Qeios

Peer Review

Review of: "MixLLM: Dynamic Routing in Mixed Large Language Models"

Evgenii Arsentev[1]

1. Independent researcher

What the paper claims

MixLLM routes each query in a stream to one model out of a mixed pool, trading response quality against dollar cost against latency. The pipeline has four parts: a BERT encoder fine-tuned with an unsupervised intra-domain and inter-domain objective over InsTag-derived tags (Section 3.3), one lightweight quality predictor and one length-or-cost predictor per candidate model (random forest for quality, a mix of MLP, RF and KNN for length, Section 4.1.4), a contextual-bandit meta decision maker that adds a latency penalty, and a continual-learning loop that keeps updating from refined or binary feedback. Evaluation runs on RouterBench, 36,497 queries over 11 models, split 80/20 (Section 4.1.1), extended by the authors with Llama 3.1 8B and 70B plus prompt and response lengths. The headline is in Section 4.2: at lambda = 1.4 MixLLM reaches 97.25% of GPT-4 quality at 24.18% of GPT-4 cost, against the best baseline OptLLM at 96.39% and 32.94%; with the Llama 3.1 models added, Section 4.6 reports 98.55% at 16.79% at lambda = 1.8.

Two things set this apart from the routing papers it compares against. Latency is treated as a first-class constraint rather than an afterthought, which is what actually breaks routing systems once a popular model saturates. And the per-model predictors are independent, so adding or retiring a candidate does not force a system-wide retrain (Section 4.6) - a deployment property, not a benchmark trick. The ablations in Sections 4.4 to 4.8 are the kind a reader can argue with, which is a compliment.

Does the conclusion hold on the data?

1. The headline pair of numbers is one hand-picked point on a swept curve. Section 4.1.3 sweeps lambda from 1e-6 to 1e6; Section 4.2 then reports the result at lambda = 1.4 and Section 4.6 at lambda = 1.8. Nothing says how lambda would be chosen in deployment, where the quality-cost frontier is not

 doi.org/10.32388/GZL4AH

known in advance. Two fixes, both cheap: report the frontier as a table rather than only as Figure 4, and give the two comparisons that need no tuning at all - quality at exactly OptLLM cost (32.94%) and cost at exactly OptLLM quality (96.39%). Then add a sentence on picking lambda from a held-out slice.

2. Cost and latency are simulated, and latency is the paper novelty. Section 4.1.4 takes prices and speeds from a public aggregator, and Section 4.5 estimates response time as output length times generation speed, with open-source models simulated "under ideal hardware conditions, assuming sufficient memory and stable network connections". That model has no queueing, no batching, no cold starts, no retries and no tail. Real serving latency is heavy-tailed, and a scheduler tuned against a mean-speed model behaves differently when p95 is three times p50. One measured slice would settle it: a few hundred queries end to end against the real APIs, reporting the p50 and p95 wall-clock latency next to the simulated numbers.

3. The router's own cost is never subtracted. Each query passes through the fine-tuned BERT encoder, then 13 quality predictors and 13 length predictors, then the bandit update, at a stream rate of 100 queries per 10 seconds (Section 4.1.4). The paper calls these lightweight and notes an MLP under 2MB, but gives no milliseconds per query and no dollars per thousand queries for the router itself. Until that is counted, 24.18% is the cost of the routed answers, not the cost of the system. Please report router overhead on the stated 12GB Titan-V and fold it into the reported cost.

4. One split, one seed, and the ablation effects are small. Section 4.1.4 sets all seeds to 42, and Section 4.1.1 uses a single 80/20 split. The differences that carry the ablation claims are of a size where that matters: Table 1 moves 75.54% to 76.45% in the 80:20 setting, Table 2 moves 53.14% to 56.18% at the low-cost level, Table 3 drops 75.54% to 71.43% under the out-of-domain split. Those are 0.91, 3.04, and 4.11 percentage points. Five splits with five seeds, means with standard deviations, and a paired test would turn three suggestive tables into three results, and the experiments are cheap enough to allow it.

5. Two reporting conventions are mixed inside the same tables. In Table 1, the cells are absolute qualities while the "Improvement" row is relative: 75.54 to 76.45 is 0.91 points absolute and is printed as 1.21%. Table 2 does the same, printing 5.72% for a move from 53.14% to 56.18%, and Table 3 prints a 5.44% decrease for a 4.11 point drop. The arithmetic is right and the convention is defensible, but a reader who takes 1.21% as percentage points will misread all three tables. State the convention in each caption and give both numbers.

6. The timeout rule does a lot of work and is never isolated. Section 4.1.3 assigns quality 0 to any query that exceeds the maximum tolerable waiting time, and that single rule is what produces the baselines

falling away at higher budgets in Figure 4. It merges two different failures, a wrong answer and a late answer. Report the timeout rate per method separately, and show the frontier once more with late queries excluded rather than zeroed, so a reader can see how much of the advantage is routing quality and how much is queue management.

Reproducibility

Strong in places. Section 4.1.4 gives seeds fixed at 42, alpha 0.01, beta 0.1, gamma 0.1, learning rates 1, 1, and 0.001, the stream rate of 100 queries per 10 seconds, a tolerance of 30 seconds, and the full software stack down to CUDA 12.1 and Ubuntu 18.04, with a 12GB Titan-V for most runs and two 80GB H100s for the Llama work. Papers in this area rarely go that far.

What blocks an actual replication:

- No link to code, and none to the extended dataset, although the extension with Llama 3.1 answers and prompt and response lengths is listed as a contribution in Section 1. That artefact is arguably the most reusable part of the work.
- The BERT variant is not named - base or large, cased or uncased - and the fine-tuning schedule for the encoder is not given.
- Section 4.4 says the InsTag tags were "manually categorized into 20 domains". A manual mapping that is not published cannot be reproduced; release the tag-to-domain file and name the InsTag checkpoint.
- Prices and speeds come from a live aggregator page (footnotes 2 and 3). Prices for these models moved substantially during 2024 and 2025, so every cost percentage in the paper is tied to an undated snapshot. Give the access date and, better, a frozen copy of the price table.
- The bandit is described as contextual, but the exploration schedule, the arm parametrisation, and the update rule for changing candidate sets are not stated at the level needed to reimplement them.

What to fix

1. Sections 4.2 and 4.6: report the quality-cost frontier as a table and add the two tuning-free comparisons at matched cost and matched quality; state how lambda is selected without oracle access.
2. Sections 4.1.4 and 4.5: add one end-to-end measured run against the real endpoints and report p50 and p95 latency beside the simulated values.

3. Section 4.1.4: report router overhead in milliseconds per query and dollars per thousand queries, and include it in the cost axis of Figure 4.
4. Sections 4.3, 4.4, and 4.7: repeat over at least five seeds and splits, report mean and standard deviation, and add a paired test for Tables 1 to 3.
5. Tables 1, 2, and 3: state in the captions that the improvement rows are relative, and print absolute percentage points alongside.
6. Section 4.1.3: report the per-method timeout rate, and repeat the main figure with late queries excluded instead of scored as 0.
7. Section 4.2: the Oracle is defined as the cheapest model meeting a quality threshold, but the threshold value is never given, and a single Oracle point is drawn where a threshold sweep would give an Oracle frontier. State the threshold and draw the frontier.
8. Section 4.6: say exactly how the quality scores for the newly added Llama 3.1 answers were computed, and whether the per-dataset metric of the original RouterBench was applied unchanged. If the scoring differs, the cross-model comparison in Figure 6 is not on one scale.
9. Release code, the extended dataset, the tag-to-domain mapping, and a dated price snapshot; name the BERT and InsTag checkpoints.
10. Section 4.3: "we believe our one-time finding can be generalized to recurrent tests" is an assumption stated where a result belongs. Either run the recurrent test or mark it plainly as untested.
11. Section 2.2: the sentence listing medical, bioinformatics, material-science, and industrial applications with citations [22] to [28] sits inside a paragraph about predictive routing and does not serve it. Move it to the introduction or drop it.
12. Abstract: "artificial generic intelligence" should be "artificial general intelligence". Figure 1 is referenced in Section 1 before the routing task it illustrates has been defined.

Recommendation

Major revisions, of the tractable kind. The design is sound and the framing is right: latency belongs inside the objective, and per-model predictors are the correct answer to a candidate set that keeps changing. What the evaluation does not yet support is the precision of the headline claim, because the cost model is simulated, the router's own cost is missing from it, the operating point is chosen after the fact, and every number rests on one split with one seed. None of that requires new machinery - it requires the frontier reported in full, a handful of extra seeds, one measured latency run, and the artefacts

released. With those, this becomes a paper people can build a serving stack on rather than one they have to re-measure first.

Evgenii Arsentev, PhD

Declarations

Potential competing interests: No potential competing interests to declare.