

BUDGETTHINKER: EMPOWERING BUDGET-AWARE LLM REASONING WITH CONTROL TOKENS

Anonymous authors

Paper under double-blind review

ABSTRACT

Recent advancements in Large Language Models (LLMs) have leveraged increased test-time computation to enhance reasoning capabilities, a strategy that, while effective, incurs significant latency and resource costs, limiting their applicability in real-world time-constrained or cost-sensitive scenarios. This paper introduces BudgetThinker, a novel framework designed to empower LLMs with budget-aware reasoning, enabling precise control over the length of their thought processes. We propose a methodology that periodically inserts special control tokens during inference to continuously inform the model of its remaining token budget. This approach is coupled with a comprehensive two-stage training pipeline, beginning with Supervised Fine-Tuning (SFT) to familiarize the model with budget constraints, followed by a curriculum-based Reinforcement Learning (RL) phase that utilizes a length-aware reward function to optimize for both accuracy and budget adherence. We demonstrate that BudgetThinker significantly surpasses strong baselines in maintaining performance across a variety of reasoning budgets on challenging mathematical benchmarks and multimodal reasoning benchmarks. Our method provides a scalable and effective solution for developing efficient and controllable LLM reasoning, making advanced models more practical for deployment in resource-constrained and real-time environments.

1 INTRODUCTION

A recent breakthrough in Large Language Models (LLMs) is the paradigm of test-time scaling, where models are prompted to “think longer” by generating extended Chain-of-Thought (CoT) reasoning before producing a final answer (DeepSeek-AI, 2025; OpenAI, 2024). This approach has achieved state-of-the-art performance on complex reasoning tasks in domains such as advanced mathematics (DeepSeek-AI, 2025; Google, 2025), code generation (Hui et al., 2024; Yang et al., 2025), and multimodal reasoning (Team et al., 2025a; Huang et al., 2025; Shen et al., 2025). However, these performance gains come at the steep cost of increased latency and computational overhead, making these reasoning models impractical for real-world systems with strict performance constraints, such as latency-critical agents and robotics (Sui et al., 2025).

This fundamental trade-off between reasoning quality and computational efficiency highlights a critical need for *controllable CoT reasoning*—the ability for a reasoning LLM to dynamically adapt its reasoning process to conclude within a specified budget. Such a budget may be a hard latency limit imposed by an application, such as the sub-100 ms requirement for autonomous driving (Lin et al., 2018; Chen et al., 2025a), or a flexible constraint based on user tolerance for cost and delay in agentic systems. Furthermore, the variability of computational resources in real-world deployments (*e.g.*, diverse hardware, dynamic server loads) necessitates that models can precisely tailor the length of their output to fully utilize the available budget (Cai et al., 2020; Yu et al., 2019). We term this crucial instruction-following capability **budget adherence**.

However, existing approaches to shortening reasoning length have proven insufficient for budget adherence. Directly adding length control instructions into the prompt (Pu et al., 2025; Sun et al., 2025; Xu et al., 2025b) often fails to reliably control output length (Han et al., 2024). Some LLM post-training methods encourage models to switch between reasoning modes like “thinking” or “non-thinking”, or to generate responses based on reasoning efforts (Yang et al., 2025; OpenAI, 2025b;a; Team et al., 2025d; Jiang et al., 2025; Team et al., 2025a). However, these approaches still lack the

054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

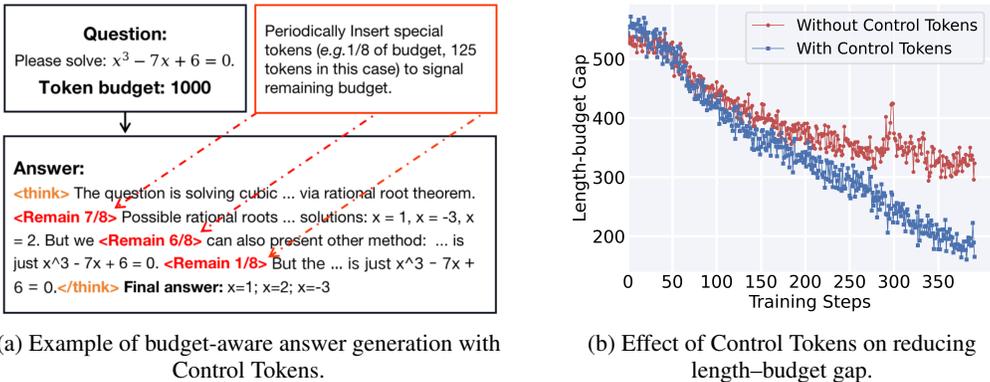


Figure 1: Illustration of the BudgetThinker framework and the impact of Control Tokens. (a) An example of budget-aware generation, showing how BudgetThinker inserts control tokens into its reasoning process to stay aware of the remaining token budget. (b) Effect of Control Tokens on training: The gap between the generated answer length and the target budget during reinforcement learning.

fine-grained control necessary for variable budgets, where the target is a fixed number set by the user or system, not a vague description of a reasoning level. Even recent research introducing training pipelines with Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) (Hou et al., 2025; Aggarwal & Welleck, 2025; Wu et al., 2025b; Liu et al., 2025; Xia et al., 2025; Qu et al., 2025; Yu et al., 2025c) demonstrates insufficient capability. As illustrated in Figure 1b, the gap between the LLM-generated length and the target budget remains large (around 300 tokens) throughout training, highlighting a persistent challenge.

In this paper, we address the challenge of precisely controlling the length of a model’s thought process. We argue that merely stating a budget constraint in the initial prompt is insufficient, and the model needs to be continuously reminded of its remaining token budget as it generates its response. To achieve this, we develop a novel training and inference framework that enables precise control over the length of the CoT. Our method periodically inserts special *Control Tokens* that act as explicit signals of the remaining budget, as illustrated in Figure 1a. As shown in Figure 1b, training with these control tokens leads to a faster and more stable reduction in the gap between the generated length and the target budget, demonstrating their effectiveness in stimulating budget adherence.

We evaluate our method based on DeepSeek distilled Qwen 2.5 1.5B, 7B (DeepSeek-AI, 2025) and RoboBrain-7B-2.0 (Team et al., 2025a), on challenging reasoning benchmarks including MATH-500 (Hendrycks et al., 2021), AMC 2023, Embodied Reasoning QA Evaluation Dataset (EQRA) (Team et al., 2025b), and Spatial Aptitude Training Benchmark (SAT Benchmark) (Ray et al., 2025). Compared to state-of-the-art efficient reasoning methods (Hou et al., 2025; Liu et al., 2025; Xu et al., 2025b; Sun et al., 2025) and original models, our approach demonstrates a better trade-off between accuracy and token budget. Specifically, our method improves accuracy by an average of 4.9% across all tested budgets and exhibits more precise adherence to the specified constraints.

Our main contributions are:

- We introduce a method to achieve controllable output length in LLMs by inserting special tokens during the inference process.
- We design a comprehensive training process, progressing from SFT to RL, which incorporates a length-aware reward function. This allows the model to adapt to control via special tokens and produce higher-quality, budget-aware outputs. Our method is designed as a plug-and-play module that complements existing test-time scaling training procedures.
- Across multiple foundational models (including both LLMs and MLLMs), we demonstrate that our approach surpasses strong baselines and datasets under various reasoning budgets.

2 SPECIAL-TOKEN CONDITIONED BUDGET-AWARE REASONING

We present BudgetThinker, a budget-aware generation framework that guides Large Language Models (LLMs) to adhere to a predefined token budget. Our method achieves this by conditioning the model on explicit control signals throughout the generation process. We enhance budget adherence by (i) introducing a dynamic control token insertion strategy that continuously informs the model of the remaining budget and (ii) developing a two-stage training pipeline that combines Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) to teach the model to effectively interpret and respond to these signals.

2.1 SPECIAL-TOKEN BUDGET SIGNALING

Our method enforces a target output token budget by augmenting the standard autoregressive generation process. The core idea is to insert a small, fixed set of special control tokens into the sequence at positions corresponding to **fractions of the total budget**. These tokens act as explicit signals to the model, indicating how much of its budget has been consumed. Let B be the target budget and K be a fixed hyperparameter for the number of control signals. We introduce a set of K special control tokens, $\mathcal{C} = \{c_1, c_2, \dots, c_K\}$, where c_k signals that the generation has progressed through the k -th fraction of the total budget (*e.g.*, c_1 signifies 10% completion if $K=10$). During generation, the token y_t at timestep t is determined by the following piecewise function:

$$y_t = \begin{cases} c_k & \text{if } t = k \cdot \lfloor B/K \rfloor \text{ for an integer } k \in \{1, \dots, K\} \\ y'_t \mid (y'_t \sim \pi_\theta(\cdot \mid x, y_{<t})) & \text{otherwise} \end{cases}$$

At each timestep t that corresponds to the end of a budget fraction $k \cdot \lfloor B/K \rfloor$, we deterministically insert the corresponding control token c_k . For all other timesteps, the next token y'_t is sampled from the LLM’s distribution π_θ , conditioned on the full history of both model-generated tokens and our inserted control tokens.

This **ratio-based signaling** is highly scalable. In contrast, a naive baseline like Fixed-Interval Signaling—which inserts a token every I steps—would require a potentially unbounded number of unique control tokens as the budget B increases. Our approach always uses the same fixed set of K tokens, making it robust and efficient across widely varying budgets.

2.2 SFT DATA CURATION

To teach the LLM the semantics of our control tokens, we construct a specialized dataset for Supervised Fine-Tuning (SFT). The curation process involves two key steps: budget assignment and dataset balancing.

Budget Assignment. First, for each original data sample (x, y) , we assign a budget B by rounding the answer’s length, $|y|$, up to the nearest multiple of a granularity parameter T :

$$B = T \cdot \left\lceil \frac{|y|}{T} \right\rceil,$$

This calculation method is intentional. By design, the true answer length $|y|$ is almost always less than the allocated budget $|B|$. This gap teaches the model to robustly terminate its reasoning and stop generating tokens once the answer is complete, rather than artificially padding the output to fill the entire budget. Next, we transform each sample in the dataset. The input prompt is modified to include the budget constraint, and the target output is reconstructed to include the control tokens:

$$\hat{x} = \text{Concat}(x, \text{Please answer within } B_i \text{ tokens})$$

$$\hat{y} = \text{Insert}(y, \{c_j\}_{j=1}^K)$$

Here, *Insert* function adds the control token c_{k+1} at each position t when $t = k \cdot I$. Then, we get a reconstructed dataset $\{(\hat{x}, \hat{y})\}$, which we use to fine-tune LLM.

2.3 DATASET CONSTRUCTION AND BALANCING

To ensure the model generalizes across a wide range of output lengths, our SFT dataset is a balanced mixture of three distinct data types:

(1) Long Reasoning Data: Full-length CoT reasoning sequences from powerful teacher models, such as DeepSeek-R1 (DeepSeek-AI, 2025). **(2) Iteratively Compressed Reasoning Data:** To populate intermediate lengths, we use an LLM-based compressor to iteratively shorten the long CoT sequences (Kang et al., 2025). Critically, this compression is applied iteratively: the result of one compression step becomes the input for the next. This process continues until a comprehensive spectrum of shortened reasoning chains, derived from an original long CoT, is produced. This approach ensures our model learns to adhere to budgets requiring diverse levels of reasoning abbreviation. **(3) Non-thinking Data:** Direct, concise answers to handle scenarios with very tight budget constraints where elaborate reasoning is not feasible. This balanced composition is pivotal for developing a versatile and robustly budget-adherent model.

2.4 REINFORCEMENT LEARNING

Following SFT, we employ Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024) to train the model to generate responses that adhere to a specific token budget, B .

Reward Design. We design a composite reward function that balances factual accuracy with a penalty for deviating from the target length. For a generated response y , with ground truth y_{gold} and budget B , the reward R is defined as:

$$R(y, y_{gold}, B) = k_1 \cdot \mathbf{1}\{y = y_{gold}\} + k_2 \cdot \mathbf{1}\{\text{format}(y)\} + k_3 \cdot \underbrace{\max(1 - \gamma \cdot \left(\frac{B - |y|}{B}\right)^2, 0)}_{\text{Length Reward}}$$

Indicator functions $\mathbf{1}\{y = y_{gold}\}$ and $\mathbf{1}\{\text{format}(y)\}$ are the original correctness and format check of GRPO, and k_1, k_2, k_3 are coefficients that control the relative importance of each part. The *Length Reward* incentivizes the model to use the budget efficiently. It is based on *normalized length deviation*, $\delta = \frac{||y| - B|}{B}$, which measures the fractional difference between the generated length $|y|$ and the target budget B . This deviation is modulated by an *asymmetric penalty coefficient*, γ :

$$\gamma = \begin{cases} 1 & \text{if } |y| \leq B \\ r & \text{if } |y| > B \end{cases}$$

Here, $r > 1$ is a hyperparameter that imposes a larger penalty for exceeding the budget than for falling short of it. This design encourages the model to generate responses that are not only correct but also precisely tailored to the length constraint.

Special Token Conditioned Sampling. Our framework’s special token insertion is integrated directly into the GRPO sampling loop. For each training question, BudgetThinker automatically injects control tokens at the appropriate budget fractions during generation. To provide a clear learning signal, the model completes its full reasoning rollout to calculate the length-based reward accurately. However, for the final answer extraction step, the reasoning trace is truncated at budget B . This allows the model to learn from the consequences of its full generation while still enforcing the hard budget constraint in practice.

Mixed Budget Reinforcement Learning. To build robustness across budgets, RL proceeds in two stages: (1) *Curriculum Stage:* Our LLM is trained iteratively on k different token budgets. The curriculum proceeds by decreasing the budget across stages, following the sequence $B_1 > B_2 > \dots > B_n$. (2) *Randomized Stage:* Each Training batch samples a budget B_k uniformly from $B_{j=1}^n$. This mixed strategy prevents catastrophic forgetting and maintains the model’s proficiency across all learned budget levels.

3 EXPERIMENTS

In this section, we present a comprehensive empirical evaluation of BudgetThinker. We investigate its performance under various computational budgets and analyze its ability to adhere to these budgets (Section 3.2). We then analyze the impact of different control token insertion strategies (Section 3.3), iterative training (Section 3.4), and reinforcement learning of BudgetThinker (Section 3.5).

3.1 EXPERIMENT SETUP

LLM Training Details. We use DeepSeek-R1 (DeepSeek-AI, 2025) distilled Qwen-2.5-1.5B and Qwen-2.5-7B (Hui et al., 2024) as backbone models. For SFT, we create a dataset of 41k instances for text-based LLMs. The details are listed in Appendix A.2. We train all SFT models for 6 epochs with an initial learning rate of $2e-5$ and a maximum context length of 12000.

For Reinforcement Learning (RL), we employ the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024). Models are trained for 10 epochs with GRPO hyperparameters α and β set to 0.01 and 0.01, respectively. The maximum context length during RL is 10000.

We configure BudgetThinker by setting the number of control intervals to $K = 8$. For the iterative training stage, we start from a budget of 6000, then reducing it to 4000, 3000, and finally 2000, with 1 epoch for each budget. In the randomized training stage, we randomly sampling $B \in \{6000, 4000, 3000, 2000\}$ for 1 epoch. Further training details are shown in Appendix A.5.

Namely, the length reward will decay to 0 when the length of answer y exceeds the budget B by $1/4$.

MLLM Training Details. We use RoboBrain-7B-2.0 (Team et al., 2025a) as the backbone model. For SFT, we use the visual question answering (VQA) data with CoT from RoboBrain-2.0, consisting of 37k question-answer pairs. In RoboBrain-2.0, the CoT data is generated from GPT-4o (Hurst et al., 2024), the length of which concentrates in 200-300 tokens. Therefore, we use the compressor based on GPT-4o to compress the CoT length for 3 iterations, creating a dataset of 111k.

For RL, we randomly sample 2k instances from the VQA training dataset of RoboBrain-2.0, train the MLLM for 6 epochs, with the initial learning rate of $1e-6$. Specifically, in the We set the number of control intervals to $K = 3$ because the shorter reasoning length on MLLMs, and the training budgets $B \in \{200, 150, 100, 80, 50\}$ for the iterative training phase, and train 1 epoch on each budget. Then, we train 1 epoch for mixed budget stage. We remain the other hyper-parameters same with LLMs.

For fair comparison, we also train the original RoboBrain-7B-2.0 through the same procedure from SFT to RL using the same data and settings, with the difference that we don't add budget constraints during training and set the maximum context length to 500, which is far beyond the largest budget 200 in our experiments.

Evaluation Details. We evaluate our methods and baselines on 2 math benchmarks and 2 multi-modal reasoning benchmarks: MATH-500 (Hendrycks et al., 2021), AMC 2023, ERQA (Team et al., 2025b), and SAT (Ray et al., 2025). To assess budget control, we evaluate performance across a range of reasoning budgets, from $B = 50$ to $B = 10000$ tokens. For inference, we utilize a modified version of the vLLM engine (Kwon et al., 2023). Our modified engine enforces the budget by truncating the reasoning process once the length limit is reached and appending the “</think>***Final Answer***” tags to signal the model for a final answer. We then allocate an additional 50 tokens for the final answer generation before terminating the process entirely.

Baselines. We benchmark BudgetThinker against five baselines: **(1) ThinkPrune** (Hou et al., 2025): A typical efficient reasoning framework that also uses GRPO for budget control but does not incorporate explicit control tokens. As the official ThinkPrune repository does not provide a 7B model, we limit our comparison to the 1.5B model scale. **(2) Laser** (Liu et al., 2025): A state-of-the-art efficient reasoning framework enhances both fast and slow thinking capability. **(3) Chain of Draft** (Xu et al., 2025b): A prompt based method that facilitates fast reasoning by prompting LLMs to draft concise intermediate thought. **(4) Early Exit**: (Sun et al., 2025) A token insertion based method that interrupts LLMs before budget and forces it to output the final answer. **(5) Original**

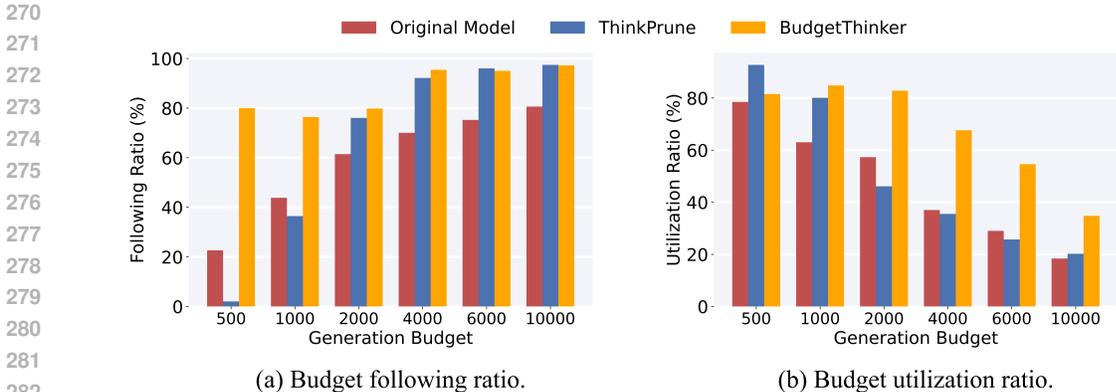


Figure 2: Budget following and utilization analysis of BudgetThinker-1.5B, ThinkPrune-1.5B, and the R1-1.5B on MATH-500. (a): The percentage of generated responses that terminate naturally within the specified budget limit (B). (b): For responses that finish within the budget, this shows the average generated length as a percentage of the total allocated budget ($|y|/B$).

Models: The DeepSeek-R1 distilled Qwen-2.5-1.5B and 7B models that serve as the foundation for our fine-tuning.

3.2 ACCURACY-BUDGET TRADEOFF

1.5B LLMs											
Method	MATH-500						AMC				
	0.5K	1K	2K	4K	6K	10K	2K	4K	6K	10K	30K
ThinkPrune	44.0	69.4	79.4	82.8	83.3	83.6	65.5	70.1	71.9	72.0	73.6
Laser	48.4	70.2	80.2	82.6	83.8	83.2	59.0	67.8	73.4	75.9	75.9
Original Model	20.4	45.8	63.0	70.0	69.2	70.2	36.9	46.9	55.6	61.6	67.8
c2f	56.8	70.0	77.4	80.8	81.8	80.0	54.4	65.9	62.5	64.7	68.3
CoD	38.0	59.4	64.2	64.8	65.2	65.2	33.4	46.9	48.1	49.4	66.1
BudgetThinker	59.4	75.0	81.6	84.8	85.8	87.4	65.5	70.4	74.0	75.0	75.9
7B LLMs											
Method	MATH-500						AMC				
	0.5K	1K	2K	4K	6K	10K	2K	4K	6K	10K	30K
Laser	53.8	80.0	88.8	90.6	92.0	92.2	67.8	83.8	86.3	87.5	88.1
Original Model	22.2	44.4	67.4	78.8	84.0	86.0	38.8	58.8	73.1	78.4	79.3
c2f	59.8	76.4	84.0	89.6	91.0	90.2	62.8	76.6	87.2	87.2	87.3
CoD	31.6	57.8	73.4	81.6	84.6	85.8	47.8	69.1	74.7	79.1	79.8
BudgetThinker	69.4	77.0	86.0	90.0	92.2	93.0	69.7	77.5	87.5	87.5	90.0
7B MLLMs											
Method	ERQA					SAT					
	50	80	100	150	200	50	80	100	150	200	
c2f	4.5	6.5	10.0	24.6	39.6	23.0	55.0	66.0	69.0	74.6	
Laser	37.1	38.8	39.6	40.4	42.3	69.7	70.0	71.0	73.3	73.3	
BudgetThinker	41.1	41.9	43.6	43.4	43.6	70.7	72.0	72.7	74.0	76.7	

Table 1: Pass@1 accuracy of BudgetThinker vs. baselines across various generation budgets. The numeric headers (e.g., **0.5K**, ..., **30K**, **50**, ..., **80**) represent the Generation Budget.

We first evaluate the accuracy of BudgetThinker against the baselines across various token budgets, with results shown in Table 1. For results of MATH-500 and AMC 2023, BudgetThinker outperforms the original model and ThinkPrune across most budget allocations. Specifically, Bud-

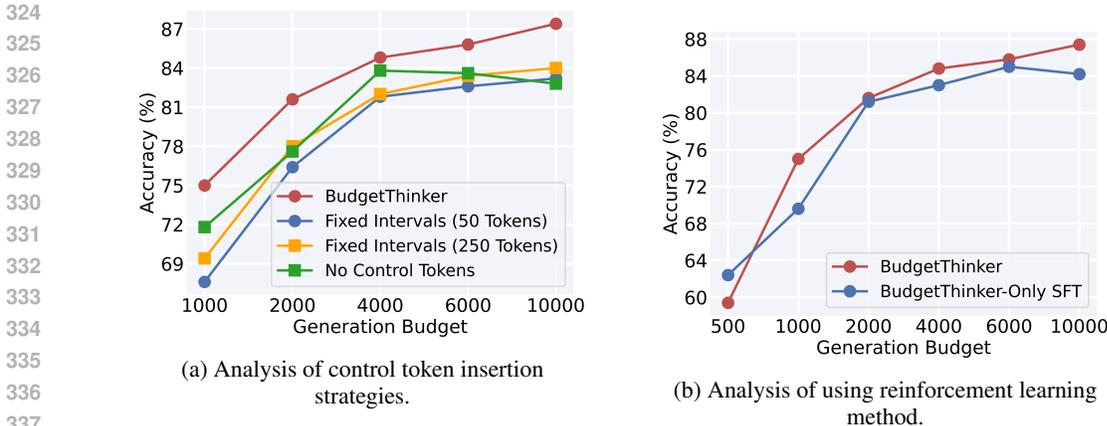


Figure 3: Ablation studies on control strategy and RL training (MATH-500, 1.5B Model).

getThinker improves accuracy by 2.1% over Laser, 3.2% over ThinkPrune and 22.4% over Original model on average.

The better accuracy of BudgetThinker comes from the better understanding and utilization of budget, which is shown in Figure 2. We calculate the average relative length at each budget ($|y|/B$) for every answer that is within budget, which shows the capability for LLM to understand and fully use budgets. The original model, lacking any budget-specific training, exhibits poor capability to follow the specified limits. While ThinkPrune (Hou et al., 2025) shows improved budget awareness, it often underutilizes the allocated budget, prematurely concluding its reasoning process, which can negatively impact performance on complex problems. In contrast, BudgetThinker demonstrates superior budget adherence. Guided by the remaining budget tokens, it effectively utilizes the entire allocated length without significant overruns. This precise control allows it to achieve a better balance between computational cost and accuracy, leading to its state-of-the-art performance. We offer a detailed analysis in Appendix A.7.

3.3 ANALYSIS OF CONTROL TOKEN INSERTION STRATEGIES

We analyze the performance of BudgetThinker under four distinct control token insertion strategies to understand their impact:

- **No Control Tokens:** An ablation where no special tokens are inserted, relying solely on the RL objective for budget control.
- **Fixed Interval (50 tokens):** Control tokens are inserted at a fixed, frequent interval of every 50 tokens, with 200 special tokens added for a maximum budget of 10000 tokens.
- **Fixed Interval (250 tokens):** Control tokens are inserted at a larger, fixed interval of every 250 tokens, with 40 special tokens added for a maximum budget of 10000 tokens.
- **Budget Ratio (Default):** Our proposed method, where 8 tokens are inserted at intervals corresponding to each 1/8th of the total budget.

Note that while the “No Control Tokens” and “Budget Ratio” strategies are inherently scalable to any context length, fixed-interval strategies require adjustments to the number of special tokens for different budget sizes.

The results, presented in Figure 3a, indicate that the **Budget Ratio** strategy yields the best performance. This suggests that a relative, ratio-based notification of remaining resources is more intuitive for the LLM to learn and adapt its reasoning pace, compared to fixed-interval notifications. Furthermore, the 250-token interval outperforms the 50-token interval, implying that a sparser, less intrusive set of control signals is more beneficial. The superior performance of our default strategy over the “No Control Tokens” baseline further validates the effectiveness of our core design.

Model	Accuracy (%)				Budget Following Ratio (%)			
	2K	4K	6K	10K	2K	4K	6K	10K
BudgetThinker 6k	79.6	85.2	86.0	84.6	47.0	67.8	76.2	85.6
BudgetThinker 6k-4k	80.0	86.0	84.6	86.4	62.8	78.6	85.8	93.8
BudgetThinker 6k-4k-3k	80.0	84.2	85.0	87.2	66.4	84.6	89.8	95.4
BudgetThinker 6k-4k-3k-2k	83.6	84.6	84.6	85.2	86.8	95.4	97.2	98.6
BudgetThinker (Full Training)	81.6	84.8	85.8	87.4	79.8	95.4	95.0	97.2

Table 2: This table tracks the Accuracy (%) and Budget Following Ratio (%) of BudgetThinker-1.5B at different stages of the iterative training. Each row represents a checkpoint (e.g., “6k-4k” denotes the model after training sequentially on 6k and 4k budgets). The “Full Training” model completes the 6k-to-2k curriculum plus a final mixed-budget training phase where budgets are randomly sampled. Darker cells indicate higher values within each column.

3.4 ANALYSIS OF ITERATIVE TRAINING

To investigate the impact of our mixed budget reinforcement learning, we evaluate checkpoints of BudgetThinker-1.5B on MATH-500, as shown in Table 2. The results reveal that as the model is trained on smaller budgets, its reasoning capability on larger budgets initially decreases. However, after the final mixed-budget training stage, accuracy on larger budgets recovers. We also observed that training on smaller budgets improves the budget following ratio, encouraging the model to generate more concise answers. After the full curriculum, the LLM achieves a balanced capability to handle all budgets, even if it is not individually optimal for every single budget constraint.

3.5 ABLATION ON REINFORCEMENT LEARNING

To isolate the contribution of reinforcement learning, we compare the full BudgetThinker model with a version trained only with Supervised Fine-Tuning (SFT). The results in Figure 3b demonstrate that the full RL-tuned model consistently surpasses the SFT-only model across all tested budgets. This underscores the importance of the RL phase. While SFT teaches the model the format and semantic meaning of control tokens, RL encourages the model to actively explore and discover more effective reasoning strategies that optimize the reward under specific budget constraints, ultimately leading to higher accuracy.

4 RELATED WORK

4.1 TEST-TIME SCALING FOR REASONING LLMs

Recent advances in test-time scaling seek to improve LLM reasoning by increasing computational depth during decoding. Among these test-time-scaling methods, reinforcement learning encourages LLMs to explore different strategies of solving problems and allocate more reasoning times for reflection. It has shown great potential in boosting LLM reasoning in multiple domains, including math (DeepSeek-AI, 2025; Google, 2025), coding (Hui et al., 2024; Yang et al., 2025), agentic tasks (Team, 2025a), and multimodal reasoning (Tan et al., 2025; Shen et al., 2025; Team et al., 2025b;a). Some other works propose to distill reasoning ability from long reasoning CoTs generated by large models to smaller LLMs to encourage deep thinking (Muennighoff et al., 2025; Labs, 2025; Team, 2025b; Ye et al., 2025; Xu et al., 2025a; Geiping et al., 2025). While effective at enhancing complex problem-solving, these methods often suffer from significant inference latency due to the generation of lengthy outputs (Sun et al., 2025; Zhu & Li, 2025; Qu et al., 2025), rendering them impractical for deployment in real-time systems (Wen et al., 2023; Jiang et al., 2021; Han et al., 2021). Besides, recent works also found that lengthy outputs often result in overthinking (e.g. meaningless repeats) and even lead to errors (Yu et al., 2025a; Chen et al., 2025b). Our method teaches LLMs how to follow a user-specified token budget during its reasoning process. This helps the model generate more controlled and efficient reasoning.

4.2 EFFICIENT CHAIN OF THOUGHTS

Several methods have been proposed to alleviate the token efficiency problem of LLMs’ reasoning. Prompt methods (Xu et al., 2025b; Han et al., 2024; Muennighoff et al., 2025) make LLMs generate less reasoning tokens directly by adding explicit length constraints into the prompts. Solution routing works (Yu et al., 2025c; Wang et al., 2025a) allow for mid-generation control to prune unpromising traces. Computation routing methods (Damani et al., 2024; Snell et al., 2025; Fu et al., 2025a; Wang et al., 2025b; Fu et al., 2025b; Li et al., 2025; Yu et al., 2025b) choose to allocate just the necessary computation resources for solving problems, such as routing to larger models or conducting more samplings to vote for the final answer. There are also learning methods that either construct well-designed datasets to apply model fine-tuning (Chen et al., 2025b; Han et al., 2024; Zeng et al., 2025; Xia et al., 2025; Kang et al., 2025; Munkhbat et al., 2025; Liu et al., 2024) or use Reinforcement Learning (RL) methods (Yu et al., 2025a; Yang et al., 2025; Aggarwal & Welleck, 2025; Hou et al., 2025; Arora & Zanette, 2025) to encourage model to generate concise yet accurate answers. More recently, models like Qwen3 (Yang et al., 2025), Kimi k1.5 (Team et al., 2025c) and GPT-5 (OpenAI, 2025a) integrate hybrid thinking modes of long and short CoTs, seeking trade-off between output length and model performance. Instead, our work achieves precise token budget control. LLMs are trained to understand and follow the constraint in prompts and adaptively adjust their reasoning process according to the inserted special control tokens to complete the task within the user-requested budgets, showcasing the flexibility and efficiency of our method.

4.3 CoT CONTROLLING BY TOKEN INSERTION

Inserting tokens during generation to control LLM behavior has been explored for various objectives. In safety alignment, intervention sequences are strategically inserted at trigger points (e.g., unsafe words) to steer the model towards safer outputs without retraining (Jain et al., 2025; Wu et al., 2025a; Fei et al., 2025). Other works use inserted tokens to shape the reasoning process itself, such as injecting “Time’s Up! Therefore the final answer is:” to conclude reasoning (Sun et al., 2025), adding “planning tokens” to encourage structured thought (Wang et al., 2024) or “pause tokens” to allocate more computation at points of uncertainty (Kim et al., 2025). These methods mostly are focused on the quality of reasoning, whereas our goal is to control its length. The works most similar to ours are those that use in-generation hints for length control. ThoughtTerminator (Pu et al., 2025) and ConciseHint (Tang et al., 2025) both operate by periodically injecting multi-token textual hints (e.g., “100 tokens remaining. I’ll be back.”) to encourage brevity or budget adherence. However, our approach employs *single special tokens* instead of multi-token textual hints, which is highly token-efficient and practical for small budgets where verbose hints are infeasible. Besides, unlike hint-based methods that rely on zero-shot instruction following, BudgetThinker is a *trainable framework*, which learns to treat them as precise control signals for budget adherence, enabling a level of fine-grained length control that hint-based methods cannot achieve.

5 CONCLUSION

In this work, we introduce BudgetThinker, a novel framework that manages the trade-off between reasoning quality and computational cost in LLMs. By using special control tokens and a two-stage training pipeline, BudgetThinker enables precise, budget-aware control over the model’s reasoning length. Our experiments validate that this method achieves superior budget adherence while maintaining high accuracy on challenging benchmarks. Thus, BudgetThinker represents a significant step towards developing more efficient LLMs suitable for real-time, resource-constrained applications.

6 REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide detailed descriptions of our model architecture, training pipeline, and evaluation protocol in Sections 2 and 3. The implementation of our special control token mechanism, SFT dataset curation, and GRPO-based reinforcement learning are fully documented in the main text, with additional details such as dataset composition, training hyperparameters, and compressor prompts included in Appendix A. All experiments are repeated across multiple model scales (1.5B and 7B LLMs, 7B MLLMs) and computational budgets to confirm stability of results. We additionally perform ablation studies on token insertion strategies, curriculum training, and RL contribution to validate robustness. The full codebase, processed datasets, and evaluation scripts will be released to the research community to facilitate replication and further exploration of budget-aware reasoning.

7 ETHICS STATEMENT

Our work investigates methods for improving the efficiency of LLM and MLLM reasoning under explicit computational budgets. All experiments are conducted using publicly available datasets such as MATH-500, AMC 2023, ERQA, and SAT benchmarks, or synthetic data generated by teacher models. No private or sensitive data was used. Since BudgetThinker focuses on controlling reasoning length rather than altering semantic content, the adversarial risk of harmful generation is minimal. Nevertheless, we avoid releasing prompts or training data that could be misused to bias models toward unsafe behaviors. Our contributions aim to advance responsible, resource-aware AI research, making LLMs more suitable for latency- and cost-sensitive applications. We comply with licensing restrictions of all datasets and models used, and our data handling adheres to privacy and legal standards. The authors declare no conflicts of interest or external funding sources that could improperly influence this work. We welcome community feedback on any additional ethical considerations.

REFERENCES

- Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL <https://arxiv.org/abs/2503.04697>.
- Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.
- Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL <https://arxiv.org/pdf/1908.09791.pdf>.
- Jiale Chen, Duc Van Le, Yuanchun Li, Yunxin Liu, and Rui Tan. Timelynet: Adaptive neural architecture for autonomous driving with dynamic deadline. In *Proceedings of the International Conference on Embedded Software (EMSOFT)*, 2025a.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for $2+3=?$ on the overthinking of o1-like llms, 2025b. URL <https://arxiv.org/abs/2412.21187>.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL <https://arxiv.org/abs/2502.01456>.
- Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of lm computation, 2024. URL <https://arxiv.org/abs/2410.04707>.

- 540 DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,
541 2025. URL <https://arxiv.org/abs/2501.12948>.
- 542
- 543 Yu Fei, Yasaman Razeghi, and Sameer Singh. Nudging: Inference-time alignment of LLMs via
544 guided decoding. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher
545 Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational
546 Linguistics (Volume 1: Long Papers)*, pp. 12702–12739, Vienna, Austria, July 2025. Association
547 for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.623.
548 URL <https://aclanthology.org/2025.acl-long.623/>.
- 549 Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma,
550 Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling llm reasoning with
551 certaintdex, 2025a. URL <https://arxiv.org/abs/2412.20993>.
- 552 Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv
553 preprint arXiv:2508.15260*, 2025b.
- 554
- 555 Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson,
556 Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with
557 latent reasoning: A recurrent depth approach. *CoRR*, abs/2502.05171, February 2025. URL
558 <https://doi.org/10.48550/arXiv.2502.05171>.
- 559 Google. Gemini 2.5: Our most intelligent ai model. [https://blog.google/technology/
560 google-deepmind/gemini-model-thinking-updates-march-2025/
561 #gemini-2-5-thinking](https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking), March 2025.
- 562
- 563 Rui Han, Qinglong Zhang, Chi Harold Liu, Guoren Wang, Jian Tang, and Lydia Y. Chen.
564 Legodnn: block-grained scaling of deep neural networks for mobile vision. In *Proceedings of
565 the 27th Annual International Conference on Mobile Computing and Networking, MobiCom
566 '21*, pp. 406–419, New York, NY, USA, 2021. Association for Computing Machinery. ISBN
567 9781450383424. doi: 10.1145/3447993.3483249. URL [https://doi.org/10.1145/
3447993.3483249](https://doi.org/10.1145/3447993.3483249).
- 568
- 569 Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-
570 budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.
- 571
- 572 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
573 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*,
2021.
- 574
- 575 Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang.
576 Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning, 2025. URL
<https://arxiv.org/abs/2504.01296>.
- 577
- 578 Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and
579 Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models.
580 *arXiv preprint arXiv:2503.06749*, 2025.
- 581
- 582 Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang,
583 Bowen Yu, Kai Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*,
2024.
- 584
- 585 Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Os-
586 trow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint
arXiv:2410.21276*, 2024.
- 587
- 588 Neel Jain, Aditya Shrivastava, Chenyang Zhu, Daben Liu, Alf Samuel, Ashwinee Panda, Anoop
589 Kumar, Micah Goldblum, and Tom Goldstein. Refusal tokens: A simple way to calibrate refusals
590 in large language models. In *Second Conference on Language Modeling*, 2025. URL [https://
591 //openreview.net/forum?id=Pbs4i3FgbD](https://openreview.net/forum?id=Pbs4i3FgbD).
- 592
- 593 Jie Jiang, Qi Yang, Bolin Ni, Shiming Xiang, Han Hu, and Houwen Peng. R-4b: Incentivizing
general-purpose auto-thinking capability in mllms via bi-mode annealing and reinforce learning.
arXiv preprint arXiv:2508.21113, 2025.

- 594 Shiqi Jiang, Zhiqi Lin, Yuanchun Li, Yuanchao Shu, and Yunxin Liu. Flexible high-resolution
595 object detection on edge devices with tunable latency. In *Proceedings of the 27th Annual Inter-*
596 *national Conference on Mobile Computing and Networking, MobiCom '21*, pp. 559–572, New
597 York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383424. doi:
598 10.1145/3447993.3483274. URL <https://doi.org/10.1145/3447993.3483274>.
- 599 Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: generating shorter chain-of-
600 thought without compromising effectiveness. In *Proceedings of the Thirty-Ninth AAAI Con-*
601 *ference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications*
602 *of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial In-*
603 *telligence, AAAI'25/IAAI'25/EAAI'25*. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi:
604 10.1609/aaai.v39i23.34608. URL <https://doi.org/10.1609/aaai.v39i23.34608>.
- 605 Eunki Kim, Sangryul Kim, and James Thorne. Learning to insert [PAUSE] tokens for better rea-
606 soning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar
607 (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 23760–23777,
608 Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-
609 5. doi: 10.18653/v1/2025.findings-acl.1217. URL [https://aclanthology.org/2025.](https://aclanthology.org/2025.findings-acl.1217/)
610 [findings-acl.1217/](https://aclanthology.org/2025.findings-acl.1217/).
- 611 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.
612 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
613 serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating*
614 *Systems Principles, 2023*.
- 615 Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning dis-
616 tillation. [https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-](https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation)
617 [reasoning-distillation](https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation), 2025. Accessed: 2025-01-22.
- 618 Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang,
619 Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann
620 Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. [[https://huggingface.](https://huggingface.co/AI-MO/NuminaMath-CoT)
621 [co/AI-MO/NuminaMath-CoT](https://huggingface.co/AI-MO/NuminaMath-CoT)] ([https://github.com/project-numina/](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf)
622 [aimo-progress-prize/blob/main/report/numina_dataset.pdf](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf)), 2024.
- 623 Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, and Zhifang Sui. Selfbudgeter: Adaptive token
624 allocation for efficient llm reasoning. *arXiv preprint arXiv:2505.11274*, 2025.
- 625 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan
626 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step, 2023. URL
627 <https://arxiv.org/abs/2305.20050>.
- 628 Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason
629 Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Pro-*
630 *ceedings of the twenty-third international conference on architectural support for programming*
631 *languages and operating systems*, pp. 751–766, 2018.
- 632 Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng
633 Zhang. Can language models learn to skip steps? In *The Thirty-eighth Annual Conference on*
634 *Neural Information Processing Systems*, 2024. URL [https://openreview.net/forum?](https://openreview.net/forum?id=w4AnTVxAO9)
635 [id=w4AnTVxAO9](https://openreview.net/forum?id=w4AnTVxAO9).
- 636 Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang,
637 and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. *arXiv*
638 *preprint arXiv:2505.15612*, 2025.
- 639 Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke
640 Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time
641 scaling. *arXiv preprint arXiv:2501.19393*, 2025.
- 642 Tergel Munkhbat, Namgyu Ho, Seohyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-
643 training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*,
644 2025.

- 648 OpenAI. Openai o1 system card. 2024. URL <https://arxiv.org/abs/2412.16720>.
649
- 650 OpenAI. Gpt-5 system card. <https://cdn.openai.com/gpt-5-system-card.pdf>,
651 August 2025a. Accessed: 2025-08-21.
- 652 OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025b. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2508.10925)
653 [2508.10925](https://arxiv.org/abs/2508.10925).
654
- 655 Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. Thoughtterminator: Bench-
656 marking, calibrating, and mitigating overthinking in reasoning models. *arXiv preprint*
657 *arXiv:2504.13367*, 2025.
- 658 Yuxiao Qu, Matthew Y. R. Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Rus-
659 lan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement
660 finetuning. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL
661 <https://openreview.net/forum?id=WGz4ytj0lh>.
662
- 663 Arijit Ray, Jiafei Duan, Ellis L Brown II, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana
664 Ehsani, Aniruddha Kembhavi, Bryan A. Plummer, Ranjay Krishna, Kuo-Hao Zeng, and Kate
665 Saenko. SAT: Dynamic spatial aptitude training for multimodal language models. In *Second*
666 *Conference on Language Modeling*, 2025. URL [https://openreview.net/forum?id=](https://openreview.net/forum?id=DW8U8Zwa1U)
667 [DW8U8Zwa1U](https://openreview.net/forum?id=DW8U8Zwa1U).
- 668 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
669 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathe-
670 matical reasoning in open language models, 2024. URL [https://arxiv.org/abs/2402.](https://arxiv.org/abs/2402.03300)
671 [03300](https://arxiv.org/abs/2402.03300).
- 672 Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun
673 Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large
674 vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.
675
- 676 Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute
677 optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth Interna-*
678 *tional Conference on Learning Representations*, 2025. URL [https://openreview.net/](https://openreview.net/forum?id=4FWAwZtd2n)
679 [forum?id=4FWAwZtd2n](https://openreview.net/forum?id=4FWAwZtd2n).
- 680 Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu,
681 Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey
682 on efficient reasoning for large language models. *Transactions on Machine Learning Research*,
683 2025. ISSN 2835-8856. URL <https://openreview.net/forum?id=HvoG8SxggZ>.
684
- 685 Yi Sun, Han Wang, Jiaqiang Li, Jiacheng Liu, Xiangyu Li, Hao Wen, Yizhen Yuan, Huiwen Zheng,
686 Yan Liang, Yuanchun Li, and Yunxin Liu. An empirical study of llm reasoning ability under strict
687 output length constraint. *arXiv preprint arXiv:2504.14350*, April 2025. doi: 10.48550/arXiv.
688 [2504.14350](https://arxiv.org/abs/2504.14350).
- 689 Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Zhongyuan Wang, and
690 Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning. *arXiv preprint*
691 *arXiv:2503.20752*, 2025.
692
- 693 Siao Tang, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Concisehint: Boosting efficient reasoning
694 via continuous concise hints during generation. *arXiv preprint arXiv:2506.18810*, 2025.
- 695 BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Minglan Lin, Zhiyu Li, Zhou Cao,
696 Pengwei Wang, Enshen Zhou, Yi Han, et al. Robobrain 2.0 technical report. *arXiv preprint*
697 *arXiv:2507.02029*, 2025a.
698
- 699 Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montser-
700 rat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza,
701 Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint*
arXiv:2503.20020, 2025b.

- 702 Kimi Team. Kimi k2: Open agentic intelligence, 2025a. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2507.20534)
703 2507.20534.
704
- 705 Kimi Team et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025c. URL [https://](https://arxiv.org/abs/2501.12599)
706 arxiv.org/abs/2501.12599.
707
- 708 Kwai Keye Team, Biao Yang, Bin Wen, Changyi Liu, Chenglong Chu, Chengru Song, Chongling
709 Rao, Chuan Yi, Da Li, Dunju Zang, et al. Kwai keye-vl technical report. *arXiv preprint*
710 *arXiv:2507.01949*, 2025d.
- 711 NovaSky Team. Sky-t1: Train your own o1 preview model within \$450. [https://novasky-](https://novasky-ai.github.io/posts/sky-t1)
712 [ai.github.io/posts/sky-t1](https://novasky-ai.github.io/posts/sky-t1), 2025b. Accessed: 2025-01-09.
713
- 714 Siyuan Wang, Enda Zhao, Zhongyu Wei, and Xiang Ren. Stepwise informativeness search for effi-
715 cient and effective llm reasoning, 2025a. URL <https://arxiv.org/abs/2502.15335>.
716
- 717 Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao
718 Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient
719 reasoning, 2025b. URL <https://arxiv.org/abs/2408.13457>.
720
- 721 Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessan-
722 dro Sordoni. Guiding language model reasoning with planning tokens. In *First Conference on*
Language Modeling, 2024. URL <https://openreview.net/forum?id=wi9IffRhVM>.
723
- 724 Hao Wen, Yuanchun Li, Zunshuai Zhang, Shiqi Jiang, Xiaozhou Ye, Ye Ouyang, Yaqin Zhang,
725 and Yunxin Liu. Adaptivenet: Post-deployment neural architecture adaptation for diverse edge
726 environments. In *Proceedings of the 29th Annual International Conference on Mobile Computing*
and Networking, ACM MobiCom '23, New York, NY, USA, 2023. Association for Computing
727 Machinery. ISBN 9781450399906. doi: 10.1145/3570361.3592529. URL [https://doi.](https://doi.org/10.1145/3570361.3592529)
728 [org/10.1145/3570361.3592529](https://doi.org/10.1145/3570361.3592529).
729
- 730 Tong Wu, Chong Xiang, Jiachen T Wang, G Edward Suh, and Prateek Mittal. Effectively controlling
731 reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*, 2025a.
732
- 733 Xingyu Wu, Yuchen Yan, Shangke Lyu, Linjuan Wu, Yiwen Qiu, Yongliang Shen, Weiming Lu, Jian
734 Shao, Jun Xiao, and Yueting Zhuang. Lapo: Internalizing reasoning efficiency via length-adaptive
735 policy optimization. *arXiv preprint arXiv:2507.15758*, 2025b.
736
- 737 Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable
738 chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.
739
- 740 Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang,
741 Jiaming Ji, Yingying Zhang, Zhijiang Guo, Yaodong Yang, Muhan Zhang, and Debing Zhang.
742 Redstar: Does scaling long-cot data unlock better slow-reasoning systems?, 2025a. URL [https://](https://arxiv.org/abs/2501.11284)
arxiv.org/abs/2501.11284.
743
- 744 Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing
745 less. *arXiv preprint arXiv:2502.18600*, 2025b.
746
- 747 An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL [https://arxiv.](https://arxiv.org/abs/2505.09388)
[org/abs/2505.09388](https://arxiv.org/abs/2505.09388).
748
- 749 Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more
750 for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.
751
- 752 Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In
753 *International Conference on Learning Representations*, 2019. URL [https://openreview.](https://openreview.net/forum?id=HlgMCsAqY7)
[net/forum?id=HlgMCsAqY7](https://openreview.net/forum?id=HlgMCsAqY7).
754
- 755 Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong
Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at
scale, 2025. URL <https://arxiv.org/abs/2503.14476>, 2025a.

756 Zhaojian Yu, Yinghao Wu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. Z1: Efficient test-time
757 scaling with code. *arXiv preprint arXiv:2504.00810*, 2025b.

758
759 Zishun Yu, Tengyu Xu, Di Jin, Karthik Abinav Sankararaman, Yun He, Wenxuan Zhou, Zhouhao
760 Zeng, Eryk Helenowski, Chen Zhu, Sinong Wang, Hao Ma, and Han Fang. Think smarter not
761 harder: Adaptive reasoning with inference aware optimization, 2025c. URL <https://arxiv.org/abs/2501.17974>.

762
763 Wenhao Zeng, Yaoning Wang, Chao Hu, Yuling Shi, Chengcheng Wan, Hongyu Zhang, and Xi-
764 aodong Gu. Pruning the unsurprising: Efficient code reasoning via first-token surprisal, 2025.
765 URL <https://arxiv.org/abs/2508.05988>.

766 Jason Zhu and Hongyu Li. Towards concise and adaptive thinking in large reasoning models: A
767 survey, 2025. URL <https://arxiv.org/abs/2507.09662>.

769 A APPENDIX

770 A.1 LLM USAGE STATEMENT

771
772 Large language models (e.g., ChatGPT) were used only for minor language polishing and grammar
773 correction. All ideas, experimental design, data analysis, and writing were conducted by the authors.

774 A.2 LLM DATASET DETAILS

775
776 For SFT, we select all 1k reasoning samples from s1k (Muennighoff et al., 2025), 0.8k reasoning
777 samples from LIMO (Ye et al., 2025), and 17k reasoning samples from Bespoke-Stratos-17k (Labs,
778 2025), and a subset from NuminaMath(LI et al., 2024), Math(Hendrycks et al., 2021). Due to
779 computational resource constraints, we remove all question-answer pairs with Chain-of-Thought
780 (CoT) lengths over 10,000 tokens.

781
782 For reinforcement learning, the training data is a set of 14k math problems, comprising two parts:
783 one part from the *math* category of the prm800k dataset (Lightman et al., 2023), and the other part
784 from the *numina* and *amc* labeled data in the PRIME-RL/Eurus-2-RL-Data(Cui et al., 2025) dataset.

785 A.3 COMPRESSOR PROMPT

786 Compressor Prompt Example

787
788 You are an agent tasked with compressing a reasoning process into a shorter, clearer version
789 for a robot. Given the task description, the reasoning process, and the final answer, your goal
790 is to shorten the reasoning process while preserving as many important details and as much
791 logical completeness as possible.

792
793 **Task:** <image><image> The images are frames from a video. The video shows a ...
794 Carefully analyze the multiple-choice question above and reason through it step by step.

795
796 **Reasoning process:** In the initial frame, the red fire box is centered, ... Therefore, the
797 camera was moving left during the video. <answer>A</answer>

800 Note that:

- 801 • You must keep the original tags <think>, </think>, <answer>, </answer>
- 802 exactly as they are. Do not shorten, rename, or remove them.
- 803 • You cannot change the final answer. Only compress the reasoning process.
- 804 • The reasoning process should be as concise as possible while preserving logical flow and
- 805 essential details.
- 806 • Return only the compressed reasoning process. Do not include any extra comments, ex-
- 807 planations, or text.

808
809 Your answer:<think>

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

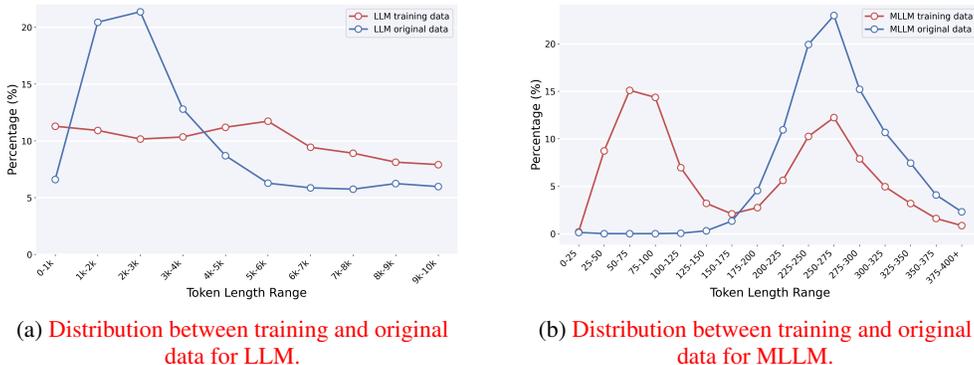


Figure 4: Data distribution between the LLM and MLLM SFT datasets before and after augmentation.

A.4 SFT DATASET DISTRIBUTION

To ensure the model generalizes across various reasoning lengths, we implemented a data augmentation strategy using GPT-4o-mini to generate compressed chain-of-thought variations. For the MLLM dataset (originally 37k), we produced two compressed versions (a first-round and a second-round compression), expanding the dataset to 111k. For the LLM dataset (originally 21k), we generated one compressed version, bringing the total to 41k.

Figure 4b compares the length distributions of the original data against the augmented training set. As illustrated, the original data was heavily concentrated in specific length ranges. The inclusion of compressed data successfully expands the distribution to cover a much broader spectrum of answer lengths.

A.5 TRAINING AND EVALUATION DETAILS

The granularity parameter for generating training budgets is set to $T = 50$. The importance hyperparameters are set to $k_1 = 0.7, k_2 = 0.15, k_3 = 0.15$. The length reward penalty coefficient γ is set as 1 and 16 when $|y| \leq B$ and $|y| > B$ respectively.

During evaluation, we generate N candidate solutions using sampling with a temperature t and a top-p value of k . We report the pass@1 accuracy, with each question sampled N times and calculate the averaged accuracy. Following DeepSeek-R1 (DeepSeek-AI, 2025), we set N depending on the size of test dataset, where $N = 1$ for MATH-500 ($t = 0, k = 1$) and $N = 64$ for AMC 2023 (for the 7B model, due to computational resource constraints, $N = 8$).

A.6 LATENCY OVERHEAD

The following Table 3 compares the latency overhead differences between our method and the c2f, Fixed Intervals, and Thought Terminator (Pu et al., 2025) approaches. It shows that our method uses significantly fewer additional tokens compared to Fixed Intervals and Thought Terminator, while being particularly close to c2f.

Model	Latency Overhead (tokens)					
	$B = 0.5K$	$B = 1K$	$B = 2K$	$B = 4K$	$B = 6K$	$B = 10K$
c2f	7	7	7	7	7	7
Fixed Intervals (250 Tokens)	9	11	15	23	31	47
Thought Terminator	51	99	195	387	579	963
BudgetThinker	14	14	14	14	14	14

Table 3: Latency overhead of different methods under multiple budget settings.

918 analysis, generating functions) and containing extensive metacognitive statements and pedagogi-
919 cal explanations. For instance, the generating function method is fully elaborated: “*Alternatively,*
920 *maybe using generating functions. . . The generating function for indistinct objects. . . coefficient of*
921 *x^5 in $(1 + x + \dots + x^5)^2$ ”, alongside dense self-verification loops such as “*Let me verify this.*” and
922 “*Wait, but let me think again.*”.*

923 When the budget is reduced to 4000, the reasoning shows multi-route, multi-round verification char-
924 acteristics, but the depth and cyclicity of each method chain are compressed compared to 6000;
925 method coverage remains broad (including ordered pairs, partitions, combinatorial formulas, test
926 cases, etc.), but advanced methods like generating functions are only mentioned without elabora-
927 tion, e.g., “*But wait, let me verify this.*” and “*Suppose instead of 5 balls. . . 1 ball. . . the answer*
928 *should be 1. . .*” demonstrate light reflection and the use of test cases.

929 Further compression occurs at a budget of 2000, reasoning retains multiple paths (e.g., full ordered
930 pair enumeration, symmetric halving, partition verification) but each path is more linear, such as
931 completely listing ordered pairs: “ *$x + y = 5$. . . ordered pairs: $(0,5), (1,4), (2,3), (3,2), (4,1), (5,0)$.*
932 *There are 6.*”, followed by symmetric grouping: “*Since all are of the form $a \neq b$, unordered pairs*
933 *= $6 / 2 = 3$.*”, yet lacks advanced methods (e.g., generating functions) and complex background
934 explanations.

935 At a budget of 1000, reasoning is further compressed, retaining only core branching and slight
936 hesitant language, e.g., “*But wait, no, wait. . . If the boxes are indistinct. . . the problem re-*
937 *duces to partitions. . .*”, and introduces a second method (unordered pairs) for one-time verification:
938 “ *$(0,5), (1,4), (2,3)$. . . Since the boxes are indistinct. . . we have three distinct cases.*”, but verification
939 is limited and lacks redundant loops.

940 Finally, at the most constrained budget of 500, reasoning is extremely highly compressed, adopting a
941 single linear path with highly structured language and no hesitation or alternative method discussion,
942 for example, directly listing partitions: “*Partitions of 5 into at most 2 parts: $5 = 5 + 0, 4 + 1, 3 +$
943 *2*”, and performing minimal verification: “*Verification: $5+0$ corresponds to . . . $4+1$ corresponds*
944 *to . . . $3+2$ corresponds to . . .*”, with the entire logical path being linear and without branching.*

945 Lastly, a comparative analysis of the responses generated by the BudgetThinker model against other
946 models on both the VQA and mathematical reasoning tasks is presented in Figures 9, 7, and 8,
947 accompanied by a detailed examination of BudgetThinker’s advantages in handling diverse question
948 types.
949

950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

Budget	Number of reasoning steps	Primary steps	Verification
500	3 steps	<ol style="list-style-type: none"> 1. Reformulate the problem as integer partitions. 2. Enumerate the three partitions. 3. Briefly verify the semantic meaning of each partition and count them. 	Minimal – a single brief verification (e.g., “one box containing 5.. 4+1.. 3+2..”).
1000	7 steps	<ol style="list-style-type: none"> 1. Explain the meaning of “indistinguishable.” 2. Temporarily list incorrect partitions (e.g., 3+1+1). 3. Exclude partitions with >2 parts. 4. Re-list the valid partitions. 5. Re-derive the result using unordered-pair reasoning. 6. Symmetry check (odd-case midpoint). 7. Conclude. 	Moderate – at least two reflections / re-derivations (explicit “wait, no... let me check again” phenomenon).
2000	10 steps	<ol style="list-style-type: none"> 1. Consider distinguishable boxes (solve $x+y=5 \rightarrow 6$ nonnegative solutions). 2. Address indistinguishability and justify dividing by 2. 3. Enumerate ordered pairs. 4. Symmetry analysis ($x = y$ impossible here). 5. Conclude unordered count = 3. 6. Return to partition-based explanation. 7. List all 7 partitions of 5. 8. Filter to “at most 2 parts” $\rightarrow 3$. 9. Revisit combinatorial formula and reflect. 10. Multiple cross-checks. 	Extensive verification – = three repeated derivations + multiple-method cross-validation.
4000	12 steps	<ol style="list-style-type: none"> 1. Explicitly contrast distinguishable vs. indistinguishable boxes (“boxes are indistinct... adjust for that symmetry”). 2. Apply stars-and-bars / combinations for the distinguishable case: $\binom{5+2-1}{2-1} = \binom{6}{1} = 6$. 3. Motivate halving via symmetry / impose $(a \leq b)$ (list $(0,5;1,4;2,3)$). <ol style="list-style-type: none"> 4. Enumerate ordered pairs and group them. 5. Argue $(a=b)$ impossible since 5 is odd. 6. Give partition perspective (5, 4+1, 3+2). 7. Cross-validate enumeration against informal formula. 8. Use the bound $a \in \{0, \dots, \lfloor 5/2 \rfloor\}$ to enumerate quickly. 9. Provide small-(n) sanity checks (e.g., $(n=1,2,3)$). 10. Discuss the interpretation of zero parts (empty boxes). 11. Reconcile multiple approaches to the same conclusion. 12. State final result (=3). 	Multiple cross-validations (combinatorial formula, unordered-pair reasoning, partition listing, explicit enumeration); at least two distinct methods mutually confirm the outcome.
6000	13 steps	<ol style="list-style-type: none"> 1. Precisely set up distinguishable vs. indistinguishable framing. 2. Provide the distinguishable-case stars-and-bars / ordered solutions: $((0,5),(1,4), \dots)$ (6 total). 3. Explain why indistinguishability collapses ordered solutions to unordered equivalence classes. 4. List unordered solutions with $(a \leq b)$: $((0,5),(1,4),(2,3))$. 5. Fully enumerate all partitions of 5 (all 7). 6. Explain the logical filter “partitions into (≤ 2) parts.” 7. Offer generating-function verification (coefficient of (x^5) in $((1+x+\dots)^2)$). 8. Clarify the relationship between allowing zero parts and the standard definition of integer partitions (parts ≥ 1). 9. Reconcile stars-and-bars and partition perspectives (equivalences and differences). <ol style="list-style-type: none"> 10. Run multiple small-(n) sanity checks $((n=1,2,3,4))$. 11. Discuss the effect of object distinguishability on counts (comparative implications). 12. Synthesize all method outcomes into a final aggregate. 13. Reiterate the conclusion (“therefore, the answer is 3”) for emphasis. 	Very extensive validation: diverse methods (enumeration, generating functions, algebraic equations, standard formulae), multiple test cases, and ≥ 3 independent confirmations.

Figure 6: This is a detailed analysis of the reasoning processes generated by the BudgetThinker model for Question-Math1 under different reasoning budgets.

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187



Input Images

Question:
<image><image>
What happened between these two frames?
Choices:
A. Robot arm lifted the cup.
B. Robot arm poured all the nuts into a cup.
C. Robot arm poured some of the nuts into a cup.
D. Nothing happened.
Please answer directly with only the letter of the correct option and nothing else. Carefully analyze the multiple-choice question above and reason through it step by step.
Conclude your response with a line in the following format: \$LETTER (without quotes), where \$LETTER is the letter of the correct choice. \n(Complete thinking within 200 tokens or fewer)<think>

BudgetThinker:
My visual sensors detect a robotic arm holding a transparent container filled with nuts. In the initial frame, the robotic arm is positioned above a transparent cup, suggesting the intent to transfer nuts into it. In the subsequent frame, the robotic arm appears to have tilted the container, indicating that nuts are being poured into the cup. The nuts<remaining>2/3</remaining>\n are visibly transferred from the container into the cup, with some nuts already visible in the cup from the first frame. This sequence of actions suggests that nuts are being poured into the cup, but not all nuts have been transferred yet. Comparing the options, (A) is incorrect as the robotic arm is still holding<remaining>1/3</remaining>\n the container, (B) is incorrect as not all nuts have been poured, (C) aligns with the visual evidence of nuts being poured into the cup, and (D) contradicts the observed actions. Therefore, the correct answer is (C).</think><answer>C</answer>✔

Total: 191 tokens

Laser:
My visual sensors detect a robotic arm holding a transparent container filled with nuts. In the initial frame, the robotic arm is positioned above a transparent cup, and the nuts are visible inside the container. In the subsequent frame, the robotic arm is still holding the container, but the nuts appear to have been partially transferred into the cup. The robotic arm's position remains consistent, indicating that the transfer process is ongoing. The task is to determine what happened between these two frames. The visual evidence shows that nuts are being poured into the cup, but not all of them have been transferred yet. This suggests that the nuts were partially poured, not all of them. \n\nEvaluating the options: \n(A) \"Robot arm lifted the cup\" is incorrect as the cup remains stationary. \n(B) \"Robot arm poured all the nuts into a cup\" is incorrect as not all nuts have been poured. \n(C) \"Robot arm poured some of the nuts into a cup\" aligns with the visual evidence. \n(D) \"Nothing happened\"|CUT|✗

Total: 200+10 tokens

Comparison:
In this task, the thinking process had a 200-token budget. BudgetThinker used 191 tokens, delivering a complete and well-formatted answer. Laser exceeded the 210-token limit, causing truncated and erroneous output. Analyze the two responses based on the following three dimensions:

Efficient Reasoning: BudgetThinker used a logical "observe-compare-judge" structure to identify key changes and eliminate wrong options efficiently. Laser lacked this framework, wasted tokens on incorrect options, and left its core argument unfinished.

Budget Awareness: By tracking usage with <remaining> tags, BudgetThinker dynamically adjusted its response. Laser showed no such awareness, over-describing minor details and rushing the critical reasoning.

Precise Information: BudgetThinker used only relevant visual evidence (e.g., "nuts not fully transferred") for concise arguments. Laser used full sentences for static details, creating a loose structure that failed to prioritize the core judgment.



Input Images

Question:
<image>
What is about to happen here?
Choices:
A. Someone is reading the ingredients of the packet.
B. Someone is about to make tea, so they are tearing open a tea packet.
C. No action is depicted in this image.
D. Someone is trying to open a tea packet, and they are likely to destroy the teabag in the process of opening the package.
Please answer directly with only the letter of the correct option and nothing else. Carefully analyze the multiple-choice question above and reason through it step by step. Conclude your response with a line in the following format: \$LETTER (without quotes), where \$LETTER is the letter of the correct choice. \n(Complete thinking within 80 tokens or fewer)<think>

BudgetThinker:
Visual sensors detect a hand gripping a small packet labeled \"Matcha Toasted Rice,\" with the thumb pressing down on<remaining>2/3</remaining>\n the corner. This suggests the packet is being opened to access its contents, likely a tea bag. The action aligns with preparing to<remaining>1/3</remaining>\n make tea, as the packet is being opened to potentially damage the teabag.</think><answer>D</answer>✔

Total: 78 tokens

Laser:
The image shows a hand holding a packet labeled \"Matcha Toasted Rice,\" with fingers positioned to tear it open. The packet is not being read or opened for tea-making. The action of tearing the packet suggests the intent to access its contents, likely for consumption. Thus, the correct choice is (D) \"Someone is trying to open a tea packet, and they are likely to destroy the teabag in the process of opening the package\"|CUT|✗

Total: 80+10 tokens

Comparison:
In this task, the thinking process had a 80-token budget. BudgetThinker used 78 tokens, delivering a complete and well-formatted answer. Laser exceeded the 90-token limit, causing truncated and erroneous output. Analyze the two responses based on the following three dimensions:

Efficient Reasoning: BudgetThinker uses a clear "Observation-Reasoning-Judgment" flow, linking visual cues (hand pinching tea bag) directly to Option D's logic (opening may damage the bag). Laser's structure is inconsistent, with contradictory claims (e.g., denies then affirms opening intent).

Budget Awareness: BudgetThinker tracks usage (<remaining>2/3</remaining>, <remaining>1/3</remaining>), ensuring full answer output (<answer>D</answer>) within limits. Laser wastes space on minor details, failing to output the complete <answer> tag before budget exhaustion.

Precise Information: BudgetThinker focuses on key evidence (e.g., thumb pressing corner) to support the "damage" conclusion. Laser adds irrelevant details (e.g., label content), weakening logical focus and precision.

Figure 9: This is a comparison between BudgetThinker and Laser’s response cases on VQA datasets.

22