

Generation-Gated Cascaded Routing for Large Language Models

Yanqiu Xu* Han Luo* Lei Feng†
Southeast University, Nanjing, China
{xyqiu, hanluo, fenglei}@seu.edu.cn

Abstract

Cost-effective deployment of large language models (LLMs) often relies on routing each query to either a low-cost local model or a stronger expert model. Existing routers typically decide either before generation, which is cheap but cannot observe the local model’s answer, or after generation, which provides a stronger correctness signal but incurs additional generation cost. We argue that this dichotomy overlooks a third option: *treating local generation itself as an optional information-acquisition action*. To this end, we propose **Generation-Gated Cascaded Routing for Large Language Models (GenGate)**, a lightweight fine-tuning method that trains a local LLM to internalize cascaded routing decisions while preserving its generation capability. A Bellman-motivated prior gate first decides whether to generate a local attempt. If an attempt is produced, a correctness-supervised posterior gate then decides whether to submit it or defer to the expert model. Both decisions are expressed through action-token probabilities in the same model, requiring no external router or auxiliary classifier at inference time. Compared with routers that decide strictly before or after local generation, GenGate reduces joint deployment cost while maintaining competitive routing accuracy across task domains, preserving the benefits of posterior routing without requiring local generation for every query. Code is available at <https://github.com/xsyhh/GenGate>.

1 Introduction

Deploying large language model (LLM) systems under a budget (Chen et al., 2024a; Ong et al., 2025; Ding et al., 2024) often requires routing each query either to a lower-cost local model or to a stronger expert. Existing routers typically decide either

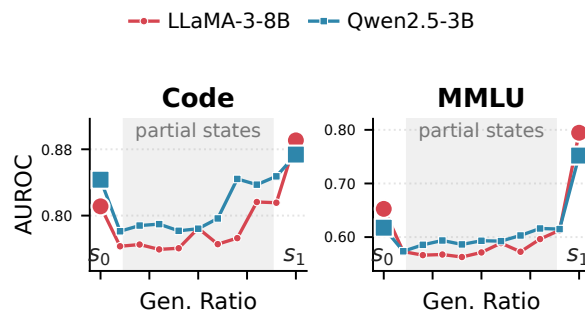


Figure 1: Ratio-wise linear probes for predicting final local-attempt correctness on Code and MMLU. Ratio 0.0 (s_0) is the prompt-only state, and ratio 1.0 (s_1) is the completed-attempt state. Code uses the held-out split built from the LeetCodeDataset test branch and HumanEvalPlus. MMLU uses the official test split.

before or after local generation. Pre-generation routers use only the query, avoiding generation cost for deferred queries but lacking answer-dependent evidence. (Ong et al., 2025; Chen et al., 2024b). Post-generation routers can judge a realized answer, but pay generation cost even for queries ultimately deferred to the expert. (Chuang et al., 2025; Kadavath et al., 2022; Manakul et al., 2023). This creates a fundamental tradeoff between acquiring stronger routing evidence and avoiding unnecessary generation.

Figure 1 examines how correctness-predictive evidence evolves as local generation progresses. Before any local tokens are produced, the input-only representation (s_0) already provides above-chance signal for predicting whether the local model will answer correctly. This signal becomes more informative after a completed local attempt is observed: probes at the completed-attempt point (s_1) achieve higher correctness-prediction AUROC on both Code and MMLU, but reaching this point requires generating the entire local response. Partial generations are not a dependable compromise, as their probe performance fluctuates across gener-

*Equal contribution.

†Corresponding author.

ation ratios and does not consistently approach the completed-attempt signal. Appendix B extends this probe analysis to all domains and both local-model families. These results suggest that routing should not be framed as a fixed choice between deciding from the prompt for every query and generating before every decision. A router should therefore decide when input-level evidence is sufficient and when the additional evidence from a realized local attempt is worth acquiring.

In this paper, we propose **Generation-Gated Cascaded Routing (GenGate)**, a lightweight method that fine-tunes a single local LLM to internalize cascaded routing decisions through action-token supervision, preserving generation capability without an external router or auxiliary classifier. GenGate consists of three steps. (i) *Rollout-based supervision construction*. For each training query, we sample multiple bounded local attempts and evaluate their correctness. These rollouts provide two supervision signals: a Bellman-motivated empirical local-success target for the initial $\langle \text{attempt} \rangle / \langle \text{defer} \rangle$ decision, and realized correctness labels for the posterior $\langle \text{self} \rangle / \langle \text{defer} \rangle$ decision. (ii) *Cascaded action-token fine-tuning*. The local LLM is fine-tuned to emit routing actions at both decision points: $\langle \text{attempt} \rangle / \langle \text{defer} \rangle$ before generation and $\langle \text{self} \rangle / \langle \text{defer} \rangle$ after a bounded attempt. (iii) *Generation-gated inference*. At test time, the model attempts only when local generation is worth its cost, attempted queries then either self-submit or defer to the expert.

Experiments across code generation, mathematical reasoning, and knowledge-intensive question answering show that GenGate reduces joint deployment cost while maintaining competitive routing accuracy. Compared with routers that decide strictly before or after local generation, GenGate preserves much of the benefit of posterior routing while avoiding local generation on queries where the prior signal already supports deferral.

2 Related Work

LLM routing and cascades. LLM routing reduces serving cost by directing each query to an appropriate model (Chen et al., 2024a; Ding et al., 2024; Ong et al., 2025; Hu et al., 2024). A central design axis is *when* the routing decision is made (Varangot-Reille et al., 2026). Pre-generation routers commit before any local answer is produced, relying on query difficulty or preference sig-

nals (Lu et al., 2024; Chen et al., 2024b). Related methods instead use response-level aggregation, hybrid select-then-cascade routing, or adaptive test-time sampling (Jiang et al., 2023; Wang et al., 2023; Shah and Shridhar, 2025; Ding et al., 2025). Post-generation cascades first produce local responses and then use response-level consistency to decide whether to escalate (Yue et al., 2024). AutoMix (Aggarwal et al., 2024) combines self-verification with a POMDP router, but every query still pays the full local generation cost. GenGate instead treats local generation as a conditional information-acquisition action: it can defer from the prompt when prior evidence is sufficient, and acquire posterior evidence only when a local attempt is worth its cost.

Confidence estimation and self-assessment.

Post-generation routing estimates local-answer correctness using signals such as token probabilities (Kadavath et al., 2022), verbalized confidence (Tian et al., 2023; Xiong et al., 2024; Geng et al., 2024), sampling-based consistency (Manakul et al., 2023; Tan et al., 2025; Farquhar et al., 2024), and hidden-state probes (Azaria and Mitchell, 2023; Khanmohammadi et al., 2025; Zhang et al., 2025b,a). These methods judge whether a completed answer should be trusted, so even queries ultimately deferred to the expert incur local generation cost. Self-REF (Chuang et al., 2025) fine-tunes an LLM to emit post-generation confidence tokens and routes by their score. GenGate uses a similar action-token interface for posterior self-assessment, but adds a prior $\langle \text{attempt} \rangle / \langle \text{defer} \rangle$ decision that determines whether the completed answer should be generated in the first place. Thus, the posterior gate is invoked only when answer-dependent evidence is expected to be useful.

Selective prediction, deferral, and abstention.

Selective prediction and learning-to-defer methods let a model abstain or pass an input to an expert when prediction risk is high (Geifman and El-Yaniv, 2017; Madras et al., 2018; Mozannar and Sontag, 2020; Verma et al., 2023; Cao et al., 2023; Liu et al., 2024; Alves et al., 2024; Wen et al., 2025; Jitkrittum et al., 2023). These frameworks typically assume a fixed information state, ignoring the option value of acquiring additional evidence before committing. Recent LLM cascade work relaxes this assumption: Zellinger et al. (2025) allow early models to abstain before expensive later stages, and Fanconi and van der Schaar (2025)

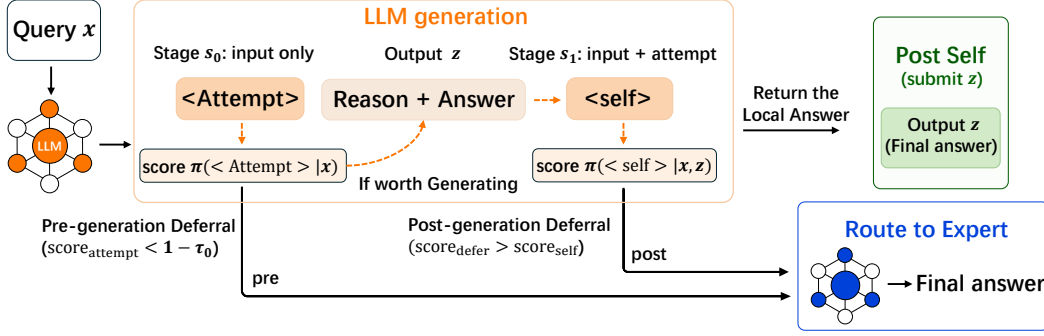


Figure 2: GenGate decision flow. At stage s_0 (input only), the model scores $\pi(\langle \text{attempt} \rangle | x)$: if $\pi(\langle \text{attempt} \rangle | x) < 1 - \tau_0$, the query is *pre-deferred* without generation. Otherwise, the LLM generates a reasoning attempt z . At stage s_1 (input + attempt), the model scores $\pi(\langle \text{self} \rangle | x, z)$: high confidence leads to local submission (output z). Otherwise the query is *post-deferred*. Both defer paths produce defer candidates scored by the same expert-routing budget. A global cutoff γ over merged defer scores determines which candidates are routed to the expert.

learn cascaded deferral policies with human fallback. These methods decide whether to continue or exit a cascade stage. GenGate focuses on a different query-level decision: whether local generation itself is worth paying for as evidence for a later submit/defer choice. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) also avoids unnecessary computation, but does so at the token level within a single generation, whereas GenGate gates the generation attempt at the query level.

3 Generation-Gated Cascaded Routing

We propose a novel routing method called GenGate (Generation-Gated Cascaded Routing), which fine-tunes a local LLM as a routing policy whose decisions condition on either prior evidence (the input alone) or posterior evidence (the input together with a bounded local attempt). The model learns whether a bounded local attempt is worth its cost, using a Bellman-motivated initial target and correctness-supervised posterior decisions. The inefficiency we target is local generation on examples that are ultimately deferred to the expert. GenGate addresses this by making the attempt itself a learned action rather than an unconditional preprocessing step. The same local model therefore acts as both generator and router, avoiding an external routing module while preserving the option to use posterior evidence when it is worth acquiring. Figure 2 illustrates the decision flow.

3.1 Generation-Gated Decision Formulation

Let x denote an input query and let z denote a bounded local attempt. GenGate makes routing decisions at two decision points. The first decision point observes only the input, denoted by $s_0 = x$,

and chooses between pre-generation deferral scoring and generating a local attempt. The second decision point observes both the input and the realized attempt, denoted by $s_1 = (x, z)$, and chooses between submitting the local answer and deferring to the expert.

We express these decisions with verbalized action markers: $\langle \text{defer} \rangle$ for deferral, $\langle \text{attempt} \rangle$ for attempting, and $\langle \text{self} \rangle$ for submission. All markers have equal token length (three tokens), avoiding length-induced bias when comparing their sequence log-probabilities. The three possible action sequences are

$$\begin{aligned} &\langle \text{defer} \rangle, \\ &\langle \text{attempt} \rangle z \langle \text{self} \rangle, \\ &\langle \text{attempt} \rangle z \langle \text{defer} \rangle. \end{aligned} \tag{1}$$

3.2 Training Objective

GenGate is trained from rollout-based supervision. For each training query x , we sample N bounded attempts $z_i \sim \pi_{\text{ref}}(\cdot | x)$ and evaluate each attempt with the task evaluator, obtaining binary correctness labels $r_i = r(x, z_i) \in \{0, 1\}$. These rollouts provide supervision for the two decision points defined in Section 3.1. At the posterior decision point $s_1 = (x, z_i)$, the target is direct: correct attempts should be self-submitted, while incorrect attempts should be deferred. At the initial decision point $s_0 = x$, however, the model must decide whether generating a local attempt is worth its cost before observing any attempt.

To construct the initial target, we use a one-step Bellman backup over the two decision points. The value of attempting at s_0 equals the expected value of the optimal posterior action at s_1 , minus

the generation cost c_A . Under the utility ordering $U_{\text{self-correct}} > U_{\text{defer}} > U_{\text{self-wrong}}$, the optimal posterior action is to self-submit when $r = 1$ and defer when $r = 0$. Therefore, the advantage of attempting over immediate deferral is (full derivation in Appendix A)

$$Q_A(s_0) - Q_D(s_0) = \Pr[r = 1] \cdot \Delta U - c_A, \quad (2)$$

where $\Delta U = U_{\text{self-correct}} - U_{\text{defer}}$. Attempting is favorable when $\Pr[r = 1] > c_A/\Delta U$. Thus, the initial gate can compare the relative preference between attempting and deferring by estimating $\Pr[r = 1]$, rather than by directly estimating the harder-to-observe values $Q_A(s_0)$ and $Q_D(s_0)$. We approximate this probability using the empirical rollout success rate

$$\hat{t}_0(x) = \frac{1}{N} \sum_{i=1}^N r_i \approx \Pr_{z \sim \pi_{\text{ref}}} [r(x, z) = 1]. \quad (3)$$

Thus, $\hat{t}_0(x)$ serves as a soft target for the prior $\langle \text{attempt} \rangle / \langle \text{defer} \rangle$ decision, while the realized correctness label r_i provides a hard target for the posterior $\langle \text{self} \rangle / \langle \text{defer} \rangle$ decision. We optimize these targets with standard binary cross-entropy on action-token log-odds rather than reinforcement learning.

We parameterize both decisions through action-phrase log-probabilities, scoring phrase a by

$$\log \pi_\theta(a | s) = \sum_{t=1}^{|a|} \log \pi_\theta(a_t | s, a_{<t}). \quad (4)$$

We compare $\langle \text{attempt} \rangle$ against $\langle \text{defer} \rangle$ at s_0 and $\langle \text{self} \rangle$ against $\langle \text{defer} \rangle$ at s_1 using $\pi_\theta(a_+ | s) = \sigma(\Delta_\theta(s))$, where $\Delta_\theta(s)$ denotes $\log \pi_\theta(a_+ | s) - \log \pi_\theta(a_- | s)$.

The initial loss matches the s_0 action log-odds to the soft initial target $\hat{t}_0(x)$:

$$\begin{aligned} \mathcal{L}_{s_0}(x) = & -\hat{t}_0(x) \log \sigma(\Delta_\theta^{(0)}(x)) \\ & - (1 - \hat{t}_0(x)) \log \sigma(-\Delta_\theta^{(0)}(x)), \end{aligned} \quad (5)$$

where $\Delta_\theta^{(0)}(x)$ denotes the prior log-odds $\log \pi_\theta(\langle \text{attempt} \rangle | x) - \log \pi_\theta(\langle \text{defer} \rangle | x)$. The posterior loss uses the realized correctness label r_i :

$$\begin{aligned} \mathcal{L}_{s_1}(x, z_i) = & -r_i \log \sigma(\Delta_\theta^{(1)}(x, z_i)) \\ & - (1 - r_i) \log \sigma(-\Delta_\theta^{(1)}(x, z_i)), \end{aligned} \quad (6)$$

where $\Delta_\theta^{(1)}(x, z_i)$ denotes the posterior log-odds $\log \pi_\theta(\langle \text{self} \rangle | x, z_i) - \log \pi_\theta(\langle \text{defer} \rangle | x, z_i)$. Combining both components, the **complete training**

objective is:

$$\begin{aligned} \mathcal{L}(\theta) = \mathbb{E}_{x, z_i} [\mathcal{L}_{s_0}(x) + \mathcal{L}_{s_1}(x, z_i)] \\ + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}(\theta), \end{aligned} \quad (7)$$

The KL term is computed only over generation tokens in posterior samples:

$$\mathcal{L}_{\text{KL}}(\theta) = \mathbb{E}_{h \in \mathcal{H}} [\text{KL}(\pi_\theta(\cdot | h) \parallel \pi_{\text{ref}}(\cdot | h))], \quad (8)$$

where $\mathcal{H}$ denotes intermediate prefixes within the generated attempt z_i . This keeps π_θ close to π_{ref} on attempt generation while leaving action logits trainable. Empirically, this preserves local answer quality: at 0% expert budget, GenGate nearly matches the reference local model across all six evaluation settings (Table 3).

3.3 Inference

At inference time, GenGate applies the two learned action-token decisions sequentially. Given a query x , the model first evaluates the initial defer score

$$d_0(x) = \pi_\theta(\langle \text{defer} \rangle | x). \quad (9)$$

If $d_0(x) > \tau_0$, GenGate skips local generation and treats the query as a pre-generation candidate. Otherwise, the model generates a bounded local attempt $z \sim \pi_\theta(\cdot | x)$ and evaluates the posterior defer score

$$d_1(x, z) = \pi_\theta(\langle \text{defer} \rangle | x, z). \quad (10)$$

The final defer score is $d_0(x)$ for pre-generation candidates and $d_1(x, z)$ for attempted queries. Although produced at different stages, both scores are normalized action-token defer probabilities. Appendix F.4 reports the calibration of their merged scores, which is comparable to that of several routing baselines across settings. GenGate therefore applies a shared cutoff γ to the merged scores. Queries above γ are routed to the expert, while the rest are answered locally. For a pre-generation candidate below γ , no local attempt exists yet, so GenGate generates the local answer at submission time. For an attempted query below γ , GenGate directly submits the existing attempt z .

For cost accounting, *ExpertRate* denotes the fraction of queries routed to the expert, and *AttemptRate* denotes the fraction with a generated local answer, including posterior-scoring attempts and submission-time generations. We define

$$\text{Cost}_\alpha = \text{ExpertRate} + \alpha \cdot \text{AttemptRate}, \quad (11)$$

where α is the cost of one local attempt relative to one expert call. This metric separately charges expert calls and local attempts, enabling a common comparison with pre-generation and post-generation routers.

4 Experiments

4.1 Experimental Setup

Datasets and models. We evaluate on Code, Math, and MMLU. Code uses LeetCodeDataset, HumanEvalPlus, and MBPPPlus (Xia et al., 2025; Chen et al., 2021; Austin et al., 2021; Liu et al., 2023). Math uses Hendrycks MATH (Hendrycks et al., 2021b). MMLU uses the auxiliary-train supervision and official test split (Hendrycks et al., 2021a). We use Qwen2.5-Coder and Qwen2.5-Instruct as Qwen-family local models (Hui et al., 2024; Yang et al., 2024), and Meta-Llama-3-8B-Instruct as the second local-model family (Grattafiori et al., 2024). Fine-tuning uses LoRA adapters (Hu et al., 2022). All local models route to the same expert: OpenAI’s publicly available GPT-5.4 API model, accessed through the OpenAI API without any fine-tuning or task-specific adaptation. Exact sizes, evaluators, and training details are in Appendix E.

Baselines. We compare GenGate against six families of routing and confidence baselines, all sharing the same expert outputs. *Reference policies* (Always Self, Always Expert, Random Routing) bound the trivial regimes. *Linear routers* (Pre-Linear and Post-Linear) fit logistic-regression probes on the prompt-only (Cencerrado et al., 2026) or completed-attempt hidden state (Azaria and Mitchell, 2023). *Confidence scores* include Answer Probability (mean token log-probability) (Mahaout et al., 2024) and Self-REF, which fine-tunes the local model to emit verbalized confidence tokens after answering (Chuang et al., 2025). Self-REF is tuned over confidence-token class ratios on validation data. *MC Two-Stage Probe* mirrors GenGate’s sequential structure but realizes both decisions as logistic-regression probes trained on the same Monte Carlo targets, isolating the contribution of action-token fine-tuning over a probe-based realization of the same target. *External routers* include a separately fine-tuned RoBERTa Router on the prompt text (Ong et al., 2025) and a Prompt Router that elicits a verbalized yes/no confidence response from the local model (Tian et al., 2023;

Chuang et al., 2025). Implementation details for every baseline are in Appendix E.

4.2 Training and Evaluation

Training setup. For each training query x , we sample $K = 10$ attempts from the reference model π_{ref} and score each with the domain evaluator to obtain binary correctness labels $r_i \in \{0, 1\}$. The *posterior target* for each attempt is its correctness label r_i , used to supervise $\pi_{\theta}(\langle \text{self} \rangle \mid x, z_i)$. The *initial target* is the empirical success rate $\hat{t}_0(x) = \frac{1}{K} \sum_{i=1}^K r_i$, which estimates the value of attempting under the oracle posterior policy (Section 3). This soft target supervises $\pi_{\theta}(\langle \text{attempt} \rangle \mid x)$. We fine-tune with LoRA, with full hyperparameters in Appendix E.

Evaluation. We construct cost–accuracy frontiers by sweeping each method’s final routing threshold. For pre- and post-generation routers, we use the convention that larger scores indicate stronger preference for expert deferral. For GenGate and MC Two-Stage Probe, we fix the initial threshold $\tau_0 = 0.7$ as a conservative default, so that the prior gate skips generation only under high defer confidence. We then sweep the deployment cutoff γ over the merged defer scores to trace the corresponding cost–accuracy frontier.

In addition to the full cost–accuracy frontiers, we report **Route@95%** as an operating-point summary. Route@95% is defined as the minimum ExpertRate required for the routed system to reach 95% of expert-only accuracy on held-out evaluation data. At this operating point, we compute the joint deployment cost Cost_{α} (Eq. 11) using the corresponding ExpertRate and AttemptRate. We report Cost_{α} for $\alpha \in \{0.05, 0.1, 0.2\}$ as a sensitivity analysis over different relative costs of local generation. Additional metrics, including frontier AUC and defer-score ECE, are reported in Appendix F.

4.3 Main Results

GenGate achieves the lowest joint deployment cost across all six settings (two local models $\times$ three domains) at $\alpha = 0.1$, and remains best or second-best in all 18 comparisons across $\alpha \in \{0.05, 0.1, 0.2\}$. Table 1 reports Cost_{α} at the Route@95% operating point. The gains are largest for Qwen-Math and Qwen-MMLU, where GenGate reduces $\text{Cost}_{0.1}$ from the strongest baseline’s 55.54 to 48.23 and from 68.18 to 58.85, respectively. For LLaMA, GenGate is strongest on Code and Math, reducing

Table 1: Joint deployment cost at Route@95%. $\text{Cost}_\alpha = \text{ExpertRate} + \alpha \cdot \text{AttemptRate}$ is the joint cost at three attempt-cost ratios. ExpertRate is the minimum expert-call rate (%) to reach 95% of expert-only accuracy. For GenGate and MC Two-Stage Probe, $\tau_0 = 0.7$. **Bold/underline** mark the best/second-best value per column within each model block.

Method	Code			Math			MMLU		
	Cost. _{0.5}	Cost. _{1.0}	Cost. _{2.0}	Cost. _{0.5}	Cost. _{1.0}	Cost. _{2.0}	Cost. _{0.5}	Cost. _{1.0}	Cost. _{2.0}
Qwen local model									
Self-REF	<u>72.86</u>	<u>77.86</u>	87.86	52.40	57.40	67.40	68.54	73.54	83.54
Answer Prob	73.11	78.11	88.11	86.70	91.70	101.70	87.43	92.43	102.43
Post Linear	83.67	88.67	98.67	51.87	56.87	66.87	63.64	68.64	78.64
Prompt Router	81.28	86.28	96.28	84.58	89.58	99.58	94.30	99.30	109.30
MC Two-Stage Probe	81.03	83.49	88.40	<u>51.25</u>	<u>55.54</u>	64.12	<u>63.36</u>	<u>68.18</u>	77.82
Pre Linear	80.22	81.26	<u>83.34</u>	54.16	56.58	<u>61.40</u>	72.39	73.85	<u>76.75</u>
RoBERTa Router	90.79	91.28	92.24	69.14	70.77	74.02	79.77	80.84	82.97
GenGate	70.20	72.04	75.71	44.43	48.23	55.85	55.38	58.85	65.78
LLaMA local model									
Self-REF	84.85	89.85	99.85	90.18	95.18	105.18	<u>56.82</u>	<u>61.82</u>	<u>71.82</u>
Answer Prob	<u>79.23</u>	<u>84.23</u>	94.23	99.32	104.32	114.32	86.90	91.90	101.90
Post Linear	85.51	90.51	100.51	87.56	92.56	102.56	58.92	63.92	73.92
Prompt Router	97.86	102.86	112.86	98.74	103.74	113.74	84.77	89.77	99.77
MC Two-Stage Probe	83.15	84.77	<u>88.02</u>	<u>84.85</u>	87.20	91.91	58.84	63.64	73.25
Pre Linear	87.13	87.81	89.16	86.07	<u>86.80</u>	88.27	70.34	71.90	75.02
RoBERTa Router	90.31	90.82	91.84	89.47	90.03	91.14	78.92	80.03	82.25
GenGate	77.72	79.41	82.81	82.01	84.84	<u>90.51</u>	56.76	61.65	71.43

Cost_{0.1} to 79.41 and 84.84, respectively, showing that GenGate consistently reaches the target accuracy with lower joint deployment cost than both pre-generation and post-generation routers.

Figure 3 further compares the cost–accuracy tradeoff across operating points. GenGate consistently lies among the strongest methods, with its largest gains appearing at moderate-to-high joint-cost budgets. This pattern reflects the limitation of pre-generation and post-generation routing methods: post-generation routers such as Self-REF and Answer Prob generate a local attempt for every query, while pre-generation routers such as Pre Linear avoid some local generation but require more expert calls to reach the same accuracy. GenGate avoids both failure modes by pre-deferring queries that the prior already determines, paying local generation cost only when the posterior signal is needed. Despite reducing local generation, GenGate maintains competitive routing accuracy, achieving best or second-best Route@95% in five of six settings and coming within 0.6 points of the best in the remaining one (Table 4, Appendix F.2).

The cost decomposition in Figure 4 provides a high-level explanation of these gains: GenGate shifts a substantial portion of deferrals to the pre-generation stage, reducing local-generation cost while still retaining posterior scoring for queries that receive a local attempt. The corresponding

LLaMA decomposition is provided in Figure 12, and full branch-level routing statistics are visualized in Figure 13 of Appendix F.3. The full routing breakdown at $\tau_0 = 0.7$ further shows that GenGate’s savings come from meaningful pre-generation deferrals rather than indiscriminate attempt reduction: early deferral covers 23.9–66.1% of queries in five of six settings (with 2.2% on LLaMA-MMLU), early-deferred examples have 57.1–93.3% defer precision, and locally submitted examples retain 62.5–86.5% accuracy.

5 Analysis and Ablations

5.1 Contribution of Each Decision Stage

The initial threshold τ_0 controls how often GenGate pays for posterior evidence from a local attempt. We sweep τ_0 to isolate two gains: avoiding unnecessary local generation at the prior gate and retaining answer-dependent evidence at the posterior gate. For each τ_0 , we sweep the expert-call budget and record accuracy and attempt-rate curves. Figure 5 shows a representative Qwen-Math sweep, and Appendix C reports the full grid (Figure 8).

The endpoints correspond to the two standard routing regimes. With $\tau_0 = 1$, every query generates a local attempt before routing, yielding a posterior-only router that always pays generation cost. With $\tau_0 = 0$, every query is scored before

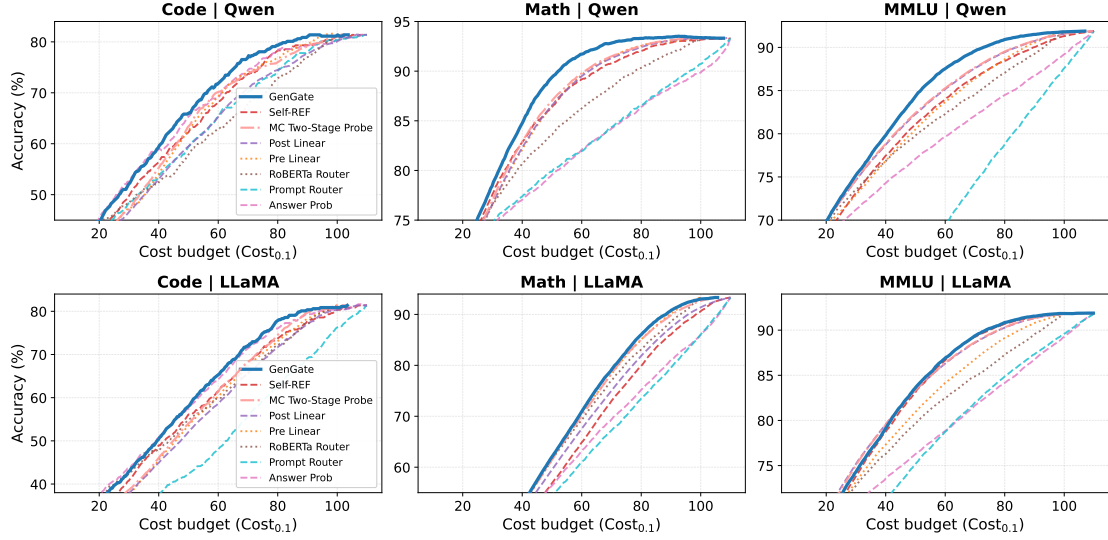


Figure 3: Accuracy under cost budget across all settings, with the accuracy axis zoomed to highlight differences among routing methods. Each curve sweeps the expert-call budget for one method. The x-axis is the cost budget $\text{Cost}_{0.1} = \text{ExpertRate} + 0.1 \cdot \text{AttemptRate}$. GenGate (blue, solid) achieves higher accuracy across broad mid-to-high cost-budget regions, where avoiding unnecessary local attempts compounds with reduced expert calls.

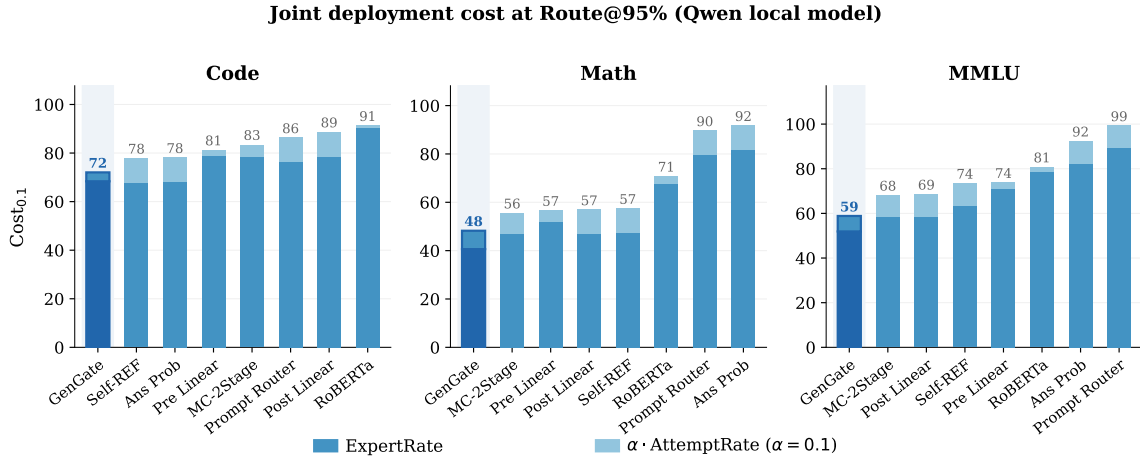


Figure 4: Cost decomposition at Route@95% for Qwen local models. Each bar decomposes $\text{Cost}_{0.1}$ into ExpertRate and $\alpha \cdot \text{AttemptRate}$ with $\alpha = 0.1$. Methods are sorted by total cost within each domain, and GenGate is highlighted. GenGate lowers joint deployment cost by reducing expert calls while avoiding unnecessary local attempts.

generation, yielding a prior-only router. Across settings, the posterior-only endpoint achieves higher raw accuracy because it conditions on the realized attempt, but the gap is usually modest. Intermediate thresholds therefore retain most posterior accuracy while reducing the attempt rate.

Under Cost_α , both endpoints are inefficient in different ways. The $\tau_0 = 1$ endpoint attempts every query, including cases already resolved by the prior signal, whereas $\tau_0 = 0$ discards posterior evidence when the prior is uncertain. Intermediate τ_0 avoids both extremes: the prior gate filters easy deferrals, and the posterior gate refines the rest. This ordering

is consistent across all six settings (Appendix C, Figure 8).

5.2 KL and Transfer

We ablate the KL regularization in the policy objective (Section 3), with full results in Appendix C.1, Figure 9. Removing KL ($\lambda_{\text{KL}}=0$) has a task-dependent effect. On Code, the routing curve shifts slightly downward with comparable accuracy at the 0% endpoint. On Math, the effect is severe: the no-KL curve rises roughly linearly from 0% at $B=0$, the signature of a model whose self-answers contribute nothing and whose accuracy scales with the

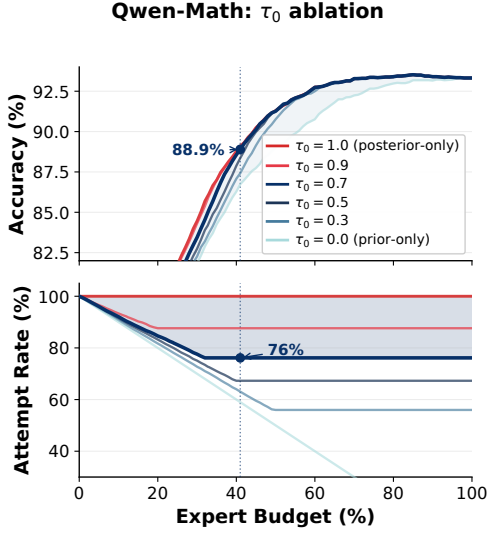


Figure 5: Effect of the initial defer threshold τ_0 on Qwen-Math (expert-budget sweep, one curve per fixed τ_0). **Top:** system accuracy vs. expert budget. The shaded band marks the accuracy gap between posterior-only routing ($\tau_0=1$) and prior-only routing ($\tau_0=0$). **Bottom:** local attempt rate vs. expert budget. The dark-blue curve is GenGate ($\tau_0=0.7$). The blue shaded area denotes its attempt-rate reduction relative to posterior-only ($\tau_0=1$) at matched expert budgets. The dotted vertical line marks the Route@95% operating point, where GenGate reaches 88.9% accuracy while attempting 76% of queries.

deferral fraction. This asymmetry likely reflects longer, less constrained Math generations and a larger Math training set, giving fine-tuning more steps to drift. The KL term is essential whenever generation is long or training is extended.

A Math-to-MMLU transfer experiment (Appendix D, Figure 10, Table 2) tests whether the learned routing behavior can be reused without re-collecting domain-specific rollout supervision. The transferred router achieves comparable AUC to in-domain training (Qwen: 82.793 vs. 83.300, LLaMA: 82.740 vs. 83.219), but requires more expert calls to reach Route@95% (Qwen: 54.95% vs. 51.91%, LLaMA: 56.43% vs. 51.87%). This suggests that GenGate learns partially reusable routing behavior, while domain-specific supervision remains beneficial under distribution shift.

5.3 Action-Token Fine-Tuning vs. Probing

MC Two-Stage Probe shares GenGate’s sequential structure and Monte Carlo targets but realizes both decisions as logistic-regression probes on frozen hidden states. The consistent gap (Table 1) isolates the contribution of action-token fine-tuning. These

results suggest that LoRA reshapes hidden states at decision boundaries to encode stronger routing signals (*representation adaptation*), joint training of s_0 and s_1 (Eq. 7) lets gradients from the initial gate benefit the posterior decision (*joint optimization*), and the fine-tuned model retains more correctness-indicative signal during generation because it anticipates the posterior routing step (*generation-aware encoding*). This advantage is consistent across all six settings, where GenGate reduces $\text{Cost}_{0.1}$ relative to MC Two-Stage Probe by 2.0–11.5 points at Route@95% for Qwen and LLaMA across Code, Math, and MMLU.

Taken together, these analyses clarify why generation-gated routing is useful. The threshold sweep shows that neither endpoint is ideal: always generating wastes local computation, while prompt-only routing discards answer-dependent evidence. The KL and probe ablations suggest that preserving the generator is important when applying routing supervision, and that the gains depend on action-token fine-tuning rather than simply using two decision points. These results support GenGate’s central design choice of training the local LLM to internalize cascaded routing decisions while preserving its generation capability when posterior evidence is needed.

6 Conclusion

We framed LLM routing as generation-gated cascaded routing, where local generation is an optional cost-bearing action rather than a mandatory step before deferral. GenGate instantiates this view by fine-tuning a single local LLM to internalize cascaded routing decisions while preserving its generation capability. The same model makes two action-token decisions: a prior gate before generation and a posterior gate after a bounded local attempt. These action probabilities decide when generation is worth paying for and when an expert call is needed. Across Code, Math, and MMLU with Qwen2.5 and LLaMA-3 local models, GenGate lowers joint deployment cost at the target accuracy by moving a substantial portion of deferrals before local generation. The ablations show why both decisions matter: prior deferral removes avoidable local attempts, posterior routing preserves answer-dependent evidence when generation is useful, and KL regularization keeps routing supervision from degrading the generator. Overall, GenGate uses local generation as optional routing evidence.

Limitations

Our evaluation focuses on automatically gradable single-turn tasks. Open-ended or multi-turn settings may require different correctness signals. GenGate also depends on rollout supervision from task evaluators or trusted labels. Its benefits are most pronounced when local generation has non-negligible cost and posterior evidence improves routing. When generation is cheap or answer-dependent evidence is weak, simpler routers may be sufficient. In deployment, GenGate may still submit incorrect local outputs, especially under distribution shift or imperfect rollout supervision. It should therefore not be relied upon as a safety mechanism in high-stakes applications without task-specific validation, monitoring, and an appropriate fallback. We leave serving effects, token pricing, and adaptive stopping to future work.

Acknowledgments

This research is supported by the Big Data Computing Center of Southeast University.

References

- Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. 2024. [AutoMix: Automatically mixing language models](#). In *Advances in Neural Information Processing Systems*, pages 131000–131034.
- Jean Vieira Alves, Diogo Leitão, Sérgio Jesus, Marco O. P. Sampaio, Javier Liébana, Pedro Saleiro, Mário A. T. Figueiredo, and Pedro Bizarro. 2024. [Cost-sensitive learning to defer to multiple experts with workload constraints](#). *Transactions on Machine Learning Research*.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. [Program synthesis with large language models](#). *Preprint*, arXiv:2108.07732.
- Amos Azaria and Tom Mitchell. 2023. [The internal state of an LLM knows when it’s lying](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976.
- Yuzhou Cao, Hussein Mozannar, Lei Feng, Hongxin Wei, and Bo An. 2023. [In defense of softmax parametrization for calibrated and consistent learning to defer](#). In *Advances in Neural Information Processing Systems*, pages 38485–38503.
- Iván Vicente Moreno Cencerrado, Arnau Padrés Masdemont, Anton Gonzalvez Hawthorne, David Demitri Africa, and Lorenzo Pacchiardi. 2026. [No answer needed: Predicting LLM answer accuracy from question-only linear probes](#). In *ICLR 2026 Workshop on Principled Design for Trustworthy AI - Interpretability, Robustness, and Safety across Modalities*.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. [Accelerating large language model decoding with speculative sampling](#). *Preprint*, arXiv:2302.01318.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2024a. [FrugalGPT: How to use large language models while reducing cost and improving performance](#). *Transactions on Machine Learning Research*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.
- Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. 2024b. [RouterDC: Query-based router by dual contrastive learning for assembling large language models](#). In *Advances in Neural Information Processing Systems*, pages 66305–66328.
- Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. 2025. [Learning to route LLMs with confidence tokens](#). In *Proceedings of the 42nd International Conference on Machine Learning*, pages 10859–10878.
- Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed H Awadallah. 2024. [Hybrid LLM: Cost-efficient and quality-aware query routing](#). In *International Conference on Learning Representations*, pages 41348–41366.
- Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. 2025. [BEST-Route: Adaptive LLM routing with test-time optimal compute](#). In *Proceedings of the 42nd International Conference on Machine Learning*, pages 13870–13884.
- Claudio Fanconi and Mihaela van der Schaar. 2025. [Cascaded language models for cost-effective human-AI decision-making](#). In *Advances in Neural Information Processing Systems*, pages 11559–11593.
- Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. [Detecting hallucinations in large language models using semantic entropy](#). *Nature*, 630(8017):625–630.

- Yonatan Geifman and Ran El-Yaniv. 2017. [Selective classification for deep neural networks](#). In *Advances in Neural Information Processing Systems*.
- Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koepl, Preslav Nakov, and Iryna Gurevych. 2024. [A survey of confidence estimation and calibration in 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)*, pages 6577–6595.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 540 others. 2024. [The Llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- 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](#). In *Agentic Markets Workshop at ICML 2024*.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayihang Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shenghaoran Quan, and 5 others. 2024. [Qwen2.5-Coder technical report](#). *Preprint*, arXiv:2409.12186.
- Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. [LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178.
- Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. 2023. [When does confidence-based cascade deferral suffice?](#) In *Advances in Neural Information Processing Systems*, pages 9891–9906.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. [Language models \(mostly\) know what they know](#). *Preprint*, arXiv:2207.05221.
- Reza Khanmohammadi, Erfan Miah, Mehrsa Mardikoraem, Simerjot Kaur, Ivan Brugere, Charese Smiley, Kundan S Thind, and Mohammad M. Ghassemi. 2025. [Calibrating LLM confidence by probing perturbed representation stability](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10448–10514.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. [Fast inference from transformers via speculative decoding](#). In *Proceedings of the 40th International Conference on Machine Learning*, pages 19274–19286.
- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. [Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation](#). In *Advances in Neural Information Processing Systems*, pages 21558–21572.
- Shuqi Liu, Yuzhou Cao, Qiaozhen Zhang, Lei Feng, and Bo An. 2024. [Mitigating underfitting in learning to defer with consistent losses](#). In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pages 4816–4824.
- 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)*, pages 1964–1974.
- David Madras, Toniann Pitassi, and Richard S. Zemel. 2018. [Predict responsibly: Improving fairness and accuracy by learning to defer](#). In *Advances in Neural Information Processing Systems*.
- Matéo Mahaut, Laura Aina, Paula Czarrowska, Momchil Hardalov, Thomas Müller, and Lluís Màrquez. 2024. [Factual confidence of LLMs: on reliability and robustness of current estimators](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4554–4570.
- Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. [SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017.

- Hussein Mozannar and David Sontag. 2020. [Consistent estimators for learning to defer to an expert](#). In *Proceedings of the 37th International Conference on Machine Learning*, pages 7076–7087.
- 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 *International Conference on Learning Representations*, pages 34433–34448.
- Soham Shah and Kumar Shridhar. 2025. [Select-then-Route : Taxonomy guided routing for LLMs](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 425–441.
- Hexiang Tan, Fei Sun, Sha Liu, Du Su, Qi Cao, Xin Chen, Jingang Wang, Xunliang Cai, Yuanzhuo Wang, Huawei Shen, and Xueqi Cheng. 2025. [Too consistent to detect: A study of self-consistent errors in LLMs](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4755–4765.
- Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. [Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442.
- 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.
- Rajeev Verma, Daniel Barrejon, and Eric Nalisnick. 2023. [Learning to defer to multiple experts: Consistent surrogate losses, confidence calibration, and conformal ensembles](#). In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 11415–11434.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-Consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. 2025. [Know your limits: A survey of abstention in large language models](#). *Transactions of the Association for Computational Linguistics*, 13:529–556.
- Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. 2025. [LeetCodeDataset: A temporal dataset for robust evaluation and efficient training of code LLMs](#). *Preprint*, arXiv:2504.14655.
- Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. [Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs](#). In *International Conference on Learning Representations*, pages 23650–23678.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2024. [Large language model cascades with mixture of thought representations for cost-efficient reasoning](#). In *International Conference on Learning Representations*, pages 21691–21728.
- Michael J. Zellinger, Rex Liu, and Matt Thomson. 2025. [Cost-saving LLM cascades with early abstention](#). *Preprint*, arXiv:2502.09054.
- Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. 2025a. [Reasoning models know when they’re right: Probing hidden states for self-verification](#). In *Second Conference on Language Modeling*.
- Zhenliang Zhang, Xinyu Hu, Huixuan Zhang, Junzhe Zhang, and Xiaojun Wan. 2025b. [ICR probe: Tracking hidden state dynamics for reliable hallucination detection in LLMs](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17986–18002.

A Preference-Based Sequential Decision Formulation

This section derives the Bellman advantage used in Section 3. The derivation uses utilities only to identify the supervision target for the initial gate. GenGate is trained with supervised action-token losses rather than reinforcement learning.

Preference formulation. We assume the utility ordering

$$U_{\text{self-correct}} > U_{\text{defer}} > U_{\text{self-wrong}}. \quad (12)$$

This states that a correct local submission is preferred to expert deferral, while expert deferral is preferred to submitting an incorrect local answer. Let c_A denote the cost of generating a bounded local attempt. The concrete value of c_A controls the deployment tradeoff, but the training target only needs to estimate whether the local attempt is likely to be correct.

Generic Bellman backup. For a one-step transition from state s under action a , the Bellman backup is

$$Q(s, a) = R(s, a) + \mathbb{E}_{s' \sim P(\cdot | s, a)}[V(s')], \quad (13)$$

where

$$V(s') = \max_{a'} Q(s', a'). \quad (14)$$

Terminal actions have value equal to their terminal utility. GenGate has two decision states. At $s_0 = x$, the model chooses between immediate deferral D and attempting local generation A . Immediate deferral is terminal:

$$Q_D(s_0) = U_{\text{defer}}. \quad (15)$$

Attempting pays the local generation cost and transitions to the posterior state $s_1 = (x, z)$ with $z \sim \pi_{\text{ref}}(\cdot | x)$:

$$Q_A(s_0) = -c_A + \mathbb{E}_{z \sim \pi_{\text{ref}}(\cdot | x)}[V_1(x, z)]. \quad (16)$$

At s_1 , the posterior value is the better of self-submission and deferral:

$$V_1(x, z) = \max\{Q_{\text{self}}(s_1), Q_{\text{defer}}(s_1)\}. \quad (17)$$

Posterior value. Let $r = r(x, z) \in \{0, 1\}$ indicate whether the local attempt is correct. The posterior action values are

$$Q_{\text{self}}(s_1) = \begin{cases} U_{\text{self-correct}} & \text{if } r = 1, \\ U_{\text{self-wrong}} & \text{if } r = 0, \end{cases} \quad (18)$$

and

$$Q_{\text{defer}}(s_1) = U_{\text{defer}}. \quad (19)$$

By the utility ordering, the optimal posterior action is self-submit when $r = 1$ and defer when $r = 0$. Thus

$$V_1(x, z) = \begin{cases} U_{\text{self-correct}} & \text{if } r = 1, \\ U_{\text{defer}} & \text{if } r = 0. \end{cases} \quad (20)$$

Taking expectation over local attempts gives

$$\begin{aligned} \mathbb{E}[V_1] &= \Pr[r = 1] \cdot U_{\text{self-correct}} \\ &\quad + \Pr[r = 0] \cdot U_{\text{defer}} \\ &= \Pr[r = 1] \cdot U_{\text{self-correct}} \\ &\quad + (1 - \Pr[r = 1]) \cdot U_{\text{defer}}. \end{aligned} \quad (21)$$

Initial advantage. Substituting this posterior value into the Bellman backup, the advantage of attempting over immediate deferral is

$$\begin{aligned} Q_A(s_0) - Q_D(s_0) &= \Pr[r = 1] \cdot U_{\text{self-correct}} \\ &\quad + (1 - \Pr[r = 1]) \cdot U_{\text{defer}} \\ &\quad - c_A - U_{\text{defer}} \\ &= \Pr[r = 1] \cdot U_{\text{self-correct}} \\ &\quad - \Pr[r = 1] \cdot U_{\text{defer}} - c_A \\ &= \Pr[r = 1] \cdot \Delta U - c_A, \end{aligned} \quad (22)$$

where $\Delta U = U_{\text{self-correct}} - U_{\text{defer}}$. Attempting is favorable when

$$\Pr[r = 1] > \frac{c_A}{\Delta U}. \quad (23)$$

Therefore, the initial decision should estimate the probability that sampling a bounded local attempt from $\pi_{\text{ref}}(\cdot | x)$ yields a correct answer. We approximate this probability with the empirical roll-out success rate

$$\hat{t}_0(x) = \frac{1}{N} \sum_{i=1}^N r_i \approx \Pr_{z \sim \pi_{\text{ref}}(\cdot | x)}[r(x, z) = 1]. \quad (24)$$

Training targets. The Bellman backup yields the supervised targets used in Section 3.2:

- **Initial decision:** $\hat{t}_0(x) \in [0, 1]$, the empirical probability that the local model can generate a correct answer. Queries with higher $\hat{t}_0(x)$ should be attempted.
- **Posterior decision:** $r_i \in \{0, 1\}$, the realized correctness of attempt z_i . Correct attempts should be self-submitted. Incorrect attempts should be deferred.

The utilities motivate the target but are not optimized directly. Training uses binary cross-entropy on action-token log-odds, and inference controls the attempt/defer tradeoff through the initial threshold τ_0 and deployment cutoff γ .

B Motivation Probe Analysis

This section expands the motivation study in Figure 1.

Per-ratio probes. The motivation experiment asks whether partial generations provide a reliable intermediate routing state between the prompt-only state s_0 and the completed-attempt state s_1 . Figure 6 extends the main-text probe study to all domains and both local-model families. The result is not that intermediate states contain no correctness signal; rather, their usefulness is inconsistent across tasks, models, and generation ratios. Some partial-generation points expose answer-dependent information, but they do not form a stable operating point that consistently improves over the boundary states. This supports using s_0 and s_1 as the two routing states instead of introducing a fixed mid-generation gate.

Mixed-ratio probe. Figure 7 tests a complementary question: whether the per-ratio pattern is caused by training separate probes at each ratio. We train a single linear probe on pooled hidden states from all ratios and evaluate it separately at each ratio. Pooling gives the classifier access to more training examples and a shared decision boundary across generation positions. Even under this setting, intermediate ratios do not become a consistently strong decision point. The mixed probe still performs reasonably across ratios, indicating that correctness-related information is present throughout generation, but the boundary states remain the most reliable choices for a simple cascaded routing design. Thus the probe analysis motivates GenGate’s two-stage design: decide from the prompt when prior evidence is sufficient, and acquire the full local attempt only when posterior evidence is worth its cost.

C Ablation Studies

This section provides the full ablation results supporting Section 5. Figure 8 extends the τ_0 sweep from the representative Qwen-Math case to all six model-domain settings.

C.1 KL Regularization Ablation

Training collects rollouts from the reference policy π_{ref} , but inference generates from the fine-tuned π_θ . Without an anchor, fine-tuning the action-token logits can pull generation-token logits along, degrading the local model’s self-answer quality and shifting the realized z away from the distribution under which the posterior decision $\pi_\theta(\langle \text{self} \rangle | s_1)$ was supervised. The KL term in Eq. 8 restricts this drift on generation tokens while leaving action-token logits unconstrained. We assess its contribution by re-training GenGate with $\lambda_{\text{KL}} = 0$ and comparing routing curves to the full objective.

The effect is task-dependent. On Code, removing KL produces a small consistent downward shift with comparable accuracy at the 0% endpoint. On Math, the effect is severe: the wo-KL curve rises roughly linearly from 0% at $B = 0$ to the always-defer accuracy at $B = 100\%$, the signature of a system whose self-answers contribute nothing and whose accuracy scales with the deferral fraction. Fine-tuning has destroyed the local model’s ability to self-answer Math queries. The asymmetry between Code and Math likely reflects both longer, less constrained Math generations and a larger Math training set, which gives fine-tuning more steps to drift. We expect Code to exhibit similar collapse under comparable training scale. The KL term is essential whenever generation is long or training is extended. Figure 9 shows the corresponding routing curves.

D Cross-Domain Transfer

A practical concern for routing methods is whether a trained router can be reused across tasks without re-collecting domain-specific rollout supervision. We test this by training GenGate on Math and applying it to MMLU without using MMLU rollout supervision for router training or calibration. MMLU labels are used only for offline evaluation of the resulting frontier of accuracy against budget. Transfer results are mixed: on Qwen, the Math-trained router requires more expert calls to reach the 95% target (54.95% vs. 51.91% for in-domain) but its AUC is comparable (82.793 vs. 83.300). On LLaMA, the transferred router shows comparable AUC (82.740 vs. 83.219) and uses more expert calls (56.43% vs. 51.87%). The residual gap is consistent with domain shift: Math and MMLU differ in difficulty distribution and answer format, so a router trained on Math does not fully

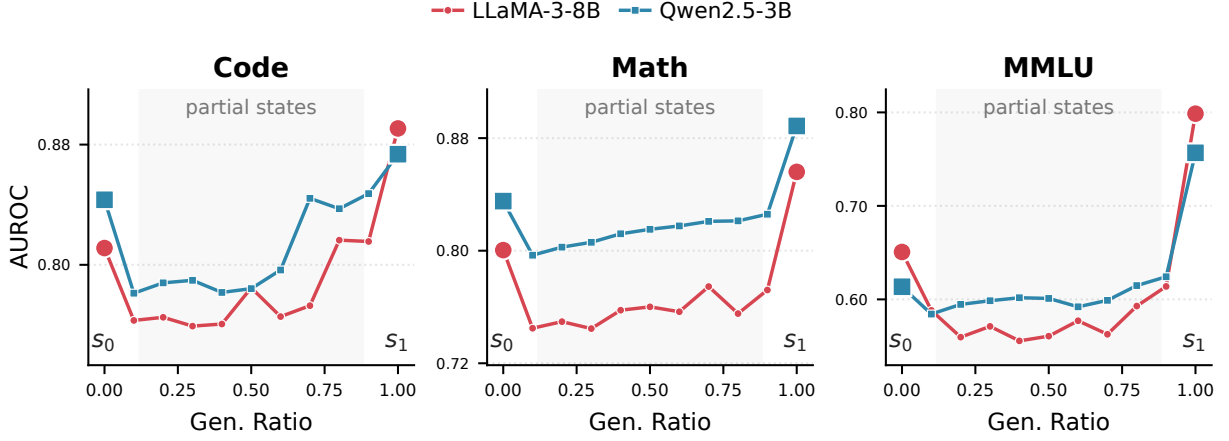


Figure 6: Per-ratio linear probes across Code, Math, and MMLU for Qwen and LLaMA local models. Each point trains and evaluates a linear probe at the same generation ratio. Intermediate partial-generation states contain correctness signal but do not provide a stable advantage over the boundary states (s_0 and s_1).

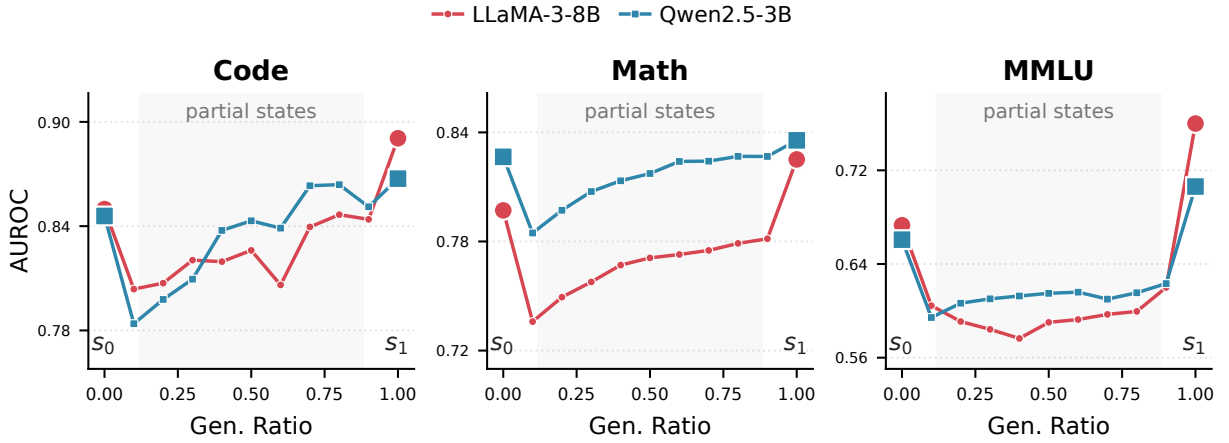


Figure 7: Mixed-ratio linear probes across Code, Math, and MMLU for Qwen and LLaMA local models. A single linear probe is trained on pooled hidden states from all ratios and then evaluated separately at each ratio. Intermediate states still do not form a consistently stronger routing point than the boundary states.

match in-domain training. The routing *behavior* partially transfers, although in-domain training remains stronger. Figure 10 shows the transfer frontiers, and Table 2 summarizes the Route@95% operating points.

Table 2: Cross-domain route rate required to reach 95% of expert-only accuracy on MMLU. Lower Route and ECE are better. Higher AUC is better.

Method	Route	ACC	AUC	ECE
Qwen local model				
MMLU $\rightarrow$ MMLU	51.91	87.30	83.300	0.2097
Math $\rightarrow$ MMLU	54.95	87.30	82.793	0.1732
LLaMA local model				
MMLU $\rightarrow$ MMLU	51.87	87.30	83.219	0.1952
Math $\rightarrow$ MMLU	56.43	87.30	82.740	0.2253

E Dataset, Training, and Baseline Details

Code availability. A reference implementation is available at <https://github.com/xsyhh/GenGate>. The repository includes prompt templates, data preprocessing, rollout annotation, routing-pair construction, and LoRA training scripts for the proposed method.

Computation infrastructure. Experiments are conducted on a GPU server with eight NVIDIA A100 GPUs, each with 80GB memory.

Dataset construction. For Code, the routing-supervision split is built from the LeetCodeDataset training branch together with MBPPPlus, and the held-out evaluation split is built from the LeetCodeDataset test branch together with HumanEvalPlus (Xia et al., 2025; Chen et al., 2021; Austin et al.,

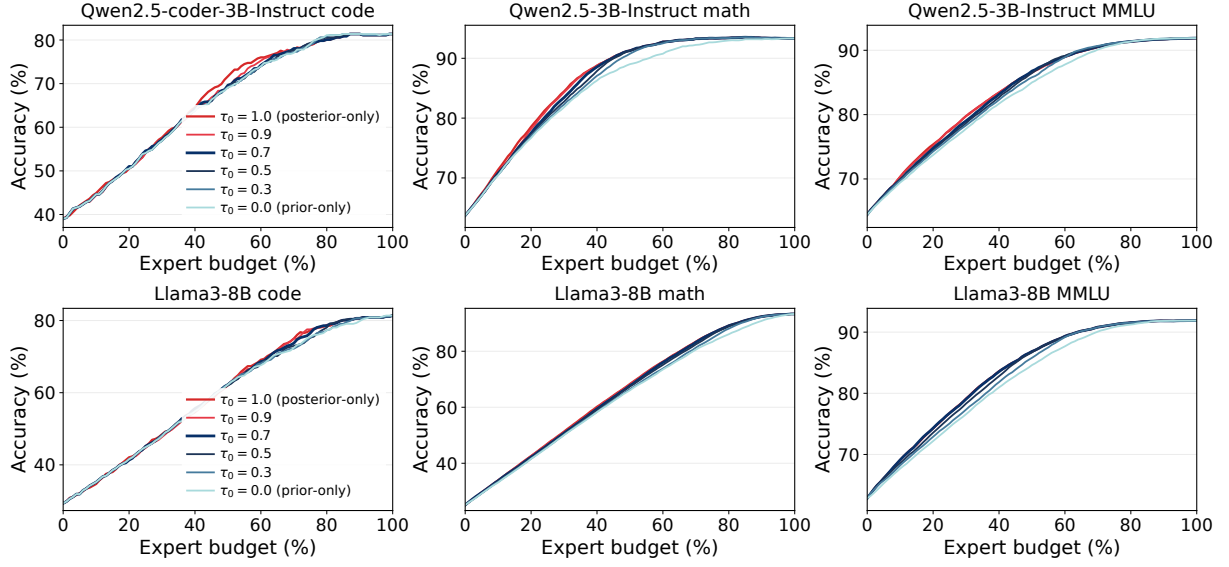


Figure 8: Full ablation of the initial defer threshold. Rows are Qwen and LLaMA. Columns are Code, Math, and MMLU. Each curve fixes τ_0 and sweeps the total expert budget.

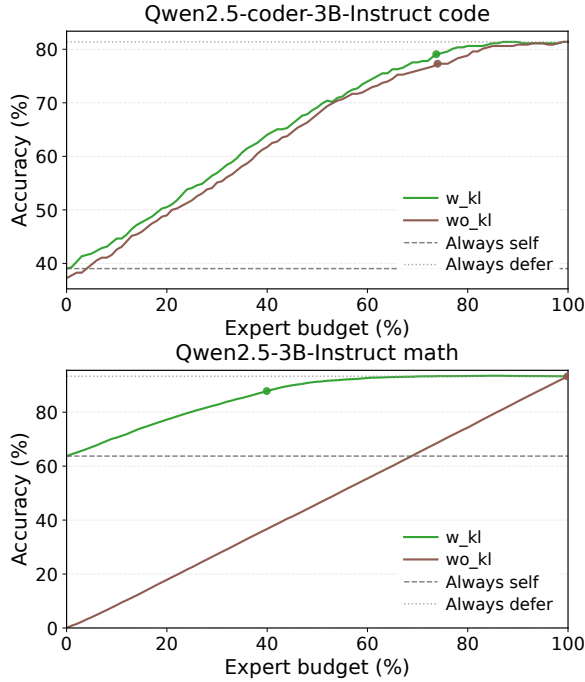


Figure 9: Effect of KL regularization on Qwen routing curves for Code (top) and Math (bottom). Green is the full GenGate objective. Brown removes the KL term.

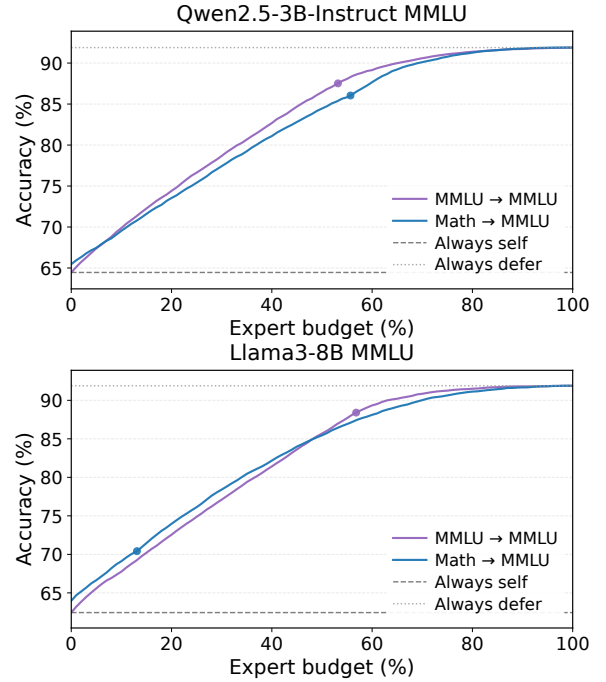


Figure 10: Cross-domain generalization from Math training to MMLU evaluation. The top panel uses Qwen and the bottom panel uses LLaMA.

2021; Liu et al., 2023). After preprocessing, this gives 2,376 LeetCode training problems, 341 MBPPPlus training problems, 228 held-out LeetCode problems, and 164 HumanEvalPlus problems. For Math, we use the official Hendrycks MATH train/test split with 7,500 training problems and 5,000 test problems. For MMLU, we use the processed MMLU auxiliary-train split for supervision,

the validation split for development checks, and the official test split for reported evaluation (Hendrycks et al., 2021b,a). These contain 99,842, 1,531, and 14,042 questions, respectively. All rollout labels used to fine-tune GenGate are produced only from the corresponding training split. Held-out score metadata is used only to compute the threshold sweeps and reported routing curves.

Model checkpoints. The Qwen local model is Qwen2.5-Coder-3B-Instruct for Code and Qwen2.5-3B-Instruct for Math and MMLU. The LLaMA local model is Meta-Llama-3-8B-Instruct in all three domains. These model families are described by Hui et al. (2024); Yang et al. (2024); Grattafiori et al. (2024). The expert model is OpenAI’s publicly available GPT-5.4 API model, accessed through the OpenAI API with no fine-tuning or task-specific adaptation. We query the expert once per held-out example and reuse the same response for every method in the same domain/model setting.

Expert API setup. Expert responses are obtained from the gpt-5.4 API. Each held-out example is sent as a single user message with no conversation history shared across examples. We use deterministic decoding. The task-specific prompt templates used for local and expert generation are listed below. Expert outputs are cached per query. Each example is queried exactly once, and the same expert response is reused across all routing methods within the same domain/model setting to ensure that any performance difference is attributable to the routing decision rather than to expert sampling variance.

Rollout annotation. For each training problem, we sample $K = 10$ bounded local attempts from the unfine-tuned local model. Sampling uses temperature 0.7 and $\text{top-}p = 0.95$. Each rollout is evaluated with the domain evaluator: executable unit tests for Code, final-answer matching for Math, and option-letter matching for MMLU. The posterior $\langle \text{self} \rangle / \langle \text{defer} \rangle$ label is the binary correctness label of the sampled attempt. The initial $\langle \text{attempt} \rangle / \langle \text{defer} \rangle$ target is the empirical success rate of the sampled attempts for the same problem.

Fine-tuning. GenGate fine-tunes the local model with LoRA adapters. We use LoRA rank 8, LoRA $\alpha = 16$, dropout 0.05, bfloat16 training, gradient checkpointing, and a cosine learning-rate schedule with peak learning rate 2×10^{-4} . All runs use an effective batch size of 32 examples. For LLaMA, this effective batch size is implemented with a smaller per-device batch size for the 8B model. Training runs for 2–3 epochs across settings. For the KL regularizer, the reference model is the corresponding local base model. We set $\lambda_{\text{KL}} = 0.08$ for Code and Math and $\lambda_{\text{KL}} = 0.32$ for MMLU.

Inference decoding. At inference time, local attempts are decoded greedily with temperature 0 and $\text{top-}p = 0.95$. All causal-LLM prompts are wrapped as a single user message with the corresponding Qwen or LLaMA chat template. GenGate uses a sequential action format: at each boundary, we score the candidate action markers, append the selected marker, and continue only when $\langle \text{attempt} \rangle$ is selected. After $\langle \text{attempt} \rangle$, the model generates a bounded solution and the posterior boundary scores $\langle \text{self} \rangle$ against $\langle \text{defer} \rangle$. Routing probabilities are computed from the normalized probabilities of the relevant action-token pair at each decision state.

Prompt and context formats. We do not rely on task-specific prompt engineering beyond enforcing output formats that the evaluators can parse. All causal-LLM prompts are wrapped as a single user message with the corresponding Qwen or LLaMA chat template. The code repository contains the exact prompt strings; here we summarize the prompt structure used to create rollouts, training pairs, and inference scores.

Rollout sampling. Rollouts used for GenGate supervision are sampled from answer-only prompts, without any routing or confidence tokens. The Code prompt gives the problem and starter code and asks the model to output only a Python fenced solution, with no explanations, tests, usage examples, or `__main__` block. The Math prompt asks for step-by-step reasoning and requires the answer to end with `Final answer:` followed immediately by the bare expression or value. The MMLU prompt similarly asks for step-by-step reasoning and requires `Final answer:` followed by a single option letter. These sampled attempts are then evaluated by the task evaluator and are not generated from the action-token prompt.

GenGate action contexts. For action-token training and inference, GenGate uses a sequential prompt that explicitly exposes two decision points. At the initial state s_0 , the user prompt contains the task instance and asks the model to choose either $\langle \text{attempt} \rangle$ or $\langle \text{defer} \rangle$ before producing any answer. If $\langle \text{attempt} \rangle$ is chosen, the same prompt specifies the bounded attempt format: Code must place the draft inside a Python fenced block, Math must end the reasoning with `Final answer:` and a bare value, and MMLU must end with `Final answer:` and one option letter. After the completed attempt, the prompt asks the model to choose $\langle \text{self} \rangle$ or

$\langle \text{defer} \rangle$. Thus the Code posterior context is the s_0 prompt followed by $\langle \text{attempt} \rangle$, the opening code fence, the sampled code, and the closing fence; for Math and MMLU it is the s_0 prompt followed by $\langle \text{attempt} \rangle$ and the sampled reasoning/answer text. Qwen-MMLU uses the rendered action strings $\langle \text{Attempt} \rangle$, $\langle \text{self} \rangle$, and $\langle \text{defer} \rangle$; the other GenGate runs use $\langle \text{Attempt} \rangle$, $\langle \text{self} \rangle$, and $\langle \text{defer} \rangle$.

Training pairs and inference scoring. The training data reuses exactly the same s_0 and s_1 contexts described above. For each query, the s_0 pair compares $\langle \text{attempt} \rangle$ against $\langle \text{defer} \rangle$ and uses the empirical rollout success rate $\hat{t}_0(x)$ as its soft target. For each realized rollout, the s_1 pair compares $\langle \text{self} \rangle$ against $\langle \text{defer} \rangle$ and uses the binary correctness label r_i . At inference time, no labels are attached: the model scores the same action pairs on the same contexts and follows the routing procedure in Section 3.3.

Baseline prompt formats. Answer-only local generation, cached expert calls, and Answer Probability use the same answer-only formats as rollout sampling: Python fenced code for Code, Final answer: plus a bare value for Math, and Final answer: plus one option letter for MMLU. Self-REF uses the same answer format but appends one learned confidence token, written in the paper as $\langle \text{CN} \rangle$ or $\langle \text{UN} \rangle$. For Code and Math, the token follows the generated answer; for MMLU, the implementation uses a compact multiple-choice answer format followed by the same confidence token. Prompt Router replaces the learned confidence token with a verbalized yes/no decision after the answer, and routes by rescaling that final decision boundary. Pre-Linear, Post-Linear, and MC Two-Stage Probe do not introduce additional natural-language prompts; they use hidden states extracted at the prompt-only state, completed-attempt state, or both. RoBERTa Router does not use a causal-LLM generation prompt and is trained on the raw problem text used by the baseline loader.

Baseline implementation details. For Self-REF, we follow the confidence-token annotation and training procedure of [Chuang et al. \(2025\)](#), using the two newly added confidence tokens $\langle \text{CN} \rangle$ and $\langle \text{UN} \rangle$. To stabilize training, gradient updates on the input and output embeddings are restricted to the two decision-token rows via a row-wise mask that zeros gradients on all other vocabulary entries. Correct local outputs are annotated with the posi-

tive token and incorrect outputs with the negative token. We add these two tokens to the vocabulary and fine-tune with LoRA (rank 8, $\alpha = 16$) applied to all Q-K-V layers, using batch size 32 and learning rate 1×10^{-4} . To preserve base model quality, we freeze all token embedding parameters except the two confidence-token rows, and similarly freeze the output projection except for these two tokens. We construct the augmented training set by balancing the two confidence-token classes. Following [Chuang et al. \(2025\)](#), we sweep the $\langle \text{CN} \rangle$: $\langle \text{UN} \rangle$ ratio over $\{1:1, 1:2, 1:3, 1:4\}$ and select per-setting on validation accuracy. The selected ratio is 1:1 in five of six settings and 1:3 for LLaMA-MMLU. Differences across ratios are small, within ~ 1 expert-call-equivalent unit per setting at the Route@95% operating point. Training runs for up to 8 epochs with early stopping based on validation perplexity. For Pre-Linear and Post-Linear, we use scikit-learn’s LogisticRegressionCV with L2 regularization, selecting the regularization strength via 5-fold cross-validation on the training rollouts. For RoBERTa Router, we fine-tune roberta-base on the prompt text with a linear classification head, using learning rate 2×10^{-5} , batch size 16, and 3 epochs. Prompt Router uses the domain-specific verbalized-confidence prompts listed above and routes by rescaling the final yes/no decision boundary.

F Supplementary Results

This section gives the full per-setting audit tables and result figures behind the main summaries. Fixed-budget tables report counterfactual threshold sweeps at matched expert-call budgets. Figure 11 shows the complete cost–accuracy frontiers, while Figure 12 provides the LLaMA counterpart to the Qwen cost decomposition in the main text.

F.1 Full Fixed-Budget Results

Table 3 presents the main fixed-budget comparison discussed in Section 4.3.

F.2 Route@95 Comparison

Table 4 reports the Route@95% metric (minimum expert-call rate to reach 95% of expert-only accuracy) for all methods. GenGate achieves best or second-best Route@95% in five of six settings and is within 0.6 points of the best in the remaining one, demonstrating that aggressive pre-deferral does not sacrifice routing accuracy.

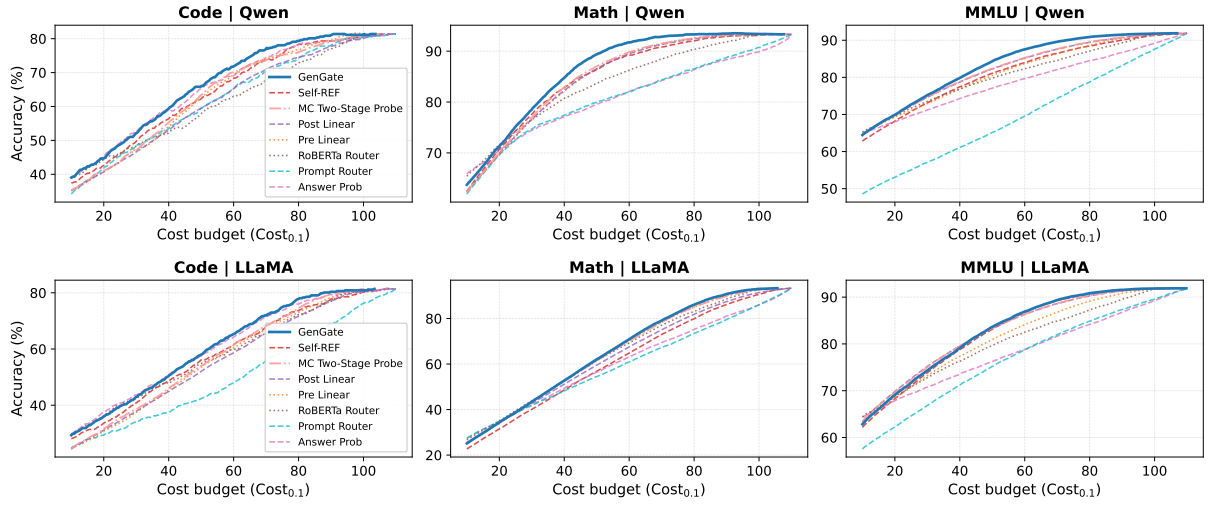


Figure 11: Complete accuracy under cost budget comparison across all settings. Each curve sweeps the expert-call budget for one method. The x-axis is the cost budget $\text{Cost}_{0.1} = \text{ExpertRate} + 0.1 \cdot \text{AttemptRate}$. GenGate (blue, solid) dominates or matches the Pareto frontier in all six panels.

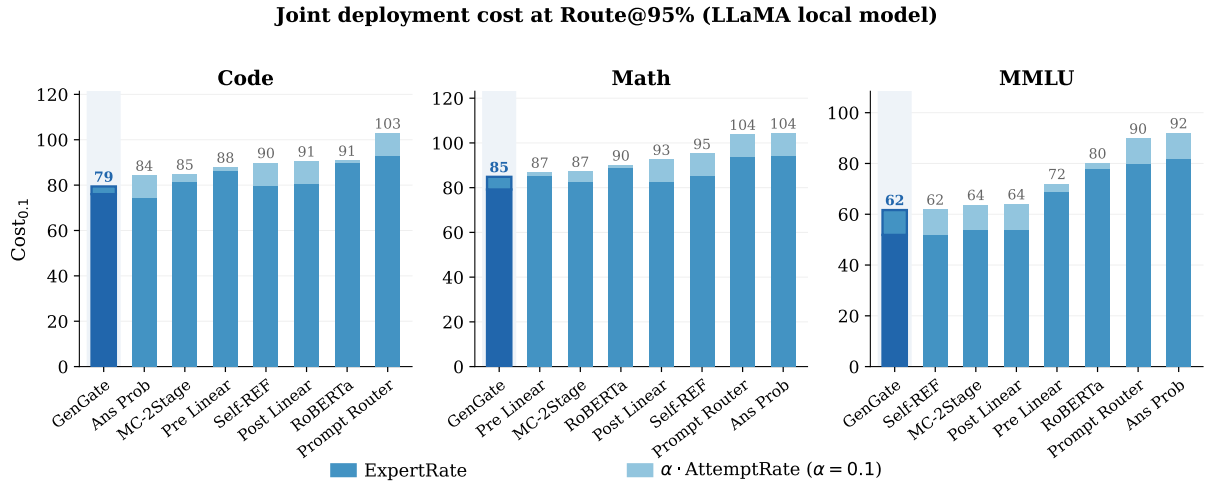


Figure 12: Cost decomposition at Route@95% for LLaMA local models. Each bar decomposes Cost_{0.1} into ExpertRate and $\alpha \cdot \text{AttemptRate}$ with $\alpha = 0.1$, matching Figure 4.

Table 3: System accuracy (%) at fixed expert-call budgets. Each column shows the accuracy achieved when exactly $B\%$ of queries are routed to the expert. This table evaluates routing quality at matched expert-call rates without charging local-attempt cost; the joint deployment cost including attempt rate is reported in Table 1.

Method	Code				Math				MMLU			
	20%	40%	60%	80%	20%	40%	60%	80%	20%	40%	60%	80%
Qwen local model												
Self-REF	49.74	62.24	73.72	79.34	77.60	86.40	90.80	92.88	73.13	81.11	86.58	90.04
Answer Prob	52.55	65.31	74.49	79.34	74.42	79.58	84.22	88.32	71.22	77.08	82.09	86.69
Post Linear	46.62	59.47	70.92	77.74	76.63	86.54	91.31	93.09	74.63	82.27	87.59	90.74
Prompt Router	48.47	59.44	70.15	78.06	74.72	79.88	84.36	88.78	57.07	65.16	74.29	83.09
MC Two-Stage Probe	46.30	60.08	72.03	77.74	76.20	86.54	91.31	93.09	74.50	82.31	87.61	90.75
Pre Linear	46.33	59.31	71.72	77.65	75.28	84.93	90.50	92.72	72.19	79.12	84.73	89.06
RoBERTa Router	47.70	54.85	64.54	73.72	75.62	82.40	87.22	90.60	72.84	78.68	83.27	87.56
GenGate	51.02	64.29	74.23	80.10	77.50	88.56	92.74	93.40	74.77	83.16	89.04	91.36
LLaMA local model												
Self-REF	40.82	56.12	67.86	77.81	40.10	56.86	73.00	86.06	73.81	83.11	89.25	91.46
Answer Prob	43.62	57.40	71.43	79.34	42.08	56.34	69.36	81.08	70.77	76.28	81.66	86.74
Post Linear	37.88	52.09	65.54	77.09	42.73	59.47	75.04	87.42	75.21	83.35	88.61	91.33
Prompt Router	33.93	42.35	55.87	69.13	41.86	54.50	67.22	79.54	66.82	75.17	81.99	87.32
MC Two-Stage Probe	37.76	51.43	64.64	76.38	42.03	58.61	74.49	87.55	74.82	83.24	88.59	91.33
Pre Linear	35.77	50.33	63.24	74.16	41.75	58.44	73.66	86.25	72.27	79.51	85.19	89.49
RoBERTa Router	41.58	51.53	62.24	72.45	42.20	57.88	72.50	84.40	71.98	78.39	83.41	87.73
GenGate	41.33	55.61	68.62	78.57	42.20	59.46	76.00	89.12	74.31	83.54	89.33	91.55

Table 4: Route@95% (%): minimum expert-call rate to reach 95% of expert-only accuracy. For GenGate and MC Two-Stage Probe, the initial threshold is fixed at $\tau_0 = 0.7$. Lower is better. **Bold/underline** mark best/second-best per column.

Method	Qwen			LLaMA		
	Code	Math	MMLU	Code	Math	MMLU
Self-REF	67.86	47.40	63.54	79.85	85.18	51.82
Answer Prob	68.11	81.70	82.43	74.23	94.32	81.90
Post Linear	78.67	<u>46.87</u>	58.64	80.51	82.56	53.92
Prompt Router	76.28	79.58	89.30	92.86	93.74	79.77
MC Two-Stage Probe	78.57	46.96	<u>58.54</u>	81.53	<u>82.49</u>	54.04
Pre Linear	79.18	51.75	70.94	86.45	85.33	68.77
RoBERTa Router	90.31	67.52	78.71	89.80	88.92	77.81
GenGate	68.37	40.62	51.91	<u>76.02</u>	79.18	<u>51.87</u>

F.3 Full Pre-Defer and Routing Breakdown Results

Figure 13 visualizes the routing breakdown at $\tau_0 = 0.7$. The segment widths report the fraction of queries handled by pre-generation deferral, post-generation deferral, and local self-submission; the labels inside the defer segments report defer precision, and the labels inside the self segments report local accuracy.

F.4 Full AUC and Calibration Results

Table 5 reports the full defer-score AUC and ECE metrics referenced in the evaluation protocol.

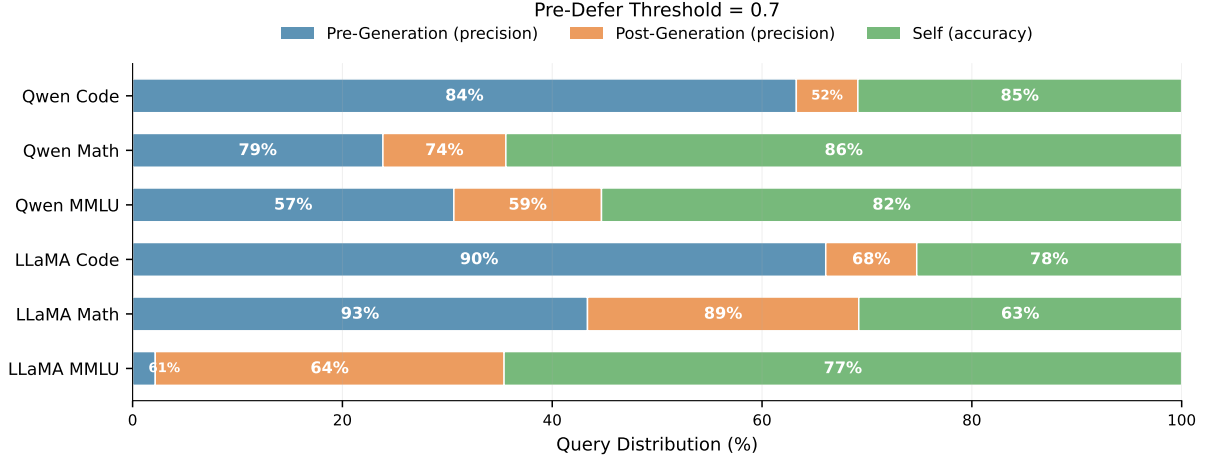


Figure 13: Routing breakdown at pre-defer threshold $\tau_0 = 0.7$. Each bar shows the query distribution across pre-generation deferral, post-generation deferral, and local self-submission. Numbers inside bars report precision for defer segments and accuracy for the self segment.

Table 5: Full routing AUC and defer-score ECE results. AUC measures defer-score ranking quality, while ECE measures defer-score calibration.

Method	Code		Math		MMLU	
	AUC	ECE	AUC	ECE	AUC	ECE
Qwen local model						
Self-REF	65.145	0.0822	85.437	0.0640	81.807	0.2304
MC Two-Stage Probe	62.789	0.0666	84.770	0.0344	82.505	0.0737
Post Linear	62.732	0.0974	85.350	0.0476	82.882	0.0838
Pre Linear	62.595	0.0767	84.432	0.0435	80.823	0.1645
RoBERTa Router	60.325	0.1600	83.218	0.2782	80.213	0.3389
Prompt Router	63.031	0.3546	81.219	0.2186	70.009	0.3818
Answer Prob	66.464	0.5227	81.174	0.2555	79.111	0.0842
GenGate	65.903	0.1076	86.154	0.0753	83.383	0.1872
LLaMA local model						
Self-REF	59.719	0.0667	63.045	0.2615	83.158	0.1555
MC Two-Stage Probe	57.302	0.0385	64.920	0.0413	82.873	0.0702
Post Linear	57.291	0.0276	65.089	0.0507	83.454	0.0729
Pre Linear	55.436	0.0424	64.115	0.0597	80.955	0.1402
RoBERTa Router	56.972	0.1584	63.487	0.2600	79.993	0.3561
Prompt Router	50.847	0.5584	60.682	0.1853	77.301	0.3190
Answer Prob	61.753	0.6233	61.750	0.5901	78.736	0.1739
GenGate	59.784	0.0947	65.095	0.1313	83.144	0.1393