselines, respectively. The AUC-AC is then defined as the integral of $A_{TS}(t)$ over the interval $[T_{SC}, T_{LC}]$:

$$\text{AUC-AC} = \int_{T_{SC}}^{T_{LC}} A_{TS}(t)\, dt. \tag{9}$$

A higher AUC-AC indicates better overall efficiency, reflecting the model's ability to maintain strong performance under stricter token budgets.

To assess the relative efficiency gain over static baselines, we define the **Normalized AUC-AC (nAUC-AC)**, which measures the gain over a linear interpolation baseline. This baseline corresponds to the linear interpolation between (T_{SC}, A_{SC}) and (T_{LC}, A_{LC}), representing the expected performance of strategies that use a fixed probability to mix SC-Only and LC-Only modes. The area under this baseline, denoted $\text{AUC-AC}_{\text{LB}}$, corresponds to the area of the trapezoid:

$$\text{AUC-AC}_{\text{LB}} = \frac{A_{SC} + A_{LC}}{2} \cdot (T_{LC} - T_{SC}). \tag{10}$$

The nAUC-AC is computed as the absolute gain over this linear reference:

$$\text{nAUC-AC} = \text{AUC-AC}_{\text{TS}} - \text{AUC-AC}_{\text{LB}}. \tag{11}$$

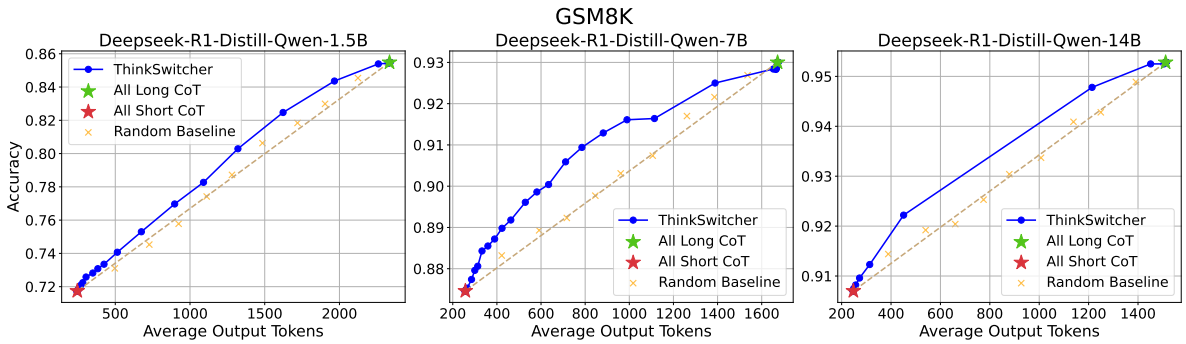
This difference captures the extent to which the model's reasoning efficiency exceeds what can be achieved by linear mixtures of static strategies.

G Per-Benchmark Trade-off Results

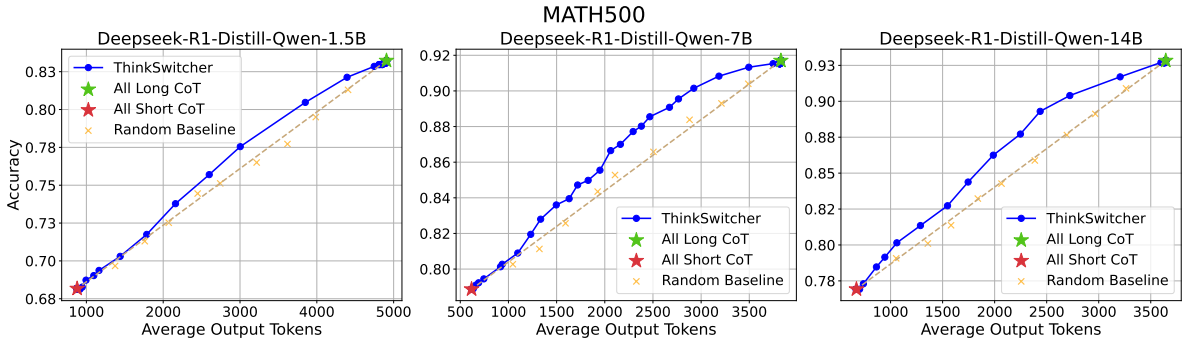
This section complements the aggregated analysis in Section 6.2 by providing a dataset-level

breakdown of the accuracy–efficiency trade-offs. While Figure 4 summarizes average performance across all benchmarks and model scales, Figures 9 and 10 present disaggregated trade-off curves for each of the seven benchmark datasets: GSM8K, MATH500, AIME24, AIME25, LiveAoPS, Omni-MATH, and OlymBench.

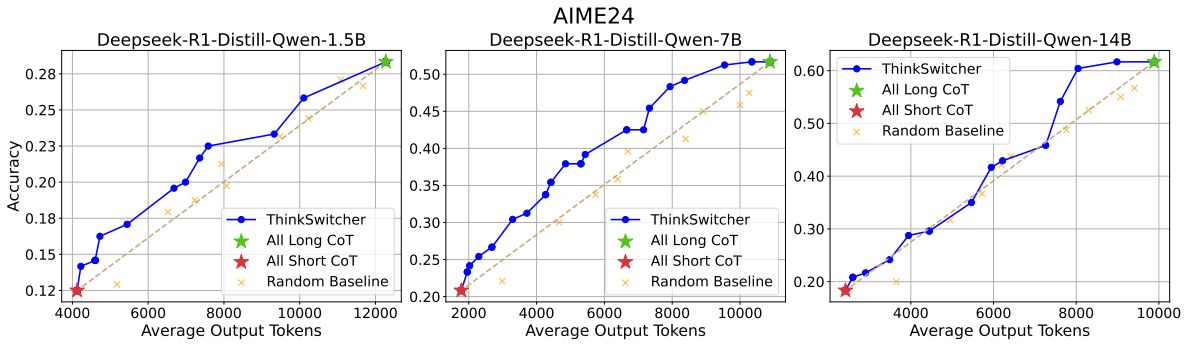
These disaggregated curves highlight how the effectiveness of ThinkSwitcher varies across task difficulties, offering further insight into when and where dynamic reasoning provides efficiency gains.



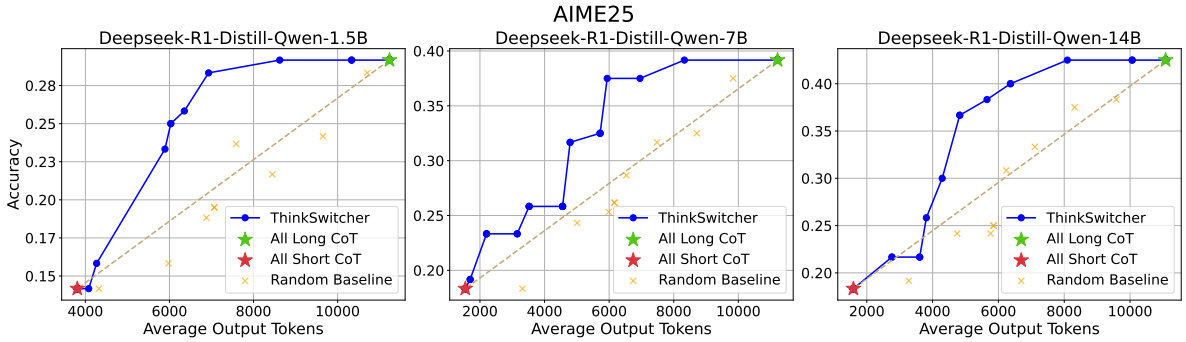
(a) Trade-off curve for GSM8K.



(b) Trade-off curve for MATH500.

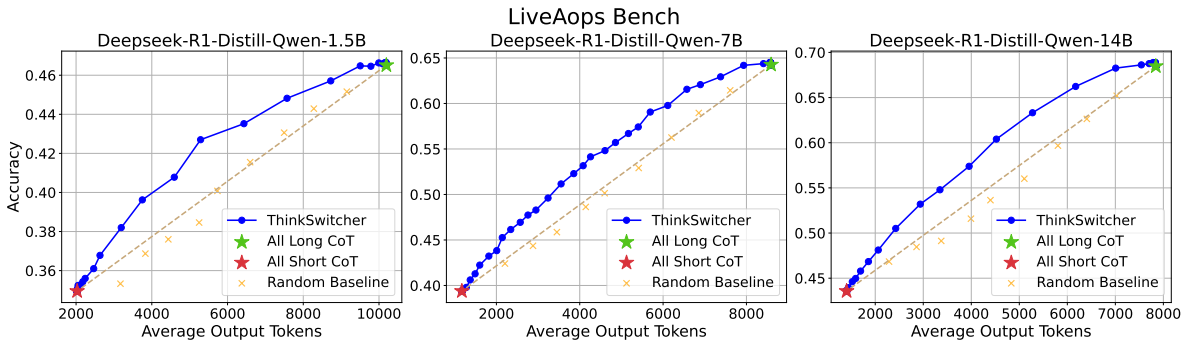


(c) Trade-off curve for AIME24.

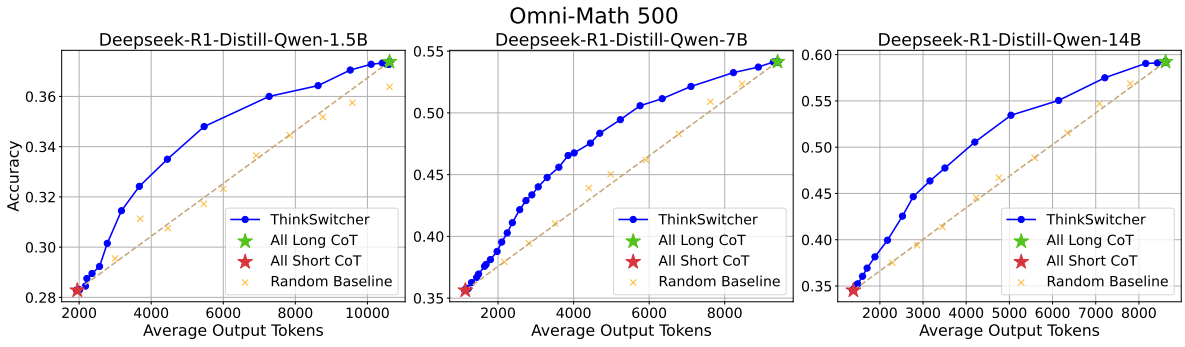


(d) Trade-off curve for AIME25.

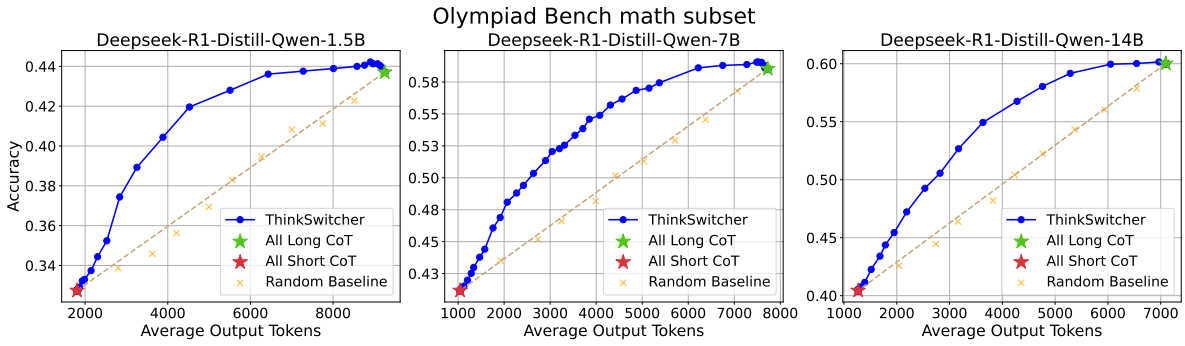
Figure 9: Trade-off between average accuracy and cost for ThinkSwitcher across various datasets (Part 1 of 2).



(a) Trade-off curve for LiveAoPS-Bench.



(b) Trade-off curve for Omni-MATH-500.



(c) Trade-off curve for OlympiadBench (math subset).

Figure 10: Trade-off between average accuracy and cost for ThinkSwitcher across various datasets (Part 2 of 2).