

# Steering LLM Thinking with Budget Guidance

Junyan Li<sup>1</sup>, Wenshuo Zhao<sup>2</sup>, Yang Zhang<sup>3</sup>, Chuang Gan<sup>1,3</sup>

<sup>1</sup>University of Massachusetts Amherst,

<sup>2</sup>Zhejiang University, <sup>3</sup>MIT-IBM Watson AI Lab

## Abstract

Recent deep-thinking large language models often reason extensively to improve performance, but such lengthy reasoning is not always desirable, as it incurs excessive inference costs with disproportionate performance gains. Controlling reasoning length without sacrificing performance is therefore important, but remains challenging, especially under tight thinking budgets. We propose *budget guidance*, a simple yet effective method for steering the reasoning process of LLMs toward a target budget without requiring any LLM fine-tuning. Our approach introduces a lightweight predictor that models a Gamma distribution over the remaining thinking length during next-token generation. This signal is then used to guide generation in a soft, token-level manner, ensuring that the overall reasoning trace adheres to the specified thinking budget. *Budget guidance* enables natural control of the thinking length, along with significant token efficiency improvements over baseline methods on challenging math benchmarks. For instance, it achieves up to a 26% accuracy gain on the MATH-500 benchmark under tight budgets compared to baseline methods, while maintaining competitive accuracy with only 63% of the thinking tokens used by the full-thinking model. *Budget guidance* also generalizes to broader task domains and exhibits emergent capabilities, such as estimating question difficulty. We release our code and model weights at <https://github.com/UMass-Embodied-AGI/BudgetGuidance>.

## 1 Introduction

With the recent success of deep-thinking large language models (LLMs) (Jaech et al., 2024; Guo et al., 2025; Yang et al., 2024a,b) that can generate long sequences of intermediate thoughts to achieve stronger performance, there is an increasing need to control their reasoning length while preserving accuracy, as such models often incur substantial inference costs with only marginal performance gains.

For example, in Figure 1, we show a response from a deep-thinking model that, while correct, is unnecessarily long. Such extensive reasoning is not always desirable, and there are cases where we need to impose a budget to limit the extent of reasoning, particularly in scenarios that demand real-time interaction, such as customer-facing chatbots, where excessive latency can degrade user experience and responsiveness.

Existing thinking budget control methods can be roughly divided into two categories with complementary strengths. The first category is fine-tuning methods, which fine-tune deep-thinking LLMs on a specially curated dataset (Han et al., 2024) or with a budget-aware reward to enable budget control capabilities (Hou et al., 2025). Fine-tuning methods are effective in changing the reasoning length while keeping competitive performance because they allow LLMs to fundamentally restructure and optimize their reasoning behavior according to the given budget. However, they come with two main drawbacks. First, fine-tuning an LLM requires substantial computational resources and time. Second, directly fine-tuning the LLM may potentially alter its behavior in unexpected ways, such as compromising safety (Qi et al., 2023).

The second category of methods is the inference time method (Ma et al., 2025; Muennighoff et al., 2025), which seeks to alter the reasoning behavior at inference time. While these approaches do not involve fine-tuning, they often result in sub-optimal reasoning behaviors and significant performance degradation, because the interventions at inference time are often heuristic and overly simple, breaking the integrity of the original reasoning process. For example, one well-known inference-time method is *budget forcing* (Muennighoff et al., 2025) which terminates the model’s reasoning as soon as the thinking budget is reached, as described in Figure 1. While this method offers strict thinking token control, abruptly interrupting the model may cut off

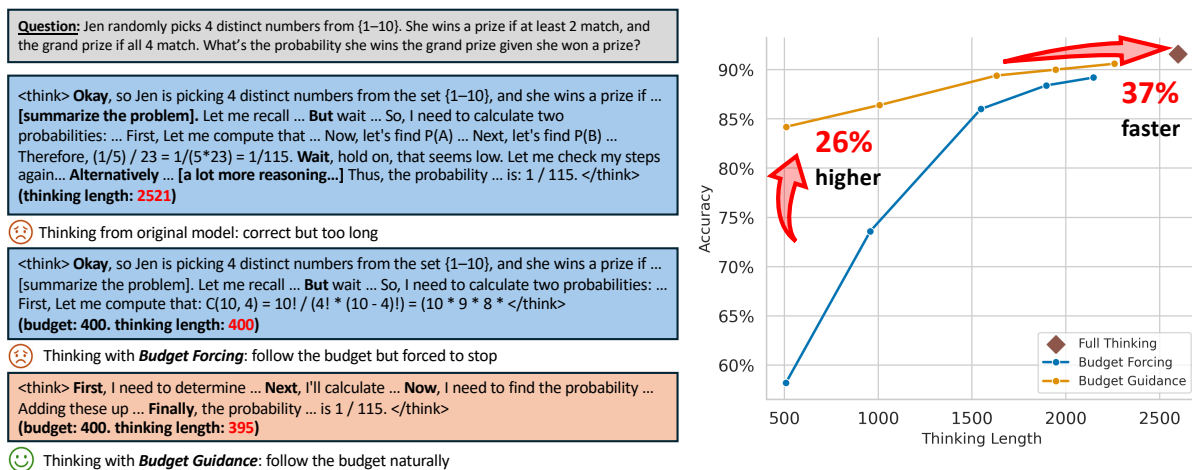


Figure 1: Deep-thinking models often produce excessively long reasoning traces, leading to high latency and unnecessary computation. Existing inference-time methods like *budget forcing* rely on simplistic heuristics such as hard stopping, which can result in incomplete reasoning and degraded answer quality. In contrast, *budget guidance* steers the reasoning process toward the target budget more smoothly and naturally, without any LLM fine-tuning.

unfinished thoughts and force premature answers, often leading to incorrect outputs.

In short, an important bottleneck in the task of thinking budget control lies in the tradeoff between *non-intrusiveness* (in inference-time approaches) and *optimality of the reasoning chain* (in fine-tuning approaches). This leads to our central research question: *Can we design a flexible inference-time budget control approach (without fine-tuning) that still allows for wholistic, principled restructuring of the reasoning process to maintain its quality under budget?*

In this paper, we introduce *budget guidance*, a novel approach that employs a lightweight auxiliary module to enable test-time control over the reasoning length of LLMs. Inspired by the principle of classifier guidance in diffusion models (Dhariwal and Nichol, 2021), we train an auxiliary predictor that predicts the probability distribution of the remaining reasoning length at each reasoning step. The predicted length distribution is then used to modulate the LLM generation probability, effectively turning it into a budget-conditional generation probability. Our method avoids the direct fine-tuning of LLMs, while providing flexible and accurate control over the reasoning process. It can be seamlessly integrated into existing inference pipelines, and adapts to a wide range of models, thinking budgets, and tasks.

Our experiments have revealed several key highlights of our method. First, *budget guidance* exhibits a remarkable trade-off between thinking length and performance. For example, as shown in

Figure 1, on MATH-500 benchmark (Hendrycks et al., 2021) *budget guidance* can reduce the full thinking length by 37% with minimal accuracy degradation, while being 26% higher in accuracy than *budget forcing* baseline under tight budget. Second, the auxiliary predictor is very successful in predicting the thinking length, effectively considering task difficulty and instruction type. Thus, it can accurately guide the thinking process under various budgets. Finally, our method demonstrates surprising generalizability across domains: an auxiliary predictor trained on one dataset can also work well in other datasets and domains. We summarize our contributions as follows:

- We propose *budget guidance*, a novel test-time method for steering the reasoning process of LLMs toward a specified thinking budget, without requiring any fine-tuning of the LLM itself.
- We design a lightweight predictor that models a Gamma distribution over the remaining reasoning length based on the current generation context, and uses this signal to guide LLM generation toward a target thinking budget.
- *Budget guidance* achieves strong trade-offs between thinking length and accuracy across multiple benchmarks, and demonstrates cross-domain generalization, enabling effective budget control and accurate thinking length prediction.

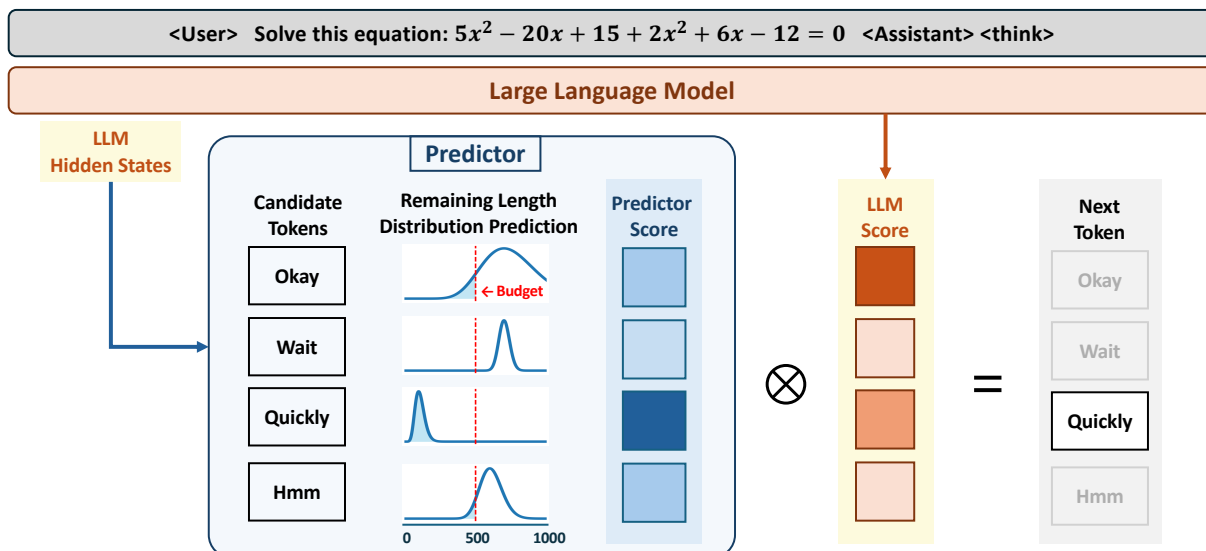


Figure 2: An overview of *budget guidance*. A lightweight predictor uses the LLM’s hidden states to predict a Gamma distribution over the remaining reasoning length for each candidate token. We then use the CDF of Gamma distribution to compute a predictor score, which is combined with the LLM’s output score to guide generation. The result is soft, token-level steering toward budget-conditioned reasoning without any LLM fine-tuning.

## 2 Related Works

### 2.1 Efficient LLM Reasoning

Efficiency in large language model (LLM) reasoning has been studied through two main paradigms: *fine-tuning based methods* and *inference-time steering*. Fine-tuning methods such as ThinkPrune (Hou et al., 2025), Z1 (Yu et al., 2025), and COCONUT (Hao et al., 2024) shorten reasoning traces via reinforcement learning, curriculum-style training on variable-length data, or continuous latent representations. While effective, these methods typically rely on expensive LLM fine-tuning and primarily aim to *reduce* the length of reasoning, rather than to *control* it. More recent approaches (Han et al., 2024; Muennighoff et al., 2025) have begun exploring methods to control the reasoning length, either through heuristic rules or model fine-tuning. In contrast, we propose a simple yet effective alternative: a fine-tuning-free approach that naturally steers the reasoning process to adhere to a specified thinking budget, enabling more efficient and flexible inference.

Inference-time steering, in contrast, intervenes directly during decoding without fine-tuning the LLM model. Dynasor (Fu et al., 2025) and DEER (Yang et al., 2025b) dynamically allocates compute by probing intermediate steps and early terminating confident cases. SEAL (Chen et al., 2025) calibrates reasoning traces by applying lightweight latent-space interventions to suppress reflection

and transition thoughts, thereby reducing redundancy during inference. While effective, these methods primarily optimize efficiency heuristically and do not offer fine-grained control over reasoning length. Simpler strategies include NoThinking (Ma et al., 2025), which bypasses reasoning altogether but typically suffers from severe accuracy loss. The most widely adopted approach that enables explicit steering is budget forcing (Muennighoff et al., 2025), used in real-world applications such as Claude 3.7 Sonnet<sup>1</sup> and Qwen3 (Yang et al., 2025a). It enforces a hard token cutoff to guarantee that reasoning length stays within a given budget. Although surprisingly effective in practice, this method leaves reasoning patterns untouched and forcibly terminates the reasoning once the budget is reached. In contrast, our approach offers smooth and fine-grained control over reasoning length, eliminating the need for heuristic rules or hard cutoffs.

### 2.2 Guidance and Guided Generation

The term *guidance* originates primarily from the diffusion model literature (Ho and Salimans, 2022), where it denotes the ability to steer the generative process, often through truncated or low-temperature sampling, by reducing the variance or range of noise inputs to the generative model at sampling time. This effectively transforms an un-

<sup>1</sup><https://www.anthropic.com/news/visible-extended-thinking>

conditional diffusion model into a conditional one, enabling it to generate targeted outputs. One of the earliest examples is *classifier guidance* (Dhariwal and Nichol, 2021), which modifies the diffusion score by incorporating the gradient of the log-likelihood from an auxiliary classifier, thereby biasing the sampling process toward desired content. This can be viewed as a form of guided generation, where image generation is conditioned on the output of a classifier.

A similar notion of guided generation has emerged in the context of LLMs (Willard and Louf, 2023), where it typically refers to constraining the model’s output to satisfy structural requirements, such as context-free grammars, to ensure syntactic correctness for downstream applications.

To the best of our knowledge, this work is the first to transfer the idea of guided generation from vision to a new dimension in NLP: *budget-conditioned generation*. We introduce a novel guidance mechanism that softly steers an LLM’s generation toward a specified thinking budget, enabling efficient and controlled reasoning while preserving output quality.

### 3 Budget Guidance

#### 3.1 Overall Framework

The overall framework of our method follows the classifier guidance framework in diffusion generation (Dhariwal and Nichol, 2021), thus we name our framework *budget guidance*. Specifically, denote  $X$  as the input question,  $Y_{<t}$  as the LLM’s output thinking process up to token  $t$ , and  $Y_t$  as the LLM’s output at token  $t$ . The vanilla LLM generation process essentially involves sampling from the following *budget-unconditional distribution*,  $p(Y_t|X, Y_{<t})$ .

However, when a budget constraint is imposed, meaning that the reasoning length must remain below a specified limit, the generation instead needs to follow a *budget-conditional distribution*. Formally, denote  $L_t$  as the random variable indicating the *remaining length* of the thinking process from token  $t$ . For example, if the overall thinking length is  $l$  (i.e., the `</think>` token occurs at token  $l$ ), then  $L_t = l - t$ . Given the thinking budget limit  $\bar{l}$ , the budget-conditional distribution is defined as  $p(Y_t|X, Y_{<t}, L_t \leq \bar{l} - t)$ .

According to Bayes’ rule, the budget-conditional distribution can be computed from the budget-

unconditional distribution as follows

$$\begin{aligned} p(Y_t | X, Y_{<t}, L_t \leq \bar{l} - t) \\ \propto p(Y_t | X, Y_{<t}) \\ \cdot \Pr(L_t \leq \bar{l} - t | X, Y_{<t}, Y_t) \end{aligned} \quad (1)$$

Therefore, at each token  $t$ , generating from the budget-conditional distribution involves three steps. First, compute the unconditional distribution, which is simply performing a forward pass of the LLM. Second, predict the remaining length distribution,  $\Pr(L_t \leq \bar{l} - t | X, Y_{<t}, Y_t)$ . Finally, use the remaining length distribution to modulate the unconditional distribution and then renormalize.

Within the budget guidance framework, the problem reduces to estimating  $\Pr(L_t \leq \bar{l} - t | X, Y_{<t}, Y_t)$ . We address this using a lightweight auxiliary thinking-length predictor, described in the following subsections.

#### 3.2 An Auxiliary Thinking Length Predictor

Denote the LLM vocabulary size as  $n$ , and denote the vocabulary as  $\mathcal{V} = \{v_1, \dots, v_n\}$ . At each token  $t$ , the LLM outputs an  $n$ -dimensional unconditional probability vector  $\mathbf{u}_t$ :

$$\mathbf{u}_t = [p(Y_t = v_1 | X, Y_{<t}), \dots, p(Y_t = v_n | X, Y_{<t})] \quad (2)$$

According to Equation 1, the predictor needs to predict an  $n$ -dimensional vector  $\mathbf{a}_t$ :

$$\mathbf{a}_t = [\Pr(L_t \leq \bar{l} - t | X, Y_{<t}, Y_t = v_1), \dots, \Pr(L_t \leq \bar{l} - t | X, Y_{<t}, Y_t = v_n)] \quad (3)$$

so that the budget-conditional probability vector, which we denote as  $\mathbf{c}_t$ , can be computed by element-wise multiplying the two vectors and renormalizing:  $\mathbf{c}_t = \text{normalize}(\mathbf{u}_t \circ \mathbf{a}_t)$ .

Equation 3 implies that the predictor must solve a highly demanding task. At each token  $t$ , given the question  $X$  and the generated context  $Y_{<t}$ , it must ❶ enumerate all possible values of  $Y_t$  in the vocabulary, ❷ for each candidate token, predict the distribution of the remaining reasoning length conditioned on that token (resulting in  $n$  distributions in total), and ❸ compute, for each distribution, the cumulative probability up to  $\bar{l} - t$ . Importantly, our goal is not to model the full reasoning dynamics or predict the entire reasoning trajectory, which is already handled by the LLM. Instead, the predictor models the remaining-length distribution for every possible next token at each generation step. The core challenge arises from the large vocabulary size

and the need for an expressive and computationally efficient estimation form for each distribution.

To address this challenge, we parameterize each predicted distribution as a Gamma distribution over  $\log(L_t)$ , *i.e.*,  $p(L_t | X, Y_{<t}, Y_t = v_i) = \Gamma(\log(L_t); \lambda_t(v_i), \alpha_t(v_i))$ , where  $\Gamma(\cdot; \lambda, \alpha)$  denotes the Gamma probability density function (PDF) with shape parameter  $\lambda$  and rate parameter  $\alpha$ . Modeling  $\log(L_t)$  rather than  $L_t$  directly improves robustness to the wide dynamic range of reasoning lengths.

Under this assumption, instead of predicting  $n$  full probability distributions, the predictor only needs to output two  $n$ -dimensional vectors,  $\boldsymbol{\lambda}_t = [\lambda_t(v_1), \dots, \lambda_t(v_n)]$  and  $\boldsymbol{\alpha}_t = [\alpha_t(v_1), \dots, \alpha_t(v_n)]$ . The cumulative probability vector  $\boldsymbol{\alpha}_t$  can then be computed efficiently using the closed-form CDF of the Gamma distribution.

### 3.3 Training the Predictor

To train the predictor, we require a dataset of reasoning chains. Formally, the data in the dataset takes the following form:  $\mathcal{D} = \{(x, y_{1:l}, l)\}$ , where  $x$  is the input question,  $y_{1:l}$  is the LLM-generated reasoning chain, and  $l$  is the length of the reasoning chain. Note that the task dataset from which reasoning chain length training data are generated is not the same as the inference dataset (not even the same task), as we will show that the trained predictor has good dataset and task generalizability.

For each training datum  $(x, y_{1:l}, l)$ , we feed the information of a partial reasoning chain to the predictor, truncated at different positions, and train the predictor to predict the remaining length. We adopt the maximum log-likelihood objective for gradient-descent training. Formally, denote the parameters of the auxiliary predictor as  $\boldsymbol{\theta}$ . For convenience, we denote the conditioning variables as  $z_t = (X = x, Y_{<t} = y_{<t}, Y_t = y_t)$ . Then the training objective can be written as

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{(x, y_{1:l}, l) \sim \mathcal{D}} \left[ \sum_{t=1}^{l-1} \log p_{\boldsymbol{\theta}}(L_t = l - t | z_t) \right] \quad (4)$$

where  $p_{\boldsymbol{\theta}}(\cdot)$  represents the predicted Gamma PDF by the auxiliary predictor.

### 3.4 Architecture of the Predictor

The predictor is designed to be lightweight so as not to introduce significant computational overhead during decoding, while remaining expressive

enough to capture the input question and the ongoing reasoning context for estimating the remaining reasoning length. We adopt BERT-base (Devlin et al., 2019) as the backbone of the predictor. Its input consists of the concatenated hidden states from all layers corresponding to the last generated token of the target LLM, which encode rich semantic information about both the question and the reasoning history. Following standard BERT usage, a [CLS] token is prepended to the sequence of hidden states, and the entire sequence is fed into BERT. The resulting [CLS] representation is then passed through a linear projection followed by a softplus activation (Dugas et al., 2000) to produce a non-negative output matrix  $M \in \mathbb{R}^{n \times 2}$ , where each row corresponds to the parameters  $\boldsymbol{\lambda}_t$  and  $\boldsymbol{\alpha}_t$  of a Gamma distribution. An ablation study of the predictor architecture is provided in Appendix D.

### 3.5 Skipping Modulation

Ideally, probability modulation would be applied at every decoding step  $t$ . Although the computation for BERT is very lightweight compared to modern LLMs, to further reduce computational overhead, we apply the modulation only at the start of each reasoning paragraph, as indicated by new-line delimiters, where the guidance for reasoning length is usually the most effective. At these positions, we compute the modulated distribution as  $c_t = \text{normalize}(\mathbf{u}_t \circ \boldsymbol{\alpha}_t)$ , while for all other decoding steps we simply set  $c_t = \mathbf{u}_t$ . Empirically, we find that this strategy is effective (Appendix C) and results in negligible latency overhead in practice (Appendix B).

## 4 Experiments

### 4.1 Settings

**Training.** We apply our method to three deep-thinking models: *DeepSeek-R1-Distill-Qwen-7B/32B* (abbreviated as DS-7B/32B) (Guo et al., 2025), and *Qwen3-8B* (Yang et al., 2024a,b). For simplicity, in this work we focus on math reasoning and use the OpenR1-Math-220k dataset (HuggingFace, 2025), which contains math reasoning chains generated by DeepSeek-R1 (Guo et al., 2025) model, for training. During training, the LLMs are frozen, and only the predictor is updated. We train for one epoch using a batch size of 8 and a constant learning rate of  $1.0 \times 10^{-4}$  after warmup. **Evaluation.** We evaluate our method on four math reasoning benchmarks: **MATH-500** (Hendrycks

	MATH-500		AIME-2024		AMC		OlympiadBench	
	Acc.	#Tokens	Acc.	#Tokens	Acc.	#Tokens	Acc.	#Tokens
<i>DeepSeek-R1-Distill-Qwen-7B</i>								
Thinking	91.6	2598	36.7	4446	78.3	4338	56.9	3960
NoThinking	74.8	-	23.3	-	47.0	-	40.4	-
Chain-of-Draft	75.2	690	30.0	2963	50.6	1535	47.7	1911
TALE-EP	85.2	1201	30.0	2718	<b>65.1</b>	1649	<u>53.9</u>	1693
SEAL	<u>87.6</u>	1741	<u>33.3</u>	3932	53.0	2661	52.4	2699
Dynasor	83.0	1653	20.0	2743	53.0	2002	48.9	1876
Budget Forcing	86.0	1547	16.7	2015	55.4	1872	47.1	1844
<b>Budget Guidance</b>	<b>88.2</b>	1329	<b>33.3</b>	2046	<u>60.2</u>	1768	<b>54.2</b>	1755
<i>DeepSeek-R1-Distill-Qwen-32B</i>								
Thinking	93.2	2226	70.0	7694	77.1	4156	61.9	3435
NoThinking	68.2	-	20.0	-	47.0	-	41.9	-
Chain-of-Draft	87.4	918	<u>50.0</u>	2919	<u>68.7</u>	1344	55.4	1283
TALE-EP	<u>88.8</u>	1449	<u>50.0</u>	3689	<u>68.7</u>	1879	56.9	1906
SEAL	87.2	1495	43.3	3416	55.4	2169	56.3	2220
Dynasor	88.0	1978	46.7	4199	67.5	3073	<u>57.5</u>	3079
Budget Forcing	86.4	1525	40.0	2936	50.6	1567	50.7	1797
<b>Budget Guidance</b>	<b>90.0</b>	1288	<b>56.7</b>	2873	<b>69.9</b>	1528	<b>57.8</b>	1820
<i>Qwen3-8B</i>								
Thinking	96.2	4613	73.3	13660	91.6	8740	71.7	9424
NoThinking	84.8	-	33.3	-	56.6	-	53.5	-
Chain-of-Draft	<u>91.6</u>	1891	<u>50.0</u>	3943	78.3	3785	<b>66.1</b>	2650
TALE-EP	90.8	3006	43.3	5353	<u>79.5</u>	5467	64.2	3604
Dynasor	91.2	3062	46.7	5164	72.3	4196	62.7	4064
Budget Forcing	90.2	2545	43.3	4010	77.1	3807	61.6	3712
<b>Budget Guidance</b>	<b>93.0</b>	2062	<b>50.0</b>	3981	<b>80.7</b>	3869	<u>65.6</u>	3639

Table 1: Evaluation results on math benchmarks. **Bold** numbers denote the best performance, and underlined numbers denote the second best in each setting.

et al., 2021), **AIME-2024** (Art of Problem Solving, n.d.a), **AMC** (Art of Problem Solving, n.d.b) (including both AMC12 2022 and AMC12 2023), and the math subset from **OlympiadBench** (He et al., 2024). We also evaluate on benchmarks from broader domains to test the out-of-domain transferability of our math-data-trained predictor, including **GPQA Diamond** (Rein et al., 2024), **FOLIO** (Han et al., 2022), **TableBench** (Wu et al., 2025), and **LiveCodeBench** (Jain et al., 2024). A detailed description of these benchmarks is provided in Appendix G. We use greedy decoding for both our method and the baselines in all evaluations for fair comparison.

**Baselines.** We compare our method with baselines that do not finetune the LLM. The first group uses prompting to shorten reasoning, including NoThinking (Ma et al., 2025), Chain-of-

Draft (Xu et al., 2025), and TALE-EP (Han et al., 2025). The second group modifies model behavior without prompting, including SEAL (Chen et al., 2025), Dynasor (Fu et al., 2025), and Budget Forcing (Muennighoff et al., 2025). We use their official implementation for evaluation, and skip SEAL evaluation for Qwen3-8B as they do not officially support this model yet. A detailed descriptive comparison between the baselines and our method is provided in Appendix A.

## 4.2 Main Results

### 4.2.1 Evaluation on Math Reasoning Benchmarks

Since the predictor is trained on math data, we first evaluate its performance on math reasoning benchmarks to assess in-domain effectiveness. We set the thinking budget to approximately half the original



Figure 3: Accuracy vs. thinking length trade-off on math benchmarks.

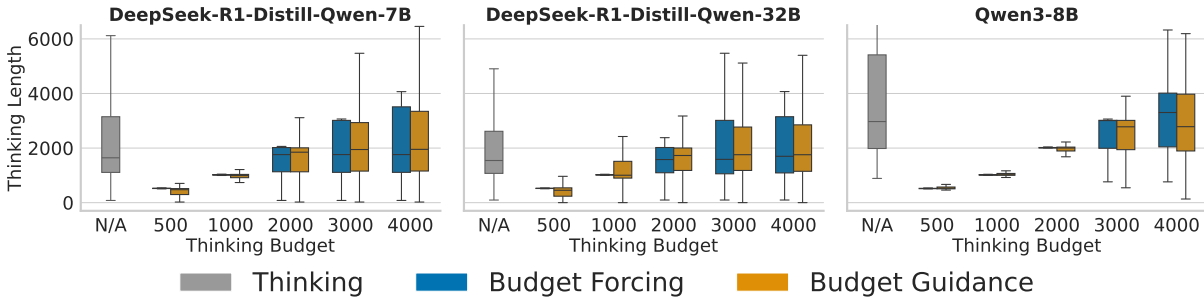


Figure 4: Thinking length controllability measured on MATH-500 benchmark.

model’s full thinking length and ensure the average thinking length (denoted as #Tokens) is comparable between our method and the baselines, and report the task accuracy. We also provide the comparison for various thinking lengths in Section 4.2.2.

Table 1 summarizes the evaluation results on math reasoning benchmarks. *Budget Guidance* consistently outperforms baselines under comparable average thinking lengths, effectively reducing the reasoning length without causing significant accuracy degradation. These improvements are consistent across different model sizes (7B to 32B) and model families (DeepSeek vs. Qwen3), highlighting the general applicability of our approach to diverse deep-thinking LLMs. Notably, even though the predictor for Qwen3-8B is trained on reasoning traces generated by DeepSeek-R1, it still performs well. This suggests that the training data can be *model-agnostic* if the target LLM exhibits a similar

reasoning style, *e.g.*, using words like “wait” or “alternatively” to structure its reasoning process.

#### 4.2.2 Accuracy–Thinking Length Tradeoff Analysis

A key indicator of effective control is the ability to achieve higher accuracy under the same thinking length, which we call *token efficiency*. To evaluate the token efficiency of our method across different reasoning lengths, we vary the token budget to obtain different average thinking lengths and record the corresponding accuracy achieved by the model. We visualize this relationship through accuracy–thinking length trade-off curves. Experiments are done on all three models across the four math benchmarks (Figure 3). We use *budget forcing* as our main baseline because it natively provides direct control over the thinking budget via a hard cutoff. We also report results for *NoThinking* as a

	<b>GPQA Diamond</b>		<b>FOLIO</b>		<b>TableBench</b>		<b>LiveCodeBench</b>	
	Acc.	#Tokens	Acc.	#Tokens	Acc.	#Tokens	Acc.	#Tokens
Thinking	49.1	5838	63.5	849	37.0	906	26.9	3509
NoThinking	38.4	-	46.3	-	16.9	-	20.7	-
Budget Forcing	39.9	1895	60.1	372	22.4	379	28.8	1135
<b>Budget Guidance</b>	<b>49.0</b>	1704	<b>61.6</b>	362	<b>26.7</b>	381	<b>29.4</b>	1138

Table 2: Evaluation on out-of-domain transferability.

reference point for performance without thinking.

From Figure 3, we observe that our method consistently achieves better token efficiency across most benchmarks, achieving higher accuracy than *budget forcing* under a range of thinking lengths. Notably, as the average thinking length decreases, corresponding to stricter budget constraints, our method yields significantly higher accuracy, particularly on benchmarks with diverse problem difficulty such as MATH-500. We attribute this to the ability of our method to adapt the reasoning pattern under strict budgets, producing concise yet complete reasoning traces. This enables the model to arrive at correct answers more efficiently, especially for questions that are relatively easy and do not require deep reasoning. This is also reflected in the occasional worse accuracy of *budget forcing* compared to the *NoThinking* baseline under strict budgets (e.g., MATH-500 on DS-7B/32B), where the reasoning trace is abruptly truncated, and the model is forced to guess prematurely. In contrast, our method avoids such incomplete reasoning and consistently outperforms the *NoThinking* baseline. An illustrative example of this guided reasoning behavior is provided in Section 4.4.

#### 4.2.3 Fine-Grained Control of Thinking Length

Our goal is to steer LLM reasoning to adhere to a specified thinking budget. To evaluate controllability, we test on MATH-500 under varying thinking budgets, measuring the actual thinking length per sample and visualizing the distributions. We compare our method to *budget forcing* and include the full-thinking baseline as a reference (Figure 4).

From Figure 4, we observe that our method behaves similar to *budget forcing*, generally respects the specified thinking budget: at least 75% of answers are within the budget, and the median thinking length closely aligns with the budget. Compared to the full-thinking baseline, our method guides the model to generate a budget-aligned rea-

soning trajectory. This behavior is notable because, unlike *budget forcing*, our approach does not enforce a hard cutoff. Instead, it softly steers the generation process to match the desired level of detail, demonstrating flexible and controllable reasoning.

#### 4.2.4 Out-of-Domain Transferability

For simplicity, we train the predictor solely on mathematical data and conduct experiments primarily on math tasks. We further seek to push the limits of this setting by exploring its generalization to broader task domains. To this end, we conduct an out-of-domain transferability analysis using the DS-7B model. Specifically, we evaluate our method on four benchmarks: **GPQA Diamond** (scientific reasoning), **FOLIO** (logical reasoning), the numerical reasoning subset from **TableBench** (tabular reasoning), and **LiveCodeBench** (code reasoning). We match the average reasoning length between our method and the baseline, and report the corresponding accuracies in Table 2.

Despite being trained exclusively on math data, our predictor generalizes well to non-math reasoning tasks. These results highlight the cross-domain generalizability of our approach and its potential applicability to a wide range of reasoning scenarios. While the gains on out-of-domain tasks are less pronounced than those on in-domain benchmarks, we believe performance can be further improved by incorporating reasoning traces from a broader range of domains during training. We leave this direction for future work.

#### 4.3 Insights into What the Predictor Learns

To probe what the predictor has learned, we analyze its estimated thinking length at the first thinking token, interpreted as the predicted number of thinking tokens needed, against **task difficulty** and **prompt type**, using the DS-7B model.

**Task Difficulty.** We evaluate on MATH-500 (in-domain) and LiveCodeBench (out-of-domain). Figure 6 shows that estimated thinking length in-

**Question:** The graph of  $f(x) = \frac{2x}{x^2 - 5x - 14}$  has vertical asymptotes  $x = a$  and  $x = b$ , and horizontal asymptote  $y = c$ . Find  $a + b + c$ .

**Thinking 1 (budget = 300)**

To determine the vertical asymptotes of the function  $f(x) = \frac{2x}{x^2 - 5x - 14}$ , I first need to find the values of  $x$  that make the denominator zero, as these will be the points where the function is undefined and thus where the vertical asymptotes occur.

I'll start by factoring the denominator:

$$x^2 - 5x - 14 = (x - 7)(x + 2)$$

Setting each factor equal to zero gives the solutions  $x = 7$  and  $x = -2$ . Therefore, the vertical asymptotes are at  $x = 7$  and  $x = -2$ . Next, to find the horizontal asymptote, I'll compare the degrees of the numerator and the denominator. The numerator has a degree of 1, and the denominator has a degree of 2. Since the degree of the denominator is higher, the horizontal asymptote is  $y = 0$ . Finally, I'll add the values of  $a$ ,  $b$ , and  $c$ :

$$a + b + c = 7 + (-2) + 0 = 5$$

(thinking length: 260)

**Thinking 2 (budget = 600)**

**Okay**, so I have this function here:  $f(x) = (2x)/(x^2 - 5x - 14)$ . I need to find the vertical asymptotes and the horizontal asymptote, and then add their  $x$ -values and the  $y$ -value together to get  $a + b + c$ . **Hmm**, let me think about how to approach this.

First, vertical asymptotes occur where the denominator is zero, right? Because the function will be undefined at those points, and the graph will shoot off to infinity or negative infinity. So, I need to factor the denominator to find those  $x$ -values.

The denominator is  $x^2 - 5x - 14$ . Let me try to factor that. I'm looking for two numbers that multiply to -14 and add up to -5. **Hmm, let's see.** The factors of 14 are 1 and 14, 2 and 7. Since the product is negative, one has to be positive and the other negative. Let me try 2 and -7:  $2 * (-7) = -14$ , and  $2 + (-7) = -5$ . Perfect! So, the denominator factors as  $(x + 2)(x - 7)$ . So, the denominator is zero when  $x + 2 = 0$  or  $x - 7 = 0$ , which means  $x = -2$  and  $x = 7$ . Therefore, the vertical asymptotes are at  $x = -2$  and  $x = 7$ . So,  $a$  and  $b$  are -2 and 7, respectively.

**Wait, hold on**, the problem says vertical asymptotes  $x = a$  and  $x = b$ . So,  $a$  is -2 and  $b$  is 7? Or is it the other way around? **Hmm**, actually, the order doesn't matter because we're just adding them together. So,  $a + b$  will be  $(-2) + 7$ , which is 5. Okay, that's straightforward.

Now, onto the horizontal asymptote. Horizontal asymptotes depend on the degrees of the numerator and the denominator. The numerator is  $2x$ , which is degree 1, and the denominator is  $x^2 - 5x - 14$ , which is degree 2. So, the degree of the numerator is less than the degree of the denominator. I remember that when the degree of the numerator is less than the degree of the denominator, the horizontal asymptote is  $y = 0$ . So,  $c$  is 0.

**Double-checking:** If the degree of the numerator is less than the denominator, horizontal asymptote is  $y = 0$ . If they are equal, it's the ratio of the leading coefficients. If the numerator is higher, there's no horizontal asymptote, but maybe an oblique one. So, in this case, since numerator is degree 1 and denominator is degree 2, horizontal asymptote is  $y = 0$ . So,  $c = 0$ .

(thinking length: 602)

Figure 5: Sample reasoning traces generated with *budget guidance* under different thinking budgets.

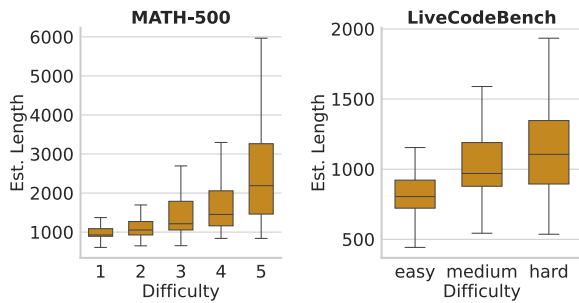


Figure 6: Correlation between question difficulties and estimated thinking lengths.

creases with difficulty in both cases. This suggests that the predictor captures a general understanding of difficulty, enabling effective difficulty estimation.

**Prompt Type.** We evaluate on MATH-500 and compare two prompts: one encouraging long reasoning and one encouraging concise reasoning (listed in Appendix H). As shown in Figure 7, the long reasoning prompt yields longer estimated thinking lengths. A t-test gives a  $p$ -value of 0.0028, confirming the difference is statistically significant and indicating that the predictor is prompt-aware.

#### 4.4 Case Study

Figure 5 shows a case study from MATH-500 illustrating reasoning traces under different thinking budgets. Rather than truncating output, our method adapts the reasoning style to the budget. With a stricter budget (left), the model generates

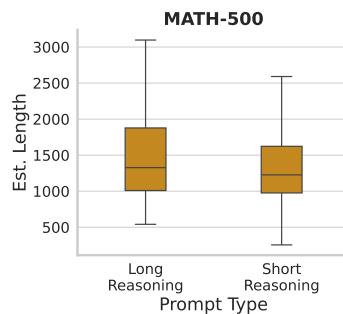


Figure 7: Correlation between prompt types and estimated thinking lengths. The prompts are provided in Appendix H.

concise answers without reflection. With a larger budget (right), it mirrors full-length reasoning: it starts with problem analysis and using reflective phrases like “wait” and “double-checking.” In both settings, the trace ends appropriately, highlighting our method’s flexibility and controllability.

## 5 Conclusion

*Budget Guidance* enables natural control over reasoning length without requiring any LLM fine-tuning, while substantially improving token efficiency on challenging benchmarks. These findings demonstrate that the classifier-guidance paradigm from vision research can be successfully adapted to NLP, and further highlight budget-conditioned next-token generation as a promising direction for efficient and controllable LLM reasoning.

## Limitations

Our predictor is trained primarily on math-domain reasoning traces; although it demonstrates encouraging cross-domain transfer, the performance improvements outside mathematics are comparatively smaller, suggesting that incorporating a broader range of reasoning domains during training may further enhance robustness and generalization. Moreover, our method assumes that the remaining-length distribution follows a Gamma parameterization, which simplifies estimation but may introduce modeling bias and potential risk of incorrect or biased reasoning trace, motivating future work on more expressive and accurate distributional modelings.

## Ethical considerations

This work proposes a test-time steering method for controlling the length of model reasoning traces in order to improve computational efficiency. The method does not modify model parameters or collect new data, and all experiments are conducted on publicly available benchmarks without human subjects or personally identifiable information. As such, the study does not directly raise privacy or data-collection risks.

However, we acknowledge several potential ethical considerations. The ability to shorten or reshape reasoning traces may affect transparency in contexts where full reasoning steps are expected, and the predictor is trained primarily on math-domain traces, which may introduce domain-specific biases or uneven generalization. Therefore, the method should not be applied in high-stakes or real-world decision-making scenarios without additional validation and safety review. Our evaluations are restricted to offline research settings, and future deployments should carefully assess domain appropriateness and communicate when reasoning traces are modified for efficiency.

## References

Art of Problem Solving. n.d.a. [Aime problems and solutions](#).

Art of Problem Solving. n.d.b. [Aimc 12 problems and solutions](#).

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025. Seal: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2504.07986*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. 2000. Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 13.

Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. 2025. Reasoning without self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

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

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

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

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng 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. *arXiv preprint arXiv:2402.14008*.

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

- Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.
- Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.
- HuggingFace. 2025. [Open r1: A fully open reproduction of deepseek-r1](#).
- 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. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- 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*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.
- Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*.
- 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*.
- Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.
- Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, and 1 others. 2025. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506.
- 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, Anpeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.
- Zhaojian Yu, Yinghao Wu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. 2025. Z1: Efficient test-time scaling with code. *arXiv preprint arXiv:2504.00810*.

## A Descriptive Comparison for Baselines

In Table 3, we present an additional descriptive comparison between the baselines and our method from four perspectives: whether the method requires LLM training or fine-tuning, whether the control mechanism relies on prompt engineering, whether it provides direct control over the thinking length (i.e., the method natively supports specifying a thinking budget as input), and whether the resulting reasoning trace is complete.

As we can see, *NoThinking*, *Chain-of-Draft*, and *TALE-EP* are prompt-based methods and thus heavily rely on rigorous prompt engineering for good performance, and they do not control the reasoning length well according to our experiments. Although *SEAL* does not rely on prompt engineering and uses a steering vector to guide the reasoning, it does not provide a direct token budget option to control the reasoning under a specific budget. Both *Dynasor* and *Budget Forcing* offer direct budget control; however, their controllability stems from early stopping and hard cutoffs, which prevent the full reasoning trace from being preserved. Our method is both LLM-finetuning-free and prompt-engineering-free, provides direct budget control, and preserves the full reasoning trace by softly steering the entire reasoning process to meet the budget, making it a unique, elegant, and higher-accuracy solution for budget-conditioned generation.

## B Predictor Latency Overhead Analysis

To better quantify the latency overhead introduced by our predictor, we conduct precise measurements on three representative models using the HuggingFace transformers library on a single NVIDIA H100 GPU. We generate 1000 thinking tokens per run on the full AIME-2024 test set, with each question executed 10 times to compute an average latency. Final results are averaged across questions, and we report 95% confidence intervals for clarity.

As shown in Table 4, the latency overhead for all three models is below 0.8%, and decreases further for larger models (only 0.17% for a 32B LLM). This demonstrates that our predictor introduces negligible additional cost relative to standard inference, and thus does not compromise the efficiency of the overall system.

## C Ablation on Skipping Strategies

We further conduct an ablation study to examine different strategies for applying budget guidance. We consider three settings:

- Apply the modulation at the beginning of every sentence.
- Apply the modulation at every token (no skipping).
- Apply the modulation at the beginning of every paragraph (this is the strategy used in our main paper).

We evaluate these strategies on the MATH-500 benchmark using the DeepSeek-R1-Distill-Qwen-7B model. Results are summarized in Table 5.

We observe that there is no significant difference between applying guidance at the beginning of a sentence or a paragraph. Since the paragraph-level strategy achieves comparable accuracy while saving more computation, we adopt it in our main experiments. Interestingly, applying guidance at every token leads to lower accuracy. Closer inspection shows that guidance applied mid-sentence is less stable, often disrupting semantic coherence. This instability can accumulate across token generations and result in random or repeated outputs. For both inference stability and efficiency, we therefore do not recommend modulating every token.

## D Ablation on Predictor Architecture

We also examine whether a more lightweight alternative to the BERT-based length predictor is feasible. Specifically, we remove the BERT encoder and use only a single linear layer to predict the parameters of the Gamma distribution. We apply the same token budget to this variant and evaluate its performance on the MATH-500 benchmark.

As shown in Table 6, the BERT-based predictor clearly outperforms the simple linear layer. In particular, the linear layer is less effective in controlling reasoning length, as evidenced by its longer outputs compared to the BERT-based predictor under the same budget. This suggests that estimating the remaining reasoning length is a challenging task, and a larger model such as BERT offers stronger predictive capacity. There exists a trade-off between predictor accuracy and efficiency; in this paper, we adopt the BERT-based predictor due to its robust performance and widespread use across many tasks.

Method	Freeze LLM	Prompt-Free Control	Direct Budget Control	Preserve Full Think Trace
NoThinking	✓	✗	✗	✗
Chain-of-Draft	✓	✗	✗	✓
TALE-EP	✓	✗	✗	✓
SEAL	✓	✓	✗	✓
Dynasor	✓	✓	✓	✗
Budget Forcing	✓	✓	✓	✗
<b>Budget Guidance</b>	✓	✓	✓	✓

Table 3: Comparison between the baselines and our method.

Model	Latency Overhead	95% Confidence Interval
DeepSeek-R1-Distill-Qwen-7B	0.72%	[0.60%, 0.83%]
DeepSeek-R1-Distill-Qwen-32B	0.17%	[0.12%, 0.21%]
Qwen3-8B	0.48%	[0.27%, 0.68%]

Table 4: Measured latency overhead of our predictor across different LLMs.

Strategy	Accuracy (%)	#Tokens
start of sentence	88.0	1333
every token	86.0	1448
start of paragraph	<b>88.2</b>	<b>1329</b>

Table 5: Ablation study on different skipping strategies for applying budget guidance.

Predictor	MATH-500 Acc.	#Tokens
Linear Layer	85.8	1617
BERT	<b>88.2</b>	<b>1329</b>

Table 6: Comparison of predictor architectures on the MATH-500 benchmark.

## E Analysis of Interventions on Samples with Different Reasoning Length Requirements

To further understand the behavior of Budget Guidance, we analyze its effect on samples requiring shorter versus longer reasoning traces. We compute the relative percentage change in reasoning length on the MATH-500 dataset using the DeepSeek-R1-Distill-Qwen-7B model, defined as

$$\Delta = \frac{|\text{length}_{\text{vanilla}} - \text{length}_{\text{BG}}|}{\text{length}_{\text{vanilla}}}$$

We find that Budget Guidance more significantly reduces reasoning length for samples originally above the budget, while having minimal effect on those below. Notably, 100% of samples already under the budget remain correct, indicating no accuracy loss in these cases.

Sample Category	Change in Length
Short reasoning	15.3%
Long reasoning	52.3%

Table 7: Length change for samples with different reasoning requirements on MATH-500.

## F Quantitative Reasoning Behavior Analysis

To quantitatively analyze how the predictor influences the reasoning behavior of LLMs under different budget settings, we follow the methodology proposed by (Hou et al., 2025). Specifically, we count the frequency of reasoning-related keywords such as “wait” and “alternatively”, which are indicative of deeper reasoning processes. We compare the keyword frequencies for thinking budget of 500, 2000, and 4000 tokens using the DS-7B model on the MATH-500 benchmark. These results are contrasted with a full-thinking baseline (*i.e.*, without applying our method). The comparison is illustrated in Figure 8.

As shown in the figure, a smaller budget substantially reduces the frequency of reasoning-related keywords, indicating a more concise reasoning process. As the budget increases, the model is encouraged to engage in deeper reasoning. Notably, when the budget is set sufficiently high, the behavior closely matches that of the full-thinking baseline, suggesting minimal loss in reasoning capability. These findings demonstrate that our method can effectively steer the reasoning behavior of LLMs, while still preserving their reasoning ability under

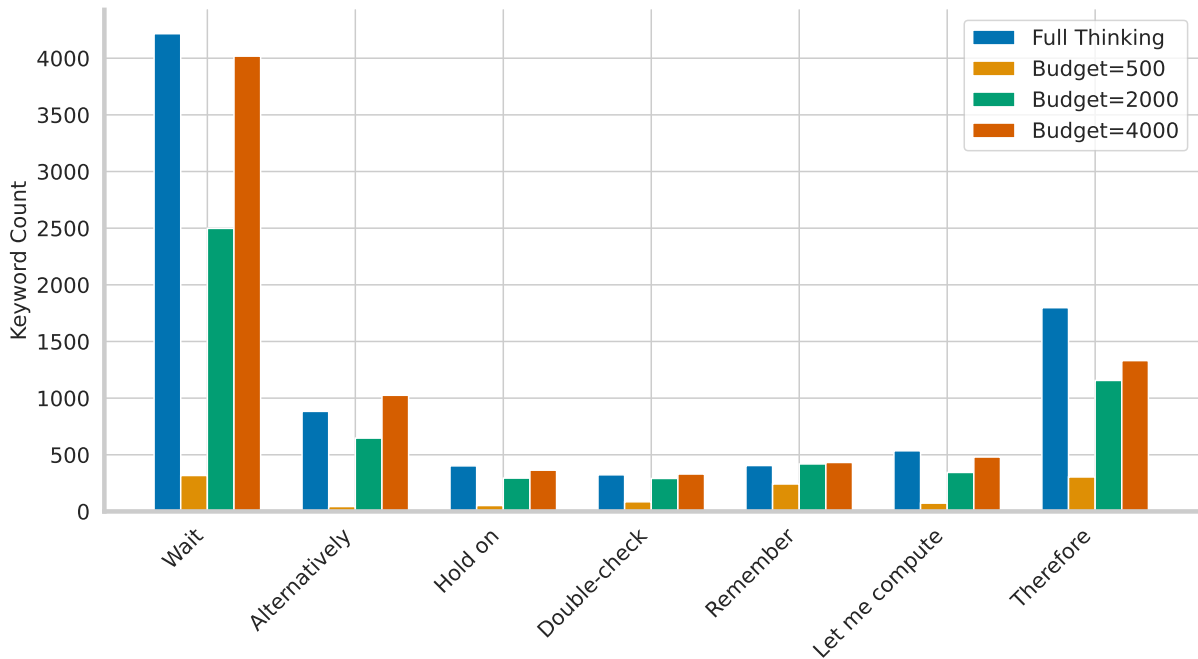


Figure 8: Reasoning keywords frequency comparison under different budget settings.

higher budget constraints.

## G Dataset Description

We provide detailed information about the evaluation datasets used in our paper.

**MATH-500** (Hendrycks et al., 2021) is a 500-problem subset of the MATH dataset, selected by (Lightman et al., 2023). Each problem is labeled with a difficulty level from 1 to 5.

**AIME-2024** (Art of Problem Solving, n.d.a) contains 30 problems from the 2024 American Invitational Mathematics Examination, covering topics such as algebra, combinatorics, geometry, number theory, and probability. Following *budget forcing* (Muennighoff et al., 2025), we retain only the essential ASY figure code required to solve each problem, omitting non-essential diagrams.

**AMC** (Art of Problem Solving, n.d.b) includes all 83 problems from AMC12 2022 and AMC12 2023.

**OlympiadBench** (He et al., 2024) is a challenging benchmark aimed at advancing AGI through Olympiad-level, bilingual, multimodal scientific problems. We use its math subset, which contains a total of 675 problems.

**GPQA Diamond** (Rein et al., 2024) consists of 198 high-quality, extremely difficult questions spanning a broad range of scientific domains, including biology, physics, and chemistry.

**FOLIO** (Han et al., 2022) is a human-annotated dataset designed to evaluate complex logical rea-

soning in natural language. It features 1,430 unique conclusions paired with 487 sets of premises, all validated using first-order logic (FOL) annotations. We use the test set, which contains 203 unique problems.

**TableBench** (Wu et al., 2025) is a benchmark for evaluating LLMs on real-world tabular data challenges. We evaluate all models on the numerical reasoning subset, which comprises 493 problems.

**LiveCodeBench** (Jain et al., 2024) offers a holistic and contamination-free evaluation of LLM coding capabilities. Following (Guo et al., 2025), we select problems from the August 2024 to January 2025 period, totaling 323 problems.

## H Prompt Description

In Section 4.3, we analyze the predictor’s estimated thinking length across different prompt types to demonstrate its prompt awareness. Below, we list the specific prompts used in our experiment.

The prompt for long reasoning is: Think step by step and provide thorough reasoning before reaching a conclusion.

The prompt for short reasoning is: Think quickly and provide a concise reasoning with minimal steps.

We add these prompts as the system prompt.

## **I Use of Large Language Models for Writing Assistance**

Portions of the writing in this paper, specifically at the level of grammar refinement, sentence polishing, and shortening of paragraphs for conciseness, were assisted by an external large language model (OpenAI ChatGPT). The model was not used to generate original ideas, experimental design, or analysis; all scientific contributions are the authors' own. The assistance was limited to improving clarity, readability, and presentation quality of the manuscript.