

TokenRouter: Uncertainty-Guided Token-Level Routing for Efficient LLM Reasoning

Zhenning Shi^{1,2*}, Yijia Zhu^{1,2*}, Junhan Shi², Xun Zhang³, Congcong Miao⁴,
Guorui Xie¹, Yong Jiang^{1,2}, Qing Li^{1†}

¹Peng Cheng Laboratory, Shenzhen, China

²Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China

³Fuzhou University, Fuzhou, China ⁴National University of Singapore, Singapore

{shizn23, shijh23}@tsinghua.org.cn, zyg26@mails.tsinghua.edu.cn,

102304229@fzu.edu.cn, mccmiao@163.com, xiegr@pcl.ac.cn,

jiangy@sz.tsinghua.edu.cn, liq@pcl.ac.cn

Abstract

Large Language Models (LLMs) are strong at multi-step reasoning, but running them for every generated token is expensive. Small language models (SLMs) are much cheaper, yet less reliable at the hardest reasoning steps. Existing routing methods mostly make a single query-level decision, assuming that response difficulty is uniform once decoding begins. For chain-of-thought reasoning, however, difficulty varies during generation, making token-level routing a more appropriate granularity. This setting introduces three challenges: token-level uncertainty is noisy, local routing decisions must satisfy global efficiency constraints, and frequent switching can disrupt reasoning coherence. We present TokenRouter, a black-box runtime controller for uncertainty-guided token-level routing between small and large language models. TokenRouter combines a robust entropy detector, a self-calibrating control policy, and a dynamic intervention commitment mechanism for stable switching over difficult spans. Across three reasoning benchmarks, TokenRouter maintains over 99% of pure-LLM accuracy while reducing large-model activation to 56.2%–76.0% of generated tokens. Code is available at <https://github.com/sznzns/TokenRouter>.

1 Introduction

Large Language Models (LLMs) have become strong reasoners on multi-step tasks, but this capability is expensive to deploy because every generated token requires a full forward pass through a large model (Kojima et al., 2022; Wei et al., 2022b; Yao et al., 2023; Liu et al., 2025). This cost is especially pronounced for chain-of-thought reasoning, where solving a single problem may require long autoregressive traces. Small language models

(SLMs) offer a much cheaper alternative, but relying on them throughout decoding can hurt accuracy at the most difficult reasoning steps. This creates a practical tension in efficient LLM reasoning: always using the large model wastes computation, while always using the small model risks failure at the most critical moments.

Model routing is a natural response to this trade-off. Prior work typically routes at the query level: a router examines the input once and then assigns the entire request to a single model (Hu et al., 2024; Ong et al., 2025; Tran et al., 2025). This design is effective when difficulty is largely determined by the input as a whole, but reasoning traces are rarely uniform. Within a single response, easy procedural spans and difficult reasoning steps often alternate (Yu et al., 2025; Kim et al., 2025). As a result, query-level routing uses the wrong granularity for chain-of-thought reasoning. It cannot selectively allocate stronger model capacity to the parts of the generation that actually need it.

Our starting point is therefore to treat efficient reasoning as a runtime collaboration problem between small and large language models. The key question is not only which model should answer a query, but also when the system should escalate from the SLM to the LLM during generation, and for how long. This perspective motivates token-level routing. It also distinguishes our setting from one-shot routing and from training-heavy token routing schemes: we seek a black-box test-time controller that can operate on unmodified models using signals that are available at decoding time.

Making token-level routing work in practice is challenging. First, token-level uncertainty signals are noisy, so naive switching based on raw scores can trigger spurious interventions (Raposo et al., 2024; Wu et al., 2025). Second, local routing decisions must still satisfy a global efficiency target rather than overusing the LLM whenever uncer-

*Equal contribution.

†Corresponding author.

tainty spikes. Third, per-token switching can oscillate and degrade the overall quality of reasoning (Pan et al., 2026; Zheng et al., 2025). A useful routing mechanism must therefore be uncertainty-guided, budget-aware, and stability-aware at the same time.

To address these challenges, we introduce TokenRouter, a closed-loop framework for uncertainty-guided token-level routing. TokenRouter uses the SLM’s output entropy as a deployable uncertainty signal for black-box runtime control. A robust entropy detector first smooths the raw entropy stream into a stable estimate of local reasoning difficulty, making the routing signal less sensitive to token-level noise. A self-calibrating control policy then converts a target usage budget into an adaptive switching threshold, allowing the system to regulate LLM usage online rather than relying on a fixed trigger. Finally, a dynamic intervention commitment mechanism keeps the LLM active long enough to cover difficult spans, reducing oscillation and preserving reasoning coherence. Together, these components form a unified runtime policy for stable small-large collaboration during decoding.

We evaluate TokenRouter on three reasoning benchmarks spanning different difficulty levels (Cobbe et al., 2021; Hendrycks et al., 2021; Li et al., 2024). Across these benchmarks, TokenRouter maintains over 99% of the reasoning accuracy of a powerful LLM (e.g., DeepSeek-R1-Distill-Qwen-14B) while activating it for only 56.2% to 76.0% of the generated tokens, with the remaining tokens handled by a much smaller model (e.g., DeepSeek-R1-Distill-Qwen-1.5B). These results show that uncertainty-guided small-large collaboration can achieve a strong accuracy-efficiency trade-off for reasoning without modifying the two underlying models.

Our contributions are as follows:

- We formulate efficient reasoning as an uncertainty-guided token-level routing problem between small and large language models, targeting the granularity mismatch of query-level routing for chain-of-thought generation.
- We propose TokenRouter, a black-box closed-loop runtime controller that integrates robust uncertainty estimation, self-calibrated efficiency control, and stability-aware intervention during decoding.
- We present a deployment-oriented implemen-

tation that keeps routing lightweight through CPU-side decisions, KV-cache reuse, and minimal additional memory cost, together with end-to-end runtime analysis.

- We show strong accuracy-efficiency trade-offs on primary reasoning benchmarks and further evaluate cross-domain and cross-model transfer robustness.

2 Related Work

2.1 Efficient LLM Inference

To reduce the deployment cost of LLMs, prior work has explored model compression and dynamic computation (Shi et al., 2025b). Most of these methods are white-box, requiring access to model internals. Static compression techniques (Wang et al., 2024; Fang et al., 2024), including quantization (Xiao et al., 2023), pruning (Sun et al., 2024), and knowledge distillation (Gu et al., 2024), permanently shrink models but typically require retraining or fine-tuning. Dynamic methods such as early exiting (Bae et al., 2023; Chen et al., 2024) adapt computation at runtime, yet remain tightly coupled to specific architectures and do not naturally support collaboration across independent models. In contrast, our method is a black-box framework that coordinates unmodified models at inference time and is therefore complementary to these approaches.

2.2 Collaborative Computing Architectures

Combining multiple computational units is a common strategy for balancing performance and efficiency, including deployment settings that coordinate on-device small models with cloud-based large models or other heterogeneous inference resources (Chen et al., 2026; Niu et al., 2025). The Mixture-of-Experts (MoE) architecture (Dai et al., 2024; Yun et al., 2024) scales model capacity through sparse expert activation within a single system, while speculative decoding (Xia et al., 2024) pairs a small draft model with a large verifier to reduce latency. Beyond LLM inference, continual systems also adapt matching rules as network conditions evolve (Shi et al., 2025a). TokenRouter instead schedules independent models based on real-time reasoning difficulty, targeting selective capacity allocation over a generation trace while preserving quality.

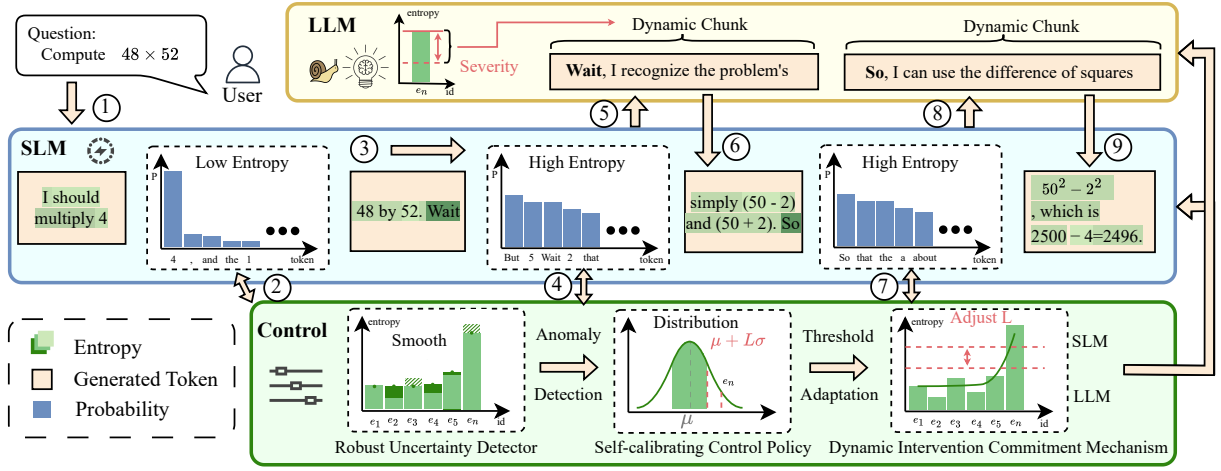


Figure 1: An overview of the TokenRouter framework, where numbered steps illustrate the adaptive token-level routing process for a sample reasoning task. Its control module continuously monitors the SLM’s output entropy, triggering an intervention from the more capable LLM only at moments of high local reasoning difficulty to ensure both efficiency and accuracy.

2.3 Model Routing

Model routing directs an input request to the most suitable model within a model suite (Jitkrittum et al., 2026; Varangot-Reille et al., 2026). Most existing methods are query-level (Hu et al., 2024): a classifier assesses the query once and routes it to a single model (Ong et al., 2025; Tran et al., 2025). Because this decision is fixed before generation, it cannot adapt to complexity that changes within a response. Our work instead performs token-level routing, re-evaluating decisions during decoding to enable finer-grained allocation. Unlike probe- or classifier-based adaptive triggers, TokenRouter targets per-token control and therefore prioritizes signals already available from decoding.

3 Methodology

We propose TokenRouter, an adaptive runtime framework for efficient LLM reasoning via token-level routing. As shown in Figure 1, the framework combines three components: a robust entropy detector that turns noisy token entropy into a stable estimate of local reasoning difficulty, a self-calibrating control policy that converts a target compute budget into adaptive activation thresholds, and a dynamic intervention commitment mechanism that prevents oscillatory switching and preserves coherence during difficult spans.

3.1 Problem Formulation

We formulate adaptive inference as a constrained sequential decision process that dynamically se-

lects between a large language model (LLM) and a small language model (SLM). At each timestep t , the policy observes the current prefix state $s_t = (y_1, \dots, y_{t-1})$, chooses an action $a_t \in \{\text{LLM}, \text{SLM}\}$, generates the next token $y_t \sim p_{a_t}(\cdot | s_t)$, and transitions to s_{t+1} . The objective is to find a policy π^* that maximizes the expected utility $U(Y)$ of the final sequence $Y = (y_1, \dots, y_T)$ subject to a global budget on the LLM usage rate R_{target} :

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{Y \sim \pi} [U(Y)] \quad (1)$$

$$\text{subject to } \frac{1}{T} \sum_{t=1}^T \mathbb{I}(\pi(s_t) = \text{LLM}) \leq R_{\text{target}} \quad (2)$$

where $\mathbb{I}(\cdot)$ is the indicator function and $U(Y)$ is an external metric, such as final-answer accuracy. The key challenge is partial observability: the reasoning complexity required at each state s_t is latent and must be inferred online from noisy proxy signals.

3.2 Robust Entropy Detector

The first component addresses the perception challenge: obtaining a reliable real-time estimate of local reasoning difficulty. We use the Shannon entropy of the model’s output distribution P_t as an observable proxy. Here, black-box means that the controller does not access hidden states, attention maps, gradients, or model parameters, and only requires next-token log probabilities exposed by the decoding interface. The raw entropy at timestep t ,

denoted as z_t , is:

$$z_t = H(P_t) = - \sum_{v \in V} p_t(v) \log p_t(v) \quad (3)$$

The raw signal is highly volatile because it conflates genuine shifts in reasoning complexity with stochastic noise from autoregressive generation and local word variation. As a result, raw entropy alone is an unreliable basis for stable control.

We therefore model the underlying reasoning difficulty x_t as a latent variable in a state-space model. Assuming that this difficulty evolves smoothly over time, we use the following state evolution equation:

$$x_t = Fx_{t-1} + \epsilon_{t-1}, \quad \epsilon_{t-1} \sim \mathcal{N}(0, Q) \quad (4)$$

The observation equation links this latent state to the noisy entropy measurement:

$$z_t = Hx_t + \nu_t, \quad \nu_t \sim \mathcal{N}(0, R) \quad (5)$$

We set $F = 1$, reducing the latent dynamics to a random walk, and $H = 1$, treating the observed entropy z_t as a direct linear measurement of the underlying difficulty x_t . The process noise ϵ_{t-1} captures unpredictable shifts in reasoning difficulty, while the measurement noise ν_t accounts for stochasticity in the entropy signal.

This formulation yields an efficient closed-form estimate of the posterior distribution $p(x_t|z_{1:t})$ through a standard predict-update cycle. The prediction step forecasts the current prior:

$$\hat{x}_{t|t-1} = F\hat{x}_{t-1|t-1} \quad (6)$$

$$P_{t|t-1} = FP_{t-1|t-1}F^T + Q \quad (7)$$

The update step then incorporates the new observation z_t :

$$K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1} \quad (8)$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H\hat{x}_{t|t-1}) \quad (9)$$

$$P_{t|t} = (I - K_tH)P_{t|t-1} \quad (10)$$

The resulting posterior mean $\hat{x}_{t|t} = \mathbb{E}[x_t|z_{1:t}]$ provides a smooth estimate of latent reasoning difficulty and serves as the uncertainty signal for the subsequent control policy.

3.3 Self-Calibrating Control Policy

Having established a robust estimator for latent reasoning difficulty $\hat{x}_{t|t}$, we next address when to trigger LLM intervention while respecting a global compute budget. The main difficulty is that

a fixed activation threshold does not generalize well across heterogeneous and non-stationary problem instances, leading to inconsistent performance and less predictable cost.

To address this, we introduce a dynamic self-calibrating policy that detects deviations from an evolving difficulty baseline rather than reacting to absolute complexity. We first transform the filtered estimate $\hat{x}_{t|t}$ into a standardized anomaly score so that routing decisions remain comparable across domains. To do so, we maintain a moving reference for typical reasoning difficulty with an Exponentially Weighted Moving Average (EWMA), updating the signal mean μ_t and variance σ_t^2 as:

$$\mu_t = \lambda\hat{x}_{t|t} + (1 - \lambda)\mu_{t-1} \quad (11)$$

$$\sigma_t^2 = \lambda(\hat{x}_{t|t} - \mu_t)^2 + (1 - \lambda)\sigma_{t-1}^2 \quad (12)$$

where $\lambda \in (0, 1]$ is the smoothing factor. We then compute the standardized anomaly score S_t :

$$S_t = \frac{\hat{x}_{t|t} - \mu_t}{\sigma_t} \quad (13)$$

This score measures how strongly the current state deviates from its recent history and provides a normalized basis for intervention decisions.

The standardized score is the input to a self-calibrating trigger that determines the intervention threshold L_t at each timestep. We engage the LLM if $S_t > L_t$, and update L_t through a proportional feedback controller based on the current usage rate R_t and target budget R^* :

$$L_t = \text{clip}(L_{t-1} + \beta \cdot (R_t - R^*), L_{\min}, L_{\max}) \quad (14)$$

This formulation creates an adaptive closed-loop system in which $(R_t - R^*)$ acts as the error signal and β controls the trade-off between convergence speed and threshold stability. The controller therefore adapts online to the observed trajectory instead of relying on a preset intervention schedule.

If the system is over-budget ($R_t > R^*$), the threshold rises and future LLM activations become less likely. If it is under-budget, the threshold falls. In this way, the global budget is translated into an adaptive online decision rule without manual threshold tuning.

3.4 Dynamic Intervention Commitment

Having addressed when to intervene, we now address reasoning oscillation. A purely reactive policy that makes an independent routing decision at

each timestep can alternate rapidly between the LLM and SLM, fragmenting the chain of thought and degrading reasoning quality.

Our core insight is that intervention duration should scale with the severity of the anomaly that triggered it. Minor fluctuations may require only brief LLM assistance, whereas sharp spikes often mark the start of a difficult reasoning phase that benefits from sustained intervention. This turns routing from isolated token decisions into span-level support for contiguous hard segments.

Motivated by this insight, we design the dynamic intervention commitment mechanism. When an anomaly at timestep t^* triggers an intervention with a standardized score S_{t^*} (i.e., $S_{t^*} > L_{t^*}$), the system commits to engaging the LLM for at least k subsequent tokens. This commitment duration k is dynamically determined by the following rule:

$$k = \lceil k_{\text{base}} + \alpha \cdot \max(0, S_{t^*} - L_{t^*}) \rceil \quad (15)$$

To further formalize stability, we define a commitment decay function that gradually releases LLM engagement once the observed complexity stabilizes. Let D_t denote the residual commitment strength at step t :

$$D_t = D_{t-1} \cdot e^{-\gamma \Delta t} + \mathbb{I}(S_t > L_t) \quad (16)$$

where γ controls the decay rate and $\mathbb{I}(\cdot)$ is the indicator function for triggered anomalies. This formulation provides a smooth transition between intervention states and avoids premature disengagement during difficult spans.

Finally, to ensure continuity of reasoning, the effective routing decision at each timestep incorporates both the current anomaly signal and the ongoing commitment:

$$a_t = \begin{cases} \text{LLM}, & \text{if } D_t > \tau \\ \text{SLM}, & \text{otherwise} \end{cases} \quad (17)$$

This mechanism mitigates oscillation and preserves chain-of-thought coherence by adapting intervention duration to the severity of the reasoning challenge. It applies stronger stabilization to major anomalies and only brief intervention to minor ones, safeguarding solution quality while maintaining efficiency. In effect, the mechanism introduces temporal hysteresis that keeps the model assignment stable until the reasoning state has stabilized over subsequent reasoning steps.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate TokenRouter on three reasoning datasets that collectively span a wide spectrum of problem difficulty. The first is *GSM8K* (Cobbe et al., 2021), a dataset of grade school problems requiring multi-step arithmetic. The second is *MATH500* (Hendrycks et al., 2021), a benchmark of high school competition-level problems requiring symbolic manipulation and algebraic reasoning. The third is *AIME24* (Li et al., 2024), which contains challenging problems from the American Invitational Mathematics Examination, emphasizing creative problem-solving and advanced logical deduction. Together, these datasets provide a comprehensive evaluation landscape, covering diverse complexities from basic reasoning to sophisticated multi-hop inference.

Baselines. We compare TokenRouter against several baselines: Pure SLM and Pure LLM establish performance bounds using exclusively small or large models. Uniform Routing applies naive stochastic selection with fixed LLM probability matching our target usage ratio. We include CITER (Zheng et al., 2025), a confidence-based token-level method, and RouteLLM (Ong et al., 2025), which routes entire queries to a single model via its BERT (Devlin et al., 2019) or CASUAL (Wei et al., 2022a) classifiers. These baselines comprehensively evaluate routing strategies across different granularities and allocation mechanisms.

Model and Implementation Details. We evaluate TokenRouter using the DeepSeek-R1-Distill-Qwen (Guo et al., 2025) family of models. Unless otherwise specified, we use the 1.5B model as the SLM and the 14B model as the LLM. We set target ratios R_{target} to 0.7 for GSM8K and 0.8 for MATH500 and AIME24. The entropy detector uses $Q = 0.1$, $R = 0.5$. The control policy employs smoothing $\lambda = 0.05$, gain $\beta = 1.0$, initial threshold $L_0 = 2.0$ clipped to $[1.0, 4.0]$. The commitment mechanism sets $\alpha = 2.0$ and k_{base} of 20 for GSM8K and 50 for MATH500 and AIME24. All experiments use a temperature of 0.7 sampling on a server with 80GB NVIDIA H100 GPUs. The routing controller runs on CPU; both models retain reusable KV-cache states for the accepted prefix, entropy is computed from existing output logits without extra GPU memory, and the controller state requires less than 10MB of CPU memory.

Evaluation Metrics. We report final-answer ac-

Table 1: Comparison of TokenRouter against baselines across datasets using two different SLM-LLM combinations. TokenRouter consistently achieves performance comparable to the pure LLM while significantly reducing the LLM usage ratio.

Method	Combination: 1.5B & 7B				Combination: 1.5B & 14B			
	Acc (%) $\uparrow$	Average Tokens $\downarrow$	LLM Tokens $\downarrow$	LLM Ratio (%) $\downarrow$	Acc (%) $\uparrow$	Average Tokens $\downarrow$	LLM Tokens $\downarrow$	LLM Ratio (%) $\downarrow$
GSM8K								
Pure SLM	83.78	2190.7	0.0	0.0	83.78	2190.7	0.0	0.0
Pure LLM	90.86	1406.1	1406.1	100.0	94.62	1101.1	1101.1	100.0
Uniform Routing	87.58	1599.7	939.0	58.7	92.87	1222.5	762.8	62.4
CITER	86.73	1588.7	1129.8	71.1	93.26	1149.9	723.4	62.9
RouteLLM _{BERT}	88.85	1788.7	1256.6	70.2	88.38	1360.9	767.2	56.4
RouteLLM _{CASUAL}	87.52	1743.5	1062.4	60.9	91.82	1443.2	896.2	62.1
TokenRouter	90.07	1515.3	913.7	60.3	94.09	1070.6	635.9	59.4
MATH500								
Pure SLM	82.60	4500.0	0.0	0.0	82.60	4500.0	0.0	0.0
Pure LLM	88.40	3527.1	3527.1	100.0	89.00	2893.4	2893.4	100.0
Uniform Routing	84.20	3556.7	2041.5	57.4	85.40	4214.4	2414.9	57.3
CITER	84.40	4051.1	2536.7	62.6	86.20	4127.2	2539.0	61.5
RouteLLM _{BERT}	86.20	4208.4	2804.3	66.6	85.80	4036.2	2368.9	58.7
RouteLLM _{CASUAL}	84.80	4012.2	2616.1	65.2	86.80	4107.5	2850.9	69.4
TokenRouter	88.00	3880.9	2545.8	65.6	88.40	3974.0	2233.4	56.2
AIME24								
Pure SLM	26.67	11503.3	0.0	0.0	26.67	11503.3	0.0	0.0
Pure LLM	43.33	6811.2	6811.2	100.0	46.67	6363.5	6363.5	100.0
Uniform Routing	30.00	7253.2	4968.4	68.5	36.67	7067.8	5534.1	78.3
CITER	33.33	7372.4	5505.3	74.7	36.67	7774.7	5843.4	75.1
RouteLLM _{BERT}	36.67	8803.4	6097.8	69.3	40.00	8825.0	5885.4	66.7
RouteLLM _{CASUAL}	36.67	9023.7	6245.2	69.2	40.00	9061.9	6414.7	70.8
TokenRouter	43.33	6874.3	4981.8	72.4	46.67	6513.6	4948.3	76.0

curacy as the primary quality metric, using dataset-specific answer extraction followed by exact-match evaluation against the gold answer. To characterize efficiency, we report average generated tokens, total LLM tokens, and the LLM token ratio, defined as the fraction of generated tokens produced by the large model. The LLM ratio provides a hardware-agnostic proxy for large-model compute and token-metered API cost, while average generated tokens captures changes in output length. Because wall-clock latency also depends on SLM generation, hardware characteristics, routing decisions, and KV-cache synchronization, we report end-to-end runtime separately in Appendix B.

4.2 Main Results

Table 1 compares TokenRouter with baseline methods across benchmarks. TokenRouter maintains near-LLM accuracy while substantially reducing large-model usage and overall computational cost. On the challenging AIME24 benchmark, TokenRouter matches the pure 14B LLM at 46.67% accuracy while reducing LLM token usage by 24%. The

same pattern appears on GSM8K and MATH500, where TokenRouter recovers over 99% of pure-LLM accuracy (94.09% vs. 94.62% on GSM8K, and 88.40% vs. 89.00% on MATH500) while activating the LLM for under 60% of tokens. Compared with CITER and query-level RouteLLM, TokenRouter achieves higher accuracy under similar or lower budgets across all datasets, suggesting more effective capacity allocation when reasoning difficulty changes during generation. Appendix B reports end-to-end runtime, showing that wall-clock savings vary with dataset difficulty and hardware. Overall, the results show that TokenRouter preserves reasoning quality while substantially reducing large-model usage.

4.3 Budget-Controlled Trade-off

Figure 2 illustrates the trade-off between accuracy and efficiency under TokenRouter’s budget control across the three datasets. The actual LLM usage ratio closely tracks the target ratio, showing that the self-calibrating policy enforces the specified budget reliably. Accuracy then improves smoothly as

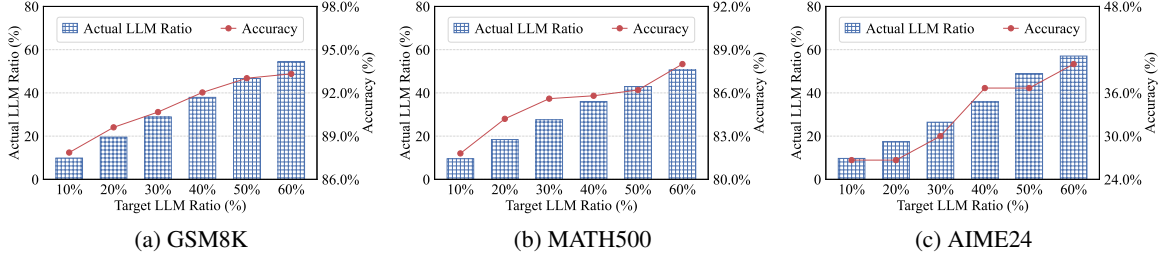


Figure 2: TokenRouter’s accuracy and actual LLM usage ratio under different target ratio budgets.

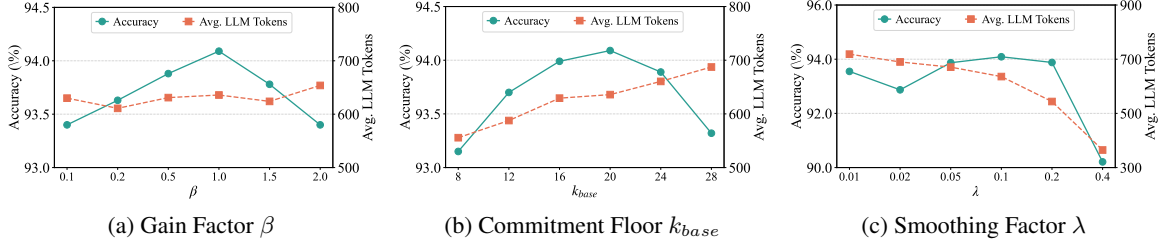


Figure 3: Effect of varying key hyperparameters on TokenRouter’s final accuracy and average LLM token usage.

more budget is allocated. On GSM8K, increasing the LLM budget from 10% to 60% raises accuracy from 87.86% to 93.32%. On MATH500, accuracy rises from 81.80% to 88.00% as the LLM activation ratio grows, and AIME24 shows the same monotonic trend. The resulting curves are smooth rather than erratic, suggesting that the controller adjusts capacity in a stable way as the target budget changes. These results indicate that TokenRouter lets users move predictably along the accuracy-cost curve, which is important for deployment under varying resource constraints. This behavior also indicates that budget tracking remains calibrated across problem difficulties.

4.4 Qualitative Analysis

To better understand TokenRouter’s decisions, we analyze representative moments when the controller activates the LLM. Table 2 shows four common scenarios: self-correction (“Wait, I made a mistake...”), planning under uncertainty (“Hmm, what is the next step?...”), pivotal logical deduction (“Therefore...”), and strategy adjustment (“But the question asks for...”). These examples suggest that the entropy-driven detector captures diverse high-difficulty reasoning moments rather than a single failure mode. Across all cases, entropy spikes align with conceptual transitions where continued SLM decoding may cause reasoning drift or premature conclusions. The triggers are therefore semantically meaningful rather than arbitrary local fluctu-

Table 2: Examples of critical reasoning moments identified by TokenRouter. The table highlights distinct scenarios that need an LLM intervention to ensure reasoning integrity.

Reasoning Step	Entropy	S_t
<i>Case 1: Self-Correction</i>		
... change is $20 - 17.25 = 2.75$. Wait , I made a mistake. ...	3.21	4.87
<i>Case 2: Uncertainty & Planning</i>		
... So $A = \pi \times 5^2 = 25\pi$. Hmm , what is the next step? ...	4.11	5.23
<i>Case 3: Logical Deduction</i>		
... The total distance is 300 miles. Therefore , the average speed is. ...	4.53	5.98
<i>Case 4: Strategy Shift</i>		
... the fraction is $30/90 = 1/3$. But the question asks for the percentage. ...	2.98	4.12

ations. Selectively escalating to the LLM at these points helps preserve coherence while avoiding unnecessary computation. This pattern is consistent with using entropy as an online proxy for local reasoning difficulty.

4.5 Parameter Sensitivity Analysis

We further analyze the sensitivity of TokenRouter to its key hyperparameters on GSM8K, with results shown in Figure 3. Overall, the framework remains robust across a broad range of settings. Varying the gain factor β yields stable, non-oscillatory control, with the best balance between responsiveness

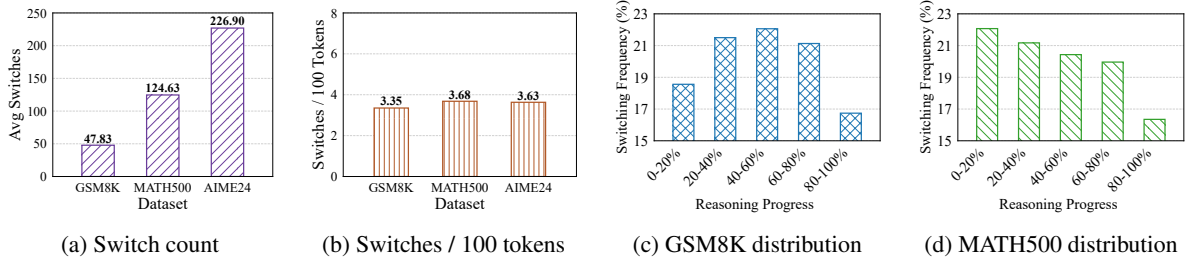
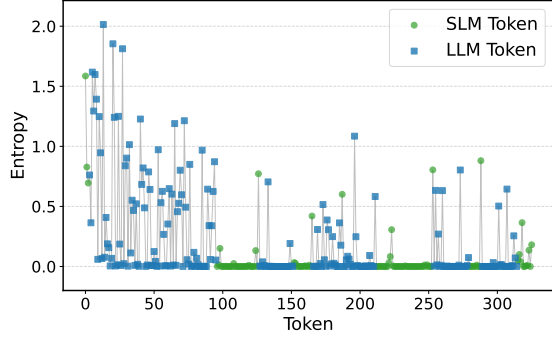
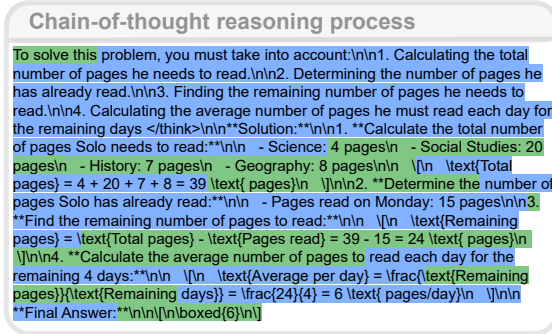


Figure 4: TokenRouter’s switching behavior across datasets.



(a) Entropy trajectory where peaks mark complex reasoning moments.



(b) Reasoning trace showing LLM engagement aligned with key reasoning shifts.

Figure 5: Reasoning dynamics and entropy-guided token routing. TokenRouter activates the LLM only at difficult reasoning steps, selectively allocating computational capacity to preserve reasoning coherence while maintaining high accuracy.

and stability around $\beta = 1.0$. Adjusting the commitment floor k_{base} trades coherence against cost: smaller values can fragment reasoning, while larger ones increase computation, with $k_{base} = 20$ giving the best trade-off. The framework is somewhat more sensitive to the smoothing factor λ : very small values overreact to token noise, whereas large values respond too slowly to real complexity shifts. A setting around $\lambda = 0.1$ performs best. Taken together, these trends indicate that the method be-

Table 3: Ablation study of TokenRouter’s core components.

Configuration	GSM8K		MATH500	
	Acc (%) $\uparrow$	Ratio (%) $\downarrow$	Acc (%) $\uparrow$	Ratio (%) $\downarrow$
w/o Robust Perception	93.04	67.1	86.60	62.9
w/o Budget Control	91.92	39.5	85.20	37.9
w/o Adaptive Commit.	87.86	10.9	83.20	9.7
w/o Commitment Floor	90.00	20.5	84.80	18.4
TokenRouter	94.09	59.4	88.40	56.2

haves predictably under moderate parameter variation and does not require careful tuning.

4.6 Ablation Study

We conduct an ablation study to evaluate the contribution of each core component in TokenRouter, with results summarized in Table 3. All modules contribute to the final balance between accuracy and efficiency. The adaptive commitment mechanism is the most critical: removing it drops GSM8K accuracy from 94.09% to 87.86%, indicating that fragmented intervention harms multi-step reasoning. Removing budget control also lowers accuracy by misallocating LLM usage, while removing robust perception makes the system more vulnerable to entropy noise, increasing the LLM usage ratio on MATH500 by nearly 7% while reducing accuracy. The ablations therefore align closely with the design motivation of the method: stable perception, budget regulation, and sustained intervention each address a distinct failure mode. Together, these results show that perception, control, and commitment play complementary roles in reliable fine-grained routing.

4.7 Analysis of Switching Behavior

To examine runtime behavior, we analyze TokenRouter’s switching dynamics across datasets. Figure 4 shows that TokenRouter makes a moder-

Table 4: Task-generalization evaluation using the DeepSeek-R1-Distill-Qwen 1.5B–7B pair.

Dataset	Pure SLM Acc (%)	Pure LLM Acc (%)	TokenRouter Acc (%)	LLM Ratio (%)	Avg. Tokens
ARC	51.88	79.03	76.60	67.69	1394.3
CommonSenseQA	42.59	63.31	61.59	70.30	1126.2
HumanEval	49.39	75.83	74.50	65.50	3479.5
OpenBookQA	43.80	72.60	72.80	70.62	1232.5

ate and consistent number of switches, typically fewer than five per problem. This indicates that the controller avoids oscillation and instead performs chunk-level routing aligned with reasoning structure. The switching distributions further show that most interventions occur in the middle stages of reasoning, where entropy rises and local reasoning difficulty is higher. Later arithmetic or summary stages trigger fewer LLM activations. This pattern indicates that TokenRouter deploys the large model selectively to difficult spans rather than switching indiscriminately, which is consistent with the intended role of the commitment mechanism. The moderate switching frequency also aligns with the lightweight routing overhead reported in Appendix B, further supporting a stable span-level routing pattern.

4.8 Reasoning Dynamics Analysis

Figure 5 provides a detailed examination of the reasoning dynamics, revealing how TokenRouter adaptively allocates computational resources throughout the reasoning process.

As illustrated in Figure 5a, the entropy trajectory exhibits structured fluctuations that correlate with local reasoning difficulty. Sharp entropy spikes emerge at moments of uncertainty, signaling transitions into high-complexity reasoning phases that trigger LLM activation. In contrast, low-entropy segments correspond to routine narrative transitions, which the SLM handles efficiently.

Figure 5b aligns these entropy variations with the actual chain-of-thought trace. The LLM is predominantly engaged during conceptual abstraction, intermediate verification, or strategic adjustment, while the SLM dominates procedural computation and connective text generation. This alternation pattern shows that TokenRouter allocates additional capacity at spans where uncertainty and reasoning difficulty intensify. Overall, this alignment makes TokenRouter’s control interpretable: entropy tracks local difficulty without auxiliary probes or additional router calls, while span-level intervention

preserves coherence and avoids redundant computation.

4.9 Generalization Beyond Math

To assess task generalization, we evaluate TokenRouter beyond mathematical reasoning on ARC (Clark et al., 2018), CommonSenseQA (Talmor et al., 2019), HumanEval (Chen et al., 2021), and OpenBookQA (Mihaylov et al., 2018). These experiments use the DeepSeek-R1-Distill-Qwen 1.5B–7B pair. As shown in Table 4, TokenRouter retains 96.9%–100.3% of pure-LLM accuracy while reducing large-model activation to 65.5%–70.6% of generated tokens. The results indicate that entropy-guided token-level routing transfers beyond mathematical reasoning to science QA, commonsense QA, code generation (Xiao et al., 2025a,b, 2026a,b,c), and open-book QA. Because output lengths vary by dataset, we use the generated-token LLM ratio as the efficiency metric and report wall-clock latency separately under the same routing policy.

5 Conclusion

This work introduces TokenRouter, an adaptive runtime framework for token-level model routing. It combines a robust entropy detector, a self-calibrating control policy, and a dynamic commitment mechanism for stable intervention. Together, these components enable fine-grained allocation between large and small models. Our experiments show that TokenRouter retains over 99% of pure-LLM reasoning accuracy while activating the LLM for only 56.2% to 76.0% of generated tokens. Because the controller relies only on decoding-time uncertainty, it remains compatible with independently deployed models while making the accuracy-efficiency trade-off explicit during generation. These results support uncertainty-guided token-level routing as a practical mechanism for small-large collaboration in multi-step reasoning.

Limitations

This work has several limitations. First, our evaluation does not exhaust all workload and model-pair regimes. TokenRouter assumes a useful default-escalation gap, where the default model is efficient on routine spans and the escalation model improves difficult spans. This assumption is supported across the evaluated tasks, model families, and hardware settings, but may be weaker when the default model is highly domain-specialized or when uncertainty is poorly aligned with task difficulty. Second, TokenRouter is designed as a lightweight black-box controller and relies on next-token log probabilities rather than model-internal states. This choice favors compatibility and low overhead, but may miss signals available in tightly integrated inference stacks. White-box states, auxiliary probes, or larger cascades could provide more routing evidence, yet they also add memory, integration, and latency cost. Efficient use of such signals is left to future work.

Acknowledgments

This work was supported by the Basic and Frontier Research Project of Peng Cheng Laboratory (PCL) under Grant No. 2025QYA001, the Open Project of the National Engineering Laboratory for Technology of Internet Domain Name under Grant No. KF202512, and the Guangdong S&T Programme under Grant No. 2024A0101010001.

References

- Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. 2023. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5910–5924. Association for Computational Linguistics.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. EE-LLM: Large-scale training and inference of early-exit large language models with 3D parallelism. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 7163–7189. PMLR.
- Zhijun Chen, Xiaodong Lu, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Ming Li, Likang Xiao, Dingqi Yang, Xiao Huang, Yikun Ban, Hailong Sun, and Philip S. Yu. 2026. Harnessing multiple large language models: A survey on LLM ensemble. *arXiv preprint arXiv:2502.18036*.
- 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 preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, et al. 2024. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. 2024. MaskLLM: Learnable semi-structured sparsity for large language models. *Advances in Neural Information Processing Systems*, 37:7736–7758.
- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638.
- 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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer,

- and Shriyash Kaustubh Upadhyay. 2024. Router-Bench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*.
- Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. 2026. Universal model routing for efficient LLM inference. In *International Conference on Learning Representations*.
- Juno Kim, Denny Wu, Jason Lee, and Taiji Suzuki. 2025. Metastable dynamics of chain-of-thought reasoning: Provable benefits of search, RL and distillation. *arXiv preprint arXiv:2502.01694*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213.
- Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, et al. 2024. NuminaMath. Hugging Face repository.
- Yue Liu, Jiaying Wu, Yufei He, Ruihan Gong, Jun Xia, Liang Li, Hongcheng Gao, Hongyu Chen, Baolong Bi, Jiaheng Zhang, Zhiqi Huang, Bryan Hooi, Stan Z. Li, and Keqin Li. 2025. Efficient inference for large reasoning models: A survey. *arXiv preprint arXiv:2503.23077*.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.
- Chaoyue Niu, Yucheng Ding, Junhui Lu, Zhengxiang Huang, Hang Zeng, Yutong Dai, Xuezhen Tu, Chengfei Lv, Fan Wu, and Guihai Chen. 2025. Collaborative learning of on-device small model and cloud-based large model: Advances and future directions. *arXiv preprint arXiv:2504.15300*.
- 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 with preference data. In *International Conference on Learning Representations*.
- 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*, pages 5568–5578.
- David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*.
- Zhenning Shi, Dan Zhao, Yijia Zhu, Guorui Xie, Qing Li, and Yong Jiang. 2025a. Helios: Learning and adaptation of matching rules for continual in-network malicious traffic detection. In *Proceedings of the ACM on Web Conference 2025*, pages 2319–2329. Association for Computing Machinery.
- Zhenning Shi, Yijia Zhu, Yi Xie, Junhan Shi, Guorui Xie, Haotian Zhang, Yong Jiang, Congcong Miao, and Qing Li. 2025b. Reasoning under uncertainty: Efficient LLM inference via unsupervised confidence dilution and convergent adaptive sampling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32204–32218. Association for Computational Linguistics.
- Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158. Association for Computational Linguistics.
- Co Tran, Salman Paracha, Adil Hafeez, and Shuguang Chen. 2025. Arch-Router: Aligning LLM routing with human preferences. *arXiv preprint arXiv:2506.16655*.
- Clovis Varangot-Reille, Christophe Bouvard, Mathieu Ciancone, Antoine Gourru, Marion Schaeffer, and François Jacquenet. 2026. Doing more with less: A survey on routing strategies for resource optimisation in large language model-based systems. *Journal of Artificial Intelligence Research*, 86. Article 14.
- Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Hang Wu, Jianian Zhu, Yinghui Li, Haojie Wang, Biao Hou, and Jidong Zhai. 2025. SpecRouter: Adaptive routing for multi-level speculative decoding in large language models. *arXiv preprint arXiv:2505.07680*.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 38087–38099. PMLR.

Jingyu Xiao, Jiantong Qin, Shuoqi Li, Man Ho Lam, Yuxuan Wan, Jen-tse Huang, Yintong Huo, and Michael R. Lyu. 2026a. Component-based reusable UI code generation for complex websites via semantic segmentation and fine-grained feedback. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5638–5649.

Jingyu Xiao, Yuxuan Wan, Yintong Huo, Zixin Wang, Xinyi Xu, Wenxuan Wang, Zhiyao Xu, Yuhang Wang, and Michael R. Lyu. 2025a. Interaction2Code: Benchmarking MLLM-based interactive webpage code generation from interactive prototyping. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 241–253. IEEE Press.

Jingyu Xiao, Ming Wang, Man Ho Lam, Yuxuan Wan, Junliang Liu, Yintong Huo, and Michael R. Lyu. 2025b. DesignBench: A comprehensive benchmark for MLLM-based front-end code generation. *arXiv preprint arXiv:2506.06251*.

Jingyu Xiao, Zhongyi Zhang, Haoran Hou, Yuxuan Wan, Yuan Jiang, Yintong Huo, and Michael R. Lyu. 2026b. VisualRepair: Dynamic tool calling and region focusing for visual software issue repair. *arXiv preprint arXiv:2607.14075*.

Jingyu Xiao, Zhongyi Zhang, Yuxuan Wan, Yintong Huo, Yang Liu, and Michael R. Lyu. 2026c. EfficientUICoder: A bidirectional token compression framework for efficient MLLM-based UI code generation. *Proceedings of the ACM on Software Engineering*, 3(FSE):2396–2418.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Zishun Yu, Tengyu Xu, Di Jin, Karthik Abinav Sankararaman, Yun He, Wenxuan Zhou, Zhouhao Zeng, Eryk Helenowski, Chen Zhu, Sinong Wang, et al. 2025. Think smarter not harder: Adaptive

reasoning with inference aware optimization. *arXiv preprint arXiv:2501.17974*.

Longfei Yun, Yonghao Zhuang, Yao Fu, Eric P Xing, and Hao Zhang. 2024. Toward inference-optimal mixture-of-expert large language models. *arXiv preprint arXiv:2404.02852*.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P Xing, Hongyi Wang, and Huaxiu Yao. 2025. CITER: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*.

A Cross-Family Transfer

To examine cross-family transfer, we pair Qwen3-8B (Yang et al., 2025) as the LLM with DeepSeek-R1-Distill-Qwen-1.5B as the SLM. This setting changes the large-model family without retraining the router and tests whether entropy-based control transfers beyond a single model-size ladder. Table 5 shows that TokenRouter remains close to the Qwen3-8B LLM on GSM8K and MATH while substantially reducing large-model activation. AIME exhibits a larger gap, suggesting greater sensitivity to model-pair calibration. These results instantiate the hierarchy assumed by TokenRouter. The escalation model is validated to improve the target workload relative to the default model, making the hierarchy a workload-calibrated design choice.

Table 5: Cross-family transfer using Qwen3-8B as the LLM and DeepSeek-R1-Distill-Qwen-1.5B as the SLM.

Dataset	Pure SLM Acc (%)	Pure LLM Acc (%)	TokenRouter Acc (%)	LLM Ratio (%)
GSM8K	75.00	97.00	93.00	57.05
MATH	77.00	83.00	81.00	74.31
AIME	20.00	60.00	53.33	79.08

B Latency and Overhead

End-to-end latency. Table 6 reports wall-clock latency across datasets and hardware platforms. TokenRouter achieves the largest gains on GSM8K, reducing total latency by 21.7% on L20, 15.0% on RTX 4090, and 12.1% on H100. Gains vary more on MATH and AIME because these benchmarks contain longer high-uncertainty spans, which increase LLM residency and leave less replaceable large-model decoding. The smaller proportional gain on H100 is consistent with faster pure-LLM decoding on that platform, which reduces the latency portion that routing can remove.

Table 6: End-to-end wall-clock latency in a self-hosted deployment setting. TR denotes TokenRouter.

Dataset	Hardware	Pure LLM Total (s)	TR LLM Time (s)	TR SLM Time (s)	TR Total (s)	Improve. (%)	Decision (ms)
GSM8K	H100	17.235	13.475	1.679	15.154	12.1	6.253
GSM8K	L20	53.679	36.021	6.008	42.029	21.7	25.700
GSM8K	RTX 4090	41.106	30.363	4.574	34.937	15.0	23.042
MATH	H100	55.613	37.775	16.651	54.426	2.1	53.911
MATH	L20	128.126	84.691	22.190	106.881	16.6	86.285
MATH	RTX 4090	95.589	69.500	22.606	92.106	3.6	91.836
AIME	H100	133.359	89.597	34.482	124.079	7.0	112.636
AIME	L20	282.222	203.455	58.924	262.379	7.0	173.650
AIME	RTX 4090	265.592	191.045	57.236	248.282	6.5	169.068

KV-cache synchronization. The routing decision logic is lightweight and runs on CPU. Decision latency remains in the millisecond range, from 6.253ms on GSM8K/H100 to 173.650ms on AIME/L20, and is small relative to total generation time. TokenRouter retains reusable KV-cache states for both models. When the inactive model resumes, a catch-up step reuses the retained prefix cache and extends only the missing accepted span instead of recomputing the full prompt and generation history. The reported wall-clock latency includes this synchronization cost.

Memory footprint. TokenRouter introduces no additional GPU memory for entropy computation because entropy is computed directly from existing output logits. The additional CPU memory required for controller state is less than 10 MB.

C Routing Design Scope

Retrieval-style triggers. TokenRouter is conceptually related to active retrieval and adaptive RAG systems, which also use uncertainty to decide when to invoke an auxiliary capability. However, retrieval triggers are typically invoked only a few times per query, whereas TokenRouter may evaluate routing at every decoding step. Semantic probes or trained meta-classifiers can therefore incur sub-

stantial per-token overhead, require extra supervision, or introduce additional model calls. We use entropy because it is already available from the decoding distribution.

White-box signals. Richer internal signals such as hidden states or attention patterns may improve difficulty estimation in white-box deployments. However, they require model-internal access, inference-engine integration, and handling of intermediate tensors. This requirement conflicts with our goal of a lightweight controller for unmodified models and standard decoding interfaces.

Model hierarchy. The two-model hierarchy is a calibrated deployment choice for the target workload. If the lower-cost model has specialized strengths, it can serve as the default model, while the escalation model is selected by validation to cover the remaining difficult spans. Multi-layer cascades such as Small-Medium-Large are possible, but they increase GPU residency, KV-cache state, routing thresholds, commitment decisions, and scheduling overhead. We therefore focus on a two-model setting as a practical real-time design point for separating low- and high-difficulty spans.