Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Difficulty-Aware Query Routing for Tiered Language-Model Serving: A Confidence-Gated Cascade for Cost-Efficient Open-Weight Edge Inference

Goutham Reddy Aenugu
San Francisco Bay University
Shekar Reddy Nimmathi
nimmathishekar7@gmail.com

Southern Arkansas University

Research Article

Keywords: LLM serving, query routing, model cascades, difficulty estimation, edge inference, discrete event simulation, queueing, cost efficient scheduling

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11447925/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Difficulty-Aware Query Routing for Tiered Language-Model Serving: A Confidence-Gated Cascade for Cost-Efficient Open-Weight Edge Inference

*Goutham Reddy Aenugu
Independent Researcher
USA
* Corresponding author. E-mail: Gouthamreddyaenugu@gmail.com

Shekar Reddy Nimmathi
Independent Researcher
USA
nimmathishekar7@gmail.com

***Abstract*—LLMs are often served from a pool of models that differ greatly in size, cost and speed, yet most systems send every query to one large model. We present a difficulty aware router that answers each query with the cheapest model first, reads that model's confidence in its own answer, and escalates only when confidence is too low. Predicting difficulty from the query text works only weakly: five router designs, including RouteLLM style routers, plateau at AUC 0.59 to 0.69, whereas the small model's single pass logprob confidence reaches test AUC 0.75 (95% CI 0.71 to 0.78) at tier 1 and 0.82 (0.79 to 0.85) at tier 2. Routing brings no per query accuracy gain: at matched latency, always using the mid sized model is 1.8 percentage points more accurate (CI 0.3 to 3.6). Its value appears under load. On open weight models served through Ollama on one 8 GB GPU, a discrete event simulation replaying measured service times shows the cascade sustaining 12.9 times (CI 11.4 to 14.0) the throughput of always using the largest model, and real concurrent serving confirms 9.8 to 12 times, because the slow, memory bound top tier stays lightly loaded. An oracle bound of 93.4% accuracy marks the remaining headroom. All results come from an untouched test split at zero API cost.**

***Index Terms*—LLM serving, query routing, model cascades, difficulty estimation, edge inference, discrete event simulation, queueing, cost efficient scheduling.**

I. Introduction

Serving a large language model is expensive, and much of that expense goes to questions that never needed a large model. When models of different sizes are available, the sensible policy is to give each question to the smallest model that can still answer it correctly. The hard part is knowing, before any expensive model runs, how difficult a question is; we call a scheduler that makes this decision a *difficulty aware router*.

Prior routers were built for paid, cloud hosted models: FrugalGPT stops a chain of API models with a learned rule [1], and RouteLLM and Hybrid LLM read the question to choose between a weak and a strong model [2], [3], with RouteLLM trained on Chatbot Arena preferences [4]. Serving platforms such as Clipper and INFaaS choose among models whose cost per request is known in advance [5], [6]. Selective classification [7], calibration [8] and self-consistency [9] instead suggest letting the cheap model answer first, reading how sure it is, and escalating only when it is unsure.

We test this alternative on local hardware: every model is served through Ollama [10] on one 8 GB consumer GPU, on a workload combining ARC [11], MMLU [12] and GSM8K [13], with Nomic Embed [14] for query only routers and Little's law [15] and multi-server queueing theory [16] for analysis under load. Our contributions are: 1) a controlled comparison in which five leakage free query only routers plateau at AUC 0.59 to 0.69 while the cheap model's single pass confidence reaches test AUC 0.75 to 0.82; 2) a confidence gated cascade, with the negative result that it does not beat a strong mid tier on single query accuracy at matched cost; 3) a reproducible, instrumented tiered serving layer; 4) a queueing analysis driven by measured service times whose 12.9 times throughput gain is confirmed by real serving at 9.8 to 12 times; and 5) a mixed open workload evaluation with validation chosen thresholds, an untouched test split and bootstrap intervals [17], at zero API cost.

II. Literature Review

A. LLM Cascades and Routing

A recent survey organizes LLM routing methods by when the decision is made and what information it uses [18]. FrugalGPT cascades API models with a learned scoring function [1]; RouteLLM learns a strong or weak router from Arena preferences with similarity weighted, matrix factorization and classifier architectures [2], [4]; and Hybrid LLM routes between an on device and a cloud model [3]. AutoMix adds self verification and a POMDP escalation policy [19], Gupta et al. defer on token level uncertainty [20], and Yue et al. cascade on answer consistency across reasoning representations [21]. RouterBench and LLMRouterBench find that most routers perform similarly and rarely beat a simple baseline [22], [23], and difficulty aware routing has been extended to long context inference [24] and multi-agent workflow selection [25]. This work measures cost in dollars per API token and ignores queueing.

B. Confidence, Calibration and Selective Prediction

Selective classification abstains under uncertainty [7], modern networks are often miscalibrated [8], expected calibration error (ECE) is the standard measure of the gap between confidence and accuracy [26], and isotonic regression is a standard post hoc correction [27]. Kadavath et al. report that large models are reasonably calibrated on multiple choice questions when confidence is read from the answer token probability [28]; Geng et al. survey logit based, consistency based and verbalized confidence estimators [29]; and Xiong et al. show that verbalized confidence tends to be overconfident [30]. Self-consistency uses agreement across sampled paths as a proxy for correctness [9].

C. Model Serving, Queueing and Resource Scheduling

Clipper and INFaaS address batching, replica scaling and latency objectives [5], [6] on classical queueing results [15], [16]. For transformer decoding, vLLM pages the key value cache [31], Orca batches per decoding iteration [32], Sarathi-Serve chunks long prefills [33] and DistServe disaggregates prefill and decode [34], while queueing theory for LLM inference gives throughput optimal schedulers [35] and stability under cache memory limits [36]. Broader scheduling work covers distributed LLM allocation [37], slice type prediction [38], compliant and carbon conscious multi-cloud placement [39], resource exhaustion prediction [40], reinforcement learning schedulers [41], container middleware [42], graph learning offloading [43], multi-cloud orchestration for AI workloads [44], vehicular [45] and industrial [46] edge scheduling, and fog heuristics [47]. All of these speed up one model or schedule tasks whose demand is observable at admission, whereas whether a cheap model will answer a query correctly stays hidden until a model has run.

TABLE I
POSITIONING AGAINST REPRESENTATIVE ROUTING STUDIES

Work	Pool / cost basis	Difficulty signal	Serving analysis	Local
FrugalGPT [1]	Hosted APIs; $ per token	Learned scorer on cheap output	None	No
RouteLLM [2]	Hosted pair; $ per token	Query only, preference trained	None	Partly
Hybrid LLM [3]	Device plus cloud; nominal budget	Query only router	None	Partly
AutoMix [19]	Hosted APIs; cost per call	Self verification plus POMDP	None	No
Token cascades [20]	FLAN-T5 family; size proxy	Token level uncertainty	None	Yes
This work	**Three open tiers, 8 GB GPU; measured latency, FLOPs**	**Single pass logprob confidence**	**Coupled DES plus measured serving**	**Yes**

D. Edge and On Device LLM Inference

Reviews catalogue the compression, offloading and runtime techniques for edge and on device language models [48], [49], and mobility and cost aware acceleration [50] and edge cloud cooperative inference under WAN limits [51] address placement. On memory bound edge hardware, the cost of a mixture of experts (MoE) model follows its total rather than its active parameter count [52]. The open weight models we use span this range: gpt-oss:20b, an MoE model that fits in 8 GB only with partial CPU offload [53], Llama 3.2 3B [54] and Qwen2.5 7B [55].

E. Research Gap and Positioning

Routing work was built on paid APIs, serving work assumes known demand, and edge work does not route between models. A fully local, open weight router that owns its tiered serving stack, measures its own costs and is analysed as a distributed system has not been built and evaluated, nor has anyone compared query text and cheap model confidence as difficulty signals on such a system. Table I contrasts this study with representative routing work; only the last row has measured costs, a serving analysis and a fully local stack.

III. METHODOLOGY

A. System Architecture

The system has four components (Fig. 1): a tiered model pool, an instrumented serving layer over Ollama, a difficulty signal and an escalation controller that runs the cheapest tier and accepts its answer when confidence clears the threshold, otherwise escalating, with tier 3 terminal.

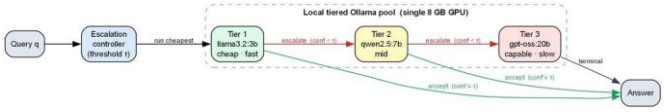


Fig. 1. System architecture of the confidence based escalation controller over the local three tier Ollama pool.

Tiered model pool. Three tiers span capability and cost under the 8 GB VRAM budget: tier 1 is llama3.2:3b (3.2 B parameters, fully resident, relative cost 1.0), tier 2 is qwen2.5:7b (7.6 B, resident but tight, 2.5) and tier 3 is gpt-oss:20b (20.9 B with about 3.6 B active, MoE, partial CPU offload, 6.0). gpt-oss:20b was chosen because it sustains about 15 tokens/s under offload, whereas a dense 27 B model would manage only 2 to 4. The relative costs are only a hardware independent reference; all policy comparisons use measured costs.

Serving layer. A thin client (serve.py) issues deterministic completions (temperature 0, fixed seed) to Ollama's /api/generate endpoint and records the timing breakdown it returns: model load, prompt evaluation (prefill) and generation (decode) durations plus token counts. Service time (prefill plus decode) feeds the queueing model.

B. Problem Formulation

A query q answered by tier t has unknown correctness $c_t(q) \in \{0, 1\}$, cost k_t and latency $\ell_t(q)$. A routing policy π assigns q an ordered path of tiers; the returned answer is that of the last tier on the path, and cost and latency are summed over the path, so escalation is paid for. Over a workload Q,

$$\mathrm{Acc}(\pi) = (1/|Q|)\,\Sigma_q\, c_{\mathrm{last}(\pi,q)}(q), \quad \mathrm{Cost}(\pi) = (1/|Q|)\,\Sigma_q\,\Sigma_{t\in\pi(q)}\, k_t, \quad (1)$$

$$\mathrm{Lat}(\pi) = (1/|Q|)\,\Sigma_q\,\Sigma_{t\in\pi(q)}\,\ell_t(q), \quad (2)$$

and we seek policies on the Pareto frontiers of accuracy against cost and against latency. The oracle policy, which sends each query to the cheapest tier that is actually correct on it, upper bounds every realizable policy.

C. Three Difficulty Signals

We compare three estimates of whether a cheap tier will be correct. 1) *Query embedding router (a priori):* a classifier on a nomic-embed-text embedding plus length and symbol density features, trained on cached correctness labels with five architectures (logistic regression, gradient boosting, an MLP as RouteLLM classifier, similarity weighted kNN as RouteLLM SW, and matrix factorization as RouteLLM MF); all scores are stratified 5 fold out of fold, and only RouteLLM's architectures, not its weights, are reused. 2) *Self-consistency (sampling):* majority vote agreement over $K = 5$ runs at temperature 0.7. 3)

Logprob confidence (single pass): the probability of the cheap tier's own answer, as the answer letter probability for multiple choice (MCQ) items and the geometric mean token probability of the final answer span for numeric items.

D. Confidence Cascade and Baselines

The controller runs tier 1 and reads its confidence $s_1(q)$; if $s_1(q) \geq \tau$ it returns tier 1's answer, otherwise it escalates to tier 2, reads $s_2(q)$, and returns or escalates to the terminal tier 3 (Algorithm 1). Sweeping $\tau \in [0, 1]$ traces a curve from always cheapest ($\tau = 0$) to always largest ($\tau = 1$). The gate adds no inference, since confidence comes from logprobs of an answer already produced.

Algorithm 1. Confidence gated escalation cascade

Input: query q; ordered tiers T = (1, 2, 3); threshold $\tau \in [0, 1]$
Output: answer a; path P; accumulated cost K; latency L
$P \leftarrow \emptyset$; $K \leftarrow 0$; $L \leftarrow 0$
for t in T do
  $(a, s_t(q), \ell_t(q)) \leftarrow$ Serve(t, q) // one deterministic pass with logprobs
  $P \leftarrow P \cup \{t\}$; $K \leftarrow K + k_t$; $L \leftarrow L + \ell_t(q)$
  if $s_t(q) \geq \tau$ or t is the terminal tier then return (a, P, K, L)
end for

Baselines are always-tier-1, always-tier-2 and always-tier-3 (a fixed tier for every query), random tier choice, the oracle, and the embedding router cascade as the a priori comparison point; static tier mixing is the reference line any informative router must beat. Cost is taken from direct measurements per query, wall clock service time and a FLOPs proxy (2 × active parameters × measured tokens), summed along the invoked path.

E. Distributed Serving and Queueing Model

One 8 GB node cannot host three GPU tiers at once, so we measure real per tier service times and drive a discrete event simulation (DES) of a multinode pool with them. Requests arrive as a Poisson process of rate λ; tier t is an M/G/c station with $c = r_t$ FIFO servers, requests traverse the stations on their path in tandem, and occupancy follows Little's law, $L_t = \lambda_t W_t$. Each simulated arrival replays a real query's own measured service times along its own cascade path, so difficulty, escalation, output length and service time stay correlated as observed. With visit ratio v_t and mean service time $E[S_t]$, station t is stable only while

$$\rho_t = \lambda\, v_t\, E[S_t] / r_t < 1, \quad \text{so} \quad \lambda_{max} = \min_t r_t / (v_t\, E[S_t]). \quad (3)$$

Because the terminal tier is slowest, the share of traffic reaching it sets the capacity ceiling. We sweep λ to near saturation for always-tier-3, always-tier-2 and the cascade with replicas (1, 1, 1) over 8 seeds, defining sustainable throughput as the largest load before p95 latency exceeds three times its unloaded value, and ask how many tier 3 replicas always-tier-3 needs to match one cascade node. To check the simulation, we measure real saturated throughput per tier with Ollama's continuous batching active by holding a fixed number of real queries in flight, and combine these capacities with each policy's visit shares.

F. Dataset, Protocol and Metrics

Workload. ARC, MMLU and GSM8K are unified into one trace with a common schema (prompt, choices, gold, answer type, difficulty label, subject, steps). From 18,909 unified queries we draw a fixed, stratified evaluation set of 2,400 queries with seed 42, used for every policy and for the cached inference: 800 ARC MCQ items (400 Easy and 400 Challenge, the official split used for external router validation), 800 MMLU MCQ items proportional across 57 subjects, and 800 GSM8K items with exact numeric answers stratified by chain of thought step count, a secondary difficulty label.

Answering protocol. All tiers receive identical prompts. MCQ items request the answer letter, with extraction tolerant of common wrappers; GSM8K items request brief reasoning followed by a final numeric line, graded on the normalized number. gpt-oss:20b returns its chain of thought in a separate thinking channel; we budget enough tokens for it to think and answer and count thinking tokens toward its cost.

Metrics and protocol. We report accuracy, measured cost (mean latency and FLOPs per query), escalation rate, tier traffic share and, under load, sustainable throughput and tail latency. Tier profiles use all 2,400 queries; every policy result uses a stratified 50/50 validation and test split, with thresholds chosen on the validation half, numbers from the untouched test half (n = 1,200), and bootstrap 95% intervals from 2,000 paired query resamples.

Hardware and reproducibility. One workstation with an NVIDIA RTX 5060 (8 GB VRAM) serves all models through Ollama 0.24.0. Seeds are fixed, decoding is deterministic, and a run once inference cache (results.parquet) lets every experiment be recomputed through the pipeline build_workload, make_eval_set, run_inference, embed_queries, router, evaluate and simulate_queue.

IV. Results and Discussion

A. Tier Capability and Cost Profile

Table II shows a clear capability ladder, from 68.3% accuracy at tier 1 to 83.7% at tier 2 and 87.5% at tier 3, but cost depends sharply on how it is measured. Relative to tier 1, tier 3 is 22.1 times more expensive by mean latency but only 2.24 times by FLOPs (tier 2: 3.76 and 2.51 times), so tier 3 is cheaper than tier 2 by FLOPs. Median service times (44, 116 and 9,613 ms) widen the gap to about 218 times. Tier 3 is therefore memory bandwidth and offload bound rather than compute bound, and wall clock latency, which sets serving throughput, is the binding cost on this hardware. MMLU stays far below ARC and GSM8K at every tier.

TABLE II
ACCURACY, LATENCY AND COMPUTE OF THE THREE TIERS

Tier	Acc.	ARC / GSM8K / MMLU	Mean ms	Median ms	GFLOP
1	68.3%	76.4 / 73.0 / 55.6	560	44	1,099
2	83.7%	93.2 / 88.1 / 69.8	2,106	116	2,754
3	87.5%	95.5 / 93.9 / 73.1	12,361	9,613	2,461

The oracle reaches 93.4% [CI 92.0 to 94.8] at 3.2 s mean latency on the test split, well above always-tier-3 (87.7% at 12.3 s), so a large headroom exists in principle.

B. Which Difficulty Signal Works

Table III compares the signals. Every query only router plateaus at AUC 0.59 to 0.69, and because five architectures sharing only their input fail together, an underpowered router cannot explain it. Self-consistency is weak for MCQ (0.52), since a 3 B model repeats the same wrong answer under sampling. Logprob confidence reaches test AUC 0.75 [0.71, 0.78] at tier 1 and 0.82 [0.79, 0.85] at tier 2 (0.74 on the full set, 0.85 on clean MCQ), and is monotonic: tier 1 answers with confidence of at least 0.95 are 83% correct, against 32% below 0.5. The embedding router barely separates ARC Easy from Challenge (+0.039).

TABLE III
AUC OF DIFFICULTY SIGNALS FOR PREDICTING TIER CORRECTNESS

Signal	Coupling	Tier 1	Tier 2
Embedding, logistic regression	a priori	0.59	0.62
Embedding, gradient boosting	a priori	0.64	0.68
Embedding, MLP (RouteLLM)	a priori	0.62	0.65
Embedding, kNN SW (RouteLLM)	a priori	0.64	0.69
Embedding, matrix factorization	a priori	0.61	0.61
Self-consistency (MCQ, K = 5)	sampling	0.52	n/a
Logprob answer confidence (test)	**single pass**	**0.75**	**0.82**

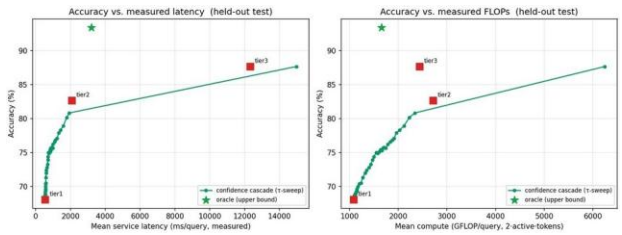


Fig. 2. Accuracy versus measured latency (left) and FLOPs (right) on the test split.

C. Accuracy Versus Measured Cost

Fig. 2 plots the cascade (τ sweep) against the static tiers and the oracle under both measured cost bases on the test split. The cascade traces a smooth frontier between tiers 1 and 3, but tier 2 sits on or slightly above the curve near its operating region, so single query routing does not dominate the mid tier. The latency and FLOPs panels also diverge for tier 3, which lies far right by latency but left of tier 2 by FLOPs, the memory bound signature of Table II.

D. Single Query Operating Point

We choose the threshold on the validation half by a fixed criterion, maximum accuracy subject to mean latency no greater than that of always-tier-2, which gives $\tau^* = 0.975$, and report on the test half (Table IV). The cascade attains 80.8% [78.6, 83.2] at 1.94 s, against 82.7% [80.5, 84.8] at 2.08 s for always-tier-2; the paired difference of −1.8 pp [−3.6, −0.3] excludes zero, so the mid tier is significantly more accurate at matched cost. The oracle gap shows that the shortfall lies in the imperfect gate rather than the policy class.

TABLE IV
SINGLE QUERY OPERATING POINT AT MATCHED LATENCY

Policy	Accuracy [95% CI]	Latency	Δ vs. tier 2 (pp)
always-tier-2	82.7% [80.5, 84.8]	2.08 s	reference
Cascade, $\tau^* = 0.975$	80.8% [78.6, 83.2]	1.94 s	−1.8 [−3.6, −0.3]
Oracle (upper bound)	93.4% [92.0, 94.8]	3.2 s	+10.7
always-tier-3	87.7%	12.3 s	+5.0

E. Distributed Serving Under Load

Table V reports stable throughput from the coupled DES and under real serving. At τ^*, which sends only 6.6% of traffic to tier 3, the cascade sustains 12.9 times [11.4, 14.0] the throughput of always-tier-3 and 2.3 times that of always-tier-2, and $\tau = 0.50$ reaches 21 times. Fig. 3 shows always-tier-3 saturating first and the embedding router cascade saturating well before the confidence cascade, whose weak gate sends more traffic to tier 3; Fig. 4 shows that always-tier-3 needs about 12 tier 3 replicas to match one cascade node; and Fig. 5 shows utilization and occupancy spread across tiers rather than pinned on the bottleneck.

TABLE V
STABLE THROUGHPUT IN THE DES AND UNDER REAL SERVING

Policy	Tier 3	DES req/s [95% CI]	Gain	Real req/s	Gain
always-tier-3	100%	0.058 [0.058, 0.059]	1.0×	0.143	1.0×
always-tier-2	0%	0.324 [0.304, 0.345]	5.6×	0.729	5.1×
Cascade, $\tau^* = 0.975$	6.6%	0.750 [0.663, 0.812]	12.9×	1.40	9.8×
Cascade, $\tau = 0.50$	0.5%	1.220 [1.144, 1.299]	21.0×	1.73	12.1×

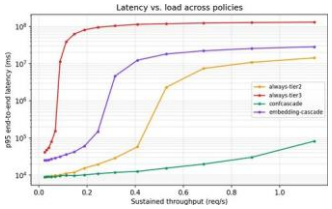


Fig. 3. p95 latency versus sustained throughput in the coupled DES.

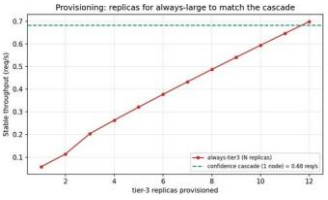


Fig. 4. Tier 3 replicas needed to match one cascade node.

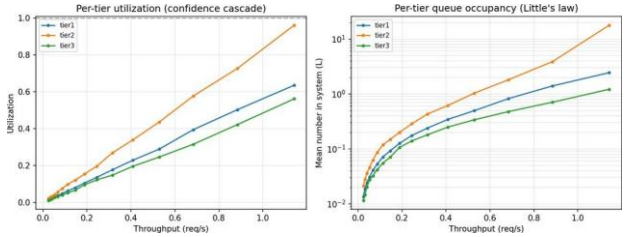


Fig. 5. Per tier utilization (left) and Little's law occupancy (right) under the cascade.

Empirical validation. Measured capacities with continuous batching are 1.73, 0.73 and 0.143 req/s for tiers 1 to 3 (from 80, 48 and 8 completions), giving a real advantage of 9.8 times at τ^* and 12.1 times at $\tau = 0.50$, which confirms the DES order of magnitude.

F. Ablations, Calibration and Robustness

At matched latency, the confidence gate beats the best query only router and the embedding router, so the gate rather than the cascade structure drives the gain. Logprob confidence is discriminative but mildly overconfident (Fig. 6), with ECE 0.14 and 0.15 at tiers 1 and 2, reduced to 0.024 and 0.012 by out of fold isotonic recalibration without changing the ranking. Fig. 7 shows accuracy varying by less than 1.7 pp across $\tau \in [0.80, 0.925]$, a broad operating band, with cost and escalation rising steeply only near $\tau = 1$. The oracle beats always-tier-2 within every source (ARC 98.6%, GSM8K 96.6%, MMLU 87.5%), and a gradient boosted meta gate adding query features to confidence matches confidence alone (tier 2 AUC 0.827 against 0.832).

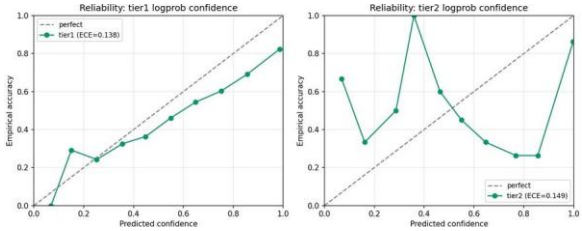


Fig. 6. Reliability diagrams of tier 1 (left) and tier 2 (right) logprob confidence.

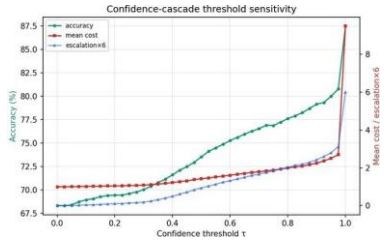


Fig. 7. Cascade accuracy, mean cost and escalation rate versus threshold τ.

G. Discussion and Limitations

A question's embedding encodes topic and surface form but only weakly whether a particular 3 B model will answer it, whereas the answer probability is produced after the model commits and costs nothing extra. In practice, a mid tier is the better default for one query at a time, while under concurrent traffic the cascade is worth about twelve nodes.

The DES maps each measured tier onto a dedicated node and abstracts batching and cache reuse, which the real concurrent measurement addresses; a physical multinode cluster remains the main open validation. RouteLLM baselines reuse its architectures on our labels, not its preference pipeline, and the study covers three English benchmarks with short answers, three models and one GPU, without wall plug energy.

V. Conclusion

This work tested whether LLM query difficulty can be predicted cheaply and whether difficulty aware routing pays off on local hardware, using a fully local, open weight cascade that runs the cheapest of three tiers first and escalates on the cheap model's own answer confidence. Query only difficulty proved weakly predictable (AUC at most 0.69), whereas single pass logprob confidence reached test AUC 0.75 to 0.82; a strong mid tier beat the cascade on single query accuracy at matched cost by 1.8 pp, but under concurrent load the cascade sustained 12.9 times the throughput of always using the large model in a measurement driven queueing analysis, confirmed at 9.8 to 12 times by real serving, with one cascade node matching about twelve large tier replicas. Difficulty aware routing is therefore a serving capacity lever rather than a per query cost lever, shown with a reproducible pipeline at zero API cost; future work should validate on a physical multinode cluster, close the 93.4% oracle gap with stronger a posteriori signals, extend confidence to open ended generation and replicate the RouteLLM preference pipeline.

Declarations. Data from public ARC, MMLU and GSM8K; code, traces and cache will be released on publication. No specific funding. The authors declare no competing interests.

References

[1] Chen, L., Zaharia, M., & Zou, J. (2023). *FrugalGPT: How to use large language models while reducing cost and improving performance*. arXiv. https://doi.org/10.48550/arXiv.2305.05176

[2] Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2024). *RouteLLM: Learning to route LLMs with preference data*. arXiv. https://doi.org/10.48550/arXiv.2406.18665

[3] Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., Lakshmanan, L. V. S., & Awadallah, A. H. (2024). Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations (ICLR)*.

[4] Chiang, W.-L., Zheng, L., Sheng, Y., et al. (2024). Chatbot Arena: An open platform for evaluating LLMs by human preference. In *International Conference on Machine Learning (ICML)*.

[5] Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., & Stoica, I. (2017). Clipper: A low-latency online prediction serving system. In *Proceedings of USENIX NSDI*.

[6] Romero, F., Li, Q., Yadwadkar, N. J., & Kozyrakis, C. (2021). INFaaS: Automated model-less inference serving. In *Proceedings of USENIX ATC*.

[7] Geifman, Y., & El-Yaniv, R. (2017). Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[8] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*.

[9] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

[10] Ollama. (2024). *Ollama: Get up and running with large language models locally*. https://ollama.com

[11] Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). *Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge*. arXiv. https://doi.org/10.48550/arXiv.1803.05457

[12] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

[13] Cobbe, K., Kosaraju, V., Bavarian, M., et al. (2021). *Training verifiers to solve math word problems*. arXiv. https://doi.org/10.48550/arXiv.2110.14168

[14] Nussbaum, Z., Morris, J. X., Duderstadt, B., & Mulyar, A. (2024). *Nomic Embed: Training a reproducible long context text embedder*. arXiv. https://doi.org/10.48550/arXiv.2402.01613

[15] Little, J. D. C. (1961). A proof for the queuing formula: $L = \lambda W$. *Operations Research, 9*(3), 383–387.

[16] Harchol-Balter, M. (2013). *Performance modeling and design of computer systems: Queueing theory in action*. Cambridge University Press.

[17] Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman and Hall/CRC.

[18] Moslem, Y., & Kelleher, J. D. (2026). *Dynamic model routing and cascading for efficient LLM inference: A survey*. arXiv. https://doi.org/10.48550/arXiv.2603.04445

[19] Madaan, A., Aggarwal, P., Anand, A., Potharaju, S. P., Mishra, S., Zhou, P., Gupta, A., Rajagopal, D., Kappaganthu, K., Yang, Y., Upadhyay, S., Mausam, & Faruqui, M. (2024). AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv. https://doi.org/10.48550/arXiv.2310.12963

[20] Gupta, N., Narasimhan, H., Jitkrittum, W., Rawat, A. S., Menon, A. K., & Kumar, S. (2024). Language model cascades: Token-level uncertainty and beyond. In *International Conference on Learning Representations (ICLR)*. arXiv. https://doi.org/10.48550/arXiv.2404.10136

[21] Yue, M., Zhao, J., Zhang, M., Du, L., & Yao, Z. (2024). Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. In *International Conference on Learning Representations (ICLR)*. arXiv. https://doi.org/10.48550/arXiv.2310.03094

[22] Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., & Upadhyay, S. K. (2024). RouterBench: A benchmark for multi-LLM routing

system. In *Proceedings of COLM*. arXiv. https://doi.org/10.48550/arXiv.2403.12031

[23] Li, H., Zhang, Y., Guo, Z., et al. (2026). *LLMRouterBench: A massive benchmark and unified framework for LLM routing*. arXiv. https://doi.org/10.48550/arXiv.2601.07206

[24] Gu, Z., Xiao, Z., & Lin, O. (2026). Difficulty-aware hybrid routing for cost-efficient long-context LLM inference. In *2026 6th International Conference on Intelligent Communications and Computing (ICICC)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICICC71012.2026.11637729

[25] Su, J., Lan, Q., Xia, Y., Sun, L., Tian, W., Shi, T., & He, L. (2026). Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference 2026 (WWW '26)* (pp. 2060–2070). ACM. https://doi.org/10.1145/3774904.3792240

[26] Naeini, M. P., Cooper, G. F., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of AAAI*.

[27] Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of ACM SIGKDD*.

[28] Kadavath, S., Conerly, T., Askell, A., et al. (2022). *Language models (mostly) know what they know*. arXiv. https://doi.org/10.48550/arXiv.2207.05221

[29] Geng, J., Cai, F., Wang, Y., Koeppl, H., Nakov, P., & Gurevych, I. (2024). A survey of confidence estimation and calibration in large language models. In *Proceedings of NAACL*. arXiv. https://doi.org/10.48550/arXiv.2311.08298

[30] Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., & Hooi, B. (2024). Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations (ICLR)*. arXiv. https://doi.org/10.48550/arXiv.2306.13063

[31] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. In *Proceedings of ACM SOSP*.

[32] Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., & Chun, B.-G. (2022). Orca: A distributed serving system for transformer-based generative models. In *Proceedings of USENIX OSDI*.

[33] Agrawal, A., Kedia, N., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B., Tumanov, A., & Ramjee, R. (2024). Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *Proceedings of USENIX OSDI*.

[34] Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., Jin, X., & Zhang, H. (2024). DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *Proceedings of USENIX OSDI*.

[35] Dai, J. G., Deng, T., Li, Y., & Peng, T. (2025). *Throughput-optimal scheduling algorithms for LLM inference and AI agents*. arXiv. https://doi.org/10.48550/arXiv.2504.07347

[36] Nie, C., Si, N., & Zhou, Z. (2026). *A queueing-theoretic framework for stability analysis of LLM inference with KV cache memory constraints*. arXiv. https://doi.org/10.48550/arXiv.2605.04595

[37] Patel, T. P., Shivam, S., & Ranganathan, V. (2026). Compute-optimal resource allocation for distributed large language model inference in cloud-scale intelligent systems. In *2026 14th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE. https://doi.org/10.1109/ISDFS69419.2026.11458958

[38] Kumar, N., et al. (2026). DeepSliceNet: A deep learning framework for network slice type prediction with augmentation and overfitting control. In Y. Koucheryavy, N. Kumar, A. Muthanna, & S. Sharma (Eds.), *Internet of Things, Smart Spaces, and Next Generation Networks and Systems (ruSMART NEW2AN 2025)* (Lecture Notes in Computer Science, Vol. 16461). Springer. https://doi.org/10.1007/978-3-032-19978-2_14

[39] Dubey, A., Sharma, S., & Kumar, N. (2025). SmartComply-RL+: An AI and ML-driven framework for compliant, cost-optimal, and carbon-conscious workload placement across multi-cloud environments. In *2025 6th International Conference on Smart Electronics and Communication (ICOSEC)* (pp. 1535–1540). IEEE. https://doi.org/10.1109/ICOSEC67334.2025.11459486

[40] Swati, S., Kumar, N., Dubey, A., Saini, V., Sharma, S., & Pranav, A. (2025). Predicting system performance and resource exhaustion with machine learning: Tackling class imbalance. In *2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT)* (pp. 658–662). IEEE. https://doi.org/10.1109/CE2CT64011.2025.10941221

[41] Zhang, W., & Ou, H. (2025). Reinforcement learning based multi-objective task scheduling for energy-efficient and cost-effective cloud-edge computing. *Scientific Reports, 15*, 41716. https://doi.org/10.1038/s41598-025-25666-1

[42] Kumar, N., Sharma, S., Dubey, A., Devi, K., & Balusamy, B. (2026). A lightweight AI-enabled container middleware for edge cloud architectures. In N. Kumar, S. Pratap Singh, B. Balusamy, P. Samuel, & C. Prabha (Eds.), *Integrating cloud, fog, and edge computing in healthcare: Federated learning and blockchain approaches* (Advances in Science, Technology and Innovation). Springer. https://doi.org/10.1007/978-3-031-96265-3_6

[43] Dai, Y., Rao, X., Gu, B., Qu, Y., Yang, H., & Lu, Y. (2025). Graph learning-based multiuser multitask offloading in wireless computing power networks. *IEEE Internet of Things Journal, 12*(15), 29230–29239. https://doi.org/10.1109/JIOT.2025.3565782

[44] Sharma, S., Kumar, N., Dash, Y., Dubey, A., & Devi, K. (2024). Intelligent multi-cloud orchestration for AI workloads: Enhancing performance and reliability. In *2024 7th International Conference on Contemporary Computing and Informatics (IC3I)* (Vol. 7, pp. 1421–1426). IEEE. https://doi.org/10.1109/IC3I61595.2024.10828941

[45] Tang, C., Li, Z., Wu, H., Xiao, S., & Li, R. (2025). Deep reinforcement learning-based collaborative computation offloading for distributed vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems, 26*(12), 22344–22356. https://doi.org/10.1109/TITS.2025.3611955

[46] Li, Y., Zhang, X., Sun, Y., Wang, W., & Lei, B. (2025). Spatiotemporal non-uniformity-aware online task scheduling in collaborative edge computing for industrial Internet of Things. *IEEE Transactions on Mobile Computing, 24*(10), 10169–10185. https://doi.org/10.1109/TMC.2025.3567615

[47] Alsadie, D. (2024). Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: Challenges and prospects. *PeerJ Computer Science, 10*, e2128. https://doi.org/10.7717/peerj-cs.2128

[48] Zheng, Y., Chen, Y., Qian, B., Shi, X., Shu, Y., & Chen, J. (2024). *A review on edge large language models: Design, execution, and applications*. arXiv. https://doi.org/10.48550/arXiv.2410.11845

[49] Xu, J., Li, Z., Chen, W., Wang, Q., Gao, X., Cai, Q., & Ling, Z. (2024). *On-device language models: A comprehensive review*. arXiv. https://doi.org/10.48550/arXiv.2409.00088

[50] Yuan, X., Li, N., Wei, K., Xu, W., Chen, Q., Chen, H., & Guo, S. (2025). Mobility and cost aware inference accelerating algorithm for edge intelligence. *IEEE Transactions on Mobile Computing, 24*(3), 1530–1549. https://doi.org/10.1109/TMC.2024.3484158

[51] Liao, J., Chang, F., Xia, T., Huang, Z., Huang, X., & Zhou, Y. (2026). Lightweight edge-cloud cooperative inference for small and medium models under WAN constraints. In *2026 5th International Conference on Electronic Information Engineering, Big Data and Computer Technology (EIBDCT)* (pp. 419–424). IEEE. https://doi.org/10.1109/EIBDCT69742.2026.11566691

[52] Alfarizy, A., Nguyen, H. T. T., Richard, R., Razavi-Far, R., & Cao, H. (2026). *Does mixture-of-experts actually help inference on consumer and edge hardware? An empirical study*. arXiv. https://doi.org/10.48550/arXiv.2606.21428

[53] OpenAI. (2025). *gpt-oss-120b and gpt-oss-20b model card*. arXiv. https://doi.org/10.48550/arXiv.2508.10925

[54] Grattafiori, A., et al. (2024). *The Llama 3 herd of models*. arXiv. https://doi.org/10.48550/arXiv.2407.21783

[55] Yang, A., et al. (2024). *Qwen2.5 technical report*. arXiv. https://doi.org/10.48550/arXiv.2412.15115