Towards Fair And Comprehensive Evaluation Of Routers In Collaborative LLM Systems

Anonymous ACL submission

Abstract

Routing in collaborative LLM systems, directing queries between a local small model and a cloud large model, is not a single homogeneous problem. We argue that routing is better organized around recurring deployment scenarios than around a single pooled benchmark, and that the router itself should remain lightweight rather than being scaled up in pursuit of broader generalization. We introduce **RouterXBench**, a benchmark organized into three scenario-specific branches (**Knowledge**, **Math**, **Agent**), each independently asking whether a router trained with limited supervision remains reliable under realistic variation within that scenario. RouterXBench pairs a router-ability metric (AUROC) with scenario-alignment metrics (LPM, MPM, HCR) that separately capture how well operating behavior matches deployment-specific cost–quality constraints. As a reference implementation, we introduce XROUTER, a set of lightweight specialist routers, one per branch, built on the small model's own Q+A hidden states. We show that branch-specific routing signals are already accessible via the small model's own Q+A representations, validating that the scenario-oriented design is both effective and practical without heavy auxiliary models.

1 Introduction

Large Language Models (LLMs) achieve remarkable performance across diverse tasks (Zhao et al., 2023; Matarazzo and Torlone, 2025), but balancing cost and accuracy under varying deployment constraints remains a key challenge. Edge–cloud collaboration addresses this by processing simpler requests on local edge devices for low latency, privacy, and reduced cost, while escalating more difficult queries to stronger cloud models (Ding et al., 2024a; Zhang et al., 2025a; Barrak et al., 2025). This setup motivates *routing*: directing each query to the smallest model that suffices. The resulting trade-off can substantially reduce computational cost, but may also sacrifice accuracy (Kassem et al., 2025; Shafran et al., 2025; Lin et al., 2025).

However, this trade-off is not equally acceptable across applications. Different settings have different tolerances: some deployments are more cost-sensitive, while others require stronger reliability guarantees. Real systems also span heterogeneous task structures and interaction patterns, from knowledge assistance and document analysis to single-shot reasoning and multi-turn agentic workflows (Busch et al., 2025; Yu et al., 2025). A router that works under one configuration may fail dramatically under another.

This fragility makes evaluation itself a central problem. If routing behavior changes with deployment setting, then measuring router quality requires more than a single pooled comparison: the benchmark must reflect the design objective we actually want the router to serve. In collaborative systems, routing decisions are made inside concrete applications, and those applications differ in what they ask the router to optimize.

This evaluation challenge can make routing appear to require broad generalization. Yet many collaborative deployments remain anchored in fairly specific application settings, such as healthcare workflows, enterprise support, or tool-using assistants. Pushing the router toward the same breadth of accuracy and generalization as a foundation model would instead move routing toward model-scale judging, for example through an LLM judge or a reward-model-like scorer. That direction undermines the edge–cloud motivation for routing, since the decision itself begins to approach the cost of the escalation it is meant to control. We therefore adopt a different design principle: in collaborative LLM systems, routing is better organized around recurring application scenarios, with lightweight specialists matched to those settings. This raises the central question: *Can a router trained with*

limited supervision for one deployment scenario remain reliable under the realistic variation expected inside that same scenario?

We instantiate this principle with a lightweight *reference specialist router* built on the small model's own Q+A hidden states, and with **RouterXBench**, which organizes routing around three recurring collaborative settings: **Knowledge**, **Math**, and **Agent**. Evaluation also requires more than a single aggregate score. End-to-end cost–performance scores conflate whether the router correctly judges when the small model is sufficient with how the collaborative system behaves under deployment-specific cost–quality trade-offs. RouterXBench therefore isolates router ability with AUROC, evaluates scenario alignment separately, and further decomposes scenario alignment into low-cost, balanced, and accuracy-critical regimes through LPM, MPM, and HCR (Section 3.3).

In summary, our contributions are:

- **Scenario-oriented router benchmark.** We introduce RouterXBench, a benchmark better organized by deployment scenario than by a single pooled testbed, enabling controlled evaluation under realistic variation within knowledge, math, and agentic settings.
- **Diagnostic metric framework.** We pair router-ability metrics (AUROC) with scenario-alignment metrics (LPM, MPM, HCR) so that intrinsic routing quality and budget-specific operating behavior can be read off independently.
- **Reference specialist router and empirical findings.** We instantiate the scenario-oriented principle with XROUTER, a simple hidden-state specialist router, and show that branch-specific routing signals are already accessible via Q+A representations without heavy auxiliary models, confirming that a lightweight modular routing stack is both effective and practical.

2 Related Work

LLM Routing. Existing routers span training-free estimators based on relative performance or weak agreement (Zhao et al., 2024; Guha et al., 2024; Aggarwal et al., 2024); learned predictors over preferences, query–model embeddings, or capability tags (Ong et al., 2025; Chen et al., 2024c; Zhang et al., 2025b); sequential decision formulations such as bandit selection and token-level deferral (Li, 2025; She et al., 2025); and budget-aware designs that couple routing with test-time cost control (Ding et al., 2024b, 2025). Internal activations have also emerged as router inputs, via prefill hidden states, multi-layer probes, and LoRA-regularized probes (Varshney et al., 2026; Meyoyan and Del Corro, 2026; Zeng et al., 2026). Benchmarking efforts such as RouterEval (Huang et al., 2025) study scaling effects across pooled models.

LLM Collaboration. Collaboration strategies complement routing by coordinating multiple models or agents. Representative directions include speculative decoding, which accelerates inference using a draft–verifier pair (Chen et al., 2023; Cai et al., 2024; Li et al., 2024), and model cascades, which escalate queries through models of increasing capacity with calibrated deferral rules (Chen et al., 2024b; Gupta et al., 2024). More recent work explores multi-agent systems with specialized roles and coordination protocols (Wu et al., 2024; Li et al., 2023; Wang et al., 2025). Related to this, recent work studies routing within multi-turn agent trajectories (Zhang et al., 2026b,a; Fang et al., 2026), where routing decisions interact with cumulative cost and error compounding across turns.

LLM Uncertainty Estimation. Uncertainty estimation provides key signals for routing. Existing methods include information-based scores such as perplexity or entropy (Fomicheva et al., 2020; Duan et al., 2024; Fadeeva et al., 2024), consistency-based signals from agreement across generations (Kuhn et al., 2023a; Lin et al., 2024b; Qiu and Miikkulainen, 2024), and introspective probes using hidden states or attention patterns (Chen et al., 2024a; Sriramanan et al., 2024; Lin et al., 2024a). These methods can be integrated into routers to improve decision reliability, though many were originally developed outside the routing context.

3 RouterXBench

3.1 Problem Setup

We consider routing between two models in an *edge–cloud collaboration* setting: a small model $\mathcal{M}_{\text{small}}$ deployed locally on edge devices for low latency and privacy, and a large model $\mathcal{M}_{\text{large}}$ deployed in the cloud for higher accuracy at greater cost. Given a query $q \in \mathcal{Q}$, the router decides which model to invoke. Let $\delta_{\text{small}}(q), \delta_{\text{large}}(q) \in [0, 1]$ denote the performance of the two models on q. The router computes a score $s(q) \in \mathbb{R}$, and the

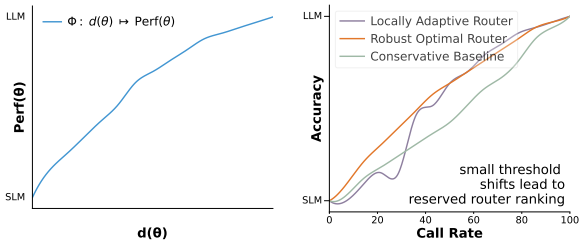

Figure 1: **Left:** Cost–performance mapping where $d(\theta)$ represents the call rate at threshold θ and $\mathrm{Perf}(\theta)$ denotes overall performance. By varying θ, this can be re-parameterized as call rate vs. performance (see §3.1). **Right:** An illustrative limitation of existing metrics.

decision is made by thresholding:

$$r(q;\theta) = \mathbf{1}\{s(q) \geq \theta\}, \tag{1}$$

where $r(q;\theta) = 1$ routes to the large model and $r(q;\theta) = 0$ uses the small model. The resulting system performance under threshold θ is

$$\delta(q;\theta) = (1 - r(q;\theta))\,\delta_{\text{small}}(q) + r(q;\theta)\,\delta_{\text{large}}(q). \tag{2}$$

For a given threshold, the large-model call rate is

$$d(\theta) \;=\; \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} r(q;\theta) \;\in [0,1], \tag{3}$$

and the corresponding overall performance is

$$\mathrm{Perf}(\theta) \;=\; \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \delta(q;\theta). \tag{4}$$

Varying the threshold θ traces out the *cost–performance curve*:

$$\Phi : \; d(\theta) \;\mapsto\; \mathrm{Perf}(\theta). \tag{5}$$

Since $d(\theta)$ is monotonic, we re-parameterize this curve as a continuous function $\Phi(x)$ of the call rate $x \in [0,1]$ via linear interpolation, which serves as the basis for our integral metrics.

3.2 Limitations of Current Metrics

As shown in Figure 1(left), the cost–performance curve introduced above provides a unified view of router behavior. Existing metrics can be seen as different ways of extracting information from this curve, which broadly fall into two categories.

Static Metrics. These methods evaluate routers at fixed thresholds or compress performance into few indicators. A common approach is the cost–accuracy trade-off: FrugalGPT (Chen et al., 2024b) fixes accuracy and reports cost savings, while HybridLLM (Ding et al., 2024b) fixes cost and measures accuracy drop. Others use single or composite indicators. TO-Router (Stripelis et al., 2024) reports total inference cost, throughput, semantic similarity, and negative log-likelihood. AutoMix (Aggarwal et al., 2024) uses Incremental Benefit per Cost, normalizing accuracy improvement by cost into a single score.

Limitation. While static metrics are simple and interpretable, they provide only a fragmented view of router behavior. As illustrated in Figure 1 (right), router rankings can be highly sensitive to threshold choice: within the call-rate range 20% to 40% , even minor shifts can lead to opposite conclusions about the Locally Adaptive Router, indicating that static evaluations may capture incidental fluctuations rather than a router's consistent behavior.

Curve-based Metrics. These methods integrate performance over the entire cost–performance curve to avoid thresholds. Examples include the AUC (area under the accuracy–cost curve) (Ramírez et al., 2024), Average Improvement in Quality (Hu et al., 2024), and Average Performance Gap Recovered (Ong et al., 2025). By summarizing global trends, these metrics provide threshold-independent evaluations of the trade-off surface.

Limitation. Aggregation, however, is scenario-blind. The Figure 1(right) also shows the limitation. Locally Adaptive Router performs poorly in low call-rate regions, but AUC scores conceal this difference and limit interpretability.

More fundamentally, cost–accuracy metrics entangle two factors: *router ability*, referring to the correctness of judgments relative to the small model's capacity, and *scenario alignment*, concerning the leverage of the large model's performance. Since end-to-end accuracy at a given cost reflects both, high scores may stem from the large model's strength rather than the router's skill, preventing faithful assessment of intrinsic routing capability.

3.3 Routing Metrics

To address this conflation, we propose two complementary metric perspectives (Figure 2) that independently evaluate distinct dimensions of routing performance. AUROC captures intrinsic discriminative ability without considering deployment costs. LPM, HCR, and MPM assess scenario alignment by quantifying how well routing matches specific cost–quality constraints.

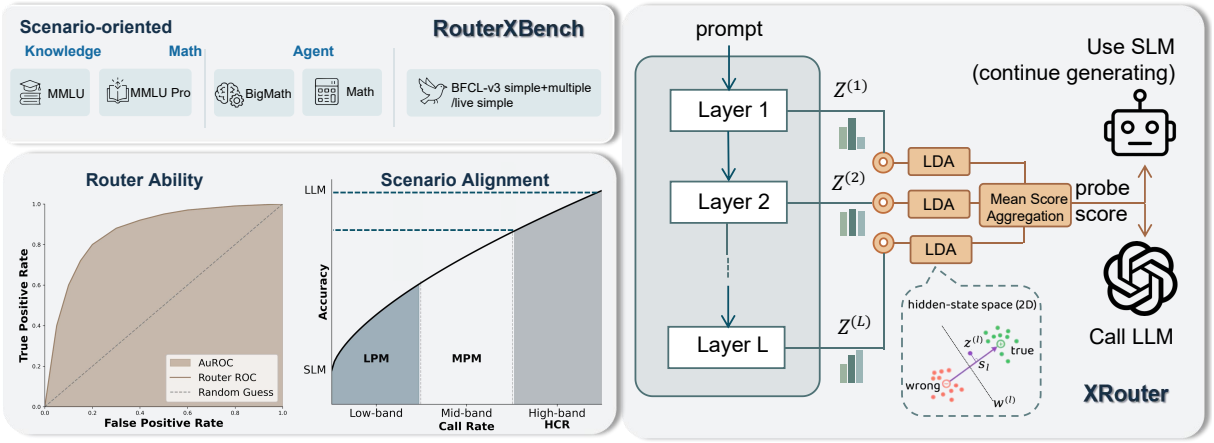

Figure 2: Overall collaboration architecture and evaluation protocol. Router ability is quantified using AUROC, measuring the router's accuracy in predicting whether the SLM can answer correctly. Scenario alignment is evaluated across three call-rate regimes: low band (Low-band Performance Mean, LPM), mid band (Mid-band Performance Mean, MPM), and high band (High-band Call-Rate, HCR).

1. Router Ability. Since the router's primary role is to decide which model to invoke, end-to-end system accuracy may blur its individual contribution. To isolate the router's discriminative power from the large model's capabilities, we define ground truth labels based on the small model's performance. Varying the decision threshold traces an ROC curve, and the area under this curve (AUROC) provides a threshold-independent measure of discriminative ability. Unlike cost-accuracy metrics, it focuses solely on the router's decision quality, and by aggregating over all thresholds, it avoids sensitivity to local fluctuations or opportunistic peaks.

2. Scenario Alignment. Routers with similar intrinsic ability can behave differently under deployment constraints. To reflect such differences, we partition the cost–performance curve into three regions: (i) low call-rate for budget-sensitive use, (ii) high accuracy for safety-critical domains, and (iii) a middle band for balanced deployment. For each region, we define a normalized mean metric: LPM, HCR, and MPM. As illustrated in Figure 2. The boundaries d_1, ρ_1, ρ_2 are practitioner-chosen, so the same router can receive different scenario alignment scores under different deployment targets.

Low-band Performance Mean (LPM). For strict budget scenarios, let $d_1 \in (0, 1]$ denote the maximum allowable call rate. The average performance in this region is defined as:

$$\mathrm{LPM} = \frac{1}{d_1} \int_0^{d_1} \Phi(x)\, dx. \tag{6}$$

High-band Call Rate (HCR). For accuracy-critical applications, we target a specific Relative Performance (RP) range. Given an RP interval $[\rho_1, \rho_2]$, we map these to absolute performance thresholds $[\tau_1, \tau_2]$ via:

$$\tau_i = \mathrm{Perf}_S + \rho_i(\mathrm{Perf}_L - \mathrm{Perf}_S), \quad i \in \{1, 2\}. \tag{7}$$

We then identify the *feasible call-rate set* $\mathcal{D}$ where the router's performance curve $\Phi(x)$ falls within this absolute band:

$$\mathcal{D} = \{x \in [0, 1] : \tau_1 \leq \Phi(x) \leq \tau_2\}. \tag{8}$$

The HCR metric computes the complement of the average call rate within this feasible set:

$$\mathrm{HCR} = 1 - \frac{1}{|\mathcal{D}|} \int_{x \in \mathcal{D}} x\, dx. \tag{9}$$

A higher HCR indicates the router maintains high accuracy while relying more on the small model.

Mid-band Performance Mean (MPM). This metric evaluates the trade-off efficiency in the transition region between the strict budget constraint (d_1) and the accuracy-critical zone. Let d_2 be the minimum call rate required to satisfy the high-accuracy threshold τ_1:

$$d_2 = \min\{x \in [0, 1] : \Phi(x) \geq \tau_1\}. \tag{10}$$

The mid-band interval is defined as $(d_1, d_2]$. Provided that a valid transition region exists, the mean performance is:

$$\mathrm{MPM} = \frac{1}{d_2 - d_1} \int_{d_1}^{d_2} \Phi(x)\, dx. \tag{11}$$

3.4 Benchmark Setup

Datasets. RouterXBench covers three deployment scenarios: Knowledge (MMLU, MMLU-Pro), Math (Big-Math, MATH), and Agent (BFCL-v3 function-calling tasks). Each branch trains on one split and evaluates on held-out data within the same scenario. Full dataset construction, split sizes, and evaluation details are in Appendix D.

4 XRouter

RouterXBench requires a reference router that is specialist (one per branch), lightweight, and cheap enough to stay well below the cost of the escalation it governs. Existing routers rely on output-based signals (Aggarwal et al., 2024; Zhang et al., 2025a) or external embeddings (Feng et al., 2025), which either suffer from calibration issues (Guo et al., 2017) or introduce a heavy auxiliary model. We instead route on **internal hidden states** from the small model: they encode task difficulty before the final answer is produced (Dong et al., 2025) and support compact linear discriminants without extra tokens. XROUTER instantiates this design with layerwise linear discriminants over mean-pooled Q+A hidden states, with one specialist probe trained per branch. Non-training baselines (SelfAsk, SemanticEntropy, ConfidenceMargin, Entropy, MaxLogits) do not depend on router training data, so we reuse their scores across the branch layout. For XROUTER, the MMLU branch is reported directly by MMLU-test plus the four MMLU-Pro subcategories: STEM, Humanities, Social Science, and Others. The Math branch reports in-domain Big-Math test performance and shifted transfer to MATH. Trainable baselines and hidden-state routers are run under the same branch-specific specialist setting.

Layerwise Mean-Discriminant Scoring. As shown in Figure 2, we first extract sentence-level representations by mean pooling over token-wise hidden states at each layer l:

$$z^{(l)} = \frac{1}{T}\sum_{t=1}^{T} h_t^{(l)}. \tag{12}$$

For every layer, we estimate class-conditional centroids from the training set:

$$\mu_+^{(l)} = \mathbb{E}[z^{(l)} \mid y=1], \quad \mu_-^{(l)} = \mathbb{E}[z^{(l)} \mid y=0], \tag{13}$$

where $y = 1$ denotes that the small model answers correctly. The layerwise discriminant direction is:

$$w^{(l)} = \Sigma_l^{-1}(\mu_+^{(l)} - \mu_-^{(l)}), \tag{14}$$

where Σ_l is a shrinkage covariance estimate. At inference time, each layer produces a scalar score:

$$s_l(x) = {w^{(l)}}^\top z^{(l)}(x), \tag{15}$$

and XROUTER averages calibrated layer scores to obtain the final routing score. This is the `pl_massmean_lda` implementation used in our experiments. It has no learned neural parameters beyond class statistics, is cheap to fit with small branch-specific training sets, and directly uses the small model's Q+A hidden states.

Specialist Training. We train one probe per branch: Knowledge and Math each use 1K training examples, and Agent uses 600 BFCL-v3 simple+multiple examples.

5 Experiments

5.1 Experiments setup

Benchmarks. We evaluate routers on the three RouterXBench branches introduced in Section 3.4: Knowledge, Math, and Agent. In the current instantiation, the Knowledge branch is built from MMLU/MMLU-Pro (Wang et al., 2024), the Math branch from Big-Math/MATH (Albalak et al., 2025; Hendrycks et al., 2021b), and the Agent branch from the Berkeley Function Calling Leaderboard (BFCL) function-calling tasks (Patil et al., 2025). Detailed data preparation, sample counts, and branch-specific evaluation protocols are provided in Appendix D.

For model selection, we use *GPT-5* as the large model and *Qwen3-8B* as the small model for evaluating router performance.

Baselines. We compare our hidden-state approach against three alternative signal modalities: **(1) Verbose-based.** Routers that depend on auxiliary generations, such as self-evaluation (Kadavath et al., 2022; Ding et al., 2025) or semantic entropy (Kuhn et al., 2023b; Zhang et al., 2025a), which are informative but incur prompt sensitivity. **(2) Logit-based.** Routers that only use the final-layer logits, such as entropy (Su et al., 2025), margin (Ramírez et al., 2024). These are efficient but brittle under scenario variation. **(3) Embedding-based.** These routers use semantic representations from

Table 1: Router ability (AUROC) under the RouterXBench setting. Knowledge and Math use 1K training examples; Agent uses BFCL-v3 simple+multiple (600 examples) and evaluates on BFCL-v3 live_simple.

Method	Knowledge					Math		Agent
	MMLU-test	STEM	Humanities	Social Sci.	Others	Big-Math	MATH	BFCL live_simple
SelfAsk	53.75	53.74	55.86	56.06	50.91	47.20	49.29	50.40
SemanticEntropy	53.93	56.27	51.72	52.90	53.95	55.81	55.25	50.36
ConfidenceMargin	46.56	54.42	46.97	54.37	49.52	56.18	50.05	39.60
Entropy	49.26	49.70	52.36	48.54	49.23	51.41	55.30	37.32
MaxLogits	43.82	50.03	50.53	41.14	46.43	47.39	47.00	50.00
EmbeddingMLP	70.09	63.78	57.12	48.80	48.87	64.65	67.95	65.07
HybridLLM	70.22	61.57	57.89	53.60	47.92	60.33	75.12	72.79
XROUTER	**79.54**	**82.86**	**64.66**	**73.45**	**68.35**	**72.49**	**79.03**	**80.43**

pretrained encoders or trainable text routers, including EmbeddingMLP (Feng et al., 2025) and HybridLLM (Ding et al., 2024b). With comparable classifier sizes, this enables direct comparison of different routing signals. The first two groups are non-training baselines and apply directly across branches.

Training Setup. For hidden-state methods, we feed the concatenation of the instruction and the small model's generated response back into Qwen3-8B and mean-pool token representations at every layer. Each specialist router is trained with a small branch-specific training split. For the MMLU branch, training uses 1K MMLU training examples and evaluation uses a fixed 2K MMLU-test subset plus the full MMLU-Pro evaluation set. For the Math branch, training uses 1K Big-Math examples and evaluation reports both Big-Math held-out test performance and transfer to MATH. For the Agent branch, training uses the BFCL-v3 simple+multiple split (600 examples) and evaluation reports AUROC on BFCL-v3 live_simple (258 examples).

5.2 Main Results

Router Ability. Table 1 reports AUROC results. XROUTER obtains the best router ability across all three branches. It reaches 79.54 AUROC on MMLU, 79.03 on MATH, and 80.43 on BFCL live_simple, outperforming both embedding-based baselines, EmbeddingMLP and HybridLLM. These results show that Q+A hidden states provide a stronger routing signal than external text embeddings or output-level uncertainty scores.

Scenario Alignment. Table 2 reports LPM, MPM, and HCR. XROUTER wins most cells and dominates every Knowledge column across the three metrics. The remaining gaps fall on Math and BFCL live_simple, where the small and large models behave similarly (both near-saturated on Math; both weak on BFCL, 51.2% vs. 66.7%): routing has less leverage, and weaker routers occasionally match or exceed XROUTER despite lower AUROC. On BFCL specifically, even when XROUTER correctly flags a small-model failure the large model often fails too, flattening the curve and absorbing the AUROC advantage.

6 Analysis

Answer-Conditioned Hidden States Help. Table 3 shows that the query alone is useful but insufficient for routing. Query-only hidden states provide a reasonable prompt-level signal, while adding the small model's own answer raises AUROC to 79.54 and 75.84. This gain comes from the model's answer-conditioned behavior rather than from extra text: shuffled or wrong answers sharply reduce performance, and a gold answer still underperforms the small-model answer. These results support our design choice to route on the small model's Q+A hidden states, since the generated answer exposes reliability signals that are not available from the query alone.

Task-Specific Layer Directions Reveal the Setting. Figure 3 removes the final cross-layer averaging step and evaluates the LDA score from each layer separately. All three branches show above-random routing ability across many layers, indicating that reliability signals are not confined to the final layer. However, the useful layers differ by scenario: Knowledge remains stronger in later layers, Math is most separable before the final generation layers, and Agent shows several local peaks. This layerwise heterogeneity explains why final-layer-only routing can be brittle, and supports both our layerwise aggregation design and the scenario-oriented evaluation in RouterXBench.

Table 2: Scenario alignment metrics (LPM, MPM, HCR) under the RouterXBench setting. Knowledge, Math, and Agent columns are reported under the same branch-specific evaluation protocol.

Method	Knowledge					Math		Agent
	MMLU-test	STEM	Humanities	Social Sci.	Others	Big-Math	MATH	BFCL live_simple
LPM								
SelfAsk	76.63	69.92	43.20	71.85	64.06	83.93	89.21	51.23
SemanticEntropy	76.21	69.95	43.26	72.13	63.99	84.25	**89.85**	51.52
ConfidenceMargin	76.28	69.77	42.85	71.79	64.00	83.71	88.25	51.80
Entropy	76.01	69.71	42.85	71.64	64.11	83.55	88.20	51.02
MaxLogits	76.20	69.83	42.71	71.61	63.78	83.57	89.19	50.99
EmbeddingMLP	75.40	69.87	42.99	71.92	63.76	94.38	84.55	**52.92**
HybridLLM	75.35	70.29	43.15	72.34	64.14	94.44	85.48	52.57
XROUTER	**77.96**	**71.06**	**43.61**	**73.05**	**65.31**	**94.68**	85.85	52.22
MPM								
SelfAsk	83.30	75.32	55.81	77.74	71.05	94.37	94.61	57.84
SemanticEntropy	83.09	75.49	55.07	77.88	71.74	94.38	**94.63**	56.64
ConfidenceMargin	82.35	74.84	55.17	77.89	71.12	94.38	94.51	56.43
Entropy	82.64	74.90	54.99	77.42	71.05	94.29	94.50	55.26
MaxLogits	82.56	74.86	54.86	77.50	70.99	94.31	94.44	57.67
EmbeddingMLP	82.01	75.14	54.51	77.37	71.17	94.53	90.53	58.60
HybridLLM	82.08	75.66	55.59	78.15	71.20	**94.99**	91.41	**58.80**
XROUTER	**84.48**	**76.36**	**56.28**	**79.07**	**73.02**	94.75	91.35	57.52
HCR								
SelfAsk	27.50	18.50	21.50	23.50	20.50	56.30	17.50	13.50
SemanticEntropy	24.50	29.00	21.00	33.00	25.00	14.00	20.00	20.50
ConfidenceMargin	14.00	16.50	19.00	20.00	19.00	15.00	16.00	16.30
Entropy	13.50	16.00	19.00	20.00	20.00	22.44	15.50	14.50
MaxLogits	22.00	24.00	21.00	20.00	17.50	14.50	18.00	13.50
EmbeddingMLP	17.00	24.50	22.00	18.00	18.00	57.14	41.50	54.30
HybridLLM	17.50	26.50	22.50	21.00	20.00	**84.50**	50.00	**55.00**
XROUTER	**54.50**	**52.50**	**27.50**	**45.00**	**33.00**	51.60	**56.00**	48.50

Table 3: Effect of hidden-state input context on router ability (AUROC). All variants use the same XROUTER scoring pipeline.

Input context	MMLU-test	MMLU-Pro
Query only	72.49	63.70
Answer only	76.82	74.41
Query + shuffled answer	67.23	59.03
Query + wrong answer	65.47	57.87
Query + gold answer	75.13	65.12
Query + small-model answer	**79.54**	**75.84**

Table 4: Specialist versus mixed training for trainable routers. Values are AUROC.

Method	Setting	MMLU-test	MATH	BFCL live_simple
EmbeddingMLP	Specialist	70.09	67.95	65.07
EmbeddingMLP	Mixed	69.70	68.52	55.98
HybridLLM	Specialist	70.22	75.12	72.79
HybridLLM	Mixed	70.44	72.47	73.18
XROUTER	Specialist	**79.54**	79.03	**80.43**
XROUTER	Mixed	74.36	**87.52**	75.47

Mixed Training Does Not Replace Specialist Routing. Table 4 compares branch-specific specialist training with a single mixed router trained across all branches. Mixed training does not provide a stable replacement for specialist routing: it can improve one branch, such as MATH for XROUTER, but often degrades others, especially MMLU-test and BFCL live_simple. This uneven trade-off supports the scenario-oriented view of RouterXBench: routing signals should be evaluated per deployment setting, and a single pooled router can hide branch-specific failures.

Lightweight Specialist Routing Is Practical. The previous analyses establish that branch-specific routing signals are real and structurally different across Knowledge, Math, and Agent. We now ask a systems question: can these specialist signals be exploited with routers that are cheap to train, easy to dispatch, and lightweight enough for deployment?

Small Specialist Training Is Sufficient. Appendix Figure 4 shows train-size scaling for XROUTER on MMLU-test. Performance improves

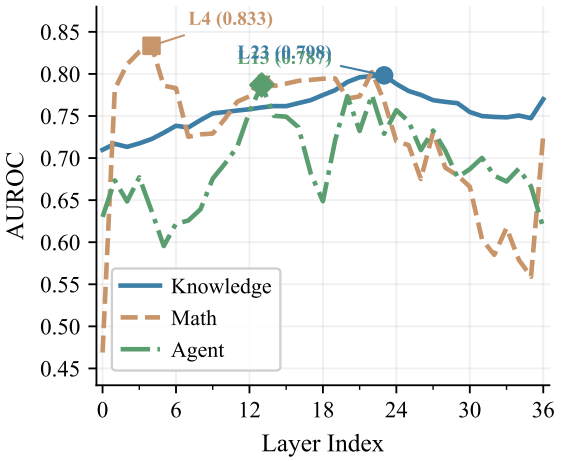

Figure 3: Per-layer XROUTER AUROC without cross-layer averaging across all three branches. All branches maintain strong AUROC beyond layer 0; the sharp divergence after layer 22 between Knowledge (stable) and Math (collapsing) reveals the most systematic structural difference in routing signal distribution.

Table 5: Branch classification accuracy and routing AUROC for the auto-select setup on a balanced mixed test set (300 per branch). Classification uses a single-letter answer (2 tokens, no thinking).

Setting	Value	
Classification accuracy	Knowledge (MMLU)	87.0%
	Math	37.0%
	Agent (BFCL live_simple)	94.6%
	Overall	**71.8%**
Routing AUROC	Oracle	0.7614
	Auto	0.7434
	Loss	0.0180

quickly with the first few hundred examples and is already near saturation by 1K examples (79.43 AUROC). We therefore use 1K as a compact default, suggesting that branch-specialized routers do not require large supervision budgets to become useful.

Branch Auto-Selection with Small Routing Loss. Our scenario-oriented framework points toward a pool of cheap, branch-specific probes coordinated by a lightweight dispatcher, where new deployment scenarios could in principle be added by training one more compact probe. Closing the loop on this picture requires the dispatcher itself to stay cheap and to tolerate misrouting, so we test whether the small model can act as the dispatcher: we prompt Qwen3-8B to classify each query as Knowledge, Math, or Agent and forward it to the matching probe (Table 5). Classification is reasonable overall (71.8%) but weak on Math (37.0%), likely because MMLU itself contains math subcategories that blur the boundary with Knowledge QA. Even so, routing AUROC drops by only 0.018 from the oracle, since specialist probes are correlated: they all score the same underlying small-model correctness, so misrouting still yields useful signal. We read this as preliminary evidence that scenario-oriented routing can be extended by adding more lightweight specialists, rather than by escalating to a heavier top-level router.

Practical Overhead of Routing Methods. XROUTER adds no extra tokens at inference and stores only 151.5K parameters, three orders of magnitude below HybridLLM's 184M, while sampling-based baselines such as SemanticEntropy cost over a thousand extra tokens per query (Appendix C). This keeps the router substantially cheaper than the escalation it controls, as the specialist principle requires.

7 Conclusion

We have argued that routing in collaborative LLM systems is better organized around recurring deployment scenarios than around a single pooled testbed, and that the router itself should remain lighter than the escalation it controls, not scaling toward LLM-judge or reward-model complexity. RouterXBench operationalizes this principle across Knowledge, Math, and Agent branches; our metric framework separates router ability (AUROC) from scenario alignment (LPM, MPM, HCR); and XROUTER serves as a reference proof-of-concept that the design is practical. The results validate the principle. Across all three branches, XROUTER outperforms text-only learned routers using only model-internal signals and 151.5K parameters, far below HybridLLM's 184M. Layer attribution reveals that branch-specific routing signals are structurally distinct and broadly distributed, confirming that specialist organization exposes signals that a monolithic pooled router would average away. Branch auto-selection adds only 2 tokens of overhead with a loss of 0.018 AUROC, making the modular stack practical. Together, these findings support a concrete alternative to monolithic routing: lightweight specialists matched to recurring scenarios, composed via cheap dispatch, already suffice, and RouterXBench is designed to remain interpretable as collaborative deployment settings evolve.

Limitations

Our routing framework assumes the large model's capability exceeds the small model's; however, both models may perform similarly or converge on the same incorrect answer in certain domains, limiting routing effectiveness. Our experiments use a small set of model configurations (mainly Qwen3-8B paired with GPT-5, with a smaller-model comparison using Qwen3-1.7B in Appendix F) and report single-run results due to computational constraints; broader validation across diverse architectures, multiple seeds, and more complex scenario shifts would further strengthen the conclusions.

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.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. 2025. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *Preprint*, arXiv:2502.17387.

Amine Barrak, Yosr Fourati, Michael Olchawa, Emna Ksontini, and Khalil Zoghlami. 2025. Cargo: A framework for confidence-aware routing of large language models. *Preprint*, arXiv:2509.14899.

Felix Busch, Lena Hoffmann, Christopher Rueger, Elon H. C. van Dijk, Rawen Kader, Esteban Ortiz-Prado, Marcus R. Makowski, Luca Saba, Martin Hadamitzky, Jakob Nikolas Kather, Daniel Truhn, Renato Cuocolo, Lisa C. Adams, and Keno K. Bressem. 2025. Current applications and challenges in large language models for patient care: A systematic review. *Communications Medicine*, 5(1):26.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *Preprint*, arXiv:2401.10774.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024a. INSIDE: LLMs' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*.

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. 2024b. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024c. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024a. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. *Preprint*, arXiv:2404.14618.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024b. Hybrid llm: Cost-efficient and quality-aware query routing. *Preprint*, arXiv:2404.14618.

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 *Forty-second International Conference on Machine Learning*.

Zhichen Dong, Zhanhui Zhou, Zhixuan Liu, Chao Yang, and Chaochao Lu. 2025. Emergent response planning in LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*. PMLR.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2024. Shifting attention to relevance: Towards the uncertainty estimation of large language models.

Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, and Maxim Panov. 2024. Fact-checking the output of large language models via token-level uncertainty quantification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9367–9385, Bangkok, Thailand. Association for Computational Linguistics.

Wenzhi Fang, Liangqi Yuan, Guangchen Lan, Dong-Jun Han, and Christopher G. Brinton. 2026. Iterative critique-and-routing controller for multi-agent systems with heterogeneous LLMs. *arXiv preprint arXiv:2605.08686*.

Tao Feng, Haozhen Zhang, Zijie Lei, Pengrui Han, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Jiaxuan You. 2025. Fusionfactory: Fusing LLM capabilities with multi-LLM log data. *Preprint*, arXiv:2507.10540.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

Neel Guha, Mayee F Chen, Trevor Chow, Ishan S. Khare, and Christopher Re. 2024. Smoothie: Label free language model routing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. 2024. Language model cascades: Token-level uncertainty and beyond. In *The Twelfth International Conference on Learning Representations*.

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 (ICLR)*.

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 *Advances in Neural Information Processing Systems (NeurIPS)*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2025. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics.

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.

Aly M. Kassem, Bernhard Schölkopf, and Zhijing Jin. 2025. How robust are router-llms? analysis of the fragility of llm routing capabilities. *Preprint*, arXiv:2504.07113.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023a. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023b. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *Preprint*, arXiv:2302.09664.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yang Li. 2025. LLM bandit: Cost-efficient LLM generation via preference-conditioned dynamic routing.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle-2: Faster inference of language models with dynamic draft trees. *Preprint*, arXiv:2406.16858.

Qiqi Lin, Xiaoyang Ji, Shengfang Zhai, Qingni Shen, Zhi Zhang, Yuejian Fang, and Yansong Gao. 2025. Life-cycle routing vulnerabilities of llm router. *Preprint*, arXiv:2503.08704.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2024a. Contextualized sequence likelihood: Enhanced confidence scores for natural language generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10351–10368, Miami, Florida, USA. Association for Computational Linguistics.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2024b. Generating with confidence: Uncertainty quantification for black-box large language models. *Transactions on Machine Learning Research*.

Andrea Matarazzo and Riccardo Torlone. 2025. A survey on large language models with some insights on their capabilities and limitations. *arXiv preprint arXiv:2501.04040*. Version 2.

Gonzalo Ariel Meyoyan and Luciano Del Corro. 2026. A BERTology view of LLM orchestrations: Token- and layer-selective probes for efficient single-pass classification. *arXiv preprint arXiv:2601.13288*.

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

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings*

of the 42nd International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.

Xin Qiu and Risto Miikkulainen. 2024. Semantic density: Uncertainty quantification for large language models through confidence measurement in semantic space. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Guillem Ramírez, Alexandra Birch, and Ivan Titov. 2024. Optimising calls to large language models with uncertainty-based two-tier selection. In *First Conference on Language Modeling*.

Avital Shafran, Roei Schuster, Thomas Ristenpart, and Vitaly Shmatikov. 2025. Rerouting llm routers. *Preprint*, arXiv:2501.01818.

Jianshu She, Wenhao Zheng, Zhengzhong Liu, Hongyi Wang, Eric Xing, Huaxiu Yao, and Qirong Ho. 2025. Token level routing inference system for edge devices. *Preprint*, arXiv:2504.07878.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. 2024. LLM-check: Investigating detection of hallucinations in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dimitris Stripelis, Zijian Hu, Jipeng Zhang, Zhaozhuo Xu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Salman Avestimehr, and Chaoyang He. 2024. Tensoropera router: A multi-model router for efficient llm inference. *Preprint*, arXiv:2408.12320.

Jiayuan Su, Fulin Lin, Zhaopeng Feng, Han Zheng, Teng Wang, Zhenyu Xiao, Xinlong Zhao, Zuozhu Liu, Lu Cheng, and Hongwei Wang. 2025. Cp-router: An uncertainty-aware router between llm and lrm. *Preprint*, arXiv:2505.19970.

Tanay Varshney, Annie Surla, Michelle Xu, Gomathy Venkata Krishnan, Maximilian Jeblick, David Austin, Neal Vaidya, and Davide Onofrio. 2026. LLM router: Rethinking routing with prefill activations. *arXiv preprint arXiv:2603.20895*.

Song Wang, Zhen Tan, Zihan Chen, Shuang Zhou, Tianlong Chen, and Jundong Li. 2025. Anymac: Cascading flexible multi-agent collaboration via next-agent prediction. *Preprint*, arXiv:2506.17784.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Shibo Yu, Mohammad Goudarzi, and Adel Nadjaran Toosi. 2025. Efficient routing of inference requests across llm instances in cloud-edge computing. *Preprint*, arXiv:2507.15553.

Yaopei Zeng, Congchao Wang, Blake JianHang Chen, and Lu Lin. 2026. ReLope: KL-regularized LoRA probes for multimodal LLM routing. *arXiv preprint arXiv:2603.24787*.

Caiqi Zhang, Menglin Xia, Xuchao Zhang, Daniel Madrigal, Ankur Mallick, Samuel Kessler, Victor Ruehle, and Saravan Rajmohan. 2026a. Budget-aware agentic routing via boundary-guided training. *arXiv preprint arXiv:2602.21227*.

Tuo Zhang, Asal Mehradfar, Dimitrios Dimitriadis, and Salman Avestimehr. 2025a. Leveraging uncertainty estimation for efficient llm routing. *Preprint*, arXiv:2502.11021.

Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. 2025b. Capability instruction tuning: A new paradigm for dynamic llm routing. *Preprint*, arXiv:2502.17282.

Yiqun Zhang, Hao Li, Zihan Wang, Shi Feng, Xiaocui Yang, Daling Wang, Bo Zhang, Lei Bai, and Shuyue Hu. 2026b. MTRouter: Cost-aware multi-turn LLM routing with history-model joint embeddings. *arXiv preprint arXiv:2604.23530*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*. Version 16, last revised March 2025.

Zesen Zhao, Shuowei Jin, and Z. Morley Mao. 2024. Eagle: Efficient training-free router for multi-llm inference. *Preprint*, arXiv:2409.15518.

A Train-Size Analysis

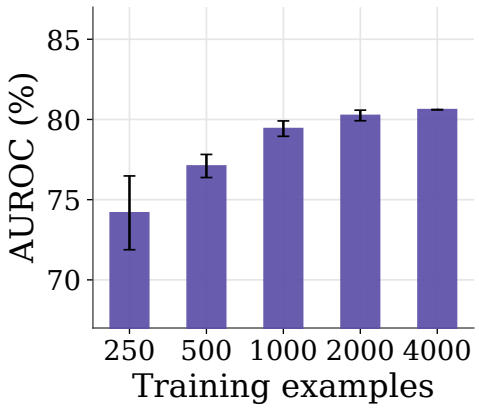

Figure 4: Train-size scaling for the main XROUTER method on MMLU-test. Performance improves quickly with modest supervision and is near saturation by 1K examples.

B Implementation Details

All main experiments use Qwen3-8B as the small model and GPT-5 as the large model. The Knowledge branch trains on 1K MMLU examples and evaluates on a fixed 2K MMLU-test subset plus MMLU-Pro. The Math branch trains on 1K Big-Math examples and evaluates on both held-out Big-Math and MATH. The Agent branch uses BFCL-v3. The main router trains on BFCL-v3 simple+multiple (600 examples) and evaluates on BFCL-v3 live_simple (258 examples). We also construct a full BFCL-v3 training split, BFCL simple+multiple+parallel+parallel_multiple, with 1K examples. Hidden-state routers use Q+A representations: the instruction and Qwen3-8B response are concatenated, passed through Qwen3-8B, and mean-pooled over tokens at every layer. All experiments use a fixed random seed (seed = 42). As acknowledged in the Limitations section, results are reported from a single run due to computational constraints; multi-seed validation across diverse architectures is left for future work. In Table 2, we instantiate the metric boundaries as $d_1 = 0.10$, $\rho_1 = 0.70$, $\rho_2 = 0.90$. For MPM, we use the branch-specific transition interval $[d_1, d_2]$ from the main definition. On Big-Math, if the mapped threshold order becomes inverted or the transition band is empty, we report a fallback MPM over the 10%–50% call-rate window.

C Routing Method Overhead

Table 6 reports the inference-time overhead of each routing method. Logit-based routers (Confidence-Margin, Entropy, MaxLogits) and EmbeddingMLP incur no extra tokens, since they operate on free by-products of the forward pass. SelfAsk and SemanticEntropy require additional generation, with SemanticEntropy costing over a thousand extra tokens per query due to its sampling procedure. Among trainable methods, XROUTER stores only 151.5K parameters (37-dimensional centroids plus covariance), three orders of magnitude smaller than HybridLLM.

Table 6: Inference overhead across routing methods. Token counts are per-query means over 100 samples.

Method	Extra Tokens	Params
SelfAsk	$+19.7 \pm 2.0$	0
SemanticEntropy	$5\times276\pm153$	0
Logit-based (3)	0	0
EmbeddingMLP	0	197K
HybridLLM	0	184M
XROUTER (ours)	0	151.5K

D Benchmark Datasets

RouterXBench is organized around three deployment scenarios. Each branch is paired with one source for training and at least one held-out source to probe scenario shift; Table 7 summarises the splits.

Knowledge (MMLU, MMLU-Pro). **MMLU** (Hendrycks et al., 2021a) is a multi-task knowledge benchmark spanning 57 subjects in STEM, humanities, social sciences, and other domains; we use 1K examples for training and a fixed 2K-example test subset for in-domain evaluation. **MMLU-Pro** (Wang et al., 2024) extends MMLU with harder distractors and broader subject coverage, and serves as a held-out evaluation set to probe scenario shift on the Knowledge branch.

Math (Big-Math, MATH). **Big-Math** (Albalak et al., 2025) is a large-scale collection of math problems with verified answers; we sample 1K examples for training and a held-out subset for in-domain evaluation. **MATH** (Hendrycks et al., 2021b) consists of competition-style problems and

serves as the shifted evaluation set, probing the Math branch under harder reasoning conditions.

Agent (BFCL-v3). **BFCL-v3** (Patil et al., 2025) evaluates function-calling and tool-use behaviour across diverse API schemas. The main router trains on 600 examples from the `simple` and `multiple` splits; we evaluate on `live_simple` (258 examples). A larger 1K training split additionally includes `parallel` and `parallel_multiple` configurations.

Table 7: RouterXBench branch structure.

Branch	Train	Test / Shift	Status
MMLU	MMLU-train 1K	MMLU-test 2K, MMLU-Pro	Complete for AUROC
Math	Big-Math 1K	MATH	Complete for AUROC
Agent	BFCL-v3 simple+multiple (600)	BFCL-v3 live_simple (258)	Complete

E Ground Truth Label Construction

For each query, the ground-truth routing label is binary: $y = 1$ if the small model is judged correct on its own (no escalation needed) and $y = 0$ otherwise. The construction differs by branch.

Knowledge and Math (MMLU, MMLU-Pro, Big-Math, MATH). Rule-based string matching is brittle due to format variation in chain-of-thought responses. We instead use **xVerify**,[1] specifically the `xVerify-9B-C` model, as a semantic verifier. Given the query q, the small model's response r_{small}, and the gold answer a_{gold}, it outputs

$$y = \text{xVerify}(q, r_{\text{small}}, a_{\text{gold}}), \tag{16}$$

where $y = 1$ indicates correctness and $y = 0$ indicates a routing target.

Agent (BFCL-v3). For tool-use queries the answer is a function call rather than free text, so we use execution-based evaluation following the BFCL-v3 protocol. A response is labelled $y = 1$ when the generated call matches the gold call in function name, argument structure, and argument values under type-aware comparison; otherwise $y = 0$. This yields a deterministic correctness signal without an auxiliary verifier model.

F Hidden-State Routing with a Smaller Small Model

Table 8 reports AUROC when Qwen3-1.7B replaces Qwen3-8B as the small model on the Knowledge and Math branches. The Agent branch is omitted because Qwen3-1.7B BFCL-v3 hidden states and results were not available. XROUTER remains the strongest trained method on both branches, suggesting that the routing signal from internal representations transfers across model scales.

Table 8: Router ability (AUROC) with Qwen3-1.7B as the small model. Agent branch is omitted; see text.

Method	Knowledge					Math	
	MMLU-test	STEM	Humanities	Social Sci.	Others	Big-Math	MATH
EmbeddingMLP	62.66	60.45	55.19	57.28	51.65	63.94	59.49
HybridLLM	64.80	59.12	55.24	54.50	50.73	53.91	59.07
XROUTER (ours)	**65.58**	**65.16**	**56.85**	**58.82**	**56.33**	**66.75**	**70.66**

G Pseudocode for XROUTER

Algorithm 1 XROUTER

procedure TRAIN($\{(H_i, y_i)\}_{i=1}^{n}$)
  for layer $l = 1, \ldots, L$ **do**
    $Z_+^{(l)} \leftarrow \{H_i[l] : y_i = 1\}$; $Z_-^{(l)} \leftarrow \{H_i[l] : y_i = 0\}$
    $\mu_+^{(l)} \leftarrow \text{mean}(Z_+^{(l)})$; $\mu_-^{(l)} \leftarrow \text{mean}(Z_-^{(l)})$
    $w^{(l)} \leftarrow \Sigma_l^{-1}(\mu_+^{(l)} - \mu_-^{(l)})$
  end for
end procedure
procedure SCORE(H)
  return $\frac{1}{L}\sum_{l=1}^{L} {w^{(l)}}^{\top} H[l]$
end procedure

[1] https://github.com/IAAR-Shanghai/xVerify