Sequential Specialist MoE Routing Architecture for Complex Queries

M. G. Shree Harsha
Dept. of Data Science and Artificial Intelligence
IIIT Dharwad, Karnataka, India
Email: 24bds037@iiitdwd.ac.in
GitHub Repo Link: https://github.com/LordKarsSama/Sequential-MoE-Router

***Abstract*—Large language models are usually trained as dense generalists that must handle mathematics, explanation, coding, and reasoning within a single network. This is computationally expensive and often inefficient for multi-stage queries such as "solve, explain, and code." In this paper I describe a simple routing architecture that coordinates several small specialist models. A two-stage loss-based router first selects a subset of relevant experts for the full query and then assigns individual query segments (e.g., solve, explain, code) to the most suitable specialist. The selected experts are executed in a specialist pipeline to produce the final response. Because each model learns only its own domain, training is much faster than for a single large generalist, while evaluation on a 50–question PhD-level benchmark shows that the specialist system comes reasonably close to a stronger 3B dense model on challenging math and computer-science tasks.**

***Index Terms*—Mixture-of-Experts, routing, language models, multi-domain reasoning, task decomposition.**

I. Introduction

Dense large language models (LLMs) are typically trained as universal generalists. A single network is expected to solve mathematical problems, explain them in natural language, and produce correct code, all with one set of parameters. This design is simple, but it makes training extremely expensive and offers limited control over how different skills are used for multi-step tasks.

Many real queries are naturally multi-stage. A typical example is: *"Solve the problem, explain the reasoning, and then write the code."* For such queries, humans intuitively separate roles: a solver, a teacher, and a programmer. A single dense LLM is forced to internally simulate this structure without any explicit routing.

In this work I explore a more direct alternative: a *sequential specialist Mixture-of-Experts (MoE) routing architecture*. Instead of one large generalist, I maintain several smaller experts: a math specialist, a code specialist, and a general instruction model. A lightweight router decides which experts are relevant for a given query and which expert should handle each segment of the query.

My goals are:

- to reduce training time by training each specialist only on its own domain;
- to maintain competitive performance on difficult math and computer science questions;
- to provide a modular architecture that can be extended with new specialists.

II. Related Work

Mixture-of-Experts (MoE) architectures route tokens or hidden states to a subset of experts inside a single large model. These systems achieve parameter efficiency but typically operate at the token level and do not explicitly decompose a query into high-level tasks.

Cascaded systems run a small model first and call a larger model only when necessary. This reduces inference cost but still depends on a single strong generalist and does not support structured multi-stage routing.

Multi-agent and ensemble approaches combine several LLMs, often in parallel, but usually rely on heuristic dispatch and do not provide a clear, loss-based mechanism for selecting specialists per segment.

In contrast, the architecture I present here performs *task-level* routing: it computes language-model loss to select relevant experts, splits the query into segments such as "solve", "explain" and "code", and assigns each segment to the best specialist. The experts then run in a simple specialist pipeline.

III. Proposed Architecture

My system consists of a pool of specialist models and a two-stage router. Stage 1 selects a subset of relevant experts for the full query. Stage 2 performs finer-grained routing over individual segments such as solve, explain, and code.

A. Stage 1: Loss-Based Expert Selection

Let N be the number of available experts. For a given input query, I compute the language-model loss of each expert on the full query prompt. Experts whose loss lies within a fixed threshold of the minimum are retained as *relevant experts*. This filters out clearly irrelevant experts while keeping a small candidate set.

Figure 1 illustrates this stage. The top row shows the full pool of experts $E_1, \dots, E_N$. The router computes loss for each expert and selects those that appear to understand the query best.

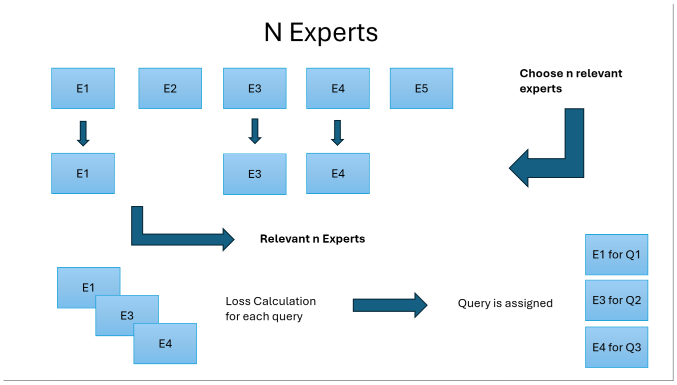

Fig. 1. Stage 1 loss-based expert selection. All N experts are evaluated on the full query, and only the experts with relatively low loss are kept as the relevant expert set for further routing.

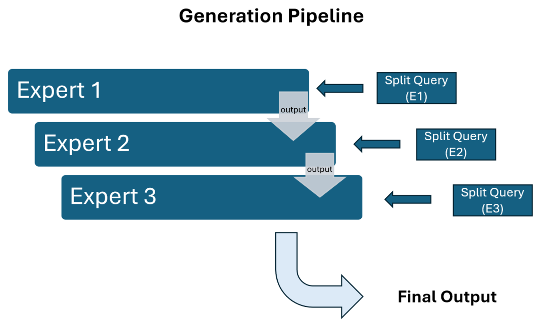

Fig. 2. Execution pipeline after expert selection. Each segment of the query is routed to the most suitable specialist, and the segment outputs are combined into a final answer.

B. Stage 2: Specialist Pipeline After Routing

After Stage 1, only the relevant experts are considered. The query is split into semantic segments such as:

- a *solve* segment describing the core mathematical task,
- an *explain* segment asking for a step-by-step explanation,
- a *code* segment requesting an implementation.

For each segment, the router computes loss using the relevant experts and assigns the segment to the specialist with the lowest loss on that segment.

The selected experts are then executed in a simple pipeline: the solve expert answers the main question, the explain expert builds a clear explanation (possibly conditioned on the answer), and the code expert writes the requested implementation. Figure 2 shows the conceptual pipeline after routing.

C. Training Compute Considerations

Training several small specialists can be significantly cheaper than training a single large dense model. Each specialist is trained only on its own domain (e.g., math, code, or general instruction), instead of learning everything at once. In practice this means fewer total effective tokens per model and simpler gradients for each domain, reducing overall training time while preserving strong performance on the targeted tasks.

TABLE I
AVERAGE SCORES OVER 50 PHD-LEVEL QUESTIONS.

Model	Solve	Explain	Code
Qwen2.5-VL-3B-Instruct	5.62	5.62	4.64
MoE Router	5.14	5.14	3.70

IV. RESULTS

To evaluate the architecture, I compared the specialist MoE router against a stronger dense model on a deliberately challenging benchmark of 50 PhD-level questions in advanced mathematics and theoretical or computational computer science. The benchmark contains:

- 25 questions in real and functional analysis, PDEs, probability, topology, approximation theory, and numerical analysis;
- 25 questions in algorithms, data structures, systems, parallel computing, compiler design, and statistical computing in R.

A. Models Compared

I compare two systems:

- **Model A:** Qwen2.5-VL-3B-Instruct, a 3B generalist instruction-tuned model.
- **Model B:** MoE Router, a routed system that dispatches query segments among three smaller specialists: *Qwen2.5-0.5B-Instruct* (general instruction model), *Qwen2.5-0.5B-Coder* (code specialist), and *Qwen2.5-1.5B-Math* (mathematics specialist).

For each question, both models are prompted once. ChatGPT Agent graded the answers manually along three axes on a 0–10 scale: (all the grading and evaluation was done by ChatGPT Agent)

- *Solve*: did the model substantially solve the main mathematical or algorithmic task?
- *Explain*: was the reasoning clear and logically structured?
- *Code*: when code was requested, did the model produce relevant and plausible code?

The grading rubric is: 0–2 = almost no progress, 3–4 = partial or badly broken solution, 5–6 = captures the main ideas but with non-trivial gaps, 7–8 = mostly correct and fixable by an expert, 9–10 = near-expert performance (extremely rare on this benchmark).

B. Average Scores

Averaging over all 50 questions yields the scores summarized in Table I.

Several observations follow from these results. On *Solve* and *Explain*, the MoE Router is only about 0.5 points behind the larger 3B generalist on a 0–10 scale, corresponding to a gap of roughly 10%. On *Code*, the gap is larger (4.64 vs. 3.70 on average), mainly because the code specialist sometimes produces partial or underdeveloped implementations rather than fully finished solutions.

Overall, despite using smaller models and a still-simple routing strategy, the specialist MoE architecture operates in a performance regime reasonably close to a much larger dense generalist on extremely difficult problems.

V. Conclusion

In this paper I presented a sequential specialist MoE routing architecture for complex multi-stage queries. Instead of relying on a single large generalist model, the system uses several small specialists coordinated by a two-stage loss-based router. Stage 1 selects a subset of relevant experts for the full query, while Stage 2 assigns individual query segments such as solve, explain, and code to the most suitable specialist. The experts then run in a simple pipeline.

On a 50-question PhD-level benchmark in mathematics and computer science, the specialist MoE system comes within about 0.5 points (on a 0–10 scale) of a 3B dense generalist on solving and explanation, while lagging somewhat more on code quality. Given its lower training cost and its modularity, I believe this architecture provides a promising direction for building multi-domain LLM systems from small, reusable components.

References

[1] S. J. Wang, H. Liu, and X. Li, "RopMura: Router-and-Planner-Based Multi-Hop Question Answering," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[2] J. Zhang, M. Qi, and L. Huang, "EMRC: Expert Model Routing for Clinical Question Answering," in *Findings of the Association for Computational Linguistics (ACL Findings)*, 2024.

[3] Y. Du et al., "Mixture-of-Agents: Learning to Share Behaviors for Multi-Agent Coordination," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[4] A. Kamradt and Y. Tay, "Chain-of-Experts: Sparse Routing in Transformer Layers," arXiv:2403.01852, 2024.

[5] H. Zhou, L. Yang, and Z. Fei, "Cascaded Large Language Models for Efficient Inference," in *Proceedings of the 2023 ACL Workshop on Efficient NLP*, 2023.

[6] S. Chen et al., "FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Accuracy," arXiv:2305.10333, 2023.

[7] A. Press et al., "Ask, Refine, Trust: Causal Decomposition Improves LLM Reasoning," in *International Conference on Learning Representations (ICLR)*, 2024.

[8] J. Zhou, T. He, and K. Cho, "Least-to-Most Prompting Enables Complex Reasoning in Large Language Models," arXiv:2205.10625, 2022.

[9] J. Liang et al., "LLMRank: Ranking, Routing, and Merging Specialized LLMs," arXiv:2406.04248, 2024.