LTMG: Less Thought, More Gain—A Token Efficient Reasoning Framework for Large Language Models

Anonymous EMNLP submission

Abstract

Large reasoning models (LRMs) demonstrate impressive logical capabilities and are able to solve complex problems through multi-step deduction. However, they often generate unnecessarily long chains of thought that significantly increase inference cost without yielding proportional gains in accuracy. We present an efficient logical reasoning framework that uses a dynamic router to instantly assess the difficulty of the question. For simple questions, the router dispatches them to a reinforcement learning–trained model that enforces strict control of the output length through learned policies during inference; for complex questions, they are dispatches to LRMs employing customized prompt strategies for different type of questions that ensures computational resources are invested only where they are truly valuable. Comprehensive experiments on arithmetic, logical, and commonsense benchmarks show that our framework reduces token usage, outperforming prior reasoning systems and demonstrating that efficiency stems from thinking just enough and only when necessary.

1 Introduction

In recent years, Large Reasoning Models (LRMs)(Xu et al., 2025), such as OpenAI's o1(Wu et al., 2024), Alibaba's Qwen3(Qwen3, 2025), and DeepSeek's R1(DeepSeek-AI et al., 2025), have achieved remarkable progress in solving complex tasks like mathematical reasoning and code generation. These models often rely on reasoning techniques such as Chain-of-Thought (CoT) prompting(Wei et al., 2022), where intermediate steps are explicitly generated to improve answer accuracy and interpretability.

Existing reasoning frameworks such as CoT, ToT(Yao et al., 2023), and GoT (Besta et al., 2024), these frameworks typically involve problem decomposition, multi-path generation, and answer aggregation. However, this step-by-step generation process often leads to substantial computational overhead and excessive token consumption. We observe that many of these frameworks are being out of date: even simple zero-shot prompts can now achieve comparable accuracy, making traditional reasoning pipelines increasingly redundant in some cases.

To validate our observation, we compare the average number of output tokens and the corresponding accuracies across multiple benchmark datasets, as shown in Figure 1. We observe a clear trend: as the number of output tokens increases, the accuracy does not improve accordingly. For instance, on AIME24, Qwen3-32B generates over 7,500 tokens per correct answer, and more than 10,000 tokens when the prediction is incorrect. This indicates that excessive token usage—rather than ensuring correctness—often reflects a behavioral pattern in which LRMs "overthink" even moderately difficult questions. In some cases, even simple problems can trigger unnecessarily long outputs, leading to significant inefficiency without accuracy gains.

To address the inefficiency and overthinking behaviors (Sui et al., 2025) , where the model generates overly elaborate reasoning chains, even for problems that may not require them that leads to excessive computational cost. Inspired by the LLM cascade(Yue et al., 2024) design, we employ a weaker LLM for simple queries and a stronger LLM for more complex questions. We propose a reasoning framework, which integrates reasoning awareness with token efficiency. It highlights the key components of our system including routing, prompt strategies, and model length control.

Experimental results on a range of benchmarks demonstrate that our framework reduces average

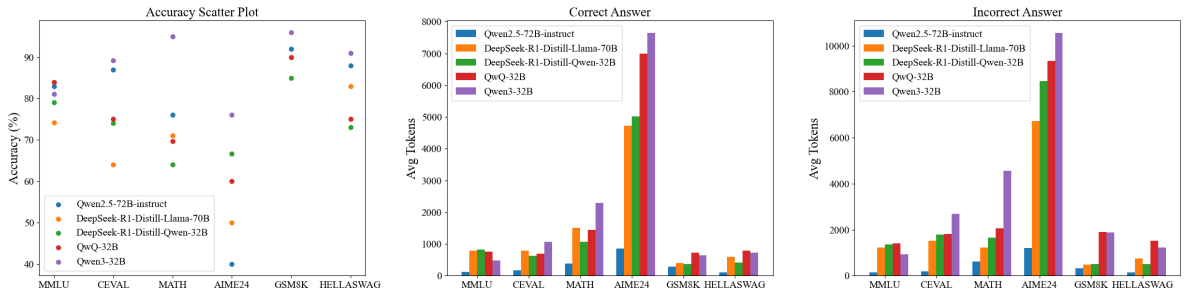

Figure 1: Comparison of token consumption and accuracy across reasoning models and datasets. Harder benchmarks (AIME24 and MATH) drive substantially higher token usage, whereas easier tasks (GSM8K and HELLASWAG) require fewer tokens. Overall, reasoning models consume more tokens for only marginal accuracy improvements, and incorrect answers incur the highest token cost.

token consumption by 10–50% compared to strong baselines, while keeping accuracy drops within acceptable bounds. As shown in Figure 2, our method achieves a competitive accuracy of 0.84 on GSM8K while using only 820 tokens significantly fewer than other methods such as COT and SPP. Our method achieves the highest reasoning efficiency among all compared approaches, as reflected by the largest circle in the plot.

To sum up, our contributions are stated as follows:

- We conduct a systematic evaluation of the logical reasoning capabilities of LRMs and find they tend to generate excessively long outputs without significant improvements in accuracy. Further analysis reveale that even simple problems often trigger unnecessary deep reasoning, resulting in wasted computational resources.

- We propose a dynamic framework that enables the system to adaptively invoke different types of models based on task difficulty, improving overall efficiency. We explore prompt strategies originally designed for LLMs and demonstrated their effectiveness when applied to LRMs.

- Our experimental results show that our framework achieves higher efficiency across diverse tasks while significantly reducing token consumption, outperforming existing reasoning frameworks.

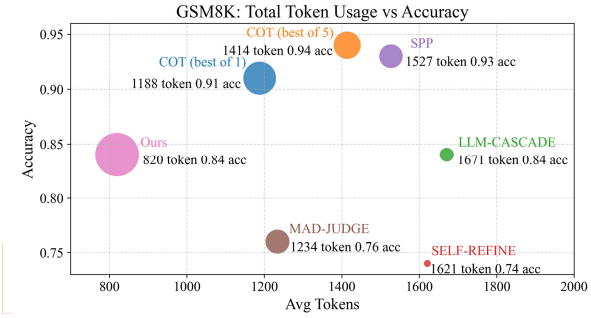

Figure 2: Comparison of our and other reasoning methods on GSM8K where the x-axis shows total token usage. Circle size reflects reasoning efficiency (accuracy per 1,000 tokens). Larger circles indicate higher efficiency with our method achieving competitive accuracy at substantially lower token cost.

2 Related work

Research can be broadly grouped into three categories. In this section, we first introduce the relevant foundational concepts, then review common reasoning frameworks, and finally examine approaches for reducing token consumption.

2.1 LRMs, Reinforcement Learning

Large Reasoning Models (LRMs)(Xu et al., 2025),such as O1(Wu et al., 2024),Qwen(Qwen3, 2025),DeepSeek(DeepSeek-AI et al., 2025) 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. By using Chain-of-Thought(Wei et al., 2022) reasoning chains, these models explicitly break down their thought process into ordered,

step-by-step sequences before delivering a final answer, which greatly enhances their performance on complex reasoning tasks.

Reinforcement learning has been proven to be effective in further improving the mathematical reasoning ability of LLMs after the Supervised Fine-Tuning(Lu et al., 2023)stage. It is used to further optimize model outputs in terms of correctness, structural coherence, and token efficiency. PPO(Schulman et al., 2017)is a widely used method that optimizes generation behavior based on a learned reward model, as applied in systems like InstructGPT(Ouyang et al., 2022). DPO(Rafailov et al., 2024)is a method that directly learns from human preference rankings between outputs, without explicitly modeling reward functions. GRPO(Shao et al., 2024)is a flexible optimization framework that supports complex reward signals, such as format consistency, content quality, and length control.

2.2 Structured Reasoning Framework

Prior works such as Chain of Thought prompting (Wei et al., 2022), Automatic Chain of Thought Prompting(Zhang et al., 2023),Tree of Thoughts(Yao et al., 2023) and Graph of Thoughts(Besta et al., 2024) frameworks have demonstrated the importance of guiding model reasoning through intermediate steps, allowing models to break down tasks into solvable steps. In addition, other reasoning frameworks include Self-Refine (Madaan et al., 2023), which iteratively produces self-feedback and refines its output for up to four rounds; LLM-Cascade (Yue et al., 2024), which dynamically routes between weaker and stronger models using a CoT-2D-Vote strategy; SPP (Wang et al., 2024b), which instantiates multiple personas of a single LLM to collaboratively solve problems; and MAD-judge (Feng et al., 2025), which orchestrates a three-agent debate with a judge module over three rounds. These methods cover a broad spectrum of Chain-of-Thought enhancements and provide a fair benchmark for evaluating our proposed framework.

2.3 Token Saving Framework

Researchers are shifting their focus toward reducing token usage. The survey Stop Overthinking(Sui et al., 2025) systematically categorizes current methods for efficient LLM reasoning, addressing the "overthinking" problem by reviewing model design, output control, prompt adaptation, and evaluation strategies.

DGoT framework(Ning et al., 2024a) aligns with this trend by dynamically optimizing the reasoning structure to achieve more cost-effective generation. SoT(Shang et al., 2024) leverages the synergy between small and large LLMs to mimic human intuitive and reflective thinking. CPO(Zhang et al., 2024) improves CoT reasoning by fine-tuning LLMs with preference signals from ToT. FrugalGPT(Chen et al., 2024) leverages LLM cascading to intelligently route queries across models. Skeleton-of-Thought(Ning et al., 2024b) introduces a parallel generation framework that reduces LLM inference latency by decoupling structure planning and content generation. TRIM(Garrachón Ruiz et al., 2024) reduces LLM inference cost by first generating a concise output with a large model, then reconstructing the full text using a smaller model. Token-Budget-Aware LLM Reasoning(Han et al., 2024) reduces unnecessary verbosity in CoT-style outputs by dynamically allocating token budgets, enabling efficient reasoning with minimal performance loss.

3 Our Method

To accomplish this, we propose our method, which consists of three part: (i) Router Dispatch, (ii) Token-Efficient Policy Optimization(TEPO) and (iii) Adaptive Prompt Optimization. This section provides detailed descriptions of them, as illustrated in Figure 3.

3.1 Problem Formulation

Let X be the universe of math problem texts and Y the set of correct solutions. A sample $(x, y) \in \mathcal{D} \subset X \times Y$ contains a problem x and its ground-truth answer y. Each problem also possesses a difficulty label $c \in \{0, 1\}$, we pre-evaluate the model on various datasets and assign a binary difficulty label:

$$c = \begin{cases} 1, & \text{if the answer on } x \text{ is correct} \\ 0, & \text{otherwise} \end{cases}$$

Here, x denotes the input math problem; s is the difficulty score output by the router $\mathcal{M}_{\text{router}}$; $\tau \in$

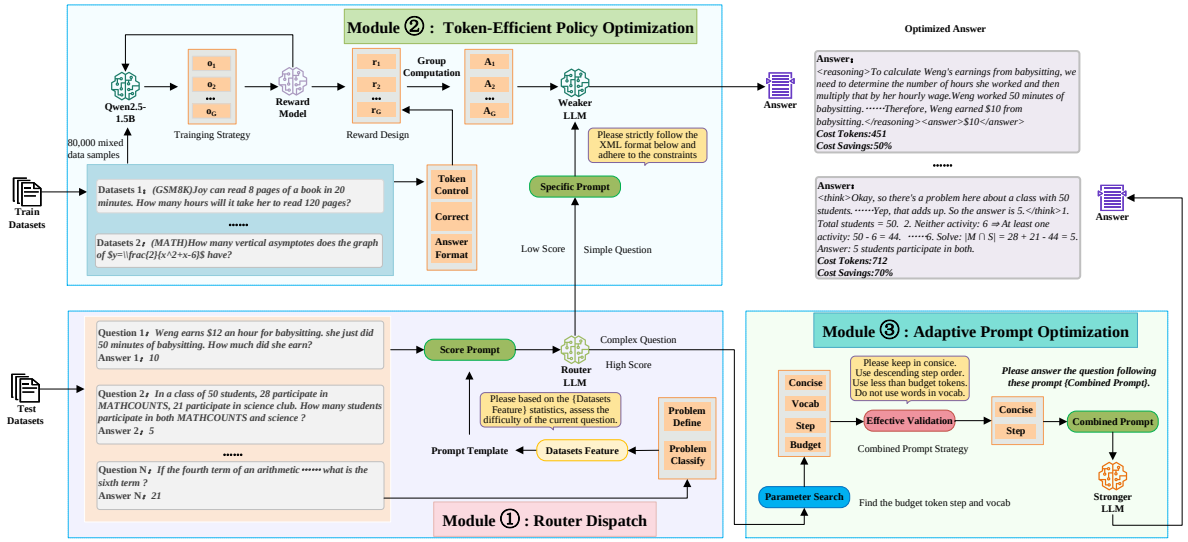

Figure 3: Our framework comprises three modules: Router dispatch for difficulty prediction, token-Efficient Policy Optimization for wearker llm, and stronger LLM controlled by adaptive prompt optimization.

$(0, 10)$ is the difficulty threshold—if $s \geq \tau$, follow the high branch, otherwise the low branch; $\mathcal{M}_{\text{weak}}$ and $\mathcal{M}_{\text{strong}}$ are the weak and strong LLMs (the former with length-control policy π_{len}, the latter with prompt strategy $\mathcal{P}$); $\hat{y}$ is the final answer; and $T_{\mathcal{R}}$, T_{weak}, T_{strong} denote the token consumption of the router, weak LLM, and strong LLM stages, respectively.

3.2 LLM Router Dispatch

We implement the router as a dedicated LLM denoted $\mathcal{M}_{\text{router}}$ with prompt $\mathcal{P}_{\text{router}}$, operating in a no-thinking mode (switching the template as in Qwen3(Qwen3, 2025) to conserve token resources) to produce an *score* $s \in [0, 10]$:

$$s = \mathcal{M}_{\text{router}}(x; \mathcal{P}_{\text{router}})$$

where larger s indicates a complex problem. Given a threshold $\tau \in (0, 10)$, we route each x via

$$\hat{y}(x) = \begin{cases} \mathcal{M}_{\text{strong}}(x), & s \geq \tau \\ \mathcal{M}_{\text{weak}}(x), & s < \tau. \end{cases}$$

We show in Algorithm 5 about how to select the optimal threshold τ^*.

To evaluate the quality of different routing strategies, we compute the **F1-score** based on a binary classification task: identifying whether a question is *simple* or *complex*.

For each input question, the routing module outputs a binary decision:

- **Positive** (complex): stronger model;
- **Negative** (simple): weaker model.

The F1-score is computed as:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1}$$

where Precision is defined as the ratio of true positives to the sum of true positives and false positives, and Recall is the ratio of true positives to the sum of true positives and false negatives. This metric reflects how effectively the routing strategy distinguishes complex reasoning tasks from simple ones.

3.3 Token-Efficient Policy Optimization

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

$$L = E\Big[\min\Big(\frac{\pi_\theta(y_t^G|x_t)}{\pi_{\theta_{\text{ref}}}(y_t^G|x_t)}\,\hat{R}_t^G,\ \text{clip}\Big(\frac{\pi_\theta(y_t^G|x_t)}{\pi_{\theta_{\text{ref}}}(y_t^G|x_t)},\ 1-\epsilon,\ 1+\epsilon\Big)\hat{R}_t^G\Big) - \lambda\, D_{\text{KL}}\big[\pi_\theta \,\|\, \pi_{\text{ref}}\big]\Big]. \tag{2}$$

where π_θ is the trainable policy, π_{ref} the fixed reference model, and $\hat{R}_t^G$ the normalised return.

To encourage concise, step-by-step reasoning while enforcing output-length constraints, we decompose the reward for a generated answer y into three interpretable components:

$$r_{\text{corr}} = \mathbf{1}\{\, y = y_{\text{gold}} \,\}$$

$$r_{\text{fmt}} = \begin{cases} 1, & \text{if format}(y) = \text{VALID}, \\ 0.5, & \text{if format}(y) = \text{PARTIAL}, \\ 0, & \text{otherwise}, \end{cases}$$

$$r_{\text{len}} = \mathbf{1}\{\, n_y \le n_{\text{budget}} \ \wedge \ n_{\text{step}} \ge 2\}.$$

where n_y is the output length in tokens and PARTIAL denotes a response that satisfies the required structure only partially (e.g., correct sections but missing delimiters). Here, n_{step} is the number of reasoning sentences in the generated response. Inspired by the Qwen3(Qwen3, 2025) design philosophy which consistently encourages step-by-step reasoning, we consider explicit multi-step thinking indispensable. To prevent the model from "lazy" behavior that outputs only the final answer without the intermediate thought process, we enforce the constraint $n_{\text{step}} \ge 2$. The overall reward $R(y)$ is the sum of the three components. It serves as the normalized return $\hat{R}_t^G$ in our TEPO surrogate objective.

3.4 Adaptive Prompt Optimization

We have analyzed the existing prompt-based strategies as illustrated in the Table 1. In our experiments, we investigate the effectiveness of different combinations of these prompt-based strategies.

Prompt method	Content
Keep Concise	Keep in concise
Step Count	Use descending step order
Token Budget	Use less than budget tokens
Vocab Banned	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.

The CCOT(Renze and Guven, 2024) method encourages the model to keep responses concise through prompt instructions.

The S1 paper(Muennighoff et al., 2025) and TALE (Han et al., 2024) proposes prompt-based strategies such as step control and token budget, which guide the model to complete reasoning within a limited number of steps or tokens. To determine an appropriate token budget for different reasoning tasks, we adopt a data-driven estimation approach based on token consumption statistics, as illustrated in Algorithm 1. We apply a step estimation and token budget procedure to determine the optimal number of reasoning steps and tokens for a given model and prompt template. As shown in Algorithm 3 and Algorithm 2.

The TRIM (Garrachón Ruiz et al., 2024) method introduces a strategy that defines a forbidden word list to guide the model away from verbose or irrelevant responses. In our approach, we collect historical responses generated by the model and feed them into a large language model to automatically generate a forbidden vocabulary list, as described in Algorithm 4.

3.5 Cost Analysis

We define $T_{\mathcal{R}}(x)$ be the number of tokens consumed by the router, $T_{\text{weak}}(x)$ the tokens consumed by the weak model $\mathcal{M}_{\text{weak}}$, and $T_{\text{strong}}(x)$ the tokens consumed by the strong model $\mathcal{M}_{\text{strong}}$. The total inference cost is then:

$$\text{Cost}(x) = T_{\mathcal{R}}(x) + \begin{cases} T_{\text{strong}}(x), & s \ge \tau \\ T_{\text{weak}}(x), & s < \tau \end{cases} \quad (3)$$

Each T includes both the input token length and the output token length, where the output consists of the reasoning trace denoted as *reasoning_content* and the final answer denoted as *content*.

$$\text{Eff}(x) = \frac{\text{Acc}(x) \times 1000}{Cost_{\text{input}}(x) + Cost_{\text{output}}(x)} \quad (4)$$

We define Eff as the amount of accuracy gained per 1,000 tokens consumed. It reflects how effectively a model utilizes its token budget to produce correct answers. A higher Eff value indicates that the model achieves better performance while consuming fewer tokens.

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, ablation analyses, and comparative evaluations.

4.1 Settings

We conduct experiments on a diverse set of datasests, including GSM8K(Cobbe et al., 2021), Game24(Yao et al., 2023), Sorting032(Besta et al., 2024), HellaSwag(Zellers et al., 2019), and

Methods	Game24			Sorting32			Gsm8k			Hellaswag			Math		
	Eff	I/T	O/T	Eff	I/T	O/T	Eff	I/T	O/T	Eff	I/T	O/T	Eff	I/T	O/T
COT (best of 1)	0.31	198	2808	0.10	655	8887	0.77	235	953	0.47	462	1234	0.21	625	2996
COT (best of 5)	0.19	694	3692	0.09	1283	8948	0.67	652	762	0.33	912	1011	0.13	1662	3550
GOT	0.09	692	9179	0.08	1904	9865	0.12	736	6653	0.08	1026	7237	0.08	1095	9331
LLM-CASCADE	0.25	212	3124	0.16	611	4612	0.50	211	1460	0.39	805	1337	0.20	1141	3010
SELF-REFINE	0.18	317	4414	0.12	350	6750	0.45	347	1274	0.32	915	1435	0.14	1037	4834
SPP	0.24	201	3144	0.18	311	4512	0.61	236	1291	0.40	445	1329	0.21	260	2768
MAD+JUDGE	0.20	307	4130	0.17	404	4923	0.62	340	894	0.36	519	1375	0.25	348	3232
Ours	**0.33**	387	**2450**	**0.21**	507	**4318**	**1.02**	403	**417**	**0.48**	641	**659**	**0.26**	518	**1984**

Table 2: Comparison of our method against baseline reasoning methods across multiple benchmarks. Our method achieves superior Efficiency (Eff) while significantly reducing both avg input tokens (I/T) and avg output tokens (O/T) compared to others. All methods use QwQ-32Bby default.

Prompt Strategy	Deepseek-R1-Distill-Llama-70B	QwQ-32b	Deepseek-R1-Distill-Qwen-32B	Qwen3-32b
Keep Concise	51.9%	35.8%	47.9%	23.0%
Step Count	61.0%	25.0%	16.6%	25.8%
Token Budget	53.2%	45.0%	33.9%	12.7%
Vocab Banned	42.2%	34.6%	42.4%	24.2%
Our Combined Strategy	**64.4%**	**47.3%**	**70.9%**	**36.8%**

Table 3: Token savings (%) of each prompting strategy across different models. Evaluation of prompt strategies on the GSM8K dataset. Each strategy was applied individually and in combination, with our combined approach achieving the best token savings.

Math500(Zellers et al., 2019).We evaluate the out-of-distribution generalization ability of our framework on the MMLUPro(Wang et al., 2024a) benchmark, which covers a wide range of domains that are not seen during training. We provide comprehensive experimental configurations and additional implementation details in the appendix B.

4.2 Main Results

As shown in Table 2, our method consistently achieves the highest reasoning efficiency (Eff) across all five benchmarks, outperforming prior approaches such as CoT, GOT, and LLM-CASCADE. Notably, on datasets like GSM8K, Game24, and Math, our method reaches Eff scores of 1.02, 0.33, and 0.26, respectively substantially higher than all baselines.

This demonstrates that our framework delivers more accurate reasoning per token consumed, achieving up to 3× greater efficiency than multi-step methods such as GOT or CoT, which require significantly more output tokens.

Unlike prior pipelines that rely on multiple sampled generations or lengthy intermediate chains, our design focuses on dynamic routing and budget-aware prompt strategies, yielding compact yet effective responses. While our method incurs slightly longer input prompts due to adaptive routing and prompt formatting, the resulting dramatic reductions in output length more than compensate for this overhead.

Overall, our framework achieves a superior trade-off between accuracy and computational cost, making it highly suitable for resource-constrained inference scenarios without compromising answer quality.

4.3 Comparative Study

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

4.3.1 Prompt Strategy Comparison

To analyze the impact of different prompt strategies on token efficiency, we conduct experiments on GSM8K using several reasoning LLMs. As shown in Table 3, each strategy—including *Keep Concise*, *Step Count*, *Token Budget*, and *Vocab Banned*—demonstrates varying degrees of token savings depending on the model architecture.

Based on the trends shown in Figure 4, we are able to identify optimal values for both token budget and step count. These observations allow us to select balanced configurations that maintain efficiency while preserving answer fidelity, and are used to guide the budget and step parameters

Reasoning Models	Game24		Sorting32		Gsm8k		Hellaswag		Math	
	Cost	Eff	Cost	Eff	Cost	Eff	Cost	Eff	Cost	Eff
Baseline(QwQ)	2837	0.33	4825	0.19	820	1.02	1300	0.48	2502	0.28
Qwen3-32B	2101	0.42	3221	0.25	1043	0.92	1202	0.52	2366	0.35
DeepSeek-R1-Distill-Qwen-32B	2390	0.36	3093	0.21	1010	0.93	**1078**	0.48	1831	0.40
DeepSeek-R1-Distill-Llama-70B	**1581**	0.56	2691	0.23	**803**	**1.13**	1096	**0.53**	**1456**	0.47
Qwen3-30B-MOE	2440	0.39	**2468**	**0.29**	956	0.99	1280	0.46	1476	**0.48**
DeepSeek-R1	1626	**0.57**	3166	0.23	953	1.00	1294	0.51	1582	0.44

Table 4: Evaluation of our prompting strategy across various RLMs and benchmarks. Shown are ACC (accuracy), Cost (average token consumption), and Efficient (accuracy per 1000 tokens). The results shows that our method consistently yields higher efficiency across different reasoning models, indicating its broad applicability.

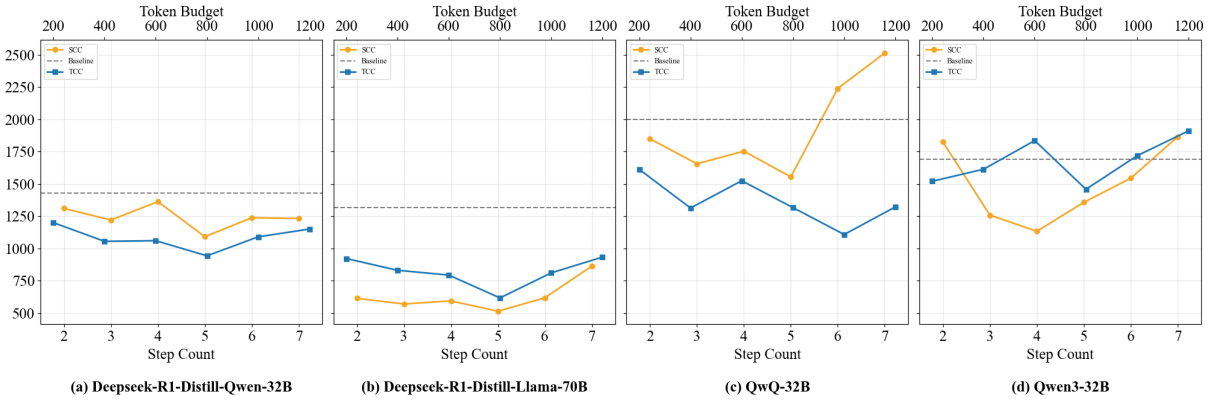

Figure 4: Conducted on the GSM8K dataset, this experiment determines the optimal token budget and step count for the following models: (a) Deepseek-R1-Distill-Qwen-32B; (b) Deepseek-R1-Distill-Llama-70B; (c) QwQ-32B; (d) Qwen3-32B. Results indicate that each model has its own specific token budget and step count.

adopted in our final combined prompt strategy.

To fully exploit the strengths of each technique, we propose the combined prompt strategy that integrates *Keep Concise* and *Step Count*.

4.3.2 Stronger LLM

As shown in Table 4, our prompting strategy exhibits strong generalization across a variety of RLMs, including models from the Qwen series (Qwen3-32B, Qwen3-30B-MOE)(Qwen3, 2025), DeepSeek-R1(DeepSeek-AI et al., 2025) variants.

We observe that applying our combined prompting strategy across different models consistently achieves the highest token savings, demonstrating its generality and effectiveness. Notably, models such as Deepseek-R1-Distill-Qwen-32B and Deepseek-R1-Distill-Llama-70B achieve 70.9% and 64.4% token savings respectively—substantially outperforming any single strategy. This indicates that our method is model-agnostic and does not rely on specific architectural features or training pipelines. Even smaller models like QwQ-32B benefit significantly 47.3% savings, reinforcing the robustness and transferability of our strategy.

4.3.3 Weaker LLM

As shown in Table 5, we evaluate the performance of several weaker language models on reasoning tasks. The Qwen2.5-1.5B model fine-tuned with our TEPO method achieves a favorable trade-off between accuracy and token efficiency compared to both the base model and the distillation-based DeepSeek-R1-Distill-Qwen-1.5B.

Further insights can be drawn from the training dynamics shown in Figure 5. As reinforcement learning progresses, the model's output length steadily decreases—dropping from over 100 to under 60 tokens—while the average reward increases consistently.

The improvement in GSM8k is particularly notable, where TEPO reduces cost to 309 tokens while maintaining competitive accuracy. Although accuracy on Sorting32 remains zero for all models, this indicates the inherent difficulty of this symbolic reasoning task for smaller-scale models.

Weaker Models	Game24		Sorting32		Gsm8k		Hellaswag		Math	
	ACC	Cost	ACC	Cost	ACC	Cost	ACC	Cost	ACC	Cost
DeepSeek-R1-Distill-Qwen-1.5B	**0.68**	2015	0	1571	**0.71**	602	**0.48**	806	**0.56**	3027
Qwen2.5-1.5B	0.62	2305	0	1068	0.62	376	0.40	452	0.40	683
Qwen2.5-1.5B-TEPO	0.65	**663**	0	**738**	0.64	**309**	0.43	**386**	0.45	**581**

Table 5: Evaluation of weaker language models on reasoning benchmarks.The Qwen2.5-1.5B model fine-tuned via TEPO achieves consistently better accuracy and lower token cost across all tasks. For the Sorting32 task, all small models exhibit zero accuracy that indicates the relative difficulty of the task for models of this scale.

Ablation Setting	Gsm8k			Math			Hellaswag			MMLUPro		
	ACC	I/T	O/T	ACC	I/T	O/T	ACC	I/T	O/T	ACC	I/T	O/T
Ours w/o tepo	**0.85**	482	593	0.60	486	2256	0.46	676	1355	0.37	141	789
Ours w/o prompt	0.82	208	824	0.62	120	2833	**0.52**	574	1440	0.33	196	1160
Ours	0.84	403	**417**	**0.70**	518	**1984**	0.50	676	**1181**	**0.38**	361	**721**

Table 6: Ablation study evaluating the impact of TEPO and prompt strategy on both in-domain tasks (Gsm8k, Math, Hellaswag) and out-of-domain task (MMLUPro). We report accuracy (ACC), input tokens (I/T), and output tokens (O/T). The full system consistently achieves favorable accuracy while significantly reducing token consumption, demonstrating strong generalization even on unseen domains.

4.3.4 LLM Router

As shown in Table 7, we compare the F1-scores of different routing strategies across three benchmarks: GSM8K, Hellaswag, and Math. Our proposed router consistently outperforms the random baseline and also slightly surpasses the Small LLM Router across all tasks. Specifically, our router outperform the Small LLM Router and Random.

These results demonstrate that our router is more effective at distinguishing between simple and complex questions, leading to more appropriate model selection and improved efficiency downstream.

Routing Strategy	Gsm8k	Hellaswag	Math
Random	0.559	0.447	0.429
Small LLM Router	0.687	0.512	0.465
Ours Router	**0.694**	**0.526**	**0.472**

Table 7: F1-score evaluation of different routing strategies. Our proposed router consistently achieves higher F1-scores that demonstrates its effectiveness in distinguishing between simple and complex questions.

4.4 Ablation Study

As shown in Table 6, we conduct an ablation study to assess the contributions of TEPO and our custom prompt strategy on both in-domain (GSM8K, Math, HellaSwag) and out-of-domain (MMLU-Pro) reasoning tasks.

Removing the TEPO component yields a modest gain in Gsm8k accuracy from 0.84 to 0.85 but inflates token usage by over 20%. Omitting the prompt strategy cuts Math accuracy from 0.70 to 0.62 and drives token consumption up sharply. In contrast, our full method maintains high accuracy with balanced token costs, illustrating the complementary benefits of TEPO and prompt design.

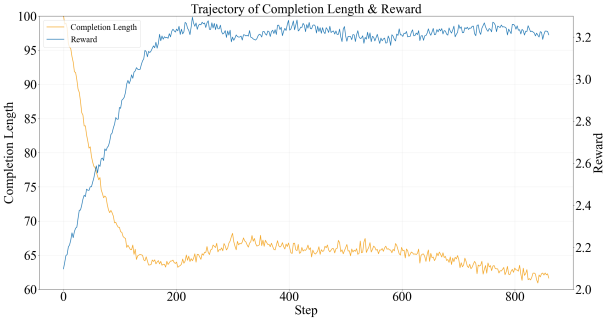

Figure 5: Reinforcement-learning dynamics: as training progresses, the model's output length (yellow axis) steadily contracts, while the reward (blue axis) consistently rises.

5 Conclusion

This paper introduces LTMG, a token-efficient reasoning framework that combines prompt design and reinforcement learning. Experiments on benchmarks show that our method achieves competitive efficiency while reducing output token usage by up to 50%. The prompt strategy generalizes well across diverse LLMs, and TEPO further improves the efficiency of smaller models. Overall, LTMG provides a scalable, cost-effective framework for enabling efficient reasoning in language models.

Limitations

We acknowledge three limitations in our study.

First, reinforcement learning is only applied to relatively small models (1.5B parameters); applying GRPO to larger models may further improve performance, but also poses significant training cost and stability challenges.

Secondly, our routing and prompting strategies are designed and evaluated under a fixed routing backbone and task types. Adapting these strategies to more diverse tasks (multi-hop QA, code generation) or dynamic router-controller architectures remains an open direction for future work.

Lastly, our experiments are conducted exclusively on English datasets. For other languages such as Chinese, the behavior of tokenization and generation may differ significantly due to linguistic and token structure differences, potentially affecting both accuracy and token efficiency.

Potential Risks

Although our framework aims to reduce token usage via output length control, excessive compression may omit critical reasoning steps. This can result in incorrect answers or shortcut behaviors, particularly in multi-step or mathematical reasoning tasks. A balance between brevity and reasoning completeness must be maintained.

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

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to use large language models while reducing cost and improving performance.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

DeepSeek-AI, Daya Guo, and Yang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv e-prints*, page arXiv:2501.12948.

Examination. 2024. Mathematical reasoning on aime24.

Zhaopeng Feng, Jiayuan Su, Jiamei Zheng, Jiahan Ren, Yan Zhang, Jian Wu, Hongwei Wang, and Zuozhu Liu. 2025. M-mad: Multidimensional multi-agent debate for advanced machine translation evaluation.

Alfredo Garrachón Ruiz, Tomás de la Rosa, and Daniel Borrajo. 2024. TRIM: Token Reduction and Inference Modeling for Cost-Effective Language Generation. *arXiv e-prints*, page arXiv:2412.07682.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-Budget-Aware LLM Reasoning. *arXiv e-prints*, page arXiv:2412.18547.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *NeurIPS Datasets and Benchmarks*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

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.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv e-prints*, page arXiv:2501.19393.

Xinyu Ning, Yutong Zhao, Yitong Liu, and Hongwen Yang. 2024a. DGoT: Dynamic graph of thoughts for scientific abstract generation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4832–4846, Torino, Italia. ELRA and ICCL.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. 2024b. Skeleton-of-thought: Prompting LLMs for efficient parallel generation. In *The Twelfth International Conference on Learning Representations*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, and Mingfeng Xue. 2025. Qwen2.5 technical report.

Qwen3. 2025. Qwen3: Think deeper, act faster.

QwQ-32B. 2025. Qwq-32b: Embracing the power of reinforcement learning.

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.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Yu Shang, Yu Li, Fengli Xu, and Yong Li. 2024. Synergy-of-Thoughts: Eliciting Efficient Reasoning in Hybrid Language Models. *arXiv e-prints*, page arXiv:2402.02563.

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.

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. *arXiv e-prints*, page arXiv:2503.16419.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024a. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2024b. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration.

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 *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

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. *arXiv e-prints*, page arXiv:2410.13639.

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. *arXiv e-prints*, page arXiv:2501.09686.

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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2024. Large language model cascades with

mixture of thoughts representations for cost-efficient reasoning.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *ACL (1)*, pages 4791–4800.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *CoRR*, abs/2406.09136.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

A Algorithm

Algorithm 1 explains how we determine the token range. Prior work(Garrachón Ruiz et al., 2024) predicted the budget with a large model, and then concatenated the resulting budget tokens into the prompt template. Our approach derives it by analyzing historical usage data.

Algorithm 1 Token Range Analysis

Global constant: *max_tokens*

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

Output: Average token usages: μ_{corr} for correct answers, μ_{wrong} for wrong answers, μ_{all} for all valid answers

Initialize: $S_{\text{corr}} \leftarrow 0,\ C_{\text{corr}} \leftarrow 0\ S_{\text{wrong}} \leftarrow 0,\ C_{\text{wrong}} \leftarrow 0$
for $(t_i, r_i) \in D$ **do**
  if $t_i =$ *max_tokens* **then**
    continue
  end if
  if $r_i = \text{true}$ **then**
    $S_{\text{corr}} \mathrel{+}= t_i,\ C_{\text{corr}} \mathrel{+}= 1$
  else
    $S_{\text{wrong}} \mathrel{+}= t_i,\ C_{\text{wrong}} \mathrel{+}= 1$
  end if
end for
$\mu_{\text{corr}} \leftarrow S_{\text{corr}} / C_{\text{corr}}$
$\mu_{\text{wrong}} \leftarrow S_{\text{wrong}} / C_{\text{wrong}}$
$\mu_{\text{all}} \leftarrow (S_{\text{corr}} + S_{\text{wrong}}) \ / \ (C_{\text{corr}} + C_{\text{wrong}})$
return $\mu_{\text{corr}},\ \mu_{\text{wrong}},\ \mu_{\text{all}}$ =0

Algorithm 2 and Algorithm 3 illustrate the process we use to select the optimal step and token budget.

Algorithm 4 describes how we determine our vocabulary. Prior approaches built it via rule-based statistical counting, our method leverages a large language model to analyze and select the vocabulary.

Algorithm 5 presents our strategy for selecting the routing threshold: by tuning this threshold, we achieve the highest F1 score.

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 $b^\star$

Initialize: *results* $\leftarrow [\,]$
for all $b = \mu_{\text{wrong}},\ \mu_{\text{wrong}} + \Delta,\ \ldots,\ \mu_{\text{corr}}$ **do**
  $C_b \leftarrow 0, \quad T_b \leftarrow 0$
  for all $c \in D$ **do**
    $P \leftarrow \text{Prompt}(c, b)$
    $(r, t) \leftarrow \mathcal{M}(P)\ \{r \in \{0, 1\},\ t \in N\}$
    $C_b \mathrel{+}= r, \quad T_b \mathrel{+}= t$
  end for
  $\alpha_b \leftarrow C_b / n, \quad \tau_b \leftarrow T_b / n$
  Append (b, α_b, τ_b) to *results*
end for
Let $\tau_{\min} \leftarrow \min\{\tau_b \mid (b, \alpha_b, \tau_b) \in \textit{results}\}$
$\mathcal{B} \leftarrow \{(b, \alpha_b) \mid (b, \alpha_b, \tau_b) \in \textit{results},\ \tau_b = \tau_{\min}\}$
$b^\star \leftarrow \arg\max_{(b,\alpha_b)\in\mathcal{B}} \alpha_b$
return $b^\star$ =0

Algorithm 3 Step Estimation

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

Output: Optimal step $k^\star$

Initialize: *results* $\leftarrow [\,]$
for all $k \in \mathcal{S}$ **do**
  $C_k \leftarrow 0, \quad T_k \leftarrow 0$
  for all $c \in D$ **do**
    $P \leftarrow \text{Prompt}(c, k)$
    $(r, t) \leftarrow \mathcal{M}(P)\ \{r \in \{0, 1\},\ t \in N\}$
    $C_k \mathrel{+}= r, \quad T_k \mathrel{+}= t$
  end for
  $\alpha_k \leftarrow C_k / n, \quad T_k \leftarrow T_k / n$
  Append (k, α_k, T_k) to *results*
end for
Let $T_{\min} \leftarrow \min\{T_k \mid (k, \alpha_k, T_k) \in \textit{results}\}$
$\mathcal{K} \leftarrow \{(k, \alpha_k) \mid (k, \alpha_k, T_k) \in \textit{results},\ T_k = T_{\min}\}$
$k^\star \leftarrow \arg\max_{(k,\alpha_k)\in\mathcal{K}} \alpha_k$
return $k^\star$ =0

B Implementation

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

Algorithm 4 Vocabulary Estimation

Input: Historical responses $D = \{c_i\}_{i=1}^{N}$; model $\mathcal{M}$; prompt template $\mathrm{Prompt}_{\mathrm{vocab}}(c)$; refinement template $\mathrm{Prompt}_{\mathrm{refine}}(V)$
Output: Final vocabulary $V^{\star}$

$V_{\mathrm{raw}} \leftarrow \emptyset$
for all $c \in D$ **do**
  $P \leftarrow \mathrm{Prompt}_{\mathrm{vocab}}(c)$
  $W \leftarrow \mathcal{M}(P)$ {W is token list}
  $V_{\mathrm{raw}} \leftarrow V_{\mathrm{raw}} \cup \mathrm{set}(W)$
end for
$P' \leftarrow \mathrm{Prompt}_{\mathrm{refine}}(V_{\mathrm{raw}})$
$V^{\star} \leftarrow \mathrm{set}\big(\mathcal{M}(P')\big)$
return $V^{\star}$ =0

Algorithm 5 Threshold Search over $\tau \in [0, 10]$

Input: Dataset $D = \{(d_i, c_i)\}_{i=1}^{N}$, $d_i \in [0, 10]$ router score, $c_i \in \{0, 1\}$ ground-truth: 0=difficult, 1=easy
Output: Best threshold τ^*

Extract all ground-truth labels: *Labels* $\leftarrow [\, c_i \,]_{i=1}^{N}$
Initialize *bestF1* $\leftarrow 0$, $\tau^* \leftarrow$ `null`
for $\tau = 0$ **to** 10 **do**
  Preds $\leftarrow []$
  for all $(d_i, c_i) \in D$ **do**
    if $d_i \geq \tau$ Append 0 to *Preds*
    else Append 1 to *Preds*
  end for
  F1 $\leftarrow \mathrm{F1}($*Labels*, *Preds*$)$
  if *F1* $>$ *bestF1* **then**
    bestF1 $\leftarrow$ *F1*, $\tau^* \leftarrow \tau$
  end if
end for
return τ^* =0

reproducibility and give readers deeper insight into the key steps of our model training and evaluation.

B.1 Dataset Statistics

The Grade School Math 8K (Cobbe et al., 2021) 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.

The 2024 AIME (American Invitational Mathematics Examination)(Examination, 2024) 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 (Hendrycks et al., 2021a) 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(Hendrycks et al., 2021a) 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.

Math500 is a subset of high-level math problems(Hendrycks et al., 2021b) 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(Zellers et al., 2019) 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(Besta et al., 2024) 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 (Yao et al., 2023) 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.

B.2 Model and Prompt Settings

We evaluate six representative models. QwQ-32B(QwQ-32B, 2025) is built on Qwen2.5-32B-Instruct(Qwen et al., 2025) and has been instruction-tuned on general QA data. Qwen3-

32B(QwQ-32B, 2025) 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(DeepSeek-AI et al., 2025) recipe—with GRPO fine-tuning applied to Qwen2.5-32B and Llama-70B, respectively—to compress model size while retaining reasoning ability. 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.

For each model, we designed a dedicated prompt template. RouterLLM:You are an expert question classifier. Please evaluate the following question and assign a difficulty score from 1 to 10. WeakerLLM: Please strictly follow the XML format below and adhere to the following constraints: <reasoning> </reasoning> <answer> </answer> StrongerLLM:Our combined Prompt. banndevocab-prompt ["Alright", "Okay", "Let me", "Alternatively", "maybe", "so", "Wait", "but", "Hmm", "however", "therefore", "next", "first", "should", "also", "yet", "that", "moreover", "meanwhile", "although", "despite", "hence", "yeah", "in addiction", "for example", "such as", "etc", "because"] token-budget-prompt "Answer the question within budget tokens" prompt-vocab "When answering the following question, please avoid using any of these words or phrases: vocab" step-prompt "Present your solution in descending step order, starting from Step step down to Step 1. Label each step exactly as "Step X: ..."." concise-prompt = "Answer the question in concise"

B.3 Experimental Setup

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, we evaluated out-of-domain generalization by measuring performance on the MMLU-Pro test set.

For the routing component, we use a RLM as a router. In head-to-head tests, this stronger 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 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(QwQ-32B, 2025) as the default model for routing decisions. We modify the model's prompt template to avoid long thinking and set the sampling temperature to 0.1 to ensure deterministic and stable routing decisions.

For the weaker LLM, we adopt the Qwen2.5-1.5B-Instruct(Qwen et al., 2025) model from the Tongyi Qianwen series 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 `128` tokens to encourage concise output. All reinforcement learning experiments are conducted on a cluster of 8 NVIDIA RTX 4090D GPUs, with a total training time of approximately 2 hours.

For the reasoning phase, we use VLLM's(Kwon et al., 2023) official default settings—temperature 0.7 and no maximum token cap—so that the model can generate full, uninterrupted answers without being cut off by external constraints, thereby preserving accuracy.