SEMIROUTER: Sparse-Data Enhanced Routing for Adaptive Multi-LLM System

Zijie Wang [1], Xinyu Yan [1], Che Wang [1], Zeng Zihao [1], Lei Xiao[2], Wei Yang Bryan Lim [1]*

[1]College of Computing and Data Science, Nanyang Technological University
[2]Alibaba-NTU Global e-Sustainability CorpLab (ANGEL)

Correspondence: bryan.limwy@ntu.edu.sg

Abstract

Large Language Models (LLMs) exhibit remarkable capabilities, but no single model optimally balances serving quality and deployment cost across diverse tasks. Multi-LLM systems address this challenge through intelligent routing mechanisms that dynamically allocate queries to the most appropriate model. However, existing routing methods suffer from two fundamental limitations: ***(i)*** dependence on extensive full-response datasets for training, and ***(ii)*** poor scalability when incorporating new models, typically necessitating retraining from scratch. In this paper, we propose **SEMIROUTER**, a novel LLM routing framework designed for data-sparse and evolving model environments. Our approach combines a data-efficient training methodology with an adaptive architecture that enables seamless integration of new models under limited supervision. As an extension, we also consider energy footprint as a potential deployment cost in our routing decision. Empirical evaluations demonstrate that our method improves data efficiency, adaptability, and routing accuracy compared to existing approaches, providing a scalable solution for dynamic multi-LLM deployment.

1 Introduction

Large Language Models (LLMs) have transformed natural language processing, yet their capabilities remain uneven across tasks, as presented in Figure 1. Each LLM occupies a distinct position in the performance-cost trade-off space, where GPT-4 achieves superior reasoning at a premium price, while open-source alternatives like Llama 3.1 (Grattafiori et al., 2024) offer cost efficiency with domain-specific limitations. This heterogeneity undermines the assumption that a single LLM can effectively serve all computational needs.

Multi-LLMs serving employs dynamic routing to select LLMs based on queries to reduce computational expense without sacrificing output quality (OpenRouter, Inc., 2025; Ong et al., 2024; OpenAI, 2025). For example, an effective routing system might direct arithmetic queries to lightweight LLMs while reserving complex analytical tasks for larger LLMs. Consequently, the economic implications are substantial: routing can reduce inference costs by an order of magnitude for mixed workloads.

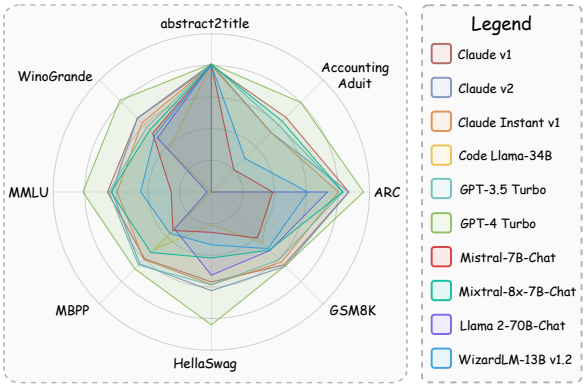

Figure 1: Radar chart of LLM performance comparison.

RouterDC (Chen et al., 2024), extended from CosineClassifier and ZOOTER (Lu et al., 2023), exemplifies current approaches, employing contrastive learning on top-k and bottom-k LLMs to enhance the training of the router. This method surpasses naive ensembling or cascading methods by requiring only a single LLM inference at deployment, yet the training process requires complete data. Therefore, current routing methods encounter several critical obstacles, as summarized below:

Costly Data Collection and Rapid Emergence of New LLMs. Given the rapid emergence of new LLMs, data acquisition represents the primary bottleneck for training the router, as existing methods require exhaustive annotation where each candidate LLM processes the a large query set (Chen et al., 2024; Ong et al., 2024; Ding et al., 2024; Zhao et al., 2024). For a modest configuration of

*Corresponding author

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 4910–4921

5 LLMs and 10,000 training queries, each training sample requires generating outputs from all candidate LLMs, followed by quality assessment through human evaluation or reward LLMs to construct the dataset (Chiang et al., 2024; Hu et al., 2024). The data augmentation process in (Ong et al., 2024) for a pair of LLMs costs about $700. This quadratic computational scaling with the number of LLMs renders comprehensive training infeasible for large LLM pools. In addition, routing LLMs naturally encounter data imbalance: popular LLMs such as the GPT series have rich training data, whereas less popular LLMs have few training queries in practice (Ong et al., 2024; Chiang et al., 2024).

Dynamic LLM Management. Current routers are architecturally inflexible and strongly coupled to their training-time LLMs set, requiring complete retraining when LLMs are added or removed. To achieve more accurate LLMs routing for a given list of LLMs and costs (API price for closed-source LLMs, compute cost for open-source LLMs), some routers (Tsiourvas et al., 2025) also include the company-determined query cost in training. These limitations make router training more frequent when the LLMs list or price changes (Jin et al., 2025). This coupling induces catastrophic forgetting when adapting to new LLMs, reducing performance on previously learned routing patterns.

Beyond Price as a Routing Target. While price is a convenient proxy and has dominated prior routing formulations, it represents only one dimension of deployment utility (Hu et al., 2024). In practice, operators must balance performance with multiple considerations, including latency, monetary cost, and energy consumption, given growing sustainability concerns. Price, being vendor-determined and volatile across product lifecycles, can be restrictive as a sole routing objective. A more flexible approach should be objective-agnostic, supporting diverse routing targets beyond price-based optimization.

The combined constraints of data acquisition overhead and integration inflexibility define the primary challenges for practical routing deployment. An efficient and adaptable routing framework of multiple LLMs is critical for real-world deployment in dynamic and data-sparse environments.

In this paper, we propose a novel framework, **SEMIROUTER**, for LLM routing (Figure 2). The framework consists of two components: a Backbone router trained on the full dataset using existing

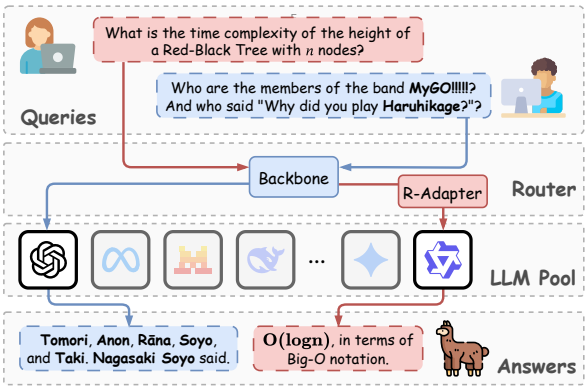

Figure 2: The workflow of our proposed general routing framework.

LLMs, and R-adapter, a lightweight module that enables efficient integration of new models without full retraining. Our approach achieves substantial cost reductions while maintaining task effectiveness. The key contributions are threefold:

- We propose a contrastive learning based training methodology to address sparse data collection while maintaining routing effectiveness in cost and task performance.
- We design an adaptive architecture (R-adapter) that allows seamless integration of new LLMs with minimal retraining, remaining effective even with insufficient data for new LLMs.
- We conduct extensive experiments to demonstrate that **SEMIROUTER** improves both data efficiency and LLM adaptability, while maintaining competitive routing performance. Moreover, we consider energy consumption as a case study to demonstrate that our routing method is objective-agnostic.

2 Related Work

Early approaches to LLM routing include deterministic methods based on predefined rules and probabilistic models that predict model performance. HybridLLM (Ding et al., 2024) builds a routing system between two models. Routing models can be classified as predictive routers and non-predictive routers. Non-predictive routers like cascading systems (Chen et al., 2023) ranked the LLMs and sequentially passed queries through models to find the best model under budget. While predictive methods such as ZOOTER (Lu et al., 2023) learn to route based on predicted quality or scores that include costs. RouterDC (Chen et al., 2024) proposed a query-based router utilizing dual contrastive learning sample-LLM and sample-sample

losses to select suitable LLMs. RouteLLM (Ong et al., 2024) trains the router by augmenting using golden-label datasets, evaluates based on human preference, and proves that data augmentation is expensive and effective in router training. Other recent advancements (Jin et al., 2025) further optimize the router through a multi-headed router and best-of-n sampling. While effective, these supervised routing models fundamentally rely on extensive "full response datasets" for training, which are impractical and costly to obtain at scale and require frequent backbone model retraining to adapt to new candidate LLMs.

3 Problem Statement

We consider a set of LLMs denoted by $\mathcal{M} = \{m_1, m_2, \ldots, m_J\}$. Among them, a subset $\mathcal{M}_A \subset \mathcal{M}$ represents the **anchor models** for which we already have abundant labeled observations, while $\mathcal{M}_S = \mathcal{M} \setminus \mathcal{M}_A$ denotes **additional models** with sparse training data coverage. For a given query x_i and model m_j, let y_i^j denote the model's response, $q_i^j \in [0, 1]$ its quality score, and c^j its associated cost (e.g., API price or energy consumption).

Ideally, we would have access to a fully observed dataset:

$$D^{\text{full}} = \{(x_i, y_i^j, q_i^j, c^j, m_j)\}_{i=1..N,\, j\in\mathcal{M}}, \quad (1)$$

which contains complete responses and labels for all models. In practice, however, only a smaller anchor dataset:

$$D^A = \{(x_i, y_i^a, q_i^a, c^a, m_a)\}_{i=1..N,\, a\in\mathcal{M}_A} \quad (2)$$

is available for training a backbone router R_{backbone}, while a sparse dataset:

$$D^S = \{(x_i, y_i^j, q_i^j, c^j, m_j)\}_{m_j\in\Omega}, \quad (3)$$

where $\Omega \subseteq \{1..N\} \times \mathcal{M}$, covers only a fraction of LLMs, especially for new models in $\mathcal{M}_S$.

Goal. Given a R_{backbone} trained on D^A, we aim to learn an adapted router R_{semi} using the sparse dataset D^S that ***(i)*** generalizes to the newly added models $\mathcal{M}_S$ without requiring the full dataset D^{full}, while ***(ii)*** introducing minimal additional parameters.

3.1 Scoring

For scoring, we use DeepSeek-R1 and GPT-4 as LLM-as-a-judge to evaluate model responses. We use DeepSeek-R1 to judge each response into ***good***, ***fair***, or ***bad*** and map as $(1, 0.5, 0)$ in quality score, and these labels are then aligned with human preference and GPT-4 scores to provide a more balanced quality evaluation. For the remaining datasets, we directly use the quality score provided where available. In the case of multiple choice benchmarks, the ground truth answer is taken as a *good* response and all other options as *bad*. For question answering datasets, the scoring follows the dataset protocol, with GPT-4 used as the judge when specified. This unified scoring pipeline ensures consistent evaluation across heterogeneous tasks, combining preference alignment from LLM-as-a-judge with objective ground truth signals.

3.2 Energy Cost

Apart from the model pricing schemes, we consider energy consumption cost as a factor considered during routing. Estimation of energy consumption follows previous works like Jegham et al. (2025) for large models such as GPT-4, and AIEnergyScore (2025) for smaller models. For models included in both evaluations, such as Llama variants, we use the intersection of results and align scaling with the AI Energy Leaderboard Standard to maintain comparability across models. Since Anthropic does not report the energy use of their models, Claude models are excluded from routing.

For GPT-3.5, no direct measurement is provided in either report. Therefore, we assume 0.2Wh per short query, half of the value of GPT-4o, consistent with the average reported by OpenAI of 0.3Wh per query in the main models. These values are based on inference deployment on A100 and V100 GPUs, as specified in the original papers. Because the routing strategy depends on the relative energy cost across the models rather than precise absolute numbers, and the energy consumption is huge between GPT-4 and those small models, these assumptions provide a sufficient and standardized starting point. The significant gap in consumption between small and large models dominates the comparison, making relative scaling more relevant than hardware-specific variation. To our knowledge, this is the first work to incorporate explicit energy assumptions into model routing.

4 Methodology

We present the framework of **SEMIROUTER** in Figure 3. It consists of two core components: a ***Back-***

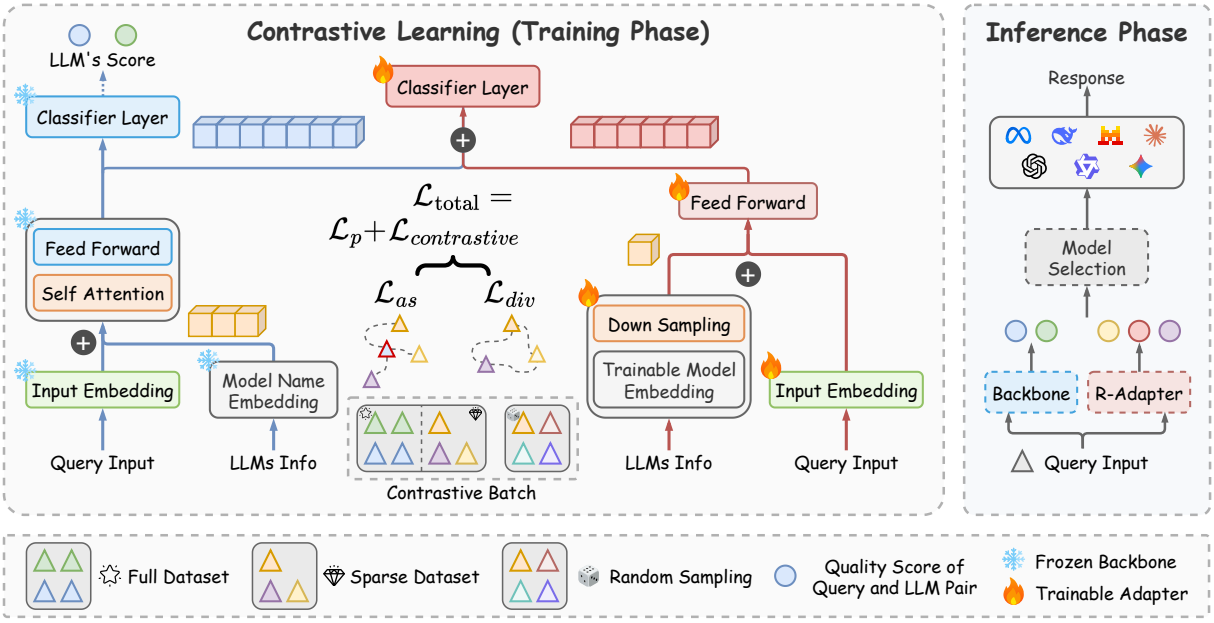

Figure 3: Overview of the proposed **SEMIROUTER** framework. **Left:** Training of the Backbone and R-Adapter; **Right:** Inference Flow.

bone for existing LLMs (denoted as the anchor model), which is trained on a rich, full dataset D^A, and a ***Routing Adapter*** (R-Adapter) for new LLMs, trained on a sparse dataset D^S. For the $R_{backbone}$, commonly used models include BERT, DeBERTa, and their variants (He et al., 2021; Devlin et al., 2019). Some approaches also employ ensemble strategies combined with clustering methods (Zhao et al., 2024).

Furthermore, the routing adapter is structured in three stages. First, a projection layer converts each candidate LLM into a trainable model information vector V_m, which captures the interrelationships between existing LLMs and new candidates, as well as the correlations among the new LLMs themselves. Second, the embedding of a query x_i is transformed into a query vector V_q, which is concatenated with the model vectors V_m. The concatenated vector is then passed through a feature extraction layer to match the hidden size of the backbone model, forming the query information vector V_f. Finally, the query information vector obtained from the R-Adapter is combined with the hidden vector V_b from the backbone. This combined representation integrates both the backbone inference and the adaptive correction, and is fed into a lightweight classification layer to produce the final quality prediction.

4.1 Backbone and R-adapter Structure

The training pipeline consists of two major phases: training the $R_{backbone}$ model using rich and complete training data, and training a sparse routing module for additional models with a batch of contrastive samples.

We first train a BERT-based backbone model (`bert-base-uncased`), which serves as $R_{backbone}$. The BERT encoder is optimized on training tuples consisting of a query x_i, its corresponding response y_i from the LLM m_a, and the associated quality score q_i^a. The quality score acts as the supervision label, while the BERT encoder provides contextualized representations of the inputs. The objective of this stage is to learn a robust mapping from queries to quality scores using a standard cross-entropy loss.

Then, during the training of the R-Adapter in **SEMIROUTER**, the parameters of $R_{backbone}$ are frozen, and a dual-path structure is introduced to enhance the router's capability. The intermediate representations from the frozen BERT backbone serve as the primary (anchor) feature point. In parallel, the model index is projected into a dense vector that captures the interrelationships among anchor and sparse models. This model vector is concatenated with the BERT embedding of x_i and passed through a small feed-forward network to obtain a representation aligned with the dimensionality of the BERT intermediate layer. The resulting

vector is concatenated with the frozen BERT feature, and the combined representation is fed into a classification head to predict the final quality score. We further explore adaptive parameterization with different structural variants, including a multi-head attention design similar to Jin et al. (2025), and simpler feed-forward architectures with or without the model projection vector V_m.

4.2 Anchor Based Contrastive Learning

The training objective of the R-Adapter combines a contrastive loss with a cross-entropy loss to guide the adaptive learning of parameters. The contrastive term, denoted as **Delta** in Figure 4, encourages the model to capture the relative differences between the anchor and new LLM predictions, thereby enhancing the discriminative capability of the learned routing parameters. Below, we present our approach from two complementary perspectives: ***(i)* Training Pair Construction** and ***(ii)* Contrastive Loss Design**.

Contrastive Training Pair. To train the adaptive parameters, we adopt a contrastive batch construction strategy (Figure 4). For each batch, a query x_i is selected, and a contrastive batch is formed as $B_c = (D_i^A, D_i^S, D^R)$. The anchor reference set $D_i^A = \{x_i, q_i^a, c_i^a, m_a\}$ contains all anchor-model training samples associated with x_i. The sparse-model set $D_i^S = \{\{x_i, q_i^s, c_i^s, m_s\} \mid x_i \in X_I, \exists q_i^s\}$, includes samples from newly introduced sparse models that have data for the same query.

Moreover, to maintain diversity and avoid query-specific bias, we randomly sample set D^R, consisting of training data from other queries, following RouterDC's sampling strategy. In each batch, the sparse-model samples D^S are paired with the corresponding anchor-model outputs from the pretrained $R_{backbone}$, ensuring a stable supervision signal.

Anchor to Sparse Contrastive Loss. For a contrastive batch B_c of anchor–sparse pairs (D_i^A, D_i^S, D^R), the anchor–sparse loss is:

$$\mathcal{L}_{\text{a}}(x_i; M^a, M^s) = \text{ReLU}(Margin - \left|\hat{q}(M_i^a) - \hat{q}(M_i^s)\right|) \tag{4}$$

The contrastive loss of query from anchor LLM to new LLM is calculated as Equation 4, then for all pairs of queries. The total contrastive loss can be calculated as Equation 5:

$$\mathcal{L}_{\text{A}}(x_i; M^A, M^S) = \frac{\alpha}{|B_c|} \sum_{D_i^A, D_i^S \in B_c} \mathcal{L}_{\text{a}}(x_i; M^a, M^s) \tag{5}$$

$$\forall a \in A \quad and \quad s \in D_i^S \tag{6}$$

Model Diversity Loss. While anchors guide alignment, it is equally important that sparse models remain diverse. Without explicit encouragement, the router may collapse to similar outputs for all sparse models, limiting complementary strengths. To address this, we introduce a small diversity loss that penalises sparse predictions that are too similar and keeps its influence smaller than contrastive loss with a scale of 0.1. For each pair (M_i^s, M_i^s) in $B_s \subseteq B_c$, we compute it as follows:

$$\mathcal{L}_{\text{div}} = \frac{0.1}{\binom{|B_c^S|}{2}} \sum_{i<j} \text{ReLU}(\delta - \left|\hat{q}(M_i^{s_1}) - \hat{q}(M_i^{s_2}))\right|) \tag{7}$$

where δ is a small threshold, same as margin in L_s, and a mild weight (*e.g.*, 0.1) to avoid excessive penalty. This ensures sparse models preserve distinguishable behaviors and improve generalization.

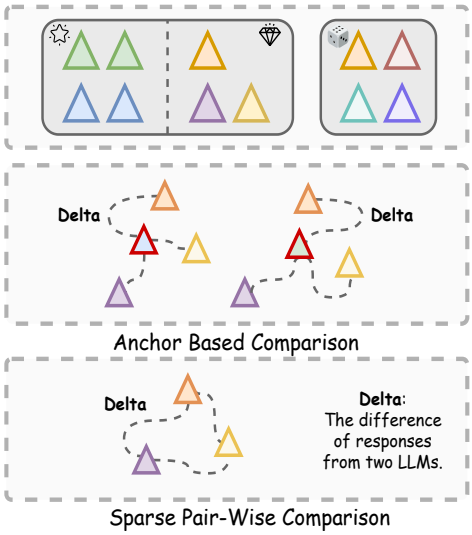

Figure 4: Training batch consists of contrastive pairs with anchor to sparse contrastive loss **(Left)** and model diversity loss **(Right)** to capture the difference between existing models and new models.

Combined Contrastive Objective. The anchor–sparse loss and diversity loss are combined:

$$\mathcal{L}_c(x_i; \theta) = \mathcal{L}_{\text{A}}(x_i) + \mathcal{L}_{\text{div}}(x_i) \tag{8}$$

The overall training objective integrates the supervised prediction loss $\mathcal{L}_p$ on anchors with the contrastive term using weight λ:

$$\mathcal{L}_{\text{total}} = (1 - \lambda)\, L_p + \lambda\, \mathcal{L}_c \tag{9}$$

This formulation transfers reliable supervision from anchors, maintains diversity among sparse models, and enables the router to achieve robust and adaptable routing under limited supervision.

Table 1: Performance across datasets for different routing methods. Notes: ● means training with data augmentation.

Method	Augmentation	abstract2title	ARC	bias_detection	GSM8K	HellaSwag	MBPP	MMLU	WinoGrande
BERT (Devlin et al., 2019)	●	99.60	77.51	77.36	62.03	65.25	62.73	76.54	59.66
ZOOTER (Lu et al., 2023)	○	99.60	78.23	77.93	57.85	**65.31**	57.03	76.06	**59.86**
LoraRetriever (Hu et al., 2022)	○	99.60	73.92	77.24	61.03	64.77	**62.26**	75.58	58.84
RouterDC (Chen et al., 2024)	○	99.60	78.18	77.93	61.37	64.15	61.56	75.98	55.51
RadialRouter (Jin et al., 2025)	○	99.60	78.07	77.58	61.32	64.23	61.25	75.74	57.53
SEMIROUTER (MLP)	○	99.60	78.14	77.82	61.33	64.35	61.02	75.93	57.93
SEMIROUTER (EMB)	○	**99.60**	**78.57**	**77.93**	**61.46**	64.25	61.88	**76.06**	57.74

4.3 Inference

We present the inference phase in the right part of Figure 3. The existing LLMs $\mathcal{M}_A$ that are pretrained on sufficient data still use the backbone to assess quality. The new LLMs $\mathcal{M}_s$, would rely on the R-Adapter trained by a sparse dataset, which takes the pretrained backbone features as input and passes them through the adaptive path head to generate predicted quality scores. This design allows the backbone to remain frozen and provide stable features while new models can be trained or updated incrementally without affecting the original routing model. The predicted quality scores from both paths are then used in the model selection stage to choose the most cost-effective model with the highest predicted quality.

Table 2: Comparison of trainable parameters across router variations.

Variation	Hidden Size	Trainable Params	Total Params
BERT	1536	109.6M	109.6M
RaidialRouter (ATT)	1536	65.7M	175.2M
SEMIROUTER (MLP)	1536	2.53M	112.0M
SEMIROUTER (EMB)	1536	**2.07M**	111.6M

5 Experiments

5.1 Experiment Settings

Candidate LLMs. We evaluate routing performance across 7 diverse large language models, covering both open-source (*e.g.*, Mistral (Jiang et al., 2023), Llama (Touvron et al., 2023), Code Llama (Roziere et al., 2023)) and proprietary systems (*e.g.*, GPT (Brown et al., 2020; Achiam et al., 2023), Claude). These models range from 7B to 70B parameters and encompass varied architectures and training objectives, ensuring our evaluation reflects heterogeneity across model families. Details of the models are provided in Appendix B.

Evaluation Datasets. We evaluate on 6 representative large-scale benchmarks: ARC (ARC-Challenge) (Clark et al., 2018), HellaSwag (Zellers et al., 2019), GSM8K (grade-school math) (Cobbe et al., 2021), MBPP (Austin et al., 2021), MMLU (Hendrycks et al., 2020), and WinoGrande (Sakaguchi et al., 2021), together with two low-resource datasets, abstract2title and Bias Detection. We mixed six large datasets and two small datasets to simulate query-level concentration: some tasks naturally produce many labelled examples while others produce fewer labelled examples. This reproduces the common production pattern where certain domains dominate the training distribution. For each dataset, we use the official training and test splits when available; otherwise, we randomly sample 70% of the data for training and hold out 30% for testing. All training subsets are combined into a unified training set $\mathcal{D}^{\text{A}}$, which is used to train the router.

We simulate a realistic sparsity setting by randomly downsampled response evaluations from training data for D^s to around 2 samples per query. Training data D^A for Anchor models are kept as original, where frequently used models have more available annotations than less common ones.

Implementation Details. All experiments are conducted on a server equipped with an NVIDIA RTX 6000 Ada GPU. For SEMIROUTER and RaidalRouter use finetuning, we trained for 2 epochs with a batch size of 16 and a learning rate of 1×10^{-5}. SEMIROUTER hyperparameter is set as $\alpha = 0.3$ and $\lambda = 0.5$. For the other baseline full fine-tuning is applied, we trained the model for 20 epochs using a batch size of 16 and a learning rate of 5×10^{-5} with a batch size of 16. We also use the `AdamW` optimizer.

To enable fair comparison across methods that utilize query embeddings (such as LoraRetriever or RouterDC, which involve clustering or similarity-based routing), we adopt a unified encoder for embedding generation. Specifically, we use the `gpt-embedding-small` model to convert each query into a fixed-dimensional vector representation, ensuring consistency in input features across

all routing approaches. Moreover, various routing methods employ different backbone architectures, including BERT, Llama, and DeBERTa-v3. To standardize our experiments and align with commonly used practices, we select BERT as the default router backbone. This choice allows for a consistent basis when comparing routing performance across different methods in our study.

5.2 Results Evaluation

We evaluate our framework across multiple routing benchmarks covering diverse tasks and query distributions. Performance is measured in terms of predicted quality, cost efficiency (inference cost) under sparse training settings.

Comparison with Baselines. We first compare our routing framework, including the full-finetuned BERT-base router, ZOOTER, and cluster-based methods such as LoraRetriever, RadialRouter, which use multiple attention heads, and RouterDC with sample-sample loss, because the top-k router required is not available under a sparse dataset. As shown in Table 3, for all queries used in training Bert, we augmented D^S by adding the quality scores of all missing new model samples for each query, resulting in a more comprehensive training dataset. The remaining baselines were trained on the original D^S without this augmentation. ZOOTER requires full training. LoraRetriever requires less training by exploiting model clustering. We use the same approximation of ZOOTER and Loraretriever. For RadialRoute, we freeze the BERT encoder path to perform finetuning. For RouterDC, we use our **SEMIROUTER** with the sample-sample contrastive loss as a comparison. Beyond baseline methods, we analyze three architectural branches within our proposed framework: ***(i)*** directly using the LLM embedding vector (denoted as **MLP**), ***(ii)*** enhancing the embedding vector with a learned projection layer (denoted as **EMB**), and **(iii)** incorporating an attention layer to adaptively weight embeddings during sparse training, which is treated as RadialRouter variants.

Efficiency of Adaptive Path. A key strength of our design lies in the adaptive path architecture. We progressively reduce the trainable parameters to perform the model routing and observe that our **SEMIROUTER** maintains strong performance with the smallest total trainable parameter without an obvious decrease in the quality of prediction accuracy. As shown in Table 2, **SEMIROUTER** (EMB) variant achieves the lowest parameter size (2.07M). The adaptive path overall contains only about 2% of the parameter size of the backbone router, making it lightweight and fast to train. We also implemented pure MLP, which does not contain the downsampling module in the adaptive layer, and RadialRouter with multiple attention head finetuning using sparse training data. For RadialRouter, we finetuned the attention head of the router, and it could serve as a parameter-dense variant of our method with approximately half of the training parameters of the backbone, as shown in the Table 2.

During inference, hidden representations from the frozen BERT backbone can be cached once and shared across all candidate models, allowing the adaptive path to run in parallel for each new LLM without repeated backbone computation. This parallelizable structure not only reduces latency but also enables efficient adaptation when integrating additional candidate LLMs, making the framework particularly well-suited for online or resource-constrained deployment scenarios.

Table 3: Cost, quality, and score across datasets for different adaptive structures.

Dataset	SEMIROUTER (Div)			SEMIROUTER (Anchor)			SEMIROUTER		
	Quality	Cost	Score	Quality	Cost	Score	Quality	Cost	Score
ARC	62.58	1.322	35.37	64.63	1.4421	**36.35**	64.17	1.515	35.91
GSM8K	53.75	1.339	31.61	54.70	1.002	**32.53**	54.00	1.242	31.86
HellaSwag	53.93	76.23	30.53	57.48	78.81	32.62	57.51	77.74	**32.68**
MBPP	50.00	3.664	27.79	52.34	5.062	28.13	53.12	5.199	**28.50**
MMLU	59.94	160.64	32.93	61.99	167.21	34.03	63.37	173.49	**34.73**
WinoGrande	53.54	0.6336	32.66	55.12	0.7194	**33.60**	55.12	0.7168	**33.60**

Ablation of Contrastive Loss. We further evaluate the effectiveness of contrastive batch in router training in Table 3. We calculate it as follows:

$$\text{Score} = 0.6 \times \text{Quality} - \frac{c_i}{|X_i|} \tag{10}$$

where $|X_i|$ denotes the number of queries. to evaluate the routing performance considering both output quality and query cost. Encouraging diversity among new LLMs using diversity loss alone does not provide significant improvement if used without anchor contrast. The anchor contrast only accentuates differences between anchor models and newly introduced LLMs, improving robustness in the presence of unseen candidates. Table 3 shows that anchor contrast yields more improvements over baselines, with the combined variant striking the best balance for more datasets between cost efficiency and accuracy. Notably, contrastive training allows the router to generalize better to low-resource settings, where direct retraining is limited.

Scalability under Variable Candidate Pools. We also conduct an ablation over the number of available LLMs to assess scalability. As shown in Table 4, performance remains stable even when subsets of models are removed, confirming that the router adapts flexibly to variable candidate pools. With new models, quality prediction is only stored in the parameter-efficient adaptive path, increasing the model or decreasing the model on the new models does not require retraining the whole router.

In the R-Adapter, if the LLM is only represented in the lightweight R-Adapter, forgetting can be avoided easily. Options include retraining the small R-Adapter (only 2% parameters), or removing the trainable model embedding corresponding to that LLM. The R-Adapter thus provides a cheap, reversible intermediate state for the router, enabling fast deployment and reducing the need for costly backbone retraining when new models are added or removed.

Table 4: Quality on variable models benchmarks.

Model	MBPP	MMLU	ARC	HellaSwag	Winogrande
-w/o Code-Llama-34B	0.5391	0.6313	0.6395	0.5695	0.5486
-w/o WizardLM-13B	0.5625	0.6918	0.8231	0.5755	0.5486
-w/o Llama-2-70B	0.5781	0.6925	0.8231	0.5702	0.5591
-w/o Mixtral-8x7B	0.6641	0.6529	0.8141	0.5904	0.5459
-w/o Claude-Instant	0.6719	0.6982	0.8367	0.6011	0.5354
-w/o Mistral-7B	0.6719	0.7018	0.8367	0.6004	0.5617

Energy Saving using Routing Model. We evaluate the energy footprint of routing decisions, where energy consumption reflects more on LLM's resource consumption than price signals. Also, energy consumption is particularly relevant for sustainable deployment, amid growing environmental concerns in LLM deployment. To quantify energy cost, we use model-level energy consumption data reported in the **AIEnergy Leaderboard** as a proxy for per-query inference cost in model selection. Since some models (*e.g.*, Claude) do not publicly release energy statistics, we exclude them from this part of the evaluation. As shown in Table 5, our approach effectively reduces energy consumption, lowering both economic and environmental costs.

GPT-4 as Backup Cost. When GPT-4 is expected to provide a higher-quality answer, the query is rerouted and executed with GPT-4 as a backup. If the answer quality of GPT-4 is higher than the LLM selected, we add the power consumption of GPT-4 for this query. We find that the energy cost is still smaller than all on GPT-4. This approach improves answer quality while reducing the average cost, since some queries are processed by both the smaller model and GPT-4.

Table 5: Energy cost, percentage savings, and GPT-4 as backup cost.

Dataset	Energy Cost (Wh)	Saving	Backup Cost (Wh)
abstract2title	0.164	98.78%	6.846
Accounting Audit	3.028	77.61%	8.278
ARC	0.231	98.29%	6.880
HellaSwag	1.346	90.06%	7.437
MBPP	1.656	87.76%	7.592
MMLU	4.143	69.38%	8.836
WinoGrande	0.474	96.50%	7.001
All at GPT-4	13.529	–	13.529

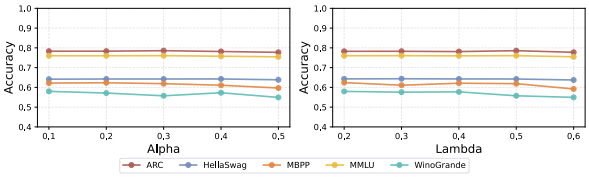

Figure 5: Sensitive analysis of α and λ for contrastive loss.

Sensitivity to Contrastive Loss Weight. We further analyze the sensitivity of our approach to the weighting of the contrastive loss in Figure 5. We evaluted the performance over λ from 0.2 to 0.5 to for L_c and α from 0.1 to 0.5 for L_A. Based on our sensitivity study in figure 5, $\lambda < 5$ to prevent the contrastive term from dominating and destabilizing training. In practice, $\lambda \in [0.1, 0.3]$ works consistently well. For $\alpha = 0.3$ is a balanced choice: larger α places more emphasis on distinguishing new models from the anchors (new LLMs more different from exsiting model), while smaller focuses on separating new models from each other(new LLMs are more different among each other). Our router maintains stable performance across a range of hyperparameter settings, without requiring fine-grained tuning. The results show that moderate variations in the contrastive weight yield consistent improvements over non-contrastive training, but the contrastive loss should be smaller than L_P. This robustness highlights that the contrastive objective acts as a general regularizer, enforcing diversity among candidate LLMs and improving generalization to unseen models, rather than being overly dependent on precise hyperparameter tuning.

6 Conclusion

We present **SEMIROUTER**, a data-efficient and adaptive routing framework that addresses the

challenges of costly data collection and inflexible LLM integration. Our approach combines contrastive learning-based training with a lightweight R-adapter module, enabling effective routing under sparse supervision and seamless integration of new models without retraining the backbone router. This design ensures compatibility with other routing methods and adaptability to evolving LLM ecosystems. Beyond cost optimization, we demonstrate the framework's objective-agnostic capability through energy consumption as a routing target, addressing growing sustainability concerns and data center power constraints. Extensive experiments validate that **SEMIROUTER** achieves superior data efficiency and model adaptability while maintaining competitive routing performance. These results establish a foundation for scalable and sustainable multi-LLM deployment in dynamic production environments.

Limitations

This paper provides an evaluation of routers performance focusing on the quality metric of the routing model. With the limited ability to annotate expensive new datasets, we resampled the existing dataset for simulation. In addition, since this is the first work to mention energy consumption as a routing metric, there are limited statistics for LLMs' energy consumption for energy cost evaluation. Moreover, although training using cost leads to certain advantages in routing, we focus more on quality prediction rather than quality–cost selection during training to provide a fairer comparison of router design methods and avoid overfitting to specific cost patterns.

Acknowledgments

This research is supported by the RIE2025 Industry Alignment Fund-Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore through Alibaba-NTU Global e-Sustainability CorpLab (ANGEL). This research is also supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award MOE-T2EP20125-0005).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AIEnergyScore. 2025. Ai energy score leaderboard. https://huggingface.co/spaces/AIEnergyScore/Leaderboard. Accessed: 2025-10-03.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios N. Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pages 8359–8388. JMLR.org.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid

llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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. *arXiv preprint arXiv:2403.12031*.

Nidhal Jegham, Marwan Abdelatti, Lassad Elmoubarki, and Abdeltawab Hendawi. 2025. How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference. *arXiv preprint arXiv:2505.09598*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ruihan Jin, Pengpeng Shao, Zhengqi Wen, Jinyang Wu, Mingkuan Feng, Shuai Zhang, and Jianhua Tao. 2025. Radialrouter: Structured representation for efficient and robust large language models routing. *arXiv preprint arXiv:2506.03880*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

OpenAI. 2025. Introducing gpt-5. https://openai.com/index/introducing-gpt-5/. Accessed: 2025-10-03.

OpenRouter, Inc. 2025. Openrouter: The unified interface for llms. https://openrouter.ai/. Accessed: 2025-10-03.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Asterios Tsiourvas, Wei Sun, and Georgia Perakis. 2025. Causal llm routing: End-to-end regret minimization from observational data. *arXiv preprint arXiv:2505.16037*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. 2024. Loraretriever: Input-aware lora retrieval and composition for mixed tasks in the wild. *arXiv preprint arXiv:2402.09997*.

Appendix

A Scoring Details

Answer quality is evaluated using GPT-4 and Deepseek-R1, following the evaluation setup defined in the datasets. For tasks with ground-truth labels, such as MMLU, responses are classified into ***good*** or ***wrong*** according to whether they match the ground-truth answer. Since the dataset itself specifies that GPT-4 is used as the reference evaluator, majority agreement is determined against this standard. For open-ended tasks, a fixed evaluation prompt is used to ensure consistency across models. Responses are categorized into three levels: ***(i) wrong***, when the answer contains factual or reasoning errors; ***(ii) fair***, when the answer is correct but suboptimal in clarity or completeness; and ***(iii) good***, when the answer is correct, clear, and near-optimal. To guarantee stable results, evaluation is performed with the temperature set to 0, and only a single evaluation pass is used. In cases where model-generated judgments and ground-truth data diverge, human verification is applied to resolve mismatches. This procedure ensures both reproducibility and alignment with the dataset's evaluation protocol.

B Model Selection

Based on the ranking in Table 6, we select GPT-4, GPT-3.5, and Mistral-7B as anchor models for routing. This set is chosen to represent different capability tiers: GPT-4 as the strongest model, GPT-3.5 as a mid-tier model with substantially lower cost, and Mistral-7B as a lightweight baseline. Together, they capture the performance–efficiency spectrum more effectively than using models from a single family.

Claude models, although ranked highly, are excluded due to the lack of reported energy usage data. By contrast, the selected models have either reported or reasonably approximated energy consumption, making them suitable for energy-aware routing. Their relatively small variation in normalized scores also contributes to stability, avoiding anchors that fluctuate significantly across tasks. While our analysis is limited to the models in Table 6, the anchor set is expected to provide a reasonable basis for routing decisions that may extend to other models of comparable scale.

C Code Details

Table 6: Model ranking based on normalized scores.

Rank	Model	Score
1	gpt-4-1106-preview	0.915
2	claude-v2	0.800
3	gpt-3.5-turbo-1106	0.741
4	claude-v1	0.724
5	claude-instant-v1	0.667
6	mistralai/mixtral-8x7b-chat	0.655
7	meta/llama-2-70b-chat	0.283
8	WizardLM/WizardLM-13B-V1.2	0.278
9	mistralai/mistral-7b-chat	0.273
10	meta/code-llama-instruct-34b-chat	0.180

C.1 Implementation Details

The overall model is constructed by combining (i) a frozen BERT encoder for contextual text representations, (ii) a variant-specific Adaptive path processor for embedding and LLM identity features, and (iii) a fusion MLP for final classification. Below, we summarize the implementation choices for each component.

C.2 Backbone and Fusion

The BERT encoder is trained using the full training samples of anchor model data with cross-entropy loss from the bert-base-uncased model and frozen during training to preserve pre-trained feature extraction ability. Its pooled 768-dimensional outputs are cached once and reused across all routing candidates, significantly reducing computation. The fusion module is a three-layer feed-forward network that accepts the concatenation of BERT features and Adaptive path outputs, with ReLU activations and dropout regularization. This module remains identical across all variants.

C.3 Adaptive Path Variants

MLP Variation. The MLP variant is realized as a two-layer fully connected network. The 1600-dimensional concatenated input (pre-computed embeddings plus LLM identity embedding) passes through a hidden layer of size 512 followed by a second hidden layer of size 256. Each layer uses ReLU activation with dropout applied between layers. This design yields approximately 2.53M trainable parameters and serves as a balanced, computationally efficient baseline.

Embedding-MLP Variation. To improve efficiency, the Embedding-MLP reduces dimensionality before processing. The 1536-dimensional

Algorithm 1 Anchor-based model router training with contrastive batches

Input: LLMs $\{M_j : j = 1, \ldots, J\}$

Parameter: Anchor model list $\{M_a : a = 1, \ldots, A\}$

Output: Predicted quality score

Train the router on anchor model data with prediction loss $L_{cross\ entropy}$.
for each epoch, for each batch of anchor data **do**
    Update router using $L_{cross\ entropy}$
end for
Construct contrastive batches between anchor and non-anchor models.
for each contrastive batch B_c **do**
    Initialize router with pretrained anchor weights
    Compute prediction loss $L_{cross\ entropy}$
    Compute anchor to sparse contrastive loss L_{as}
    Compute model diversity loss L_{div}
    Combine: $L_{contrastive} = \alpha(L_{as}) + L_{div}$
    Final objective: $L_{\text{total}} = (1 - \lambda)L_{contrastive} + \lambda L_{cross\ entropy}$
end for
Use the trained router to predict quality scores for routing.
Inference:
For a query x_i, compute the hidden representation from the frozen backbone.
For existing models $M_j \notin M_A$: predict using backbone path only.
For new models with sparse data: use adaptive path features and a classification head.
Output predicted quality scores and route to the model with the best cost-quality trade-off.

embeddings are linearly projected to a 128-dimensional space, which is then concatenated with a learnable 128-dimensional model vector and a 64-dimensional LLM identity embedding. This compact 320-dimensional representation is processed by a lightweight MLP similar in structure to the standard variant. By reducing input dimensionality, this design achieves only 2.07M trainable parameters, an 18% reduction relative to the plain MLP, while retaining factorization-based learning capacity.

Transformer Variation. The Transformer variant replaces the MLP with a multi-head self-attention processor. The concatenated 1600-dimensional input is first linearly projected to the model dimension (1536), followed by multiple Transformer encoder layers with 8 attention heads. Residual connections, layer normalization, and feed-forward sublayers are applied in standard fashion. The outputs are pooled to obtain a fixed-size representation. This design enables complex feature interactions at the cost of significantly higher parameter usage (65.7M trainable), making it most suitable for resource-rich scenarios.

For LoraRetriever, we explored different numbers of clusters among $\{10, 20, 40\}$ and report results using 40 clusters for all comparisons, as this yielded the best performance.

D Ethical Considerations

This paper presents work whose goal is to advance the field of router training. There are many potential societal consequences of our work, none of which we feel will leads to potential risk that needs to be specifically highlighted here.