

Dynamic Reasoning Budgeting: Adaptive Routing for Token Efficiency in Large Reasoning Models

Anonymous ACL submission

Abstract

Large reasoning models (LRMs) deliver strong performance on complex tasks through multi-step deliberation. However, their reliance on long chains of thought often causes excessive token consumption and elevated inference cost but only yields limited improvement in accuracy. This work frames this inefficiency as a reasoning budget allocation problem that determines how much computation and token budget should be invested per instance. To address this, we propose a dynamic reasoning budgeting framework that adaptively routes each input to an appropriate reasoning path according to estimated difficulty. Specifically, simple problems are handled by a lightweight model under strict length constraints, whereas difficult problems are processed by LRMs with optimized reasoning prompts. On benchmarks covering arithmetic, logic, and commonsense reasoning, our method substantially reduces token usage while preserving or improving accuracy and outperforms latest routing and reasoning work. The results suggest that efficiency in LRMs stems not from universally deeper reasoning chains, but from allocating only the amount of reasoning budget required for each problem.

1 Introduction

In recent years, Large Reasoning Models (LRMs) (Xu et al., 2025), including OpenAI o1 (Wu et al., 2024), Alibaba Qwen3 (Qwen3, 2025) and DeepSeek R1 (DeepSeek-AI et al., 2025) demonstrated significant advances on complex tasks such as mathematical problem solving and code generation. These models often rely on reasoning techniques such as Chain-of-Thought (CoT) prompting (Wei et al., 2022), which generates intermediate reasoning steps to boost both accuracy and interpretability.

The extensions of CoT, such as Tree-of-Thought (ToT) (Yao et al., 2023) and Graph-of-Thought

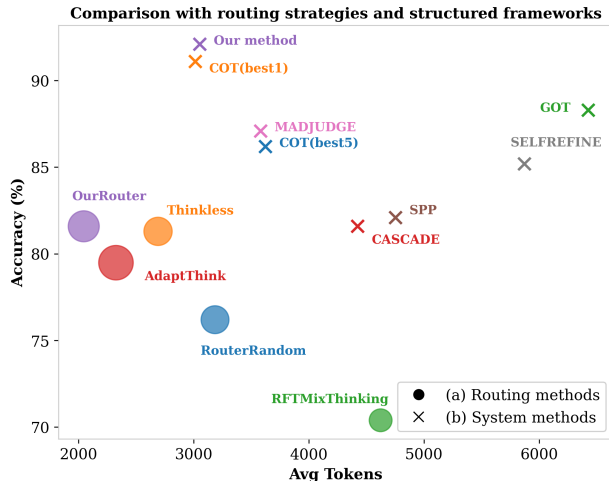


Figure 1: Performance comparison on the MATH500 dataset: (a) various routing strategies evaluated with the DeepSeek-R1-1.5B model, and (b) different structured inference methods evaluated with the QwQ-32B model. Circle size indicates the routing ratio the proportion of instances routed to the stronger model.

(GoT) (Besta et al., 2024) typically decompose problems into multiple reasoning paths and aggregate their outputs. Although these step-by-step pipelines can improve accuracy, they incur substantial computational overhead and high token usage. Recent studies indicate that even straightforward zero-shot prompts can match the accuracy of elaborate reasoning chains in typical use cases, rendering lengthy chains unnecessary.

Our empirical analysis reveals that longer reasoning chains do not lead to higher output quality. As illustrated in Figure 2, incorrect answers across several datasets tend to significantly increase token count compared to correct answers. This overthinking behavior (Sui et al., 2025) indicates that LRMs often generate unnecessary intermediate steps when facing difficult or even insolvable questions, leading to wasted computation. We conducted an in-depth analysis of the overthinking phenomenon (see Appendix B).

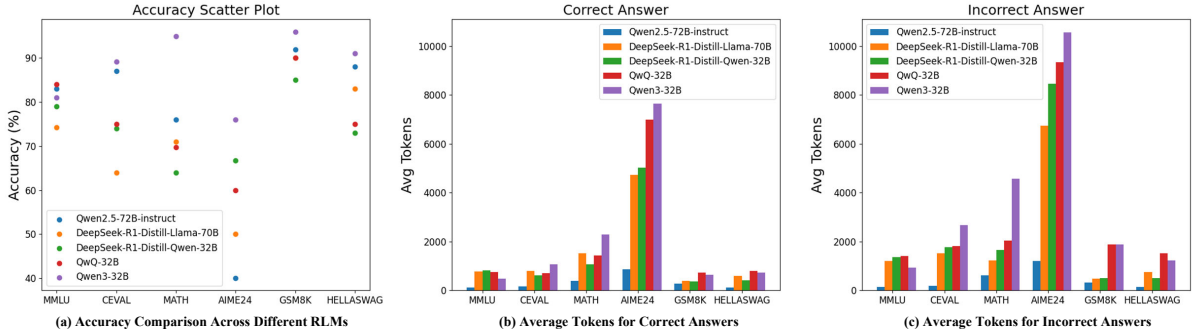


Figure 2: Comparison of token consumption and accuracy across reasoning models and datasets: (a) Accuracy improvement in reasoning models compared to non-reasoning models; (b) High-level benchmarks drive substantially higher token usage, while low-level tasks require fewer tokens; (c) Reasoning models consume more tokens for incorrect answers.

To mitigate these inefficiencies, we draw inspiration from the LLM cascade paradigm (Yue et al., 2024). Our framework first routes each query based on difficulty. That is, simple questions are handled by a smaller, faster model, while challenging ones invoke a larger, more capable model. We further integrate prompt optimization and model length control to ensure token efficiency throughout the reasoning process.

As illustrated in Figure 1, we demonstrate that on MATH500 (Lightman et al., 2023) our routing method achieves 81.6% accuracy with just 2,044 tokens, reducing token usage by up to 56% compared to strong baselines. Our structured inference framework attains 92.1% accuracy using 3,053 tokens, surpassing both COT and GOT in efficiency and final accuracy. Overall, we reduce token consumption by 71% on average across tasks while maintaining or improving accuracy.

Our contributions are summarized as follows.

- We provide empirical insights by systematically analyzing the reasoning behavior of LRMs. Our study reveals that these models often generate excessively long chains of thought without achieving notable gains in accuracy, indicating inherent inefficiency in current reasoning pipelines.
- We propose a dynamic cascade framework that adaptively routes each query to a model with an appropriate strength based on estimated difficulty. This framework further enhances efficiency through sophisticated prompt strategies and output length control, ensuring that simple questions incur minimal computational cost and complex questions

benefit from sufficient reasoning capacity.

- We validate our approach on a diverse suite of benchmarks, demonstrating reductions of 10–50% in average token consumption with minimal accuracy degradation. Across all tasks, our method achieves higher reasoning efficiency than existing frameworks, establishing a new state of the art in token efficient reasoning.

2 Related work

Several methods have been proposed to improve the performance and efficiency of large language models in reasoning tasks.

Existing research on efficient reasoning reduces inference cost from complementary angles: ZOOTER (Wang et al., 2025) routes queries to the most suitable expert LLM to improve both performance and computational efficiency across tasks; CoT-Valve (Ma et al., 2025) dynamically regulates and compresses Chain-of-Thought reasoning to avoid unnecessary long traces; and To CoT or not to CoT (Sprague et al., 2025) systematically examines when CoT helps, showing clear gains on math and logic problems but more limited improvements on other tasks. Reasoning Patterns (Zhang et al., 2025b) improves CoT prompting by leveraging reasoning patterns to reduce noise and enhance interpretability.

Our approach also aims at improving efficiency and accuracy by introducing a novel dynamic reasoning budgeting framework. This framework routes queries based on task complexity, adapting the reasoning path length and model selection to optimize computational cost. This allows us to

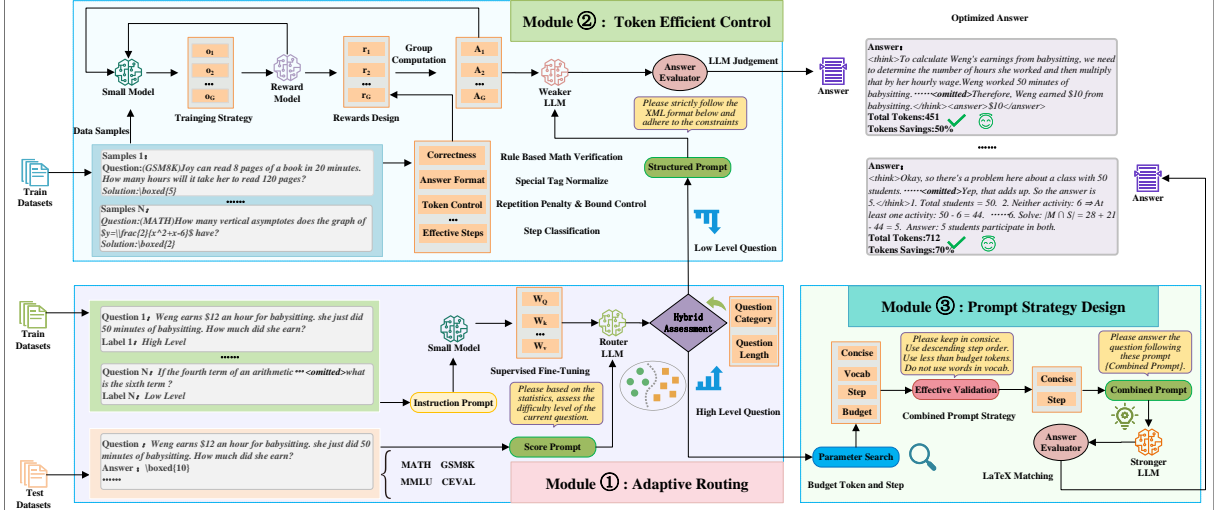


Figure 3: Overview of the proposed framework comprising three modules: Router dispatch, Token Efficient Control, and Prompt Strategy Design.

synergize performance and execution efficiency, especially in complex applications. In addition, we provide an extended discussion of related studies with a broader set of studies that further contextualize our contributions (see Appendix A).

3 Methodology

We propose a method comprising three module: (i) Adaptive Routing, (ii) Token Efficient Control, and (iii) Prompt Strategy Design. As showed in Figure 3, upon receiving a query, the router estimates its complexity and routes difficult cases to the strong-model branch and easy cases to the weak-model branch, balancing accuracy and computational cost.

3.1 Problem Formulation

We define the following components for our routing-based reasoning framework:

- $\mathcal{M}_{\text{weak}}$: a weak LLM (typically with fewer than 7B parameters) that is trainable under our reinforcement learning framework. For such a model, we apply length-control policy π_{len} and reward optimization to enhance their reasoning performance and token efficiency.
- $\mathcal{M}_{\text{strong}}$: a strong LLM (typically with more than 14B parameters) for which further training is often infeasible due to hardware and cost constraints. Instead, we study prompt optimization strategies $\mathcal{P}_{\text{strong}}$ to harness their reasoning capabilities while mitigating unnecessary token consumption.

- TN : total number of intermediate thoughts or reasoning units generated by the model before concluding the final answer.

- FCT : position of the first correct token in the generated output, measuring how early the model produces the correct answer.

We calculate the metrics TN and FCT based on the model’s reasoning traces enclosed within think tags. The total number of such discourse markers is used to estimate TN where a higher count indicates more frequent shifts. For FCT we record the position where the model first generates a correct token, hence evaluating how early the model begins to produce the correct conclusion during its reasoning process.

Our objective is to minimize the overall inference cost such as average token usage, computation time, or invocation expense while maintaining accuracy close to the upper bound achievable by the strong model. Queries that require stronger reasoning are routed to the high capacity model to reduce the risk of errors, whereas simpler queries are dispatched to the lightweight model to avoid unnecessary computation.

3.2 Adaptive Routing

Inspired by the Self-Route (He et al., 2025) framework, we design and explore the architecture and training strategy of the router. We establish the generation process of difficulty labels as follows. Similar questions refer to items within the same dataset that share comparable length, structure, or

numerical complexity. For each group, we use the median token usage of the strong model as a reference point. If the strong model answers correctly but consumes noticeably more tokens than typical questions in the same group, we label the sample as High Level; otherwise, it is labeled Low Level. All labels are then manually checked to ensure consistency.

The router is expected to reflect the decision boundary of weak models, we choose small language models as the base for router training. We perform Supervised Fine-Tuning (SFT) on several small models and evaluate their classification performance using F1 score, which reflects the model’s ability to distinguish between simple and complex samples.

3.3 Token Efficient Control

We adopt Generative RL with Policy Optimization (Shao et al., 2024) whose surrogate loss for a trajectory (x_t, y_t^G) is:

$$L = E \left[\min \left(\frac{\pi_\theta(y_t^G|x_t)}{\pi_{\theta_{\text{ref}}}(y_t^G|x_t)} \hat{R}_t^G, \text{clip} \left(\frac{\pi_\theta(y_t^G|x_t)}{\pi_{\theta_{\text{ref}}}(y_t^G|x_t)}, 1 - \epsilon, 1 + \epsilon \right) \hat{R}_t^G \right) - \lambda D_{\text{KL}}[\pi_\theta \parallel \pi_{\text{ref}}] \right]. \quad (1)$$

Explanation of Symbols:

- $\frac{\pi_\theta(y_t^G|x_t)}{\pi_{\theta_{\text{ref}}}(y_t^G|x_t)}$ is the ratio between the current and reference policies’ action probabilities.
- \hat{R}_t^G is the advantage estimate, measuring the advantage at a specific time step by comparing the actual return to the expected return
- $\text{clip}(\cdot, 1 - \epsilon, 1 + \epsilon)$ restricts the ratio to avoid excessively large updates.
- $D_{\text{KL}}[\pi_\theta \parallel \pi_{\theta_{\text{ref}}}]$ is the KL divergence that penalizes large deviations between the current and reference policies.
- λ is the hyperparameter controlling the weight of the KL divergence penalty.

We employ GRPO to guide the model toward an optimal answer format. By designing a multi dimensional reward function, the model automatically explores the best response patterns during training. The individual reward components are defined as follows:

- **Formatting Reward.** This component scores how well the response matches the required

structure:

$$r_{\text{fmt}} = \begin{cases} 1, & \text{they fully match,} \\ 0.5, & \text{they partially match,} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

- **Correctness Reward.** This component incentivizes the model to produce accurate answers. We adopt a simple binary reward: if the model’s final answer is judged as correct (by LLM-as-judge), it receives a reward; In this way, the correctness signal is disentangled from output length, and other reward components can independently regularize efficiency.

$$r_{\text{corr}} = \begin{cases} 1, & \text{if } y \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

- **Length Control Reward.** This component provides a linearly decaying score when the response length $t = |y|$ falls outside the ideal range $[L_{\text{min}}, L_{\text{max}}]$ but within the buffer b , with the score reduced to zero if the length exceeds the buffer. The score is formulated by:

$$r_{\text{len}} = \begin{cases} 1, & L_{\text{min}} \leq t \leq L_{\text{max}}, \\ 1 - \frac{L_{\text{min}} - t}{b}, & L_{\text{min}} - b \leq t < L_{\text{min}}, \\ 1 - \frac{t - L_{\text{max}}}{b}, & L_{\text{max}} < t \leq L_{\text{max}} + b, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

For simple benchmarks, the model is encouraged to produce concise reasoning traces by setting a lower target length range with a small tolerance buffer. In contrast, for more challenging problems such as those requiring multi-step mathematical reasoning, we expand both the lower and upper bounds and increase the buffer size. In Algorithm 1, we describe the process of designing the parameters.

- **Total Reward** The total reward then takes the sum of these components, calculated by:

$$r_{\text{sum}} = r_{\text{fmt}} + r_{\text{corr}} + r_{\text{len}} \quad (5)$$

3.4 Prompt Strategy Design

In this section, we explore several existing prompt-based strategies and propose that combining these

Algorithm 1 Length Control Reward Design

Input: $D = \{(t_i, r_i)\}_{i=1}^N$ - Answer records where t_i is the token count, $r_i \in \{\text{true}, \text{false}\}$, max_tokens

Output: $L_{min}, L_{max}, b // L_{min}$: average to //ken length of correct answers ; L_{max} : average //token length of wrong answers; b : length gap.

```

1:  $S_{corr}, S_{wrong} \leftarrow 0, C_{corr}, C_{wrong} \leftarrow 0$ 
2: for each  $(t_i, r_i) \in D$  do
3:   if  $t_i = max\_tokens$  then
4:     continue
5:   end if
6:   if  $r_i = \text{true}$  then
7:      $S_{corr} \leftarrow S_{corr} + t_i$ 
8:      $C_{corr} \leftarrow C_{corr} + 1$ 
9:   else
10:     $S_{wrong} \leftarrow S_{wrong} + t_i$ 
11:     $C_{wrong} \leftarrow C_{wrong} + 1$ 
12:   end if
13: end for
14:  $L_{min} \leftarrow S_{corr} / C_{corr}$ 
15:  $L_{max} \leftarrow S_{wrong} / C_{wrong}$ 
16:  $b \leftarrow L_{max} - L_{min}$ 
17: Return  $L_{min}, L_{max}, b$ 

```

strategies can lead to improved performance in model reasoning. We have analyzed the existing prompt strategies as illustrated in the Table 1.

The Concise Chain of Thought (CCOT) (Renze and Guven, 2024) strategy guides the model to generate brief and focused responses through explicit prompts and dynamic token budget (Han et al., 2024) management strategy adjusts the number of tokens used in each reasoning task based on task complexity and reasoning requirements.

By analyzing token consumption patterns from historical data, we can determine the optimal token budget for different tasks, further enhancing reasoning efficiency. We find that combining these strategies can significantly improve both the efficiency and quality of model reasoning. The specific implementation details and algorithms are provided in Appendix D.

To better understand how prompting strategies influence the efficiency of reasoning LLMs, we conduct a systematic comparison on GSM8K using three models: QwQ-32B, DS-R1-32B, and Qwen3-32B. As shown in Table 2, all four prompting strategies consistently reduce output token length across models. The resulting accuracy changes are ex-

Prompt method	Content
Concise Chain of Thought	Keep it concise
Simple Test Time Scaling	Use descending step order
Token Budget Aware	Use less than budget tokens
Token Reduction Inference	Do not use words in vocab
Our Combined prompt	Keep Concise and Step Count

Table 1: Illustrations of the other prompt methods and our combined prompts.

tremely small, often within $\pm 1\%$, indicating that prompt optimization affects token usage far more than task performance.

Prompt Strategy	QwQ-32B		DS-R1-32B		Qwen3-32B	
	Token	ACC	Token	ACC	Token	ACC
Keep Concise	35.8	-1.05	47.9	-2.13	23.0	0.00
Step Count	25.0	+1.05	16.6	+3.19	25.8	-2.08
Token Budget	45.0	0.00	33.9	-2.13	12.7	-1.04
Vocab Banned	34.6	-1.05	42.4	+1.04	24.2	+1.04
Ours	47.3	0.00	70.9	+2.13	36.8	+1.04

Table 2: Token savings and accuracy changes of different prompting strategies on the GSM8K dataset.

4 Experiments

In this section, we report experimental results that highlight the efficiency and effectiveness of our proposed framework, covering preliminary motivation studies, framework validation experiments and comparative evaluations.

4.1 Datasets

We conduct experiments on a diverse set of datasets, including GSM8K (Cobbe et al., 2021), Game24 (Yao et al., 2023), Sorting32 (Besta et al., 2024), HellaSwag (Zellers et al., 2019), CEval (Huang et al., 2023), MMLU (Hendrycks et al., 2021) and Math500 (Lightman et al., 2023). To measure the reasoning model’s ability, we select two types of datasets: reasoning datasets such as MATH500, and commonsense datasets such as CEVAL.

The evaluation metrics used in our reasoning framework include three key components: ACC , which measures the proportion of correct predictions to assess answer accuracy; $Cost$, representing the total token usage, including the router model, during the reasoning process; and ¥ , the monetary inference cost, which is calculated based on the API pricing for both input and output tokens for each model. We provide comprehensive experimental configurations and additional implementation details in the appendix C.

Methods	GAME24			SORTING32			CEVAL			MMLU			GSM8K			HELLASWAG		
	ACC	I/T	O/T	ACC	I/T	O/T	ACC	I/T	O/T	ACC	I/T	O/T	ACC	I/T	O/T	ACC	I/T	O/T
Base Model: QwQ-32B																		
COT _{best of 1}	94.1	198	3808	96.1	655	8887	88.8	284	1693	67.8	343	1354	<u>91.2</u>	235	1232	71.2	462	<u>1234</u>
COT _{best of 5}	82.9	694	3692	91.0	1283	8948	<u>88.5</u>	765	2415	66.1	797	2021	92.1	652	1421	66.1	912	1811
GOT	87.1	692	9179	92.5	1904	9865	75.2	1114	8214	64.2	806	6055	82.4	736	6653	65.3	1026	7237
CASCADE	82.2	212	<u>3124</u>	82.9	611	4612	83.6	543	2117	69.2	629	2351	84.1	211	1460	72.7	805	1337
SELFREFINE	83.2	317	4414	86.3	350	6750	82.3	422	3125	70.1	519	4851	74.9	347	1274	70.2	915	1435
SPP	79.7	201	3144	85.6	311	<u>4512</u>	81.8	271	<u>1432</u>	<u>72.6</u>	301	<u>1252</u>	83.2	236	1291	71.9	445	1329
MADJUDGE	88.4	307	4130	91.0	404	4923	82.3	392	2578	74.6	387	1901	76.4	340	<u>1194</u>	67.2	519	1375
Weaker _{DS-1.5B}	61.2	108	3407	2.0	470	3601	42.2	154	1574	42.7	206	1315	78.6	261	826	28.1	162	1772
Our method	<u>89.9</u>	243	2205	<u>95.2</u>	551	3172	86.9	332	1272	72.1	370	1001	90.3	293	877	<u>72.3</u>	265	1179

Table 3: Comparison of our method against typical reasoning framework across multiple benchmarks. I/T denotes the average number of input tokens per sample, O/T denotes the average number of output tokens, and ACC denotes overall answer accuracy. All experiments use QwQ-32B by default.

4.2 Primary Performance Achievements

As shown in Table 3, we use the QwQ-32B model alone to represent traditional, non-routing frameworks; in contrast, we introduce DeepSeek-R1-1.5B as a router to enable dynamic dispatch. To ensure fairness, all evaluations are conducted within the QwQ-32B ecosystem. The results show that our routing strategy incurs only a negligible drop in accuracy while substantially reducing both input and output token consumption: overall accuracy closely matches that of CoT, yet average token usage falls by roughly 50% across multiple datasets.

As shown in Table 4, we compares our router against two families of methods reinforcement learning classifiers and explicit router designs across GSM8K, MATH500 and AIME 2024. Here, Ratio denotes the proportion of instances dispatched to the strong model by the router. For a fair comparison, both our router and the strong expert use the same DeepSeek-R1-1.5B backbone. The results show that Our Router matches or exceeds the highest accuracies of all baselines while dramatically reducing total token cost, demonstrating its superior balance of accuracy and efficiency.

4.3 Ablation Experiments

In this comparative study, we conduct comparison experiments on each of the core modules of our system.

4.3.1 Comparing Various Prompt Strategies

As shown in Table 5, ACC denotes answer accuracy; Cost refers to total token consumption; FCT is the token position at which the first correct answer appears lower values indicate faster correctness, with remaining tokens used for verification; and TN measures the number of reasoning steps via keyword-based chain segmentation.

Method	GSM8K		MATH500		AIME2024	
	ACC	CostRatio	ACC	CostRatio	ACC	Cost Ratio
Base Model: DeepSeek-R1-Distill-Qwen-1.5B						
Original _{Thinking}	71.3	987	0	80.8	4084	0
Original _{NoThinking}	69.8	280	0	67.2	658	0
DPO _{Shortest}	77.3	913	0	81.2	3819	0
OverThink	76.1	781	0	80.1	4220	0
DAST	75.9	598	0	82.0	2611	0
O1-Pruner	73.5	<u>577</u>	0	80.2	3523	0
TLMRE	<u>79.8</u>	912	0	84.6	3523	0
ModelMerging	79.1	730	0	62.5	2932	0
THINKPRUNE	78.2	714	0	<u>82.6</u>	2953	0
RouterRandom	72.1	1124	50.0	76.2	3184	50.0
Thinkless	79.0	731	13.3	81.3	2690	51.4
RFTMixThink	74.0	1123	8.8	70.4	4623	33.4
AdaptThink	78.1	491	86.9	81.5	<u>2124</u>	76.8
Our Router	80.8	614	42.2	81.6	2044	62.0
				33.3	7521	90.0

Table 4: Evaluation of the small model DS-R1-1.5B against several reinforcement learning and router baselines. Our Router matches or exceeds their accuracy while reducing average token consumption by a substantial margin.

Our combined prompt strategy consistently improves token efficiency across a variety of LLMs achieving approximately 37% savings on QwQ-32B, 49% on Qwen3-32B, 8% on DeepSeek-R1-Qwen-32B, 17% on DeepSeek-R1-Llama-70B, and 16% on DeepSeek-R1 while also yielding modest accuracy gains. This suggests that our prompts effectively curb overthinking, allowing models to arrive at correct answers more quickly.

4.3.2 Ablation Study of weaker LLM

As shown in Table 6 which compares several weaker language models before and after training with our reward shaped GRPO framework across seven benchmarks. For each model, the top column shows accuracy and token cost without fine-tuning, while the bottom column reports the same metrics after applying our reward functions. Our results show that reward optimization substantially im-

Strong Models	CEVAL				GSM8K				MATH500						
	ACC	Cost	FCT	TN	¥	ACC	Cost	FCT	TN	¥	ACC	Cost	FCT	TN	¥
QwQ-32B	85.4	2302	26	52	14.1	94.0	1153	231	39	7.1	89.8	3180	559	100	19.7
QwQ-32B w/ prompt	89.6	1459	12	31	9.1	93.1	967	213	27	6.0	92.8	2657	488	84	16.5
Qwen3-32B	82.6	2229	12	50	18.1	94.0	1320	246	42	10.8	95.1	3641	694	98	29.7
Qwen3-32B w/ prompt	86.2	1148	10	22	9.4	93.5	1102	214	34	9.0	95.4	2569	439	68	21.1
DS-R1-Qwen-32B	83.1	1010	8	21	6.3	91.2	534	112	3	3.3	91.3	1668	663	34	10.4
DS-R1-Qwen-32B w/ prompt	85.8	933	6	19	5.8	92.2	465	95	2	2.9	91.3	1558	307	32	9.7
DS-R1-Llama-70B	77.1	1218	57	19	8.1	87.3	507	99	4	3.6	91.2	1914	340	50	13.1
DS-R1-Llama-70B w/ prompt	77.6	1010	8	15	6.3	90.1	434	93	3	3.1	91.5	1352	276	26	9.7
DeepSeek-R1	90.5	1191	11	26	19.2	87.1	834	179	24	13.4	89.1	1776	362	46	28.7
DeepSeek-R1 w/ prompt	91.5	998	9	20	16.1	89.9	592	148	16	9.5	93.0	1609	340	44	26.0

Table 5: Evaluation of the combined prompting strategy across multiple strong RLMS and benchmarks.

Weaker Models	GAME24		SORTING32		MMLU		CEVAL		GSM8K		HELLASWAG		MATH500		AVG	
	ACC	Cost	ACC	Cost	ACC	Cost	ACC	Cost	ACC	Cost	ACC	Cost	ACC	Cost	ACC	Cost
R1-1.5B	0	3515	0	2571	40.2	1004	41.5	1521	71.3	987	24.4	934	80.8	4084	36.9	2088
w/ GRPO	61.2	3230	2.1	1768	42.2	890	42.7	1052	78.6	655	28.1	730	84.1	2701	48.1	1575
Qwen3-1.7B	81.9	3465	16.2	5621	49.8	1797	51.5	2279	73.2	868	50.8	1125	80.2	3404	57.7	2651
w/o think	35.2	4857	1.8	880	51.2	656	51.8	685	65.1	255	47.2	532	75.9	722	46.9	1227
w/ GRPO	85.1	2250	17.1	5445	63.1	1129	68.7	1741	75.2	795	61.6	894	89.4	2267	65.7	2074
Qwen2.5-7B	8.2	697	4.2	997	58.0	592	81.2	390	52.4	510	44.1	428	59.4	955	44.4	653
w/ GRPO	20.3	610	7.1	724	79.1	483	82.1	316	77.3	436	58.2	665	67.5	798	56.4	576
Qwen2.5-1.5B	0	2305	0	1068	40.3	393	59.8	414	62.9	376	41.2	467	43.3	940	36.0	852
w/ GRPO	28.7	593	2.9	738	50.1	302	60.3	348	64.1	309	51.5	352	57.6	701	45.2	478

Table 6: Models trained with our rewards achieve higher accuracy and lower token cost across multiple datasets.

proves model reasoning: tasks previously unsolvable for weaker models—such as GAME24 and SORTING32. GRPO training consistently reduces input and output token usage, demonstrating more efficient inference. These gains appear across architectures from 1.5B to 7B parameters, confirming that our multidimensional reward design is broadly applicable and effective in guiding smaller models toward higher accuracy with lower computational cost.

Table 7 presents the ablation study across five difficulty levels of MATH500 for both DeepSeek-R1-7B and Qwen3-8B, highlighting the role of each reinforcement learning reward component. Removing any reward term—including the length reward r_{len} , the format reward r_{fmt} , or the correctness reward r_{corr} —leads to noticeable performance degradation, manifested as reduced accuracy, lower reasoning efficiency, or substantially increased token usage.

The absence of r_{corr} or r_{fmt} in particular causes sharp drops in accuracy and unstable reasoning behavior, while removing GRPO optimization further degrades both solution quality and token efficiency. Interestingly, our experiments reveal that reinforcement learning benefits are highly scale-dependent: for models under 7B parameters, RL produces significant accuracy gains with smaller models exhibiting larger improvements, likely due to learning new

structural priors or solution strategies. However, at the 7B scale, reinforcement learning yields limited accuracy improvement on MATH500; although ablating reward terms still harms performance, most of the observed token reductions stem primarily from the length-control reward rather than from enhanced reasoning capability.

These findings collectively indicate that the full reinforcement learning strategy is crucial for stabilizing reasoning depth, improving correctness, and regulating token consumption, while also revealing diminishing accuracy returns for larger models.

4.3.3 LLM Router

As shown in Table 8, the table reports the F1 scores and routing ratios of four router models across five benchmark tasks, both before and after supervised fine-tuning (SFT). All models achieve substantial F1 improvements following SFT. The increased proportion of instances routed to the strong model further indicates that the finetuned routers learn to better identify challenging queries and delegate them accordingly.

Table 9 evaluates the impact of different length-control settings on math reasoning tasks. Each configuration specifies a minimum length L_{min} , a maximum length L_{max} , and a bias parameter b , with the resulting token usage and accuracy reported in the table. Setting A enforces a moderate

Method	Math500 L1				Math500 L2				Math500 L3				Math500 L4				Math500 L5			
	ACC	TN	FCT	Tokens	ACC	TN	FCT	Tokens	ACC	TN	FCT	Tokens	ACC	TN	FCT	Tokens	ACC	TN	FCT	Tokens
DeepSeek-R1-7B																				
Ours	0.98	9	371	785	0.97	8	365	827	0.99	11	421	1043	0.94	18	592	1613	0.86	49	727	3052
w/o r_{len}	0.95	5	273	775	0.94	4	377	837	0.98	8	454	1278	0.88	9	504	1765	0.81	37	1430	3608
w/o r_{fmt}	0.93	3	238	657	0.87	3	309	733	0.79	4	395	1040	0.79	7	438	1507	0.65	30	1674	3112
w/o r_{corr}	0.93	11	209	1276	0.92	10	223	596	0.90	7	253	1068	0.76	13	351	941	0.64	20	378	2183
w/o GRPO	0.98	1	212	878	0.97	7	288	1060	0.92	11	558	1778	0.89	22	829	2729	0.84	72	3046	5748
Qwen3-8B																				
Ours	0.98	2	237	471	0.98	1	270	498	0.96	3	309	642	0.95	5	379	1058	0.93	11	565	1628
w/o r_{len}	0.98	16	457	1366	0.98	16	514	1446	0.96	27	700	2272	0.95	43	859	3190	0.91	81	2980	5713
w/o r_{fmt}	0.95	3	141	482	0.91	6	167	539	0.91	12	459	926	0.83	21	492	1977	0.88	32	1312	2665
w/o r_{corr}	0.99	4	256	638	0.98	3	356	647	0.96	5	356	786	0.95	11	479	1378	0.90	30	1024	2881
w/o GRPO	0.97	26	971	2082	0.99	30	1639	2526	0.96	44	1616	3500	0.95	68	2389	5397	0.93	104	5417	7656

Table 7: Ablation of the rewards on Math500 across five difficulty levels for DeepSeek-R1-7B and Qwen3-8B.

Router Models	MMLU		CEVAL		GSM8K		HELLASWAG		MATH500	
	F1	Ratio	F1	Ratio	F1	Ratio	F1	Ratio	F1	Ratio
R1-Distill-Qwen-1.5B	0.49	21.3%	0.52	21.8%	0.75	22.9%	0.63	72.3%	0.71	11.3%
R1-Distill-Qwen-1.5B _{SFT}	0.64	61.7%	0.76	71.2%	0.83	34.8%	0.73	90.4%	0.85	54.2%
Qwen3-1.7B	0.68	36.5%	0.78	70.3%	0.75	10.0%	0.64	50.1%	0.80	5.4%
Qwen3-1.7B _{SFT}	0.70	47.7%	0.84	73.6%	0.85	64.5%	0.74	77.6%	0.87	74.2%
Qwen2.5-7B	0.51	4.9%	0.81	3.8%	0.51	3.8%	0.45	3.1%	0.61	7.2%
Qwen2.5-7B _{SFT}	0.71	74.5%	0.87	71.5%	0.73	28.8%	0.76	92.0%	0.79	61.4%

Table 8: Performance of Router Models on Reasoning Benchmarks: F1 Score and Routing Ratio. It denotes the proportion of instances dispatched to the strong model.

Setting	Lmin	Lmax	b	Tokens	Acc
A	900	3000	500	2748	92.2
B	900	6000	500	5434	91.6
C	0	3000	500	3268	86.8

Table 9: Evaluation of length control on math reasoning tasks. Each setting specifies a length range, a bias parameter b , and reports the resulting token usage and accuracy.

Model	GPQA		HLE		MMLU-PRO	
	F1	Ratio%	F1	Ratio%	F1	Ratio%
7B	0.60	96	0.49	95	0.60	52
7B-SFT	0.71	98	0.56	97	0.71	78
1.5B	0.52	92	0.48	94	0.54	61
1.5B-SFT	0.61	96	0.51	96	0.63	67

Table 10: Robustness evaluation of the router on OOD reasoning tasks from Qwen2.5-Instruct. Each task reports two metrics: F1 score and routing ratio. HLE represents Human Last Exam.

length range and achieves the best balance between conciseness and accuracy, reducing tokens while maintaining a high accuracy of 92.2. Setting B allows significantly longer outputs, which increases token consumption without improving accuracy. In contrast, Setting C removes lower-bound constraints and leads to shorter but less reliable reasoning, reflected by a substantial accuracy drop to 86.8. These results demonstrate that appropriate length control can effectively regulate output verbosity while preserving model correctness.

Table 10 reports the robustness evaluation of the router on several OOD reasoning benchmarks. The results show that models fine-tuned with supervised signals generally achieve higher F1 scores and improved routing behavior compared to their base counterparts. These findings indicate that supervised fine-tuning enhances the router’s ability to generalize to OOD reasoning tasks while preserving efficient routing decisions.

5 Conclusion

In this work, we investigated the problem of reasoning budget allocation in LRMs, showing that excessively long chains of thought offer little benefit for questions the model already answers correctly, while being appropriate only for cases where additional reasoning depth is truly needed. Simple instances can be efficiently handled by a lightweight model under tight length constraints, whereas challenging ones should be processed by LRMs equipped with optimized reasoning prompts. Our approach consistently reduces token usage while preserving a competitive accuracy in comparison with strong routing and reasoning baselines. The key insight of this work is that efficient reasoning arises not from prolonged thinking, but from allocating only the computation that each problem truly requires.

472 Limitations

473 We acknowledge three limitations in our study.

474 First, reinforcement learning is only applied to
475 relatively small models; applying GRPO to larger
476 models may further improve performance, but also
477 poses significant training cost and stability chal-
478 lenges.

479 Secondly, our routing and prompting strategies
480 are designed and evaluated under a fixed routing
481 backbone and task types. Adapting these strate-
482 gies to more diverse tasks multi-hop QA, code gen-
483 eration or dynamic router-controller architectures
484 remains an open direction for future work.

485 Lastly, our experiments are conducted exclu-
486 sively on English and Chinese datasets. For other
487 languages the behavior of tokenization and genera-
488 tion may differ significantly due to linguistic and
489 token structure differences, potentially affecting
490 both accuracy and token efficiency.

491 Potential Risks

492 Although our framework aims to reduce token us-
493 age via output length control, excessive compres-
494 sion may omit critical reasoning steps. This can
495 result in incorrect answers or shortcut behaviors,
496 particularly in multi-step or mathematical reason-
497 ing tasks. A balance between brevity and reasoning
498 completeness must be maintained.

499 The effectiveness of our system heavily depends
500 on the router’s ability to assess question difficulty.
501 Misclassification could lead to complex tasks being
502 assigned to weak models, reducing answer quality.
503 If the router is biased or poorly calibrated, it may
504 introduce systemic errors in downstream inference.

505 References

506 Daman Arora and Andrea Zanette. 2025. [Training](#)
507 [language models to reason efficiently](#). *Preprint*,
508 arXiv:2502.04463.

509 Maciej Besta, Nils Blach, Ales Kubicek, Robert Ger-
510 stenberger, Michał Podstawski, Lukas Gianinazzi,
511 Joanna Gajda, Tomasz Lehmann, Hubert Niewiadow-
512 ski, Piotr Nyczyk, and Torsten Hoefler. 2024. [Graph](#)
513 [of thoughts: solving elaborate problems with large](#)
514 [language models](#). In *Proceedings of the Thirty-*
515 *Eighth AAAI Conference on Artificial Intelligence*
516 *and Thirty-Sixth Conference on Innovative Applica-*
517 *tions of Artificial Intelligence and Fourteenth Sym-*
518 *posium on Educational Advances in Artificial Intelli-*
519 *gence, AAAI’24/IAAI’24/EAAI’24*. AAAI Press.

520 Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He,
521 Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi

Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, 522
Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. [Do](#)
523 [not think that much for 2+3=? on the overthinking of](#)
524 [o1-like llms](#). *Preprint*, arXiv:2412.21187. 525

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, 526
Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias 527
Plappert, Jerry Tworek, Jacob Hilton, Reiichiro 528
Nakano, Christopher Hesse, and John Schulman. 529
2021. [Training verifiers to solve math word prob-](#)
530 [lems](#). *CoRR*, abs/2110.14168. 531

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea 532
Bobu, and Jacob Andreas. 2025. [Learning how hard](#)
533 [to think: Input-adaptive allocation of LM computa-](#)
534 [tion](#). In *The Thirteenth International Conference on*
535 *Learning Representations*. 536

DeepSeek-AI, Daya Guo, and Yang. 2025. [DeepSeek-](#)
537 [R1: Incentivizing Reasoning Capability in LLMs](#)
538 [via Reinforcement Learning](#). *arXiv e-prints*,
539 arXiv:2501.12948. 540

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. 541
[Thinkless: Llm learns when to think](#). *Preprint*,
542 arXiv:2505.13379. 543

Zhaopeng Feng, Jiayuan Su, Jiamei Zheng, Jiahao Ren, 544
Yan Zhang, Jian Wu, Hongwei Wang, and Zuozhu 545
Liu. 2025. [M-mad: Multidimensional multi-agent](#)
546 [debate for advanced machine translation evaluation](#).
547 *Preprint*, arXiv:2412.20127. 548

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu 549
Zhao, Shiqing Ma, and Zhenyu Chen. 2024. [Token-](#)
550 [Budget-Aware LLM Reasoning](#). *arXiv e-prints*,
551 arXiv:2412.18547. 552

Yang He, Xiao Ding, Bibo Cai, Yufei Zhang, Kai 553
Xiong, Zhouhao Sun, Bing Qin, and Ting Liu. 2025. 554
[Self-route: Automatic mode switching via capa-](#)
555 [bility estimation for efficient reasoning](#). *Preprint*,
556 arXiv:2505.20664. 557

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, 558
Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 559
2021. [Measuring massive multitask language under-](#)
560 [standing](#). *Preprint*, arXiv:2009.03300. 561

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, 562
Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 563
2025. [Thinkprune: Pruning long chain-of-thought](#)
564 [of llms via reinforcement learning](#). *Preprint*,
565 arXiv:2504.01296. 566

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei 567
Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu,
568 Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu,
569 Maosong Sun, and Junxian He. 2023. [C-eval: A](#)
570 [multi-level multi-discipline chinese evaluation suite](#)
571 [for foundation models](#). *Preprint*, arXiv:2305.08322. 572

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri 573
Edwards, Bowen Baker, Teddy Lee, Jan Leike,
574 John Schulman, Ilya Sutskever, and Karl Cobbe.
575 2023. [Let’s verify step by step](#). *Preprint*,
576 arXiv:2305.20050. 577

578	Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. instag: Instruction tagging for analyzing supervised fine-tuning of large language models . <i>Preprint</i> , arXiv:2308.07074.	632
579		633
580		634
581		635
582		636
583		637
584	Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning . <i>Preprint</i> , arXiv:2501.12570.	638
585		639
586		640
587		641
588	Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning Models Can Be Effective Without Thinking . <i>arXiv e-prints</i> , arXiv:2504.09858.	642
589		643
590		644
591		645
592	Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning . <i>Preprint</i> , arXiv:2502.09601.	646
593		647
594		648
595		649
596	Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhunoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback . <i>Preprint</i> , arXiv:2303.17651.	650
597		651
598		652
599		653
600		654
601		655
602		656
603		657
604	Qwen3. 2025. Qwen3: Think deeper, act faster .	658
605	Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model . <i>Preprint</i> , arXiv:2305.18290.	659
606		660
607		661
608		662
609		663
610	Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models . In <i>2024 2nd International Conference on Foundation and Large Language Models (FLLM)</i> , page 476–483. IEEE.	664
611		665
612		666
613		667
614		668
615	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models . <i>Preprint</i> , arXiv:2402.03300.	669
616		670
617		671
618		672
619		673
620		674
621	Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models . <i>Preprint</i> , arXiv:2503.04472.	675
622		676
623		677
624		678
625		679
626	Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning . <i>Preprint</i> , arXiv:2409.12183.	680
627		681
628		682
629		683
630		684
631		685
	Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models . <i>arXiv e-prints</i> , arXiv:2503.16419.	686
		687
		688
		689
	Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning . In <i>Findings of the Association for Computational Linguistics: NAACL 2025</i> , pages 6904–6917, Albuquerque, New Mexico. Association for Computational Linguistics.	690
		691
	Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2024. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration . <i>Preprint</i> , arXiv:2307.05300.	692
		693
		694
		695
	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models . In <i>Advances in Neural Information Processing Systems</i> , volume 35, pages 24824–24837. Curran Associates, Inc.	696
		697
	Han Wu, Yuxuan Yao, Shuqi Liu, Zehua Liu, Xiaojin Fu, Xiongwei Han, Xing Li, Hui-Ling Zhen, Tao Zhong, and Mingxuan Yuan. 2025. Unlocking efficient long-to-short llm reasoning with model merging . <i>Preprint</i> , arXiv:2503.20641.	698
		699
	Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, Qunshu Lin, Junbo Zhao, Zhaoxiang Zhang, Wenhao Huang, Ge Zhang, Chenghua Lin, and J. H. Liu. 2024. A Comparative Study on Reasoning Patterns of OpenAI’s o1 Model . <i>arXiv e-prints</i> , arXiv:2410.13639.	700
		701
	Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models . <i>arXiv e-prints</i> , arXiv:2501.09686.	702
		703
	Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. 2025. Dynamic early exit in reasoning models . <i>Preprint</i> , arXiv:2504.15895.	704
		705
	Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: deliberate problem solving with large language models . In <i>Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23</i> , Red Hook, NY, USA. Curran Associates Inc.	706
		707
		708
		709

- 690 Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu
691 Yao. 2024. [Large language model cascades with](#)
692 [mixture of thoughts representations for cost-efficient](#)
693 [reasoning](#). *Preprint*, arXiv:2310.03094.
- 694 Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali
695 Farhadi, and Yejin Choi. 2019. [Hellaswag: Can](#)
696 [a machine really finish your sentence?](#) In *ACL (1)*,
697 pages 4791–4800.
- 698 Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and
699 Juanzi Li. 2025a. [Adaptthink: Reasoning models can](#)
700 [learn when to think](#). *Preprint*, arXiv:2505.13417.
- 701 Yufeng Zhang, Xuepeng Wang, Lingxiang Wu, and
702 Jinqiao Wang. 2025b. [Enhancing chain of thought](#)
703 [prompting in large language models via reasoning](#)
704 [patterns](#). *Proceedings of the AAAI Conference on*
705 *Artificial Intelligence*, 39(24):25985–25993.
- 706 Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex
707 Smola. 2023. [Automatic chain of thought prompting](#)
708 [in large language models](#). In *The Eleventh Interna-*
709 *tional Conference on Learning Representations*.

A Related work

A.1 LRMs and Reinforcement Learning

LRMs have emerged as a powerful extension of large language models, specifically targeting complex reasoning tasks such as mathematical problem solving, multi step logical inference, and structured question answering.

Supervised fine-tuning (SFT) (Lu et al., 2023) aligns LRMs to human annotated data by optimizing their likelihood on target examples. Building on this foundation, reinforcement learning techniques further boost their reasoning abilities. Direct Preference Optimization (DPO) (Rafailov et al., 2024) is a method that directly learns from human preference rankings between outputs. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a flexible optimization framework that supports complex reward signals.

A.2 Structured Reasoning Framework

Prior works have shown that guiding intermediate reasoning steps significantly improves performance, such as CoT, Automatic CoT (Zhang et al., 2023), ToT and GoT. Self-Refine (Madaan et al., 2023) produces self-feedback and LLM-Cascade (Yue et al., 2024) dynamically routes between weaker and stronger models. Multi-Agent Debate (MAD) judge (Feng et al., 2025) orchestrates a three-agent debate with a judge module. Task-Solving Agent through Multi-Persona Self-Collaboration (SPP) (Wang et al., 2024) instantiates multiple personas of a single LLM to collaboratively solve problems.

learning how hard to think (Damani et al., 2025) propose an adaptive computation allocation method that dynamically adjusts computational resources for the decoding process based on the difficulty of the input, improving efficiency or quality. Experimental results show that this method can reduce computation by up to 50% without sacrificing quality, or improve quality by up to 10% at a fixed computational budget.

A.3 Token Efficiency Methods

The survey Stop Overthinking (Sui et al., 2025) systematically categorizes current methods for efficient LLM reasoning.

The work self-route (He et al., 2025) employ a dynamic routing framework. Dynamic Early Exit (Yang et al., 2025) introduce a confidence-based

early-exit mechanism. Effective Without Thinking (Ma et al., 2025) show that skipping explicit thinking in LLMs using simple nothinking prompts. As showed in appendix we compared typical long chain reasoning methods with recent token efficiency strategies and expose their inherent limitations.

OverThink (Chen et al., 2025) treats the original long chain reasoning output as a negative example. Difficulty Adaptive Slow Thinking (DAST) (Shen et al., 2025) constructs its preference dataset by ranking presampled responses with a length-based reward function. O1-Pruner (Luo et al., 2025) evaluates a reference model’s performance through pre sampling. TLMRE (Arora and Zanette, 2025) incorporates a token length penalty directly into on policy reinforcement learning training. ModelMerging (Wu et al., 2025) reduces reasoning length by linearly combining the weights of a reasoning capable model with those of a non reasoning model.

RFTMixThinking samples multiple thinking and nothinking outputs per example. THINKPRUNE (Hou et al., 2025) is an iterative RL based token limited pruning method. Thinkless (Fang et al., 2025) is a decoupled algorithm to adaptively switch between concise and detailed reasoning. AdaptThink (Zhang et al., 2025a) is a framework that adaptively balances thinking and nothinking modes through a constrained optimization objective and importance sampling.

Difficulty Adaptive Self Consistency (Wang et al., 2025) is a method that uses both prior and posterior difficulty signals to dynamically allocate sampling budgets, greatly reducing self-consistency inference cost while maintaining comparable reasoning accuracy. CoT-Valve (Ma et al., 2025) is a parameter-space tuning strategy that enables controllable and compressible chain-of-thought generation, significantly shortening reasoning chains with minimal performance loss.

As shown in Figure 4, existing long chain reasoning frameworks achieve reasoning correctness at the cost of high token consumption and computation overhead, while recent token-efficient methods aim to preserve reasoning performance while drastically reducing token cost.

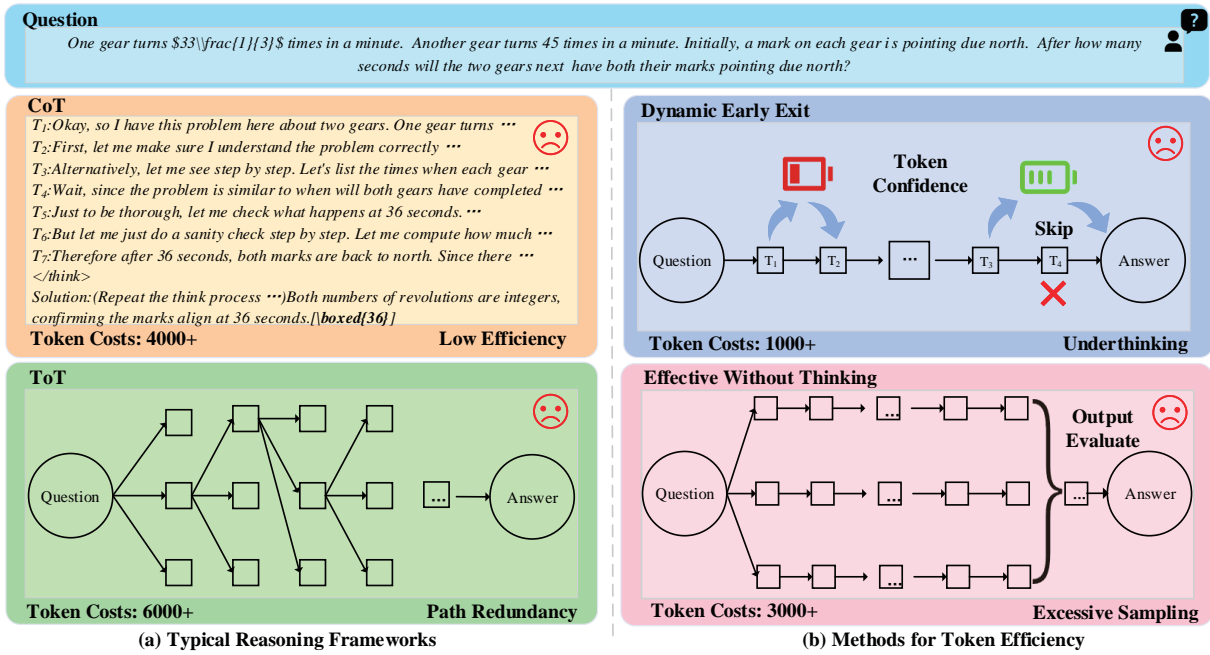


Figure 4: Comparison of typical long chain reasoning methods and recently token efficiency methods.

B Empirical Analysis of Overthinking

As shown in the figure 5 the Left Panel shows that accuracy improves in reasoning mode, but at the cost of increased token consumption. The Middle Panel highlights a significant rise in token usage when reasoning is applied. The Right Panel displays the thought chain count and the position of the first correct tokens, with reasoning mode leading to more thought chains and higher token consumption. We have analyzed the phenomenon of overthinking across multiple datasets, and from a broader perspective, reasoning mode introduces more thought chains, which, while improving accuracy, also leads to an increase in token consumption.

Overthinking can occur in both correctly answered and incorrectly answered questions, indicating that it is not merely a by-product of failure cases but rather an inherent behavioral pattern of the model’s reasoning process. When the model generates the correct key information at an early stage (reflected by a small FCT), yet continues to elaborate with lengthy, repetitive, or tangential reasoning (reflected by a large TN), this still constitutes overthinking despite the final answer being correct. Conversely, for incorrectly answered questions, the model may exhibit excessive reasoning due to repeated attempts, shifts in reasoning strategies, or uncertainty, likewise resulting in abnormally large TN values. Therefore, overthinking

is neither a sufficient condition for incorrectness nor the opposite of correctness—it is a systematic form of redundant reasoning that emerges when the model faces uncertainty or lacks an effective stopping mechanism.

The data presented in this table 11 provides a detailed analysis of the performance of the DeepSeek-R1-32B and QwQ-32B models on the MATH500 dataset, with different token budgets and difficulty levels. It shows the relationship between the token budget, accuracy, though number, and first correct token for each model.

Generally, increasing the token budget improves accuracy particularly in lower difficulty levels. For example, in Level 1, DeepSeek-R1-32B improves from 0.884 accuracy at 1024 tokens to 0.907 at 16384 tokens, before slightly decreasing to 0.884 at 32768 tokens. Similarly, QwQ-32B improves from 0.326 accuracy with 1024 tokens to 0.854 at 16384 tokens, but then slightly declines to 0.850 at 32768 tokens. However in higher difficulty levels the improvement in accuracy becomes less significant. For instance, in Level 5, DeepSeek-R1-32B shows only a modest improvement from 0.209 at 1024 tokens to 0.582 at 32768 tokens, and QwQ-32B goes from 0.030 to 0.686, indicating diminishing returns in terms of accuracy improvement with larger token budgets.

Along with the increase in token budget, the though number for both models also increase. For

example, in Level 1, DeepSeek-R1-32B’s thinking steps increase from 43 at 1024 tokens to 279 at 32768 tokens, while QwQ-32B goes from 472 to 980. This trend continues across all difficulty levels, suggesting that the models engage in more reasoning steps as the token budget increases.

The first correct token position also becomes later in the sequence with higher token budgets, meaning that the models require more tokens to generate the first correct answer. In Level 1, for instance, DeepSeek-R1-32B’s FCT moves from 43 at 1024 tokens to 279 at 32768 tokens, and QwQ-32B shows a similar shift, from 472 to 980. This trend is particularly noticeable in higher difficulty levels, where the first correct token appears much later, such as in Level 5, where DeepSeek-R1-32B’s FCT increases from 370 at 1024 tokens to 1854 at 32768 tokens.

From this data, we observe the presence of overthinking. As the token budget increases, the models show improved accuracy, but this improvement becomes less significant in higher difficulty levels. At the same time, the number of thinking steps required and the time taken to generate the first correct answer both increase. Despite the additional tokens, the accuracy does not improve proportionally, indicating that the models are engaging in unnecessary reasoning steps. This pattern suggests that in more difficult tasks, the models are spending excessive computational resources and time without achieving significant performance gains which points to the occurrence of overthinking.

C Experiments Details

In this appendix, we provide supplementary experimental details and supporting information omitted from the main text. Specifically, we cover dataset statistics and partitioning, prompt templates and parameter configurations, and the experimental environment along with hyperparameter settings. These materials are intended to enhance reproducibility and give readers deeper insight into the key steps of our model training and evaluation.

C.1 Dataset Statistics

The Grade School Math 8K dataset contains approximately 8,500 problems focused on elementary to middle-school-level word math. Each problem comes with a detailed step-by-step solution, making it ideal for evaluating a model’s arithmetic reasoning and chain-of-thought capabilities.

Level	Token	DeepSeek-R1-32B			QwQ-32B		
		ACC	TN	FCT	ACC	TN	FCT
L1	1024	0.884	0.16	43	0.326	8.86	472
	2048	<u>0.885</u>	0.56	227	0.767	12.42	668
	4096	0.837	1.00	199	0.861	13.63	633
	8192	0.860	0.58	235	0.884	15.70	732
	16384	0.907	0.65	200	0.854	18.98	750
	32768	0.884	1.00	279	0.850	19.60	980
L2	1024	0.700	1.67	230	0.256	9.79	546
	2048	0.744	1.86	257	0.644	15.04	696
	4096	0.833	2.81	407	0.789	19.98	937
	8192	0.845	4.62	404	<u>0.878</u>	20.17	1055
	16384	0.833	2.63	252	0.879	20.32	1165
	32768	<u>0.844</u>	6.24	323	0.868	20.51	1372
L3	1024	0.581	1.81	327	0.095	11.14	653
	2048	0.667	2.74	499	0.400	20.63	1057
	4096	0.695	4.23	446	0.619	27.77	1404
	8192	<u>0.733</u>	5.68	394	0.714	34.05	1414
	16384	0.724	7.03	451	<u>0.725</u>	35.63	1279
	32768	0.762	6.69	446	0.737	31.71	1372
L4	1024	0.406	2.10	327	0.063	9.30	523
	2048	0.500	5.29	459	0.289	20.52	844
	4096	0.617	7.63	431	0.562	33.77	1160
	8192	0.672	10.09	633	0.727	42.72	1733
	16384	0.727	9.34	495	<u>0.758</u>	49.01	1757
	32768	<u>0.688</u>	13.14	625	0.759	49.39	1685
L5	1024	0.209	5.23	370	0.030	11.78	425
	2048	0.261	10.67	600	0.119	25.97	853
	4096	0.433	17.14	729	0.336	44.78	1233
	8192	0.559	34.22	1880	0.515	68.31	2106
	16384	0.589	43.73	1998	<u>0.659</u>	81.43	3704
	32768	<u>0.582</u>	47.05	1854	0.686	93.97	4584

Table 11: Performance of DS-R1-32B and QwQ-32B on MATH500 across different token budgets and difficulty levels.

ties.

The 2024 AIME (American Invitational Mathematics Examination) question bank includes around 2,500 challenging algebra and combinatorics problems. It tests a model’s problem-solving ability and mathematical expressiveness in high-difficulty competition settings.

The professional version of the Multi-domain Language Understanding benchmark covers about 57 disciplines (including STEM, social sciences, and humanities) with roughly 20,000 multiple-choice questions. It assesses a model’s knowledge and reasoning in academic and professional domains.

CEVAL is a Chinese professional evaluation set containing around 13,000 multiple-choice questions drawn from licensure exams in fields like science, medicine, and law. It specifically measures a large model’s understanding and application of specialized Chinese professional knowledge.

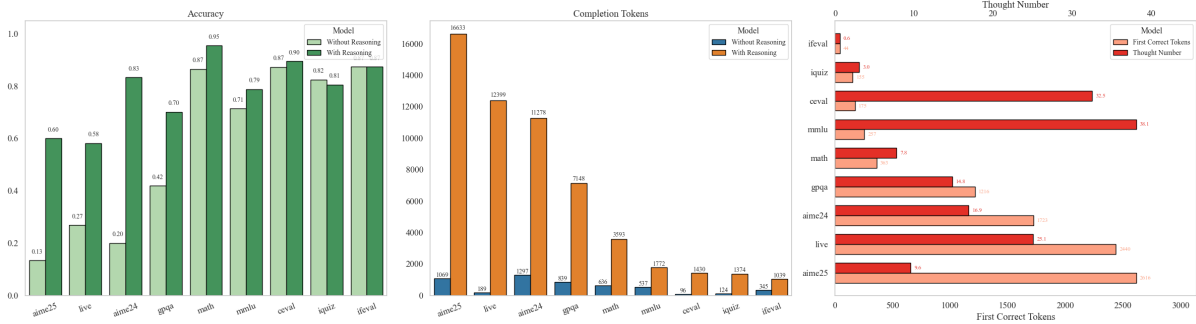


Figure 5: The performance of Qwen3-32B in reasoning and non-reasoning modes across three metrics. The Left Panel shows that accuracy improves in reasoning mode, but at the cost of increased token consumption. The Middle Panel highlights a significant rise in token usage when reasoning is applied. The Right Panel displays the thought chain count and the position of the first correct tokens, with reasoning mode leading to more thought chains and higher token consumption.

Math500 is a subset of high-level math problems consisting of about 500 high-school and competition-grade questions spanning algebra, geometry, number theory, and more. It evaluates a model’s advanced reasoning and formal expression skills on difficult math tasks.

HellaSWAG is a commonsense reasoning dataset with roughly 70,000 multiple-choice questions, each offering four possible endings. It probes a model’s understanding of everyday physical and causal scenarios, known for its highly deceptive distractors.

Sorting32 is a synthetic sequence manipulation task with about 5,000 problems requiring a model to sort sequences of up to 32 elements according to specified rules (e.g., ascending order or custom patterns). It measures algorithmic thinking and structured data processing.

Game24 is The “24-point” game dataset featuring around problems in which the model must use the four basic arithmetic operations to combine four given numbers into 24. It tests a model’s elementary arithmetic operations and search strategy integration.

C.2 Model Settings

We evaluate six representative models. QwQ-32B is built on Qwen2.5-32B-Instruct and has been instruction-tuned on general QA data. Qwen3-32B is the official instruction-tuned release of the Qwen3-32B series. DS-R1-Qwen-32B and DS-R1-Llama-70B are both distilled via the DeepSeek-R1 recipe—with GRPO fine-tuning applied to Qwen2.5-32B and Llama-70B, respectively—to compress model size while retaining reasoning abil-

ity. Qwen3-30B-MOE augments Qwen3-30B with a Mixture-of-Experts layer to sparsely activate parameters. Finally, DeepSeek-R1 is the distilled Llama-70B model produced by the DeepSeek-R1 pipeline without any additional prompt tuning, serving as a compact baseline optimized for efficient reasoning.

C.3 Prompt Settings

Instruction for Judge LLM

You are a reasoning assistant with the ability to determine whether the Model Answer matches the Reference Answer. Reference Answer: <Insert Reference Answer> Model Answer: <Insert Model Answer> Instructions: - If the core content is the same (even if phrased differently), output: Match. - If the content differs, output: Mismatch. Only return "Match" or "Mismatch" without further explanation.

Instruction for Strong LLM

You are a reasoning assistant. Please answer the following question in the following manner: 1. Follow the steps in descending order, starting from the most general to the most specific. 2. Keep your response concise, ensuring that each step is clear and to the point. 3. Use less than the given token budget and avoid unnecessary words or repetition. 4. Do not include terms outside the vocabulary required for the answer. Provide your answer in a clear, logical sequence while maintaining brevity.

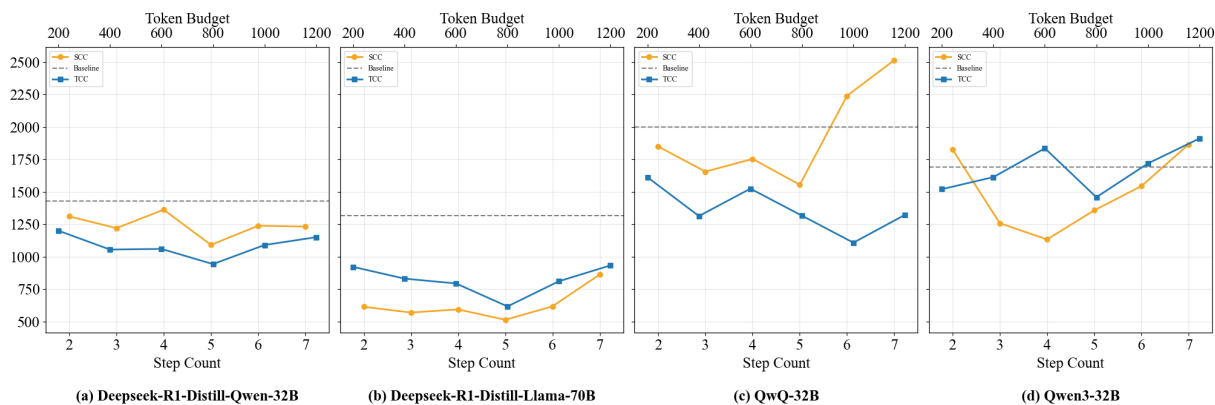


Figure 6: Model Specific Optimal Token and Step Budgets.

Instruction for Weaker LLM

You are a disciplined reasoning assistant. Follow these rules:
 In `<think>...</think>`, reason step-by-step with full details. After reasoning, provide the final answer in `<answer>...</answer>`. The final answer should be enclosed in boxed for clarity.
 Example structure:
`<think>...</think>`: Step-by-step reasoning process.
`<answer>...</answer>`: The final answer, enclosed in boxed

Instruction for Router LLM

You are tasked with analyzing the following problem:
 1. Assess the difficulty of the problem based on its structure, content, and any specific keywords that may indicate complexity.
 2. Classify the problem into one of two categories:
 Simple: The problem can be solved using straightforward methods and does not require advanced reasoning or multiple steps.
 Complex: The problem requires deeper reasoning, multiple steps, or specialized knowledge to solve.
 Please provide your classification as either "Simple" or "Complex," and briefly explain your reasoning for this classification.

C.4 Experimental Setup

Our experiments were conducted on a machine equipped with eight H800 GPUs, a 32-core CPU, and 512 GB of memory, running Ubuntu 22, with the TRL and MS-SWIFT packages installed.

For reinforcement learning, we assembled a training corpus by concatenating the training splits of GSM8K, Math500, and HellaSWAG into a single set of 80,000 examples. After fine-tuning on this combined dataset.

For the routing component, we use LLM as a router. The strong model achieves a higher F1 score than a lighter-weight counterpart, indicating its superior ability to recognize the intrinsic complexity of a question. We quantify complexity via three heuristics in descending order of priority:

Numeric count: the number of numeric values appearing in the question (more numbers stands for higher complexity). Sentence breaks: the number of periods (.) in the text, which correlates with the number of reasoning steps. Question length: the overall character length, used as a secondary tiebreaker.

For the router module, we use QwQ-32B as the default model for routing decisions. We modify the model’s prompt template to avoid long thinking and set the sampling temperature to 0.0 to ensure deterministic and stable routing decisions.

For the weaker LLM, we adopt the Qwen2.5-Instruct model from the Tongyi Qianwen series and DeeoSeek-R1 as our base model for experimentation. To further enhance performance, we apply the GRPO reinforcement learning method. The model is trained on approximately 80,000 samples for 2 epochs, with a learning rate of 5e-6 and a maximum generation length of 32768 tokens to encourage completely output. All reinforcement learning experiments are conducted on a cluster of 8 NVIDIA RTX H800 GPUs, with a total training time of approximately ten hours.

For the strong LLM, we examine token consumption for the strong LLM through a refined lens. The left panel shows that deeper reasoning increases accuracy, the center panel reveals that this accuracy gain comes at a high token cost, and the right panel decomposes that cost into Thought Number and First Correct Tokens—demonstrating that prompt engineering to lower these two metrics produces substantial token savings. As showed in Figure 5.

For the reasoning phase, we use VLLM official default settings temperature 0.7 and no maximum token so that the model can generate full uninterrupted answers without being cut off by external

constraints thereby preserving accuracy.

D Algorithms Details

Algorithm 1 explains how we empirically analyze the average token consumption for both correct and incorrect answers, and use this analysis to determine the range for the Length Control Reward function. Algorithm 2 and Algorithm 3 illustrate our process for selecting the optimal token budget and the most efficient solution steps. Algorithm 4 describes how we determine the set of transition words, which differs from previous rule-based statistical counting methods by leveraging a large language model to analyze and select the vocabulary.

Figure 6 illustrates the design procedure used to identify optimal token budgets and reasoning step counts across four LLMs. The results show that each model exhibits distinct efficiency curves: some achieve peak performance with shallow reasoning depths, while others benefit from extended step counts.

Algorithm 2 Token Estimation

Input: Budget range $[\mu_{\text{wrong}}, \mu_{\text{corr}}]$ with step size $\Delta = 100$; evaluation set $D = \{c_i\}_{i=1}^n$; model \mathcal{M} ; prompt template $\text{Prompt}(c, b)$

Output: Optimal budget b^*

```

1: Initialize:  $results \leftarrow []$ 
2: for all  $b = \mu_{\text{wrong}}, \mu_{\text{wrong}} + \Delta, \dots, \mu_{\text{corr}}$  do
3:    $C_b \leftarrow 0, T_b \leftarrow 0$ 
4:   for all  $c \in D$  do
5:      $P \leftarrow \text{Prompt}(c, b)$ 
6:      $(r, t) \leftarrow \mathcal{M}(P) \{r \in \{0, 1\}, t \in N\}$ 
7:      $C_b += r, T_b += t$ 
8:   end for
9:    $\alpha_b \leftarrow C_b/n, \tau_b \leftarrow T_b/n$ 
10:  Append  $(b, \alpha_b, \tau_b)$  to  $results$ 
11: end for
12: Let  $\tau_{\min} \leftarrow \min\{\tau_b \mid (b, \alpha_b, \tau_b) \in results\}$ 
13:  $\mathcal{B} \leftarrow \{(b, \alpha_b) \mid (b, \alpha_b, \tau_b) \in results, \tau_b = \tau_{\min}\}$ 
14:  $b^* \leftarrow \arg \max_{(b, \alpha_b) \in \mathcal{B}} \alpha_b$ 
15: return  $b^*$ 

```

Algorithm 3 Step Estimation

Require: Candidate steps $\mathcal{S} = \{3, 4, 5, 6, 7\}$; evaluation set $D = \{c_i\}_{i=1}^n$; model \mathcal{M} ; prompt template $\text{Prompt}(c, k)$

Ensure: Optimal step k^*

```

1: Initialize:  $results \leftarrow []$ 
2: for all  $k \in \mathcal{S}$  do
3:    $C_k \leftarrow 0, T_k \leftarrow 0$ 
4:   for all  $c \in D$  do
5:      $P \leftarrow \text{Prompt}(c, k)$ 
6:      $(r, t) \leftarrow \mathcal{M}(P) \{r \in \{0, 1\}, t \in N\}$ 
7:      $C_k += r, T_k += t$ 
8:   end for
9:    $\alpha_k \leftarrow C_k/n, T_k \leftarrow T_k/n$ 
10:  Append  $(k, \alpha_k, T_k)$  to  $results$ 
11: end for
12: Let  $T_{\min} \leftarrow \min\{T_k \mid (k, \alpha_k, T_k) \in results\}$ 
13:  $\mathcal{K} \leftarrow \{(k, \alpha_k) \mid (k, \alpha_k, T_k) \in results, T_k = T_{\min}\}$ 
14:  $k^* \leftarrow \arg \max_{(k, \alpha_k) \in \mathcal{K}} \alpha_k$ 
15: return  $k^*$ 

```

Algorithm 4 Vocabulary Estimation

Require: Historical responses $D = \{c_i\}_{i=1}^N$; model \mathcal{M} ; prompt template $\text{Prompt}_{\text{vocab}}(c)$; refinement template $\text{Prompt}_{\text{refine}}(V)$

Ensure: Final vocabulary V^*

```

1:  $V_{\text{raw}} \leftarrow \emptyset$ 
2: for all  $c \in D$  do
3:    $P \leftarrow \text{Prompt}_{\text{vocab}}(c)$ 
4:    $W \leftarrow \mathcal{M}(P) \{W \text{ is token list}\}$ 
5:    $V_{\text{raw}} \leftarrow V_{\text{raw}} \cup \text{set}(W)$ 
6: end for
7:  $P' \leftarrow \text{Prompt}_{\text{refine}}(V_{\text{raw}})$ 
8:  $V^* \leftarrow \text{set}(\mathcal{M}(P'))$ 
9: return  $V^*$ 

```

Table 12: An example from Our framework on GSM8K dataset, with special symbols used in the thinking process. The Thought is highlighted in blue box and the position of correct tokens is highlighted in green box respectively.

Example #1 from GSM8K Dataset
<p>Question: Brinley is in Mr. Bert’s math class. Mr. Bert gives six tests each semester. He allows the students to remove the lowest score from the average each semester. If Brinley has scores of 89, 71, 92, 100 and 86 on the first five tests, what score does she need on the sixth test to get an average of 93?</p> <p>Labeled Answer: 98</p>
Original Output - Overthinking
<p><think>Okay, so I need to figure out what score Brinley needs on her sixth test to average 93, considering she can drop her lowest score. Let me break this down step by step. First, Brinley has taken five tests with scores: 89, 71, 92, 100, and 86. She’s about to take the sixth test, and Mr. Bert allows her to drop the lowest score when calculating the average. So, I need to determine the sixth score that would make her average 93 after dropping the lowest score. I think the first thing to do is identify which of her current scores is the lowest because that’s the one she’ll likely drop. Looking at the scores: 71 is the lowest. So, if she drops 71, the remaining scores are 89, 92, 100, and 86. ... Let me double-check that:</p> $89 + 92 = 181, \quad 181 + 100 = 281, \quad 281 + 86 = 367.$ <p>Yes, that’s correct. So, if the sum of the five tests (including the sixth) needs to be 465, and the sum of the four highest current tests is 367, then the sixth test score needed is:</p> $465 - 367 = \mathbf{98}.$ <p>Wait, that seems high. Let me verify. $367 + 98 = 465$, which divided by 5 is:</p> $\frac{465}{5} = 93.$ <p>Yes, that makes sense. But wait, what if the sixth test score is lower than 71? In that case, she would drop the sixth score instead of 71. So, in that scenario, the lowest score would be the sixth test, and the sum would be the sum of the other five, which would be the four high scores plus 71. But that would give a lower sum, so the average would be lower. Since she wants an average of 93, she needs the sixth test score to be at least 71, preferably higher, so that she can drop the 71. Therefore, the minimum she needs on the sixth test is: 98 to reach the desired average of 93 after dropping the 71.</think> To determine the score Brinley needs on her sixth test to achieve an average of 93 after dropping the lowest, follow these steps: ... Answer: Brinley needs to score 98 on her sixth test.</p>
Our Model Output - Dynamic Reasoning
<p>Router Model: This is a simple problem that involves calculating the required score on the sixth test to achieve a desired average, considering the rule that the lowest score is dropped. Model Output: <think>To calculate the score Brinley needs on her sixth test, we need to find the required sum of the highest five test scores to achieve an average of 93. Brinley needs to have a total score of $93 \times 5 = 465$ for the five tests. The sum of her first four highest scores is $89 + 92 + 100 + 86 = 367$. Therefore, she needs a score of $465 - 367 = \mathbf{98}$ on the sixth test to reach an average of 93.</think> <answer>98</answer></p>

Table 13: An example from Our framework on MATH500 dataset with special symbols used in the thinking process. The Thought is highlighted in blue box and the position of correct tokens is highlighted in green box respectively.

Example #2 from MATH500 Dataset
<p>Question: Six cars pull up to a red light, one at a time. At the light, there are three lanes, one left-turn lane, one straight-going lane, and one right-turn lane. How many ways can the cars stack up so that all three lanes are occupied? Note that if the first car turns left and the second goes straight, this is considered different from the first car going straight and the second car turning left. In other words, the cars are distinguishable, but pull up to the intersection in a fixed order ?</p> <p>Labeled Answer: 540</p>
Original Output - Overthinking
<p><think>Okay, so I have this problem where six cars are approaching a red light, and they can choose to go left, straight, or right. ... The cars are distinguishable, meaning each car is unique, and the order in which they arrive is fixed. First, I need to understand the problem clearly. Each car has three choices: left, straight, or right. But we want to count the number of ways all three lanes are used, meaning at least one car must choose each lane. So, it's a problem of counting the number of onto functions from the set of six cars to the three lanes, considering that each car independently chooses a lane. ... Now, summing these up: $729 - 192 + 3 + 0 =$540. Wait. But let me verify if this is correct. Alternatively, another way to think about it is using permutations. Since each car can choose any of the three lanes, but we need all lanes to be used. So, it's similar to assigning each car to one of the three categories (left, straight, right) with no empty categories. ... Therefore, the correct number is 540. But let me think differently. Maybe using multiplication principle. We can model this as assigning each car to one of the three lanes, with the condition that all lanes are used. ... Another way to calculate this is to use the formula: So, yes, the result is 540. Wait, but another thought: since the cars are distinguishable, each different assignment is unique, so the count is correct. ... Let me compute $S(6, 3)$. The Stirling numbers of the second kind can be calculated using the recurrence relation or using the formula: So, this also gives 540. Therefore, this confirms the previous result.</think> Final Answer The number of ways is 540.</p>
Our Model Output - Dynamic Reasoning
<p>Router Model: This problem is complex because it involves the arrangement and combination of vehicles, as well as the inclusion exclusion principle to ensure that all three lanes are occupied. Model Output: <think>Okay, so I have this complex problem here where six cars are pulling up to a red light one by one. There are three lanes at the light: left-turn, straight-going, and right-turn. The question is asking how many ways the cars can stack up so that all three lanes are occupied. The cars are distinguishable, meaning each one is different, but they pull up in a fixed order. Hmm, let me think about this. ... Let me calculate that.</p> $3^6 = 729$ $2^6 = 64, \quad \text{multiplied by 3 gives } 3 \times 64 = 192$ $1^6 = 1, \quad \text{multiplied by 3 gives } 3 \times 1 = 3$ $729 - 192 = 537, \quad \text{plus 3 gives } 537 + 3 = $ 540 . <p>Wait, maybe I'm overcomplicating it. The cars are arriving in a fixed sequence, so for each car in order, we assign it to one of the three lanes, but ensuring that all three lanes have at least one car. So the problem reduces to counting the number of onto functions from the set of six cars to the three lanes. Yes, that's exactly what I did earlier with inclusion-exclusion. So the calculation should be correct. So the answer should be 540</think> <answer>540</answer></p>