Adaptive Model and Strategy Routing for Cost-Efficient LLM Services

Zhihong Pan
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
panzh@mail.ustc.edu.cn

Kai Zhang*
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
kkzhang08@ustc.edu.cn

Yuze Zhao
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
yuzezhao@mail.ustc.edu.cn

Yupeng Han
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, Anhui Province, China
yupenghan@mail.ustc.edu.cn

Abstract

In web-based AI services, providers typically host multiple large language models (LLMs) that exhibit diverse capabilities and incur different API costs. Meanwhile, LLM's performance depends not only on its inherent capacity but also on the reasoning strategy it employs, which together influence both answer quality and computational cost. A key challenge is therefore how to adaptively allocate models and strategies to achieve high-quality responses under constrained costs. To address this challenge, we propose **Route-To-Reason (RTR)**, a unified routing framework that simultaneously selects suitable LLMs and reasoning strategies according to query complexity and user budget. Specifically, RTR learns dense vector representations of models and strategies that capture their behavioral characteristics in handling different queries. Leveraging these embeddings, RTR builds a routing table that estimates the cost and performance of different model–strategy pairs. During inference, RTR consults this routing table to dynamically assign the most appropriate pair, enabling adaptive and cost-efficient reasoning tailored to query difficulty and budget scenarios. Extensive experiments across multiple reasoning benchmarks show that RTR achieves comparable or higher accuracy than the best single LLM while substantially reducing both token usage and API cost (by up to 60%), achieving a superior trade-off between performance and efficiency. By lowering the overhead of large-scale LLM inference, RTR contributes to cost-aware and environmentally sustainable deployment of web-based AI services.

CCS Concepts

• **Computing methodologies → Natural language generation**; **Neural networks**.

*Corresponding author.

Keywords

Large Language Models; Adaptive Routing Framework; Cost-Efficient Inference; LLM Reasoning

ACM Reference Format:
Zhihong Pan, Kai Zhang, Yuze Zhao, and Yupeng Han. 2026. Adaptive Model and Strategy Routing for Cost-Efficient LLM Services. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3774904.3792556

1 Introduction

Large Language Models (LLMs) have been widely deployed as web-based applications for tasks such as code generation, logical reasoning, and knowledge-intensive question answering, where they demonstrate human-like and even superhuman capabilities [1, 29, 48]. Within this ecosystem, reasoning ability has emerged as the core driver of intelligent agent behavior [49]. Consequently, an increasing number of reasoning-oriented models [14, 43] and reasoning strategies [13, 23, 42] have been developed. These models and strategies evolve synergistically, collectively advancing LLM reasoning capabilities.

This raises a critical question: *How can we efficiently identify the most suitable model-strategy pairing from this combinatorial space?* In practical web-based deployments, API providers typically host models with different capabilities and costs. Selecting appropriate combinations improves user experience, reduces costs, and enhances overall service quality.

Intuitively, one might prefer combining powerful reasoning models (e.g., o3 [31]) with sophisticated strategies (e.g., Chain-of-Thought [40]), and under the test-time scaling paradigm, allocating a high compute budget appears natural.

However, this intuition-driven, fixed approach may face two key challenges in practice: First, prior work [4, 6, 10, 24, 25, 27, 44, 46] shows that "overthinking" can trap the model in prolonged local reasoning patterns, limiting adaptability and ultimately degrading performance. Second, applying high-capacity models and complex strategies indiscriminately to low-difficulty tasks yields little benefit while incurring excessive computational and financial costs [4, 44]. These observations motivate our central claim that **less is more**:

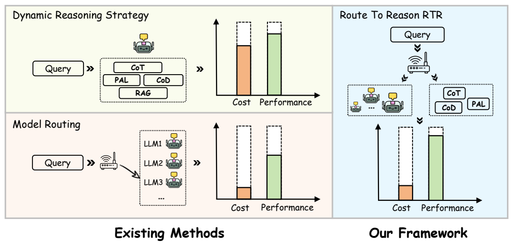


Figure 1: Comparison of existing methods and our framework. Existing approaches focus on either model routing or strategy routing in isolation, which often leads to suboptimal cost-performance balance. Our framework jointly selects both, achieving better trade-offs.

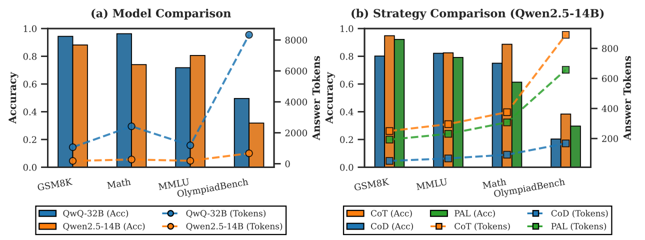


Figure 2: (a) Deep reasoning mainly benefits complex tasks but introduces unnecessary cost on simple ones; (b) Different reasoning strategies exhibit complementary strengths and diverse accuracy-token trade-offs.

adaptive use of lightweight model–strategy pairs can often achieve a better balance between accuracy and cost.

Some prior explorations have focused on *model routing* [5, 7, 12, 18, 21, 26, 28, 30, 39, 54], where the system selects the most suitable model from a pool based on input characteristics. Recent industrial systems, such as the GPT-5 series [32], also adopt this paradigm by dynamically assigning queries to either lightweight or high-capacity reasoning models during inference to achieve cost-effective adaptation.

However, existing model-routing approaches largely overlook the intricate interplay between model capability, reasoning strategy, and input complexity, often leading to suboptimal routing decisions. Complementary to this line of work, several studies [2, 34, 47] have investigated *reasoning-strategy routing*, which dynamically adjusts the reasoning process according to input characteristics. While these methods improve adaptability and support test-time scaling [10, 38], they typically operate under a fixed model configuration, leaving the joint optimization of model and strategy selection for queries of varying difficulty largely unexplored.

To address these limitations, we propose a unified framework for joint model and strategy routing, enabling efficient and accurate test-time inference through dynamic selection. Specifically, we represent each expert model and reasoning strategy using learnable vectors that capture their respective behavioral characteristics in terms of performance and computational cost. Given an input instance, it is first encoded using a pretrained language model. Two predictors are then designed to estimate the expected performance and output length of all model–strategy pairs, thereby constructing a routing table. Based on this table, the policy module selects the optimal pair that maximizes efficiency while maintaining or improving accuracy.

Compared to previous approaches, our framework dynamically adapts to queries of varying difficulty and selects the most appropriate model-strategy pair for each input, leading to better cost-performance trade-offs. As illustrated in Figure 1, by jointly selecting both the model and reasoning strategy, our framework achieves superior performance at reduced computational cost.

We conduct extensive experiments on eight diverse reasoning benchmarks (spanning mathematics, scientific reasoning, code generation, and general knowledge) under both in-distribution and out-of-distribution settings. Results demonstrate that RTR consistently improves reasoning accuracy while reducing the overall number of generated tokens by up to 60% compared with the best single model, thus validating its efficiency and effectiveness.

Our contributions are as follows[1]:

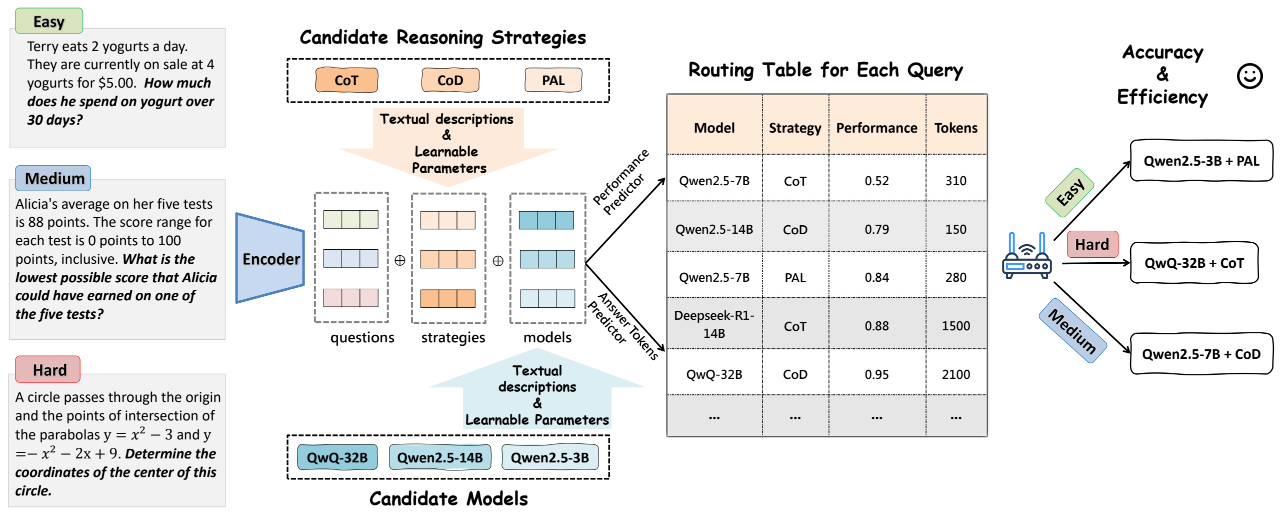
- We identify limitations of existing routing and dynamic reasoning strategy approaches, highlighting the unaddressed interaction between models, strategies, and input difficulty.
- We introduce Route-To-Reason (RTR), a unified framework that jointly learns representations of models and strategies, and constructs a routing table for dynamic selection.
- We conduct extensive experiments on eight benchmarks, showing that RTR reduces both token usage and API costs by 60% while maintaining or improving reasoning accuracy, thereby achieving more cost-efficient web-based LLM deployment.

2 Motivations and Challenges

Motivation 1: When Deep Reasoning Helps and When It Hurts. Reasoning-enhanced LLMs have achieved remarkable success on complex tasks through extended intermediate thinking [14, 43]. However, as illustrated in Figure 2(a), deep reasoning benefits diminish quickly on simpler problems. While the reasoning model (*QwQ-32B*) substantially outperforms the non-reasoning model (*Qwen2.5-14B-Instruct*) on challenging datasets such as Math [17] and OlympiadBench [15], its advantage becomes marginal on easier benchmarks like GSM8K [9] and MMLU [16]. Yet these modest gains incur a steep cost: the reasoning model generates an order of magnitude more tokens, leading to significant overhead. This pattern suggests that rather than deploying deep reasoning universally, intelligent systems should learn *when complex reasoning is necessary.*

Motivation 2: Reasoning Strategy Matters as Much as the Model. Performance disparities also stem from the choice of reasoning strategy. As shown in Figure 2(b), different prompting strategies exhibit complementary strengths: **CoT** [40] elicits step-by-step reasoning but often produces lengthy responses on simple problems; **CoD** [42] encourages concise reasoning with comparable accuracy for simpler queries; and **PAL** [13] excels at arithmetic or code-oriented tasks via symbolic computation. These observations

[1]https://github.com/goodmanpzh/Route-To-Reason



Easy: Terry eats 2 yogurts a day. They are currently on sale at 4 yogurts for $5.00. ***How much does he spend on yogurt over 30 days?***

Medium: Alicia's average on her five tests is 88 points. The score range for each test is 0 points to 100 points, inclusive. ***What is the lowest possible score that Alicia could have earned on one of the five tests?***

Hard: A circle passes through the origin and the points of intersection of the parabolas $y = x^2 - 3$ and $y = -x^2 - 2x + 9$. ***Determine the coordinates of the center of this circle.***

Model	Strategy	Performance	Tokens
Qwen2.5-7B	CoT	0.52	310
Qwen2.5-14B	CoD	0.79	150
Qwen2.5-7B	PAL	0.84	280
Deepseek-R1-14B	CoT	0.88	1500
QwQ-32B	CoD	0.95	2100
...	...	...	...

Figure 3: RTR encodes the input query, available models, and reasoning strategies. Two predictors estimate performance and token usage for each model-strategy pair, generating a routing table. The router then applies a routing policy to select the optimal pair according to user-specified preferences.

suggest that adaptively selecting both the model and reasoning strategy can enable efficient inference while maintaining high accuracy.

Challenges. Adaptive joint routing of models and reasoning strategies entails three major challenges: (1) **Modeling diversity:** capturing the behavioral characteristics of heterogeneous model–strategy combinations across different query types; (2) **Generalization:** maintaining robust routing performance on out-of-distribution (OOD) queries and unseen domains with varying task difficulty; (3) **Efficiency balance:** jointly considering reasoning performance, token length or API cost to achieve an optimal trade-off between accuracy and resource consumption.

3 Methodology

3.1 Problem Formulation

Consider a collection of language models $\mathcal{M} = \{m_j : j = 1, \dots, M\}$, each differing in size or capability, and a set of reasoning strategies $\mathcal{S} = \{s_k : k = 1, \dots, K\}$. Given a set of input queries $\mathcal{D} = \{q_i : i = 1, \dots, N\}$, applying model m_j with strategy s_k to query q_i yields a response characterized by two quantities: the performance score $a_{i,j,k}$ (e.g., accuracy, utility, or another task-specific metric), and the number of generated tokens $l_{i,j,k}$ serving as a proxy for inference cost. Our objective is to predict these two quantities and select an appropriate model-strategy pair (m_j, s_k) based on these quantities for each query q_i. Formally, we seek to learn a routing function:

$$\pi : \mathcal{D} \to \mathcal{M} \times \mathcal{S},$$

where $\pi(x_i) = (j^*, k^*)$ denotes the selected model and strategy for x_i. The objective is to optimize the trade-off between total performance and total generation cost:

$$\max_{\pi} \sum_{i=1}^{N} a_{i,j^*,k^*} - \lambda \sum_{i=1}^{N} c_{j^*} \cdot l_{i,j^*,k^*},$$

where c_j denotes the per-token cost of model m_j, $\lambda > 0$ is a hyperparameter that balances performance and efficiency, a_{i,j^*,k^*} and l_{i,j^*,k^*} denote the performance and generated length, respectively.

3.2 RTR Framework Overview

As illustrated in Figure 3, our Route-To-Reason (RTR) framework processes each incoming query q_i in three steps. First, q_i is encoded into a dense vector representation. Second, each candidate model m_j and reasoning strategy s_k is mapped into embeddings reflecting their properties. Finally, the concatenated joint representation,

$$z_{i,j,k} = h\big(f_{\text{enc}}(q_i) \oplus g(m_j) \oplus g(s_k)\big), \tag{1}$$

is forwarded to two predictor heads that estimate performance $\hat{a}_{i,j,k}$ and cost $\hat{l}_{i,j,k}$.

This procedure produces a routing table $\mathcal{T}_i$ that records, for query q_i, the predicted accuracy and token usage corresponding to every (m_j, s_k) combination. The downstream policy then converts this table into a discrete routing decision by scoring each entry. The entire framework introduces negligible overhead, with query encoding as the main cost, making RTR suitable for latency-sensitive applications (See Appendix A.4).

3.3 Model and Strategy Representation

Accurate prediction hinges on how models and strategies are represented. We employ a dual-component representation to capture both high-level semantics and fine-grained adaptivity:

$$g(m_j) = e_j^{\text{text}} \oplus e_j^{\text{learned}}, \tag{2}$$

$$g(s_k) = e_k^{\text{text}} \oplus e_k^{\text{learned}}. \tag{3}$$

Here, e^{text} is a frozen embedding from a pretrained encoder applied to textual descriptions of the model or strategy, automatically curated via an auxiliary LLM (See Appendix A.6). This enables integration of unseen models or strategies given only a textual description.

The second component, e^{learned}, is randomly initialized and updated during training, enabling adaptation to fine-grained patterns such as dataset-specific biases. Together, these components balance transferability (via textual priors) and adaptability (via task-specific learning). When adding a new model or strategy, only minimal data is needed to fine-tune its learnable embedding.

3.4 Dual Prediction Module

Given query–model–strategy representation $z_{i,j,k}$, RTR learns two predictors:

Performance prediction.

$$\hat{a}_{i,j,k} = \sigma\big(\text{MLP}_{\text{perf}}(z_{i,j,k})\big), \tag{4}$$

which estimates the probability of m_j with s_k answering q_i correctly. The corresponding loss is binary cross-entropy:

$$\mathcal{L}_{\text{perf}} = -y_{i,j,k} \log \hat{a}_{i,j,k} - (1 - y_{i,j,k}) \log(1 - \hat{a}_{i,j,k}), \tag{5}$$

where $y_{i,j,k} \in \{0, 1\}$ denotes correctness.

Token usage prediction.

$$\hat{l}_{i,j,k} = \text{MLP}_{\text{len}}(z_{i,j,k}), \tag{6}$$

estimates the expected number of tokens consumed. Since length is continuous, we adopt mean squared error (MSE):

$$\mathcal{L}_{\text{len}} = \big(\hat{l}_{i,j,k} - l_{i,j,k}\big)^2, \tag{7}$$

with $l_{i,j,k}$ the observed output length.

The two predictors are trained jointly, sharing the representation $z_{i,j,k}$. Only the MLP parameters and learnable embeddings are updated, while the query encoder remains frozen.

3.5 Routing Table and Routing Policy

After predictions, each query q_i yields a routing table:

$$\mathcal{T}_i = \left\{ \big(\hat{a}_{i,j,k}, \hat{l}_{i,j,k}\big) \;\middle|\; m_j \in \mathcal{M},\, s_k \in \mathcal{S} \right\}, \tag{8}$$

where $\mathcal{M}$ and $\mathcal{S}$ denote candidate models and strategies.

Scoring function. To select an execution route, we design a trade-off score:

$$\text{score}_{i,j,k} = \lambda \cdot \hat{a}_{i,j,k} - (1 - \lambda) \cdot \frac{c_j \cdot \hat{l}_{i,j,k}}{C}, \tag{9}$$

where $\lambda \in [0, 1]$ controls preference towards accuracy or efficiency, c_j denotes the per-token cost associated with model m_j (e.g., API pricing), and C is a normalization factor (e.g., global mean cost) to avoid unit imbalance. When per-token costs are unavailable or uniform across models, c_j can be set to 1, reducing to a token-count-based objective. The routing decision is obtained via:

$$(j^*, k^*) = \arg\max_{j,k}\ \text{score}_{i,j,k}. \tag{10}$$

Rationale. This formulation is inspired by multi-objective optimization [12, 30], where c_j enables realistic cost modeling across different API pricing. The parameter λ offers flexibility: $\lambda \approx 1$ prefers accuracy; $\lambda \approx 0$ favors efficiency; intermediate values balance both (See 4.4 and 4.5).

Algorithm 1 Training and Inference of RTR

```
1:  Input: Training set $\mathcal{D}_{\text{train}}$, encoder $E$, descriptions $\{d_j, d_k\}$,
    trade-off parameter $\lambda$
2:  Output: Routing table for selecting model–strategy pairs
3:  Initialize learnable embeddings $\{e_j\}$, $\{e_k\}$ and MLPs: $\text{MLP}_{\text{perf}}$,
    $\text{MLP}_{\text{len}}$
4:  Encode descriptions: $z_j = [E(d_j); e_j]$, $z_k = [E(d_k); e_k]$
5:  for each $(x_i, \{a_{i,j,k}, l_{i,j,k}\})$ in $\mathcal{D}_{\text{train}}$ do
6:      $q_i = E(x_i)$
7:      for each $(j, k)$ do
8:          $z_{i,j,k} = [q_i; z_j; z_k]$
9:          Update $\text{MLP}_{\text{perf}}$, $\text{MLP}_{\text{len}}$ using BCE and MSE losses
10:     end for
11: end for
12:
13: Inference phase:
14: for each test input $x_n$ do
15:     $q_n = E(x_n)$
16:     for each $(j, k)$ do
17:         $z_{n,j,k} = [q_n; z_j; z_k]$
18:         Predict $\hat{a}_{n,j,k}$, $\hat{l}_{n,j,k}$ via MLPs
19:         $\text{score}_{n,j,k} = \lambda \hat{a}_{n,j,k} - (1 - \lambda) \cdot c_j \cdot \hat{l}_{n,j,k}/C$
20:     end for
21:     Select $(j^*, k^*) = \arg\max_{j,k} \text{score}_{n,j,k}$
22: end for
```

4 Experiments

4.1 Experimental Setup

4.1.1 Candidate LLMs and Strategies. We select six open-source LLMs from two categories, (1) *Non-reasoning models* [43]: Qwen2.5-3B, Qwen2.5-7B, and Qwen2.5-14B, which are general-purpose models with performance scaling with model size. (2) *Reasoning models* [14, 43]: DeepSeek-R1-7B, DeepSeek-R1-14B, and QwQ-32B, which are optimized for complex reasoning tasks and produce longer reasoning chains.

We evaluate four reasoning strategies: **Vanilla**, which uses the original question without any additional prompting and serves as a baseline; **CoT** [40] prompts the model to generate intermediate reasoning steps before answering; **PAL** [13] prompts the model to solve questions by generating executable code; **CoD** [42] prompts the model to generate only intermediate drafts with explicit constraints on output length, encouraging concise reasoning. Details of all prompts are provided in Appendix A.5.

4.1.2 Datasets. We select subsets from four reasoning benchmarks: **(1) GSM8K** [9], a mathematical reasoning dataset with diverse grade school word problems; **(2) MMLU** [16], a general-purpose benchmark where we evaluate on a selected subset of STEM subjects; **(3) Math** [17], a large-scale dataset focusing on diverse math problems requiring numerical reasoning and problem solving; and **(4) OlympiadBench** [15], a challenging benchmark derived from olympiad-level scientific problems. The statistics of the datasets used are summarized in Table 1.

We construct our dataset by collecting responses generated by all candidate LLMs paired with each reasoning strategy. For each

Table 1: Overview of datasets used in experiments. Difficulty levels are assigned based on prior work and average baseline model performance.

	Dataset	Type	Cases	Difficulty
ID	GSM8K [9]	Math Reasoning	600	Easy
	Math [17]	Math Reasoning	600	Medium
	MMLU [16]	General Knowledge	600	Medium
	OlympiadBench [15]	Science & Math Reasoning	300	Hard
OOD	SciQ [41]	Science QA	300	Easy
	PIQA [3]	Commonsense Reasoning	300	Medium
	ARC-C [8]	Science Reasoning	300	Hard
	LiveCodeBench [20]	Code Generation	240	Hard

query, we record the correctness label and output token length for every model-strategy combination. The dataset is split into 70% for training and 30% for testing, with stratified sampling to preserve difficulty distribution.

To assess generalization, we further evaluate the trained router on four out-of-distribution (OOD) benchmarks: **(1) SciQ** [41], a multiple-choice science question dataset for middle school level; **(2) PIQA** [3], which tests physical commonsense reasoning through everyday scenarios; **(3) ARC-C** [8], a challenging subset of the ARC benchmark focusing on questions requiring multi-step reasoning and commonsense inference; and **(4) LiveCodeBench** [20], a comprehensive benchmark designed to evaluate reasoning and problem-solving abilities in code generation tasks. These datasets were excluded from training and used only for evaluation.

4.1.3 Baselines. We compare RTR against two single-model baselines and several routing methods covering both similarity-based and learning-based paradigms. Detailed settings are provided in Appendix A.1:

- **Qwen2.5-3B**: The smallest single LLM in our pool of candidate LLMs.
- **QwQ-32B**: The best single LLM in our pool of candidate LLMs.
- **Random**: Randomly selects a model-strategy pair from the pool of candidates.
- **KNN-Router** [18]: For each candidate model-strategy pair, it estimates the expected performance by averaging the observed rewards over these neighbors, and selects the pair with the highest estimated score.
- **RouteLLM** [30]: A binary router that assigns queries to either a strong or weak model based on learned routing policies over query features.
- **EmbedLLM** [54]: Constructs an embedding for each model using matrix factorization to capture its performance profile across the dataset, and uses these embeddings to route queries to the most suitable LLM.

4.1.4 Evaluation metrics. Routing methods are evaluated based on two primary criteria that capture both effectiveness and efficiency.

(1) **Accuracy.** Measures the correctness of the selected model's response for each task, indicating how well the router chooses the most suitable model for a given query.
(2) **Efficiency.** Reports the average number of generated tokens (or equivalently, the API cost in extended analysis), which reflects the computational and monetary efficiency of the routing decisions.

For tables reporting results across multiple datasets, the "Overall" column shows metrics computed over all test samples across datasets.

4.1.5 Implementation Details. For non-reasoning models, outputs are generated via greedy decoding (temperature = 0). For reasoning-based models, we follow official recommendations, using temperature = 0.6 and top-p = 0.95 for sampling. For the router, textual descriptions of each model, reasoning strategy, and input query are encoded with the `all-mpnet-base-v2`[2] model following Zhuang et al. [54], a lightweight (~110M parameters) yet effective encoder that yields 768-dimensional embeddings. Both predictors are implemented as MLPs with a hidden size of 768. The router is trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 32.

4.2 Main Results

In-Distribution Results. Table 2 presents the performance comparison of various routing methods across four reasoning tasks. All routing models outperform random routing, demonstrating the effectiveness of routing strategies. Our proposed method RTR, achieves the best overall accuracy (82.5%) while significantly reducing the overall output length (1091.3 tokens). It outperforms all baselines in terms of the trade-off between performance and cost. Specifically, RTR achieves the highest accuracy on MMLU and OlympiadBench, and the second-best results on GSM8K and Math. Compared to the best-performing baseline, EmbedLLM, RTR matches or surpasses its accuracy while reducing the overall token usage by over 39.6%. Notably, compared to the single largest model, QwQ-32B, which achieves strong performance at a very high cost, RTR improves overall accuracy by 2.5 percentage points while reducing output token length by about 60%. These results demonstrate that RTR effectively balances accuracy and efficiency through joint model-strategy selection.

Out-Of-Distribution Results. As shown in Table 3, the proposed RTR achieves the highest overall accuracy and lowest token cost among all routing-based models across out-of-distribution datasets. Compared with the best-performing individual model (QwQ-32B), RTR attains comparable accuracy (81.7% vs. 82.3%) while reducing token usage by more than 50%, demonstrating a much better efficiency–performance trade-off. This advantage mainly stems from its two-part architecture, which jointly captures high-level textual semantics and fine-grained dataset-specific characteristics of both the model and the reasoning strategy, thereby enabling the router to achieve stronger generalization to unseen domains. These results suggest that RTR learns transferable routing patterns rather than overfitting to training distributions.

4.3 Evaluation of API Cost Efficiency

To complement our token-level efficiency analysis, we evaluate inference costs under real-world settings using per-token API pricing

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

Table 2: Results on in-distribution datasets. The best router-based result is in bold, and the second-best is underlined.

Model	GSM8K		Math		MMLU		OlympiadBench		Overall	
	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens
Qwen2.5-3B	71.5	205.4	54.3	295.5	61.7	253.1	21.0	1007.2	56.0	371.7
QwQ-32B	94.4	1148.5	95.6	2583.0	71.8	1219.3	48.5	8762.3	80.0	2745.2
Random	84.4	382.4	77.7	1043.7	69.4	723.1	32.6	4219.1	69.5	1271.6
KNN-Router	89.2	**272.9**	88.3	1122.3	78.1	**347.5**	36.6	4197.7	76.9	1101.3
RouteLLM	91.0	372.0	89.3	1161.8	72.8	597.3	**45.5**	7696.8	77.3	1814.3
EmbedLLM	**95.8**	927.1	**94.8**	1898.2	80.5	508.8	41.5	5786.4	81.9	1808.3
RTR (Ours)	95.2	297.8	92.9	**982.9**	**82.5**	432.5	**45.5**	**3399.7**	**82.5**	**1091.3**

Table 3: Results on out-of-distribution datasets. The best router-based result is in bold and the second-best is underlined.

Model	PIQA		SciQ		ARC-C		LiveCodeBench		Overall	
	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens
Qwen2.5-3B	73.3	150.8	68.2	187.6	65.3	248.9	9.48	480.4	56.4	255.7
QwQ-32B	95.5	1126.4	93.8	1203.6	91.7	1831.9	39.5	5328.7	82.3	2217.1
Random	84.6	350.0	78.3	430.8	74.1	713.9	25.5	3688.1	67.7	1169.8
KNN-Router	90.3	277.4	89.2	**376.2**	86.3	**550.6**	31.6	3845.8	76.6	1126.5
RouteLLM	93.2	363.1	92.7	894.3	91.2	1103.2	33.1	4132.4	79.9	1491.2
EmbedLLM	95.1	832.8	92.2	1002.3	92.4	1631.1	**35.3**	4922.9	81.0	1948.6
RTR (Ours)	**95.3**	**222.3**	**94.2**	405.7	**93.1**	553.7	34.6	**3554.7**	**81.7**	**1059.3**

Table 4: Accuracy and Cost (USD per query, $\times 10^{-3}$) on In-Distribution Datasets. The best router-based result is in bold, and the second-best is underlined.

Model	GSM8K		Math		MMLU		Olymp.		Overall	
	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost
Qwen2.5-3B	71.5	0.021	54.3	0.030	61.7	0.025	21.0	0.101	56.0	0.037
QwQ-32B	94.4	1.378	95.6	3.100	71.8	1.463	48.5	10.515	80.0	3.294
Random	84.4	0.411	78.4	1.208	69.8	0.817	32.6	5.023	69.6	1.479
KNN-Router	89.3	0.204	90.3	**1.111**	77.6	**0.403**	40.5	6.431	77.8	1.490
RouteLLM	91.5	**0.134**	89.1	1.114	72.3	0.511	45.3	8.973	78.9	1.915
EmbedLLM	**95.8**	2.191	**93.8**	2.313	80.5	0.618	41.5	7.028	81.3	2.191
RTR (Ours)	95.2	0.331	92.2	1.454	**82.0**	0.514	**45.5**	**2.746**	**82.2**	**1.055**

from *LiteLLM*[3] (See Appendix A.3). We revise the routing score as:

$$\begin{aligned}\text{Score} = (1-\lambda)\cdot \text{PredictedPerformance} \\ - \lambda \cdot \text{PredictedTokens} \cdot \text{APICostPerToken},\end{aligned} \tag{11}$$

where λ controls the trade-off between accuracy and cost. This formulation explicitly incorporates model-specific API pricing.

As summarized in Table 4, RTR achieves the highest overall accuracy (82.2%) while incurring the lowest API cost across all router-based baselines. Compared with the strongest router-based baseline (EmbedLLM), RTR improves accuracy by 0.9% and reduces the overall API cost by approximately 52%. This substantial gain stems from our joint routing design over both models and reasoning strategies, enabling adaptive selection of reasoning depth according to task difficulty.

4.4 Ablation Studies and Analysis

Effectiveness of Multi-Strategy. We evaluate the impact of multi-strategy selection by comparing the original accuracy and the routing accuracy of our proposed RTR to those of individual models. As shown in Figure 4, we compare the performance using RTR within a single model (labeled as 'Model + RTR', marked by $\times$) to the model's original overall accuracy (labeled as 'Model Overall', marked by $\circ$). Routing across multiple strategies consistently improves performance across all models, highlighting the value of strategy selection tailored to individual query characteristics.

Effectiveness of Multi-Model. As shown in Figure 4, we plot the λ-curve (λ controls the trade-off between token usage and performance) to evaluate the effectiveness of multi-model selection. While RTR's λ-curve shows only a slight performance drop compared to using only QwQ-32B for routing, it significantly outperforms most individual models. This demonstrates that leveraging multiple models with strategy-aware routing effectively balances accuracy and efficiency.

Effectiveness of Performance Prediction. We compare three representation configurations: (1) textual embeddings only, (2) learnable embeddings only, and (3) the combination of both. As shown in Figure 5, the combined representation yields the highest accuracy

[3]https://github.com/BerriAI/litellm

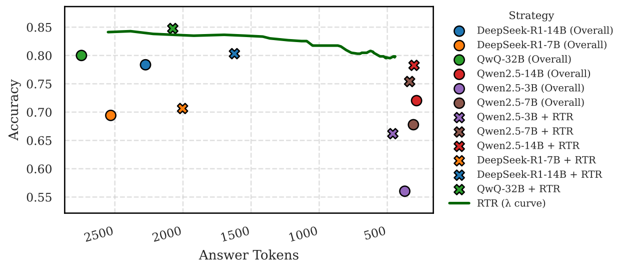


Figure 4: Performance comparison illustrating the effect of multi-model and multi-strategy routing.

in predicting model-strategy correctness, demonstrating the complementary strengths of prior knowledge from textual descriptions and task-adaptive learned embeddings.

Effectiveness of Token Usage Prediction. We evaluate the performance of our token length prediction. As shown in Figure 6, although reasoning and non-reasoning models differ substantially in average output length, the predictor achieves approximately 80% accuracy for non-reasoning models within a 200-token error margin, and about 55% accuracy for reasoning models with a 600-token margin. Given that reasoning models can generate over 10,000 tokens, this accuracy is sufficient for effective routing, as the primary goal is to distinguish between high-cost and low-cost model-strategy pairs rather than predicting exact token counts.

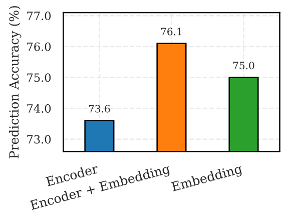


Figure 5: Ablation study on representation components for performance prediction accuracy.

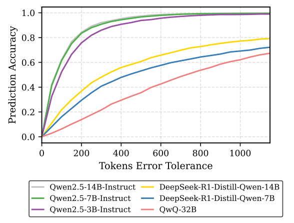


Figure 6: Token length prediction accuracy under different error tolerances for reasoning and non-reasoning models.

4.5 Different Routing Configurations

We further evaluate our model under various routing scenarios. As shown in Figure 7, relying on a single fixed strategy often leads to higher costs and suboptimal performance. For example, strategies with greater reasoning depth may achieve better accuracy but incur significantly greater cost. This underscores the advantage of dynamically selecting reasoning strategies.

We then examine three routing configurations by adjusting λ to reflect different trade-off preferences: *performance-first* ($\lambda = 2 \times 10^{-5}$), *balanced* ($\lambda = 2.5 \times 10^{-5}$), and *cost-first* ($\lambda = 3 \times 10^{-5}$), corresponding to RTR-performance, RTR-balanced, and RTR-cost in Figure 7, respectively. A larger λ assigns higher penalty to token usage, favoring efficiency over accuracy. Experimental results show

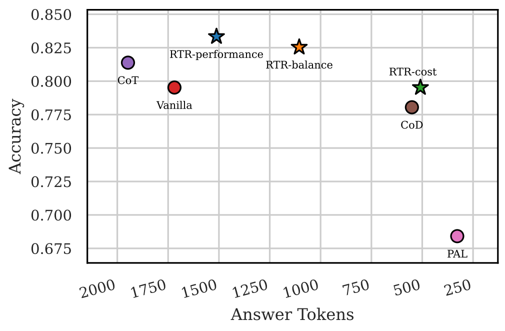


Figure 7: Distribution of different routing configurations showing the performance of RTR-performance, RTR-cost, and RTR-balanced configurations compared to single-strategy baselines.

that our framework is robust under different budget constraints and achieves competitive performance while adapting to varying priorities in cost-effectiveness and accuracy. Moreover, all three configurations outperform using only the vanilla strategy, highlighting the benefits of leveraging multiple reasoning strategies.

Table 5: Performance of dynamic routing between reasoning and non-reasoning modes for Qwen3-4B.

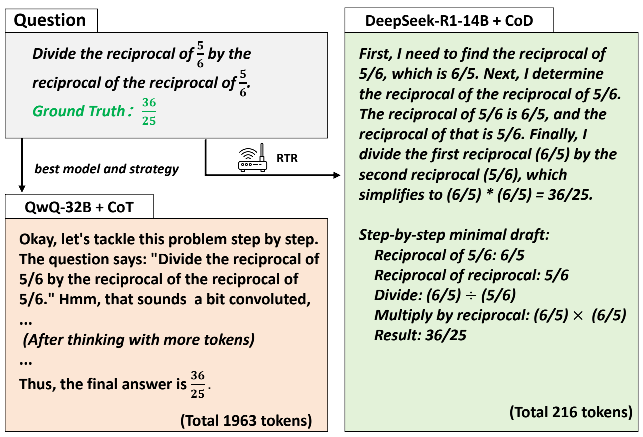
Method	Acc (%)	Tokens
Qwen3-4B (non-reasoning)	73.4	592.1
Qwen3-4B (reasoning)	82.4	3112.8
Random	76.6	1834.2
RouteLLM	82.8	2247.1
KNN-Router	80.6	1418.7
EmbedLLM	83.2	2623.5
RTR (Ours)	**83.8**	**1321.1**

4.6 When-To-Think Routing

Recent LLMs support switching between *reasoning* and *non-reasoning* modes. Our framework naturally handles such dual-mode models by representing each mode as a distinct candidate (See Appendix A.6), enabling the router to automatically determine when to invoke reasoning mode based on task requirements. To validate this capability, we conduct experiments with Qwen3-4B on a subset of the same in-distribution datasets described in Section 4.1. As shown in Table 5, our framework effectively learns to trigger the reasoning mode only when beneficial. Compared to always using the reasoning mode, RTR achieves higher accuracy (83.8% vs. 82.4%) while reducing token usage by 57%, demonstrating the effectiveness of both adaptive mode selection and dynamic strategy selection.

4.7 Case Study

Figure 8 illustrates the effectiveness of RTR through a case example. When RTR is disabled, the best-performing model (QwQ-32B) along with the reasoning strategy of CoT results in redundant and unnecessarily verbose reasoning, ultimately leading to the use of 1963 tokens. In contrast, when RTR is applied, it routes the query to DeepSeek-R1-14B and the CoD strategy. This configuration yields the correct answer while using only 216 tokens, demonstrating significantly reduced computation.



Divide the reciprocal of $\frac{5}{6}$ by the reciprocal of the reciprocal of $\frac{5}{6}$.
Ground Truth : $\frac{36}{25}$

Okay, let's tackle this problem step by step. The question says: "Divide the reciprocal of 5/6 by the reciprocal of the reciprocal of 5/6." Hmm, that sounds a bit convoluted,
...
(After thinking with more tokens)
...
Thus, the final answer is $\frac{36}{25}$.
(Total 1963 tokens)

*First, I need to find the reciprocal of 5/6, which is 6/5. Next, I determine the reciprocal of the reciprocal of 5/6. The reciprocal of 5/6 is 6/5, and the reciprocal of that is 5/6. Finally, I divide the first reciprocal (6/5) by the second reciprocal (5/6), which simplifies to (6/5) * (6/5) = 36/25.*

Step-by-step minimal draft:
Reciprocal of 5/6: 6/5
Reciprocal of reciprocal: 5/6
Divide: (6/5) ÷ (5/6)
Multiply by reciprocal: (6/5) × (6/5)
Result: 36/25
(Total 216 tokens)

Figure 8: Without using RTR, selecting the best model (QwQ-32B) and the CoT strategy leads to redundant reasoning steps. In contrast, RTR routes the query to DeepSeek-R1-14B with the CoD strategy and obtains the correct answer using only 216 tokens.

5 Related Work

5.1 LLM Routing

Model routing has emerged as one of the most cost-effective approaches in LLM ensembles [7, 11, 12, 18], as it selects a single model to generate responses for individual queries. RouteLLM [30] proposes four distinct strategies for routing between small and large models, effectively reducing cost. EmbedLLM [54] proposes learning universal model embeddings to facilitate routing decisions across diverse models. RouterBench [18] introduces a benchmark dataset for routing tasks and implements a range of routing baselines to balance response quality and computational cost. Division-of-Thoughts (DoT) [35] introduces a collaborative reasoning framework that coordinates on-device smaller-scale models with cloud-based LLMs through dynamic task decomposition and scheduling, substantially reducing inference cost while preserving reasoning accuracy. Recent works [21, 28, 39] have further explored collaborative multi-model deployments and the development of general-purpose routing systems.

5.2 Efficient LLM Reasoning

Recently, LLMs [14, 19, 43] have demonstrated increasingly powerful reasoning capabilities, but this progress has come at the expense of significantly higher inference-time computation. Recent studies have shown that such improvements often result in the problem of overthinking [4, 6, 10, 24, 25, 27, 37, 44], where models tend to generate unnecessarily long and redundant reasoning steps, especially for simple queries. This has brought growing attention to the challenge of dynamically adapting inference based on problem difficulty. Existing approaches typically address this by fine-tuning [22, 27, 33] the model or applying reinforcement learning [6, 37, 44] to shorten reasoning trajectories and reduce unnecessary token generation, thereby enabling adaptive computation during inference.

5.3 Dynamic Reasoning Strategies in LLM

Chain-of-Thought (CoT) prompting [40] and its variants, such as decomposition-based methods [23, 53] and Tree-of-Thought (ToT) [45], have significantly improved performance across a wide range of reasoning tasks. Recent studies [34, 36, 47] have shown that these strategies exhibit complementary strengths depending on the characteristics of the task [50–52]. For instance, while CoT performs well on mathematical and logical reasoning, it may not be universally effective for all question types. The Program-aided Language (PAL) approach [13] has shown superior performance on arithmetic-intensive tasks by incorporating code execution into the reasoning process. Recent work [2, 34, 47] has studied how to select the most suitable reasoning strategy for different types of questions. However, these methods assume a fixed model configuration and do not explore how different models interact with various strategies. In contrast, our work jointly optimizes both model and strategy selection, enabling more efficient and accurate routing decisions.

6 Discussion

Conclusion. In this paper, we present RTR, a unified framework for jointly selecting the optimal model and reasoning strategy for each query. To the best of our knowledge, RTR is the first approach to simultaneously address both model and strategy selection in a single routing framework. By representing all candidate models and strategies as learnable embeddings, our framework predicts two key metrics for each query: the expected performance score and the answer token usage across all available model-strategy pairs. These predictions are used to construct a routing table, enabling RTR to dynamically determine the best combination of model and reasoning strategy for each query. Extensive experiments demonstrate that RTR consistently makes effective routing decisions, achieving a favorable trade-off between overall performance and computational cost across all baselines.

Limitations and Future Directions. Our current evaluation focuses on query-level routing among individual models, while incorporating collaborative decisions across multiple models and extending evaluation beyond reasoning tasks are promising directions. We believe RTR can facilitate cost-aware and environmentally sustainable deployment of large-scale web-based AI services, advancing intelligent model orchestration within the broader Web ecosystem.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (Grants No.62406303), Anhui Province Science and Technology Innovation Project (202423k09020010).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179* (2025).

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 7432–7439.

[4] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567* (2025).

[5] Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems* 37 (2024), 66305–66328.

[6] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187* (2024).

[7] Zhijun Chen, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Dingqi Yang, Hailong Sun, and Philip S Yu. 2025. Harnessing Multiple Large Language Models: A Survey on LLM Ensemble. *arXiv preprint arXiv:2502.18036* (2025).

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv:1803.05457v1* (2018).

[9] 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. *arXiv preprint arXiv:2110.14168* (2021).

[10] Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2025. Learning How Hard to Think: Input-Adaptive Allocation of LM Computation. In *The Thirteenth International Conference on Learning Representations*.

[11] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *The Twelfth International Conference on Learning Representations*.

[12] Tao Feng, Yanzhen Shen, and Jiaxuan You. 2025. GraphRouter: A Graph-based Router for LLM Selections. In *The Thirteenth International Conference on Learning Representations*.

[13] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*. PMLR, 10764–10799.

[14] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[15] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008* (2024).

[16] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

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

[18] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. RouterBench: A Benchmark for Multi-LLM Routing System. In *Agentic Markets Workshop at ICML 2024*.

[19] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

[20] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and Contamination-Free Evaluation of Large Language Models for Code. *arXiv preprint arXiv:2403.07974* (2024).

[21] Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Zifeng Wang, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. 2025. Universal Model Routing for Efficient LLM Inference. *arXiv preprint arXiv:2502.08773* (2025).

[22] Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 24312–24320.

[23] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. In *The Eleventh International Conference on Learning Representations*.

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

[25] Yue Liu, Jiaying Wu, Yufei He, Hongcheng Gao, Hongyu Chen, Baolong Bi, Jiaheng Zhang, Zhiqi Huang, and Bryan Hooi. 2025. Efficient Inference for Large Reasoning Models: A Survey. *arXiv preprint arXiv:2503.23077* (2025).

[26] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. Routing to the Expert: Efficient Reward-guided Ensemble of Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 1964–1974.

[27] Rohin Manvi, Anikait Singh, and Stefano Ermon. 2024. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725* (2024).

[28] Kai Mei, Wujiang Xu, Shuhang Lin, and Yongfeng Zhang. 2025. Eccos: Efficient capability and cost coordinated scheduling for multi-llm serving. *Available at SSRN 5159339* (2025).

[29] Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao yong Wei, Shanru Lin, Hui Liu, Philip S. Yu, and Qing Li. 2025. A Survey of WebAgents: Towards Next-Generation AI Agents for Web Automation with Large Foundation Models. arXiv:2503.23350 [cs.AI] https://arxiv.org/abs/2503.23350

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

[31] OpenAI. 2024. *Introducing GPT-4o and GPT-4o-mini.* https://openai.com/index/introducing-o3-and-o4-mini/ Accessed: 2025-05-16.

[32] OpenAI. 2025. GPT-5 System Card. https://openai.com/index/gpt-5-system-card.

[33] Jiabao Pan, Yan Zhang, Chen Zhang, Zuozhu Liu, Hongwei Wang, and Haizhou Li. 2024. DynaThink: Fast or slow? A dynamic decision-making framework for large language models. *arXiv preprint arXiv:2407.01009* (2024).

[34] Tanmay Parekh, Pradyot Prakash, Alexander Radovic, Akshay Shekher, and Denis Savenkov. 2024. Dynamic Strategy Planning for Efficient Question Answering with Large Language Models. *arXiv preprint arXiv:2410.23511* (2024).

[35] Chenyang Shao, Xinyuan Hu, Yutang Lin, and Fengli Xu. 2025. Division-of-Thoughts: Harnessing Hybrid Language Model Synergy for Efficient On-Device Agents. arXiv:2502.04392 [cs.CL] https://arxiv.org/abs/2502.04392

[36] Zayne Rea 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. In *The Thirteenth International Conference on Learning Representations*.

[37] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, et al. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419* (2025).

[38] Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2024. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. *arXiv preprint arXiv:2408.13457* (2024).

[39] Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025. Mixllm: Dynamic routing in mixed large language models. *arXiv preprint arXiv:2502.18482* (2025).

[40] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[41] Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209* (2017).

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

[43] An Yang, Baosong Yang, Beichen Zhang, bnyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[44] Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2025. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080* (2025).

[45] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems* 36 (2024).

[46] Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, et al. 2025. Don't Overthink It: A Survey of Efficient R1-style Large Reasoning Models. *arXiv preprint arXiv:2508.02120* (2025).
[47] Murong Yue, Wenlin Yao, Haitao Mi, Dian Yu, Ziyu Yao, and Dong Yu. 2024. DOTS: Learning to Reason Dynamically in LLMs via Optimal Reasoning Trajectories Search. *arXiv preprint arXiv:2410.03864* (2024).
[48] Wei Zeng, Hengshu Zhu, Chuan Qin, Han Wu, Yihang Cheng, Sirui Zhang, Xiaowei Jin, Yinuo Shen, Zhenxing Wang, Feimin Zhong, and Hui Xiong. 2025. Multi-level Value Alignment in Agentic AI Systems: Survey and Perspectives. arXiv:2506.09656 [cs.AI] https://arxiv.org/abs/2506.09656
[49] Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. CodeAgent: Enhancing Code Generation with Tool-Integrated Agent Systems for Real-World Repo-level Coding Challenges. arXiv:2401.07339 [cs.SE] https://arxiv.org/abs/2401.07339
[50] Kai Zhang, Qi Liu, Hao Qian, Biao Xiang, Qing Cui, Jun Zhou, and Enhong Chen. 2021. Eatn: An efficient adaptive transfer network for aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 377–389.
[51] Kai Zhang, Hao Qian, Qing Cui, Qi Liu, Longfei Li, Jun Zhou, Jianhui Ma, and Enhong Chen. 2021. Multi-interactive attention network for fine-grained feature learning in ctr prediction. In *Proceedings of the 14th ACM international conference on web search and data mining*. 984–992.
[52] Kai Zhang, Hefu Zhang, Qi Liu, Hongke Zhao, Hengshu Zhu, and Enhong Chen. 2019. Interactive attention transfer network for cross-domain sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5773–5780.
[53] Yuze Zhao, Tianyun Ji, Wenjun Feng, Zhenya Huang, Qi Liu, Zhiding Liu, Yixiao Ma, Kai Zhang, and Enhong Chen. 2025. Unveiling the magic of code reasoning through hypothesis decomposition and amendment. *arXiv preprint arXiv:2502.13170* (2025).
[54] Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. 2025. EmbedLLM: Learning Compact Representations of Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

A Additional Details and Experiments

A.1 Baseline Implementation Details

We describe the implementation details of the baseline methods as follows:

- **Random:** A model–strategy pair is randomly selected for each input. We report the average performance over 50 independent runs.
- **KNN-Router:** The number of neighbors k is set to 10. For each model–strategy pair, we compute its score as the average accuracy minus the average output length multiplied by a parameter λ. The parameter λ is tuned to balance performance and cost, and we report the best-performing configuration.
- **RouteLLM:** As a binary router, RouteLLM labels each query according to its average accuracy across candidate models, and trains a binary classifier to determine which model to select. The balance parameter is tuned, and the best result is reported. The strong and weak models used are QwQ and Qwen2.5-7B, respectively.
- **EmbedLLM:** Each model–strategy pair is assigned a unique ID, and a 768-dimensional embedding is learned for each pair. Following the original implementation, the learning process is formulated as a reconstruction task: given a matrix of model correctness across prompts, a reconstruction network is trained to recover this matrix, ensuring that the embeddings capture the behavioral characteristics of each model–strategy pair.

A.2 Effectiveness of Training Samples

We evaluate the prediction performance of the accuracy predictor under varying training sample sizes. As shown in Figure 9, an increase in the number of training samples leads to a general improvement in prediction accuracy across all datasets. This enhancement is particularly pronounced when the training sample size is relatively small. As the training size approaches around 5000 samples, the performance begins to converge. This suggests that our prediction framework is capable of effectively modeling the relationship between model-strategy pairs and problems using a relatively small amount of data, thereby enabling accurate performance prediction.

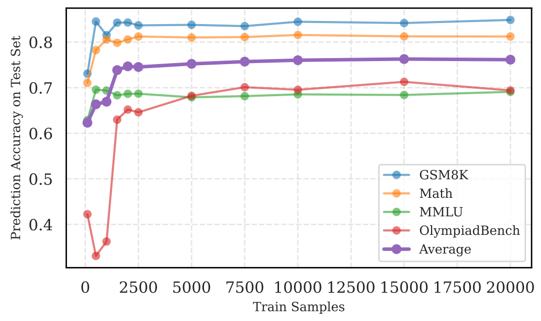


Figure 9: Prediction accuracy on the test set under different training sample sizes.

A.3 API Cost

Table 6: Candidate LLMs and their pricing (USD per 1M tokens).

LLM	Cost per 1M tokens ($)
DeepSeek-R1-Distill-Qwen-14B	1.6
DeepSeek-R1-Distill-Qwen-7B	1.2
QwQ-32B	1.2
Qwen2.5-14B	0.8
Qwen2.5-7B	0.3
Qwen2.5-3B	0.1

A.4 Inference Cost Analysis

To further analyze the real-world efficiency of our routing framework, we evaluate the computational overhead introduced by **RTR** beyond the LLM inference itself. The routing process consists of three lightweight components:

(1) **Embedder:** Each query is encoded once using a pre-trained sentence transformer (`all-mpnet-base-v2`, ~110M parameters). On a standard NVIDIA A100 GPU, this takes only a few milliseconds per query, which is negligible compared to LLM inference times that typically last several seconds.
(2) **Predictor:** For each of the M candidate models and K reasoning strategies, two small MLPs estimate the predicted performance and token usage. These $M \times K$ forward passes are highly parallelizable and collectively take less than one millisecond.

(3) **Router:** The final selection step involves a few arithmetic operations and comparisons, contributing only nanosecond-level overhead.

Overall, the routing overhead is negligible relative to LLM inference, with the vast majority of computation and cost dominated by model execution. This demonstrates that RTR enables efficient model selection with minimal additional computational burden.

A.5 Prompts Used in Experiments

CoT
Please reason step by step before providing the final answer, and put your final answer within `\\boxed{{}}`.

CoD
Think step by step, but only keep minimum draft for each thinking step, with 5 words at most, and put your final answer within `\\boxed{{}}`.

PAL
Write a Python code snippet to solve the following problem. Do not use any plotting libraries or the input() function.

A.6 Profiles of Strategies and Models

Vanilla: Vanilla prompting retains the original question content without adding any additional prompt information.

Chain-of-Thought (CoT): Chain-of-Thought (CoT) prompting guides the model to articulate a step-by-step reasoning process before providing the final answer. This results in longer responses and slower reasoning speed, typically generating the longest answers, but it performs best on complex problems such as mathematical reasoning.

Chain-of-Draft (CoD): Chain-of-Draft (CoD) prompts the model to generate only intermediate drafts with explicit constraints on output length, encouraging concise reasoning. These drafts represent the model's thinking process, often containing important calculation steps and key reasoning information. It simplifies the intermediate steps of the reasoning chain while retaining good performance, resulting in shorter answers.

Program-Aided Language (PAL): Program-Aided Language (PAL) transforms the reasoning process into executable code. This approach leverages the determinism of programming languages to ensure logical consistency and high accuracy, making it particularly effective for mathematical, symbolic, or algorithmic tasks. PAL relies on a suitable code execution environment and consistently produces results with high reliability and stable, moderately sized outputs. However, it may not well suited for commonsense reasoning tasks.

Qwen2.5-3B-Instruct: Qwen2.5-3B-Instruct is a lightweight 3B parameter model with fast inference and low resource usage. It is suitable for simple tasks such as basic question answering and short-form text generation, but is limited in handling complex reasoning or multi-step tasks.

Qwen2.5-7B-Instruct: Qwen2.5-7B-Instruct is a mid-small 7B parameter model that balances speed and performance. It is capable of multi-turn dialogue, basic code and math tasks, and offers improved language understanding over smaller models, while maintaining efficient inference.

Qwen2.5-14B-Instruct: Qwen2.5-14B-Instruct is a mid-sized 14B parameter model that excels at complex reasoning, document summarization, and structured mid-length text generation. It demonstrates strong performance on tasks requiring deeper understanding and context retention.

DeepSeek-R1-7B: DeepSeek-R1-7B is a 7B distilled model with slightly slower but stable inference compared to other models of similar size. It is well-suited for medium-length answers that require deep reasoning, and often generates more detailed and comprehensive responses.

DeepSeek-R1-14B: DeepSeek-R1-14B is a 14B distilled model with slower inference but strong logical and mathematical capabilities. It is ideal for mathematical proofs and tasks requiring rigorous step-by-step reasoning, offering robust performance in logic-intensive scenarios.

QwQ-32B: QwQ-32B is a large 32B quantized model with slow inference. It excels at complex logic, coding, and multi-step reasoning tasks, though it may produce verbose outputs. Its large capacity enables handling of challenging prompts and long-context tasks.

Qwen3-4B (thinking): Qwen3-4B in 'thinking' mode generates longer reasoning chains and detailed thought processes. While it has slower inference and higher resource usage, it excels at solving complex logic and reasoning problems, making it suitable for tasks that require step-by-step explanations or in-depth analysis.

Qwen3-4B (non-thinking): Qwen3-4B in 'non-thinking' mode is optimized for short, direct answers. It provides fast inference with low resource cost, but is limited in deep reasoning or step-by-step explanations, making it best for straightforward queries or when efficiency is prioritized.