
Eagle: Efficient Training-Free Router for Multi-LLM Inference

Zesen Zhao*, Shuowei Jin*, Z. Morley Mao

University of Michigan
{*hymanzsz, jinsw, zmao@umich.edu*}

Abstract

The proliferation of Large Language Models (LLMs) with varying capabilities and costs has created a need for efficient model selection in AI systems. LLM routers address this need by dynamically choosing the most suitable model for a given query based on task requirements and budget constraints. However, existing routers face challenges in scalability and real-time adaptation, particularly in high-volume online environments. We present *Eagle*, a novel LLM routing approach that combines global and local ELO ranking modules to overcome these limitations. By evaluating both general and specialized LLM abilities, *Eagle* provides a scalable, training-free solution that enhances model selection quality while reducing computational overhead. Our experiments across multiple datasets show *Eagle* consistently outperforms baseline methods, with improvements of up to 23.52% in Area Under Curve (AUC) scores. Moreover, *Eagle* demonstrates remarkable efficiency, requiring only 1/20 of baseline methods' time for initialization and 100-200x faster incremental updates in online scenarios, making it well-suited for dynamic, high-volume online serving environments.

1 Introduction

Large language models (LLMs) have demonstrated extraordinary performance across numerous tasks, from daily tasks like commonsense reasoning and question answering, to advanced tasks like code generation and mathematical problem-solving [3, 4, 23, 14].

Two primary strategies have emerged to improve LLM performance across various tasks. The first is scaling up [15]: increasing model size and training data to boost overall performance across a wide range of tasks. However, this method comes with the tradeoff of significantly higher inference costs due to the larger number of parameters. The second approach is specialization: developing domain-specific LLMs that can achieve performance comparable to larger models within a specific domain while maintaining a much smaller model size. For instance, CodeQwen [2] demonstrates GPT-4-level code generation capabilities with only a 7B parameter model [16].

These developments have resulted in a diverse ecosystem of LLMs, each with its own specialties and associated inference costs. To leverage the varying qualities and costs of these models effectively, researchers have proposed the concept of a "router" – a system designed to select the optimal model for a given query based on content and cost constraints [18, 11]. Compared to traditional single-model query procedures, a router enables users to obtain the highest quality answer within their budget. The primary goal of the router is to predict quality rankings, as the cost of querying each model is fixed. LLM service providers are beginning to integrate routers into their systems, with OpenAI recently releasing an "Auto" feature on the ChatGPT website to dynamically choose between their models (GPT-4o, GPT-4o mini, o1-preview) based on task requirements and performance characteristics.

*Equal contribution.

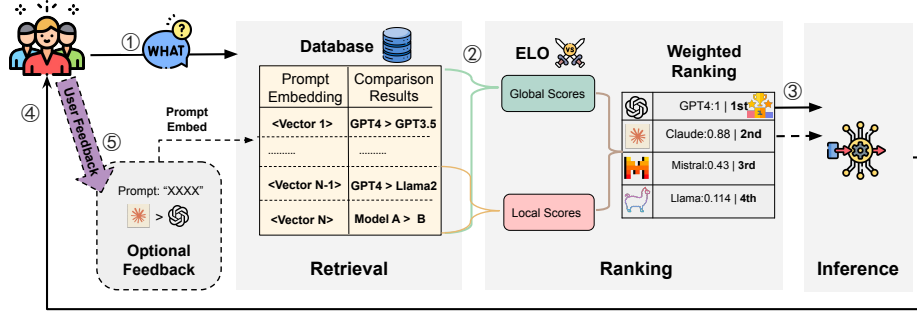


Figure 1: *Eagle* workflow: ① User request submission. ② Retrieval of relevant historical data. ③ LLM quality ranking and selection within budget. ④ Response generation and delivery. ⑤ Optional secondary model comparison and feedback collection.

However, designing a practical LLM router for online serving systems presents several challenges:

- **Scalability and Real-time Adaptation:** With millions of requests per second, the router must efficiently process and route queries while continuously adapting to new information. Traditional machine learning approaches often struggle with the computational overhead of frequent retraining.
- **Incomplete Feedback Data:** User feedback in online systems is often limited to pairwise comparisons between two responses. Extracting meaningful global rankings from this partial information is a non-trivial task that many existing methods fail to address adequately.
- **Balancing Accuracy and Efficiency:** While high prediction accuracy is crucial for optimal routing, achieving this accuracy must not come at the cost of system responsiveness. Striking the right balance between these often-competing goals remains a significant challenge in LLM routing.

Existing approaches to LLM routing have limitations. Routerbench [11] proposes using traditional machine-learning methods like MLP and KNN to predict quality rankings, but these require heavy retraining and cannot utilize real-time user feedback. RouteLLM [18] proposes a similarity-weighted ranking approach for training-free prediction. While effective for binary routing decisions, their method is limited to scenarios involving only two models and does not address cases with $n \geq 3$ models (where n denotes the number of models).

To address these challenges, we present *Eagle*, a novel approach to LLM routing with the following key contributions:

- *Eagle* achieves the most accurate prediction of model quality rankings by designing a global and local ELO module, ensuring the highest-quality answers within a given budget. Our experiments show that *Eagle* outperforms baseline methods across all datasets, with improvements of 23.52% over SVM, 5.14% over KNN, and 4.73% over MLP in terms of Area Under Curve (AUC) scores.
- *Eagle* is a training-free router that is significantly more efficient than traditional machine learning-based routers when adapting to newly collected data. Our experiments demonstrate that *Eagle* requires only 4.8% of the training time of baseline methods for initial setup, and a mere 0.5-1% of the updating time for incremental data updates.

2 *Eagle* Architecture and Design

We present the workflow of *Eagle* in Figure 1. *Eagle* is designed to optimize the selection of Large Language Models (LLMs) based on user requirements and historical performance. The system maintains a comprehensive vector database containing embeddings of previous input prompts along with their corresponding user feedback. When a user submits a request, *Eagle* leverages this database to retrieve relevant historical information and generates a predicted response quality ranking for various available LLM models. Considering the user’s budget constraints, *Eagle* then selects the highest quality model within the specified budget. The user’s query is then routed to the selected LLM for inference. After the target LLM generates the response, the result is returned to the user. To gather additional feedback, *Eagle* may optionally select another model from the list and ask the user to compare the quality of the two responses.

2.1 Eagle Design

The key idea behind *Eagle*'s design for accurately predicting different models' response quality ranking is based on two main insights:

1. **Holistic Ability Assessment:** Each LLM possesses two types of abilities: a) General Ability: A model's overall performance across diverse tasks, captured by analyzing its behavior on the entire dataset. b) Specialized Ability: Task-specific proficiency, identified through performance patterns on similar historical queries.

2. **Efficient Feedback Integration** Since user feedback typically only provides a preference between two models, obtaining a complete ranking of all models from users is challenging. To address this, we employ the ELO[9] ranking method, which transforms sparse pairwise comparisons into a comprehensive ranking system, maximizing the value of each user interaction.

To implement these principles, *Eagle* incorporates two core modules:

- *Eagle-Global*: Evaluates models' general ability using the entire historical dataset.
- *Eagle-Local*: Assesses specialized abilities by analyzing performance on similar past queries.

Both modules leverage the ELO algorithm to construct full model rankings from partial feedback data. By combining both *Eagle-Global* and *Eagle-Local*, *Eagle* achieves a more accurate and nuanced prediction of model performance, adapting to both general trends and query-specific requirements.

2.2 Details of Eagle

The ELO algorithm is a rating system used to calculate the relative skill levels of players (or models) by updating their scores based on the outcomes of pairwise comparisons. The ELO rating for a model is updated after each match using the following formula:

$$R' = R + K \times (S - E) \tag{1}$$

Where R is the current rating, R' is the updated rating, S is the actual score (1 for win, 0.5 for draw, 0 for loss), K is a constant determining the sensitivity of rating changes, higher K results in larger adjustments, and E represents the expected score or the predicted probability of a player winning a match based on their current ratings and is calculated using the rating difference between two players:

$$E = \frac{1}{1 + 10^{\frac{R_{\text{opponent}} - R}{400}}} \tag{2}$$

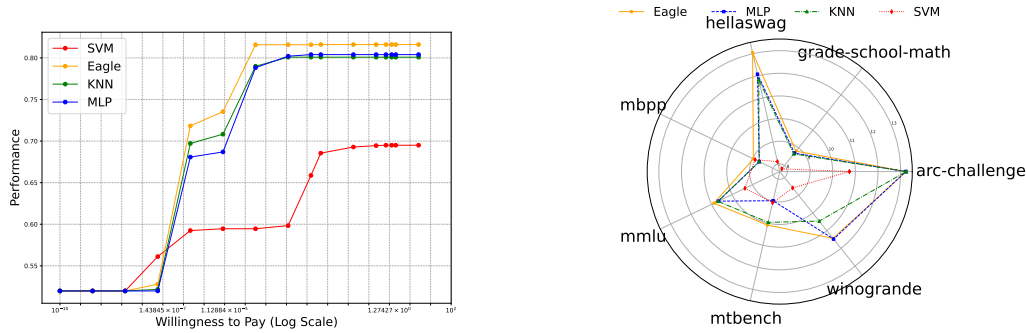
For *Eagle-Global*, we calculate the average ELO rating across all pairwise feedback information in the database. When new feedback data is collected, we can efficiently update *Eagle-Global* by performing ELO calculations on the new data only, eliminating the need for retraining.

To operate *Eagle-Local*, we first utilize a vector database to retrieve the N nearest neighbors of user feedback based on cosine semantic similarity using the prompt embedding vector. We initialize the local ELO scores for each model using the global ELO scores as background knowledge. Then, we use the retrieved local feedback to update the local ELO scores for each query.

With these two scores, we compute the weighted sum of the global and local ELO scores using the following equation: $\text{Score}(X) = P \times \text{Global}(X) + (1 - P) \times \text{Local}(X)$. This combined score provides a comprehensive quality ranking of the models, accounting for both their general and specialized abilities. *Eagle* then selects the highest-ranked model that falls within the user's specified budget constraints, ensuring an optimal balance between performance and cost-efficiency.

3 Evaluation

In this section, we present a series of experiments designed to evaluate the effectiveness and efficiency of *Eagle*. All of our experiments were conducted using the RouterBench dataset [11]. Detailed information about the experimental setup, including hardware specifications, model parameters, and baseline configurations, can be found in Appendix A. We also perform an ablation study to understand the effects of *Eagle-Global* and *Eagle-Local* on the final performance, which is detailed in Appendix B.

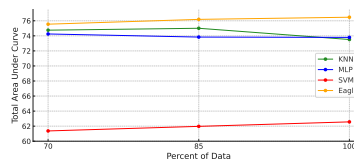


(a) Router’s performance with budget on MMLU dataset. (b) Area Under Curve across seven datasets.

Figure 2: Comparison of Baseline Models with Eagle.

Models	70%	85%	100%
KNN	176.3	180.6	193.4
MLP	248.3	253.3	260.2
SVM	114.7	143.0	150.5
<i>Eagle</i>	8.0	1.4	1.5

(a) Training time (seconds) for models at different data stages. 70%: initial training; 85% and 100%: incremental adding data simulating user feedback collection.



(b) Routers’ performance on test set when incrementally using more data.

Figure 3: Comparison of training time and quality

3.1 Overall Performance

We evaluated the performance of our method against baseline methods across all datasets in Router-Bench, including MMLU[10], Hellaswag[24], GSM8K[6], ARC Challenge[5], Winogrande[19], MBPP[1], and MT-Bench[25]. Figure 2a illustrates the willingness-to-pay versus performance graph for each router on the MMLU dataset. As observed, *Eagle* consistently outperforms other methods across various willingness-to-pay levels.

To comprehensively compare performance across all datasets, we calculated the area under the curve (AUC) using the trapezoidal rule. The AUC serves as a metric to evaluate a router’s average performance across all cost scenarios. Figure 2b presents a radar graph comparing the AUC of *Eagle* and baseline methods across the seven datasets. *Eagle* demonstrates superior performance across all datasets. Summing up the AUC scores, we find that *Eagle* achieves a 23.52% improvement over SVM, a 5.14% improvement over KNN, and a 4.73% improvement over MLP.

3.2 Online Adaptation Efficiency and Quality

In this section, we evaluate the efficiency and effectiveness of our method compared to baseline models in an online serving scenario. We simulate this by initially training all methods on 70% of the training data, then assessing the retraining time and performance when each additional 15% of data is introduced. Table 3a illustrates the efficiency (training time) of *Eagle* compared to the baseline methods. Even at the initial full data training stage, *Eagle*’s global ELO score initialization is significantly more efficient, using only 4.8% of the training time required by baseline methods. This efficiency stems from *Eagle*’s approach of updating global scores once, rather than iteratively optimizing the model. The efficiency gap widens as new data is introduced, with *Eagle* requiring only 0.5-1% of the baseline updating time for each 15% increment of new data.

Importantly, this efficiency does not come at the cost of performance. We evaluated *Eagle* against baseline methods across seven datasets and calculated the summed Area Under the Curve (AUC) metric. As demonstrated in Figure 3b, our results show that *Eagle* consistently outperforms other methods across all dataset settings, demonstrating both superior efficiency and effectiveness. At 70% of the data, *Eagle* achieves an average quality improvement of 8.65% across the three baseline routers. This improvement increases to 9.21% at 85% data and 9.92% at 100% data, demonstrating consistently superior performance across varying data volumes.

References

- [1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- [2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- [3] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- [5] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- [6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [7] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024.
- [8] Dunzhang. Stella en 1.5b, 2024. https://huggingface.co/dunzhang/stella_en_1.5B_v5.
- [9] Arpad E. Elo. The proposed uscf rating system, its development, theory, and applications. *Chess Life*, 22(8):242–247, 1967.
- [10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [11] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.
- [12] Kunal Jain, Anjaly Parayil, Ankur Mallick, Esha Choukse, Xiaoting Qin, Jue Zhang, Íñigo Goiri, Rujia Wang, Chetan Bansal, Victor Rühle, et al. Intelligent router for llm workloads: Improving performance through workload-aware scheduling. *arXiv preprint arXiv:2408.13510*, 2024.
- [13] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*, 2023.
- [14] Shuwei Jin, Yongji Wu, Haizhong Zheng, Qingzhao Zhang, Matthew Lentz, Z Morley Mao, Atul Prakash, Feng Qian, and Danyang Zhuo. Adaptive skeleton graph decoding. *arXiv preprint arXiv:2402.12280*, 2024.
- [15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

- [16] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [17] Xiaoding Lu, Zongyi Liu, Adian Liusie, Vyas Raina, Vineet Mudupalli, Yuwen Zhang, and William Beauchamp. Blending is all you need: Cheaper, better alternative to trillion-parameters llm, 2024.
- [18] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.
- [19] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- [20] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.
- [21] Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets, 2023.
- [22] Dimitris Stripelis, Zijian Hu, Jipeng Zhang, Zhaozhuo Xu, Alay Shah, Han Jin, Yuhang Yao, Salman Avestimehr, and Chaoyang He. Polyrouter: A multi-llm querying system. *arXiv preprint arXiv:2408.12320*, 2024.
- [23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [24] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- [25] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

A Experimental Setup

The experiments were performed on a server equipped with a Ryzen 3700X CPU, an RTX 4070 GPU, and 32GB of RAM. We utilized the `stella_en_1.5B_v5` [8] model for text embedding. The dataset was split into 70% for training and validation, with the remaining 30% used for testing.

A.1 Model Parameters

For *Eagle*, we set the following parameters:

- $P = 0.5$
- $N = 20$
- $K = 32$

A.2 Baseline Configurations

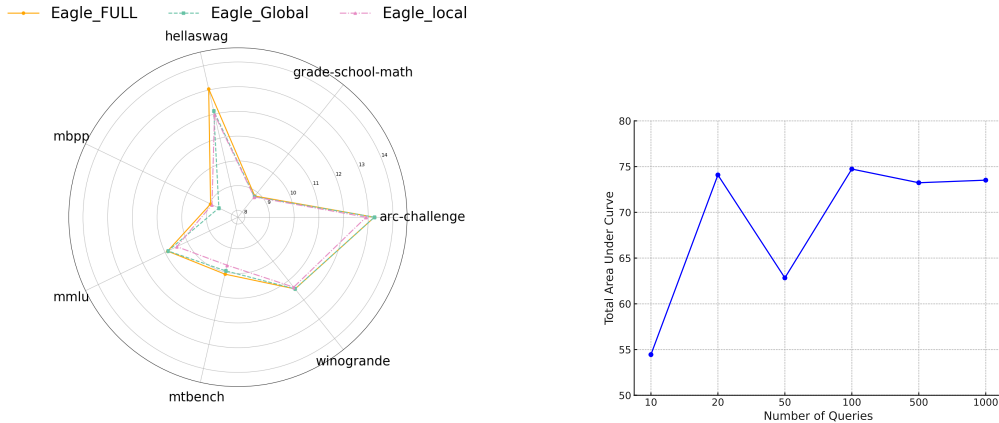
For the baseline methods, we used the following configurations:

- Common settings: neighbor size of 40 and cosine similarity as the distance function
- MLP: two layers with a hidden size of 100 and ReLU activation
- KNN: 40 nearest neighbors with cosine similarity
- SVM: LinearSVR with epsilon set to 0.0

B Ablation Studies for *Eagle*

To further understand the effectiveness of *Eagle*, we conducted ablation studies comparing the performance of *Eagle* with its individual components: *Eagle-Global* and *Eagle-Local*. Figure 4a illustrates the results of these comparisons.

Our findings reveal that neither *Eagle-Global* nor *Eagle-Local* alone can achieve optimal performance. *Eagle-Global*, while effective in capturing global information, lacks the ability to focus on different models’ specialized capabilities. Conversely, *Eagle-Local* excels at identifying model-specific strengths but may be biased due to limited sample sizes. The combination of *Eagle-Global* and *Eagle-Local* in *Eagle* leverages the strengths of both approaches, resulting in superior overall performance.



(a) Performance comparison of *Eagle-Global*-only, *Eagle-Local*-only, and *Eagle*.

(b) Effect of local neighbor size (N) on *Eagle-Local* performance.

Figure 4: Ablation studies for *Eagle* components and parameter sensitivity.

We also investigated the impact of the local neighbor size (N) on *Eagle-Local* performance. As shown in Figure 4b, we observed that when $N = 10$, *Eagle-Local* lacks sufficient information to make accurate predictions. However, increasing N beyond a certain point yields diminishing returns.

Our experiments indicate that $N = 20$ provides the optimal balance between performance and computational efficiency.

C Related Works

Large language models (LLMs) have demonstrated exceptional capabilities across a wide range of tasks. However, training and serving a single massive model is both costly and inefficient. Additionally, recent findings show that larger models do not consistently outperform smaller or specialized LLMs for all tasks. To address these issues, researchers are exploring multi-LLM approaches to enhance system performance while maintaining cost efficiency.

Mixture-of-Experts (MoE) and Ensemble Learning are two pivotal techniques for optimizing multi-LLM systems by leveraging multiple models to improve both performance and efficiency. Ensemble Learning, seen in systems like LLM Blender[13] and Blending Is All You Need[17], combines outputs from multiple models to enhance accuracy and robustness, albeit often at the cost of increased computational overhead. In contrast, MoE[20] activates only a subset of experts for each task, reducing computational demands by using only the most relevant models. While both approaches aim to boost LLM performance through the use of multiple models, MoE emphasizes scalability and resource efficiency, whereas Ensemble Learning focuses on robustness by combining model outputs. Nonetheless, challenges such as increased complexity in ensemble methods and potential inefficiencies in expert selection for MoE remain.

Router-based methods, including Route LLM[18], PolyRouter[22], hybrid LLM[7], and Intelligent Router for LLM Workloads[12], strive to enhance efficiency by dynamically routing queries to the most suitable model. These methods intelligently allocate tasks based on factors like task complexity, model performance, and system load, minimizing unnecessary computation and optimizing resource utilization. Route LLM focuses on matching queries to the most capable model, PolyRouter balances performance with cost, hybrid LLM tries to predict query complexity and route to most suitable models rather than singleton superior LLM, and Intelligent Router applies workload-aware scheduling to maximize throughput under heavy loads. While these approaches improve efficiency, they often introduce complexity in designing effective routing algorithms and managing real-time coordination among multiple models. To facilitate fair comparisons between routing strategies, benchmarks like RouterBench[11] and Large Language Model Routing with Benchmark Datasets[21] provide standardized metrics that assess performance, efficiency, and resource consumption.

However, many current systems still increase computational costs or require additional training when processing new user data. No existing approach can adaptively update route designation in real time based on a user’s recent queries.