# Confidence Estimation via Decoupled Smoothing for Dynamic LLM Routing and Aggregation

**Hao Luo**[1] **Ning Cheng**[1] **Yuhao Lin**[1] **Xiaokai Zhou**[1]
**Yongqiang Zhang**[2] **Xiao Yan**[3]* **Jiawei Jiang**[1,4]*
[1]School of Computer Science, Wuhan University
[2]Dameng Database Co., Ltd., Wuhan, China
[3]Institute for Math and AI, Wuhan University
[4]School of Computer Science, Central China Normal University
{lohozz,alian_c,yuhao_lin,xiaokaizhou}@whu.edu.cn
zyq@dameng.com, {yanxiaosunny,jiawei.jiang}@whu.edu.cn

## Abstract

The rapid emergence of heterogeneous large language models (LLMs) has highlighted the limitations of static single-model inference systems. Although model routing alleviates this issue by assigning the most suitable model to each query, existing methods often struggle to stably align query semantics with actual model performance and remain constrained by the capability ceiling of a single model. On the other hand, although multi-model aggregation can improve performance on complex queries, it also introduces extremely high computational overhead for routine tasks. In this paper, we propose PARS, a model confidence estimation method for dynamic LLM routing and aggregation. PARS first decouples the feature representation space and applies structured smoothing to local performance signals, producing more discriminative confidence score estimates for candidate models. On this basis, PARS represents inference modes such as direct routing, self-consistency sampling, and multi-model voting as test-time compute allocation actions under a budget constraint. By extracting state features such as routing confidence distributions and predictive uncertainty, a lightweight selector adaptively assigns an inference strategy to each query. Extensive experiments show that PARS improves average accuracy over the best baselines by 1.02% and 1.36% in the single-model and multi-model settings, respectively, while reducing average inference cost by 1.73× and exhibiting stable out-of-distribution generalization. The code is available at https://github.com/lohoz/PARS.

## 1 Introduction

As the number of available models continues to grow, practical systems can often access a pool of candidate models that differ in training objectives, capability profiles, and inference costs [18, 19, 58, 40]. At the same time, user queries are highly heterogeneous: some inputs are closer to factual retrieval, some require complex mathematical or symbolic reasoning, and others depend more on code generation or domain-specific knowledge [34, 52, 27]. Prior studies repeatedly show that different models have different strengths across queries, so using a fixed model for all inputs rarely remains optimal across scenarios [32, 62, 57, 60]. Existing work mainly exploits such model heterogeneity along two directions: model routing attempts to select a more suitable single model for each query, while multi-model aggregation integrates responses from multiple models to improve robustness on complex queries.

*Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In model routing, existing methods estimate the matching relation between the query and candidate models through semantic-neighborhood retrieval [61], preference learning [42], query–model representation matching [8, 63], or graph-based modeling [14]. For multi-model aggregation, common approaches include repeated-sampling consistency [9], candidate-answer voting [6], proposer–aggregator collaboration [50, 28], and expert coordination for complementary capabilities [5]; recent methods further introduce route-and-aggregate workflows, such as response-aware rerouting and route-or-aggregate switching after candidate responses have been generated [47]. These methods improve the flexibility of multi-model collaboration, but their dynamic decisions typically occur after the expert set has been determined or after candidate responses have already been generated, and thus still require multiple candidate models to be called for each query. In contrast, a better alternative is to reliably estimate the expected performance of candidate models on the current query before response generation, so as to inform model selection and multi-model collaboration.

However, when estimating the compatibility between a query and model capabilities, many methods still largely rely on pretrained text embeddings or retrieval based on semantic similarity [42, 61, 29]. Such semantic neighborhoods can capture topical or surface-level similarity among inputs, but they do not necessarily reflect whether the corresponding model-capability requirements are similar. Relying only on semantic similarity may therefore place queries with different capability requirements into similar neighborhoods, weakening the estimation of model performance. This makes reliable estimation of candidate models' expected performance a key prerequisite for dynamic LLM routing and aggregation. We therefore focus on the following research question:

*How can we estimate reliable query-level model confidence and use it to jointly guide LLM routing and aggregation?*

To this end, we propose PARS, a model confidence estimation method based on decoupled smoothing for dynamic LLM routing and aggregation. **(i)** To address the difficulty of stably aligning query semantics with actual model performance in existing methods, PARS decouples the feature representation space and applies structured smoothing to local performance signals, producing more discriminative confidence estimates for each candidate model. **(ii)** Based on this stabilized model-confidence estimate, PARS further designs a budget-aware dynamic inference mechanism, representing inference modes such as direct routing, self-consistency sampling, and multi-model voting as test-time compute allocation actions under a budget constraint. Specifically, PARS uses a lightweight selector to adaptively assign an inference strategy for each query according to state features such as the routing confidence distribution and predictive uncertainty. This design allows PARS to maintain efficient single-model inference on most routine queries, while dynamically introducing multi-model voting or self-consistency sampling for difficult inputs, thereby making more effective use of complementary strengths in a heterogeneous model pool under controlled test-time compute overhead.

To evaluate PARS, we compare it with 10 baselines, including RouterDC [8], EmbedLLM [63], MoA [50], and JiSi [47], on 19 datasets spanning five domains, with 20 open-source LLMs as candidate models. The results show that, in the single-model setting, PARS improves average accuracy by 1.02% over the strongest routing baseline. In the multi-model setting, PARS improves average accuracy by 1.36% over the strongest baseline under the same setting while reducing average inference cost by $1.73\times$, and it also exhibits stable generalization in out-of-distribution scenarios.

Overall, our contributions are summarized as follows:

- We propose a model confidence estimation method based on decoupled smoothing. By decoupling the feature representation space and applying structured smoothing to local labels, it mitigates the mismatch between textual semantics and actual model capabilities, improving both the accuracy of single-model selection and the reliability of confidence estimation.
- We design a budget-aware dynamic inference mechanism. It represents candidate inference modes as an action space constrained by a test-time compute budget, and uses a lightweight selector to adaptively switch among single-model inference, self-consistency, and multi-model voting.
- We demonstrate the effectiveness of PARS through extensive cross-domain experiments. In both the single-model and multi-model settings, PARS consistently outperforms existing baselines.

## 2 Related Work

**Single-Model Routing.** Single-model routing methods estimate query-level model suitability by selecting, from a candidate model pool, the model most likely to produce the correct answer for each query. EmbedLLM [63] supports query–model matching by learning compact model representations, while Avengers [61] explores how a training-free allocation mechanism can more effectively utilize a candidate model pool. RadialRouter [24] further emphasizes structured representations to improve routing efficiency and robustness. Other methods make the routing process more explicit and structured: RouterDC [8] uses dual-contrastive learning to align query and model representations, GraphRouter [14] organizes relations among queries, tasks, and models as a graph, and MODEL-SAT [59] improves model characterization through capability-based representations. In addition, Hybrid LLM [13] and RouteLLM [42] also learn query-level single-model selection policies, but place more emphasis on hybrid decision-making, preference prediction, or quality–cost trade-offs. Our model confidence estimation aims to obtain a more reliable score structure over candidate models, providing a finer-grained query-level signal for subsequent inference decisions.

**Multi-Model Aggregation and Dynamic Collaboration.** Multi-model aggregation methods focus on response-level complementarity, improving final output quality by integrating candidate responses from different models or generation trajectories. Majority-voting schemes [6] and self-consistency decoding [9] represent direct aggregation strategies based on voting and repeated sampling, respectively, while LLM-Blender [22] integrates multiple candidate answers through pairwise ranking and generative fusion. Subsequent work begins to adopt more structured aggregation mechanisms, for example, MoA [50] explicitly separates the roles of proposers and aggregators, SMoA [25] further sparsifies the aggregation pipeline, and Self-MoA [28] revisits repeated sampling and aggregation within a single strong model. Symbolic-MoE [5] further uses skill-aware expert recruitment to select multiple relevant models and aggregate their reasoning outputs, while JiSi [47] forms a route-and-aggregate workflow by combining response-aware routing, aggregator selection, and adaptive switching after multiple candidate model responses have been generated. Our method moves this decision before response generation: it estimates model confidence from historical evaluation signals and uses it to decide whether to perform single-model routing or introduce multi-model aggregation.

## 3 Problem Formulation

Consider a heterogeneous candidate pool of $L$ large language models, denoted by $\mathcal{M} = \{M_1, M_2, \ldots, M_L\}$. Given a test query $x$, the system needs to choose among different inference modes: it may call a single candidate model to generate an answer, or it may introduce multiple model responses and aggregate them. The goal of this paper is to improve final answer correctness while controlling test-time inference cost, namely to adaptively decide, for each query, whether to use single-model selection or multi-model inference.

During training, we are given a set of queries $\{x_i\}_{i=1}^{N}$ together with the responses and historical evaluation outcomes of each candidate model on these queries. We represent these historical evaluation signals as a model correctness matrix

$$\mathbf{Y} = [Y_{ik}] \in \{0,1\}^{N \times L}, \tag{1}$$

where $Y_{ik} = 1$ indicates that model $M_k$ answers training query $x_i$ correctly, and $Y_{ik} = 0$ otherwise. This matrix captures the empirical relationship between training queries and candidate model capabilities, providing supervision for estimating model performance on test queries.

Given a test query $x \in \mathcal{X}$, we first estimate a query-conditioned model confidence score vector:

$$\mathbf{s}(x) = [s_1(x), s_2(x), \ldots, s_L(x)] \in [0,1]^L. \tag{2}$$

Here $s_k(x) \in [0,1]$ is a continuous confidence score that estimates the expected correctness of model $M_k$ on query $x$. Based on $\mathbf{s}(x)$, the final system output can be generally written as

$$\hat{y} = \mathcal{F}(x; \mathcal{M}, \mathbf{s}(x)), \tag{3}$$

where $\mathcal{F}$ denotes a test-time inference rule conditioned on model confidence scores. This rule can select a single model to answer the query, or it can call and aggregate responses from multiple models. The following sections describe how we estimate reliable $\mathbf{s}(x)$ from historical evaluation signals (▷ Section 4.1) and how this vector is used for dynamic inference decisions (▷ Section 4.2).

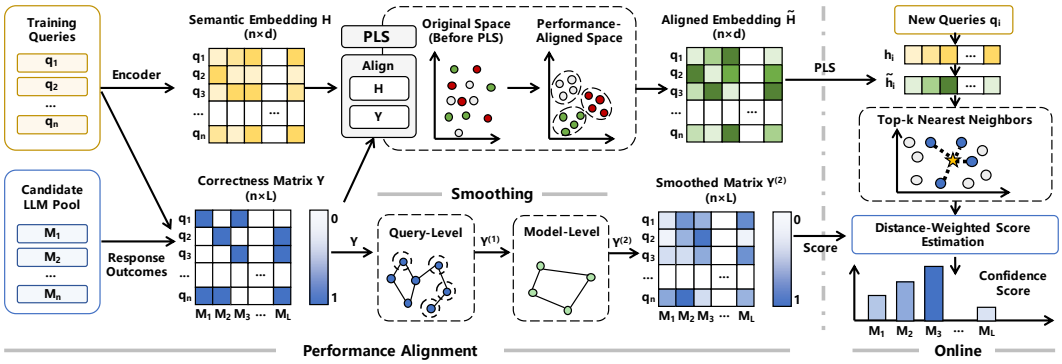



Figure 1: **Model confidence estimation via decoupled smoothing.** PARS aligns semantic embeddings with model-correctness supervision, smooths the resulting performance signals over query and model relations, and estimates model confidence scores for a test query through distance-weighted aggregation over top-$k$ neighbors in the performance-aligned retrieval space.

# 4 Method

## 4.1 Model Confidence Estimation via Decoupled Smoothing

Existing retrieval-based model routing methods usually rely directly on pretrained text embeddings to measure query similarity. To make local neighborhoods better reflect model correctness patterns, we first construct a performance-aligned representation and estimate candidate-model confidence scores in this space. Figure 1 summarizes the full model-confidence estimation pipeline.

**Representation Decoupling for Performance Alignment.** Let $\mathbf{h}_x \in \mathbb{R}^d$ denote the normalized base embedding of query $x$. We aim for the retrieval representation to preserve semantic information while adjusting the neighborhood structure along directions revealed by historical model performance. We therefore apply a controlled correction to the base embedding and normalize the result:

$$\tilde{\mathbf{h}}_x = \frac{\mathbf{h}_x + (\gamma - 1)\mathbf{h}_x^{\text{perf}}}{\left\| \mathbf{h}_x + (\gamma - 1)\mathbf{h}_x^{\text{perf}} \right\|_2}, \tag{4}$$

where $\mathbf{h}_x^{\text{perf}}$ denotes a performance-aware correction learned from historical model performance, and $\gamma > 1$ controls its relative strength. This combination shifts the retrieval space toward directions related to model correctness patterns, making queries with similar semantics but different capability requirements easier to separate. To learn $\mathbf{h}_x^{\text{perf}}$, we denote the embeddings of all training queries by

$$\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_N]^\top \in \mathbb{R}^{N \times d}. \tag{5}$$

The supervision matrix $\mathbf{Y}$ provides the correctness pattern of each training sample over the candidate model pool. To make representation learning focus more on training samples where models disagree, we first compute the average correctness and disagreement weight for each sample:

$$\bar{p}_i = \frac{1}{L} \sum_{k=1}^{L} Y_{ik}, \qquad \rho_i = 4\bar{p}_i(1 - \bar{p}_i), \qquad \omega_i = 1 + \rho_i. \tag{6}$$

We then weight $\mathbf{H}$ and $\mathbf{Y}$ by $\sqrt{\omega_i}$ and fit **partial least squares (PLS)** on the weighted pair $(\mathbf{H}, \mathbf{Y})$, so that the covariance-maximization objective of PLS places greater emphasis on training samples where models disagree. Let $\mathbf{T} \in \mathbb{R}^{N \times r}$ denote the latent scores learned by PLS, where $r$ is the number of PLS components. This low-dimensional representation therefore forms a performance subspace aligned with model correctness patterns. Given the input-side loadings $\mathbf{P}_x$, we map this low-dimensional representation back to the original embedding space and obtain the performance-aware correction term for training samples:

$$\mathbf{H}^{\text{perf}} = \mathbf{T}\mathbf{P}_x^\top. \tag{7}$$

For any test query, the same PLS projection produces the corresponding $\mathbf{h}_x^{\text{perf}}$. The resulting $\tilde{\mathbf{h}}_x$ is used for subsequent neighborhood retrieval and local performance prediction. Additional construction details are provided in Appendix A.1.

**Offline Structured Score Smoothing.** The representation $\tilde{\mathbf{h}}$ provides a neighborhood structure aligned with model performance, but the supervision matrix $\mathbf{Y}$ remains a binary observation over a finite set of training samples. Estimating model confidence scores directly from raw correctness labels in the retrieved neighborhood can be sensitive to local sample noise and evaluation sparsity. To obtain more stable training-side score signals, we propagate correctness information along the two dimensions of training samples and candidate models.

**(i)** Inter-query smoothing. We build a $k_{\mathrm{nn}}$ graph over training samples in the $\tilde{\mathbf{h}}$ space so that nearby queries can share local correctness evidence, and construct a row-normalized sparse sample propagation matrix $\mathbf{S}_{\mathrm{data}}$. Each sample is connected only to local neighbors, so the number of nonzero entries scales as $O(Nk_{\mathrm{nn}})$. Let $\mathbf{Y}^{(1)} = [Y_{ik}^{(1)}]$ denote the training score matrix after inter-query smoothing. For sample $x_i$ and model $M_k$, the fixed-point form of this smoothing is:

$$Y_{ik}^{(1)} = (1-\alpha)Y_{ik} + \alpha \sum_{j \in \mathcal{N}_{\mathrm{tr}}(i)} S_{ij}^{\mathrm{data}} Y_{jk}^{(1)}, \tag{8}$$

where $\mathcal{N}_{\mathrm{tr}}(i)$ denotes the neighbor set connected to $x_i$ in the training-sample graph, $S_{ij}^{\mathrm{data}}$ is the row-normalized propagation weight, and $\alpha \in (0,1)$ controls the propagation strength of correctness signals along the sample manifold.

**(ii)** Inter-model smoothing. We exploit correlations among model correctness patterns so that models with similar behavior can share statistical strength. Based on $\mathbf{Y}^{(1)}$, we compute non-negative correlations between model correctness vectors and retain the top-$k_{\mathrm{model}}$ strongly correlated edges that satisfy the threshold constraint, obtaining the model propagation matrix $\mathbf{S}_{\mathrm{model}} \in \mathbb{R}^{L \times L}$. Let $\boldsymbol{\mu}$ denote the column mean of $\mathbf{Y}^{(1)}$, and let $\bar{\mathbf{Y}}^{(1)} = \mathbf{Y}^{(1)} - \boldsymbol{\mu}$ be the centered matrix. The model-level smoothing step is written as

$$\bar{\mathbf{Y}}^{(2)} = (1-\beta)\bar{\mathbf{Y}}^{(1)} + \beta \bar{\mathbf{Y}}^{(2)} \mathbf{S}_{\mathrm{model}}^{\top}, \tag{9}$$

where $\beta$ controls the diffusion strength over the model propagation matrix. After restoring the column means, we clip the scores element-wise to $[0,1]$ to obtain the final training score matrix $\mathbf{Y}^{(2)}$. Further details are provided in Appendix A.1.

**Online Confidence Extraction.** For a test query $x$, the system first computes its performance-aligned representation $\tilde{\mathbf{h}}_x$ and retrieves a local neighborhood $\mathcal{N}_k(x)$ from the model-performance-aligned training representation space. Let $d_i$ denote the distance from neighbor $x_i$ to $x$ in this space, and let $\tau$ be a bandwidth estimated from training-neighborhood distances. The neighborhood weight is defined as

$$w_i(x) = \frac{\exp(-d_i/\tau)}{\sum_{x_j \in \mathcal{N}_k(x)} \exp(-d_j/\tau)}. \tag{10}$$

Using the smoothed training score matrix $\mathbf{Y}^{(2)}$, the confidence score of model $M_k$ on query $x$ is obtained by local neighborhood weighting:

$$s_k(x) = \sum_{x_i \in \mathcal{N}_k(x)} w_i(x) Y_{ik}^{(2)}. \tag{11}$$

This yields $\mathbf{s}(x)$. Sorting $\mathbf{s}(x)$ gives the ordered candidate model list $\{M_{(1)}, M_{(2)}, \ldots, M_{(L)}\}$, where $s_{(1)}(x) \geq s_{(2)}(x) \geq \cdots \geq s_{(L)}(x)$, providing the confidence order used for subsequent inference.

### 4.2 Test-Time Compute Allocation for Dynamic Inference

After obtaining confidence scores $\mathbf{s}(x)$, the key question is how to allocate compute according to the confidence structure of the current query. For queries with a steep confidence distribution, the system tends to concentrate computation on the highest-confidence model; for more complex queries with a flatter distribution, it may need to spend more test-time compute to reduce uncertainty. We therefore use $\mathbf{s}(x)$ as the decision basis for test-time compute allocation and adaptively schedule inference resources. Figure 2 illustrates the resulting policy-learning and adaptive-execution procedure.

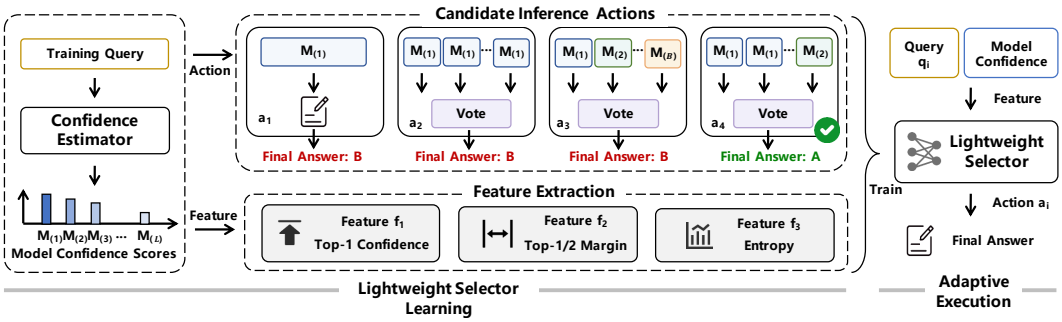



Figure 2: **Test-time compute allocation for dynamic inference.** PARS extracts confidence features from the estimated model scores, learns a lightweight selector from oracle-labeled training actions, and adaptively chooses a single-model or multi-model inference action at test time.

**Budgeted Inference Action Space.** Following the model-pool setting of LLMRouterBench under the performance-oriented protocol [26], the candidate models considered in this paper are open-source models in the 7B–9B parameter range. To characterize the resource use of different inference actions under a unified protocol, we use the number of model calls as the basic unit of the test-time compute budget, and represent candidate inference modes as an action space $\mathcal{A}_B$ constrained by this budget. Given budget $B$, any action $a \in \mathcal{A}_B$ allocates generation calls according to the candidate model ranking $\{M_{(1)}, M_{(2)}, \dots\}$ induced by $\mathbf{s}(x)$. Let $m_a^{(j)}(x)$ denote the number of calls assigned by action $a$ to the $j$-th ranked candidate model $M_{(j)}$. Its cost satisfies

$$c(a; x) = \sum_j m_a^{(j)}(x) \le B. \tag{12}$$

We consider the following four types of candidate actions:

- **Direct routing.** Calls only $M_{(1)}$, namely $m_a^{(1)}(x) = 1$.
- **Homogeneous self consistency.** Allocates the entire budget to $M_{(1)}$, namely $m_a^{(1)}(x) = B$.
- **Heterogeneous voting.** Calls each top-$B$ candidate model once and votes over their responses.
- **Hybrid resampling.** First calls each top-$K$ candidate model once, and then allocates the remaining $B - K$ calls to these $K$ models, balancing cross-model diversity with local repeated sampling.

All multi-response actions aggregate in the normalized answer space. If a tie occurs, the answer supported by the higher-confidence model is preferred. Additional details on budget allocation and consensus resolution are provided in Appendix A.2.

**Confidence-Driven Policy Learning.** For each query $x$, we construct state features $\boldsymbol{\phi}(x)$ from the model confidence scores $\mathbf{s}(x)$ and learn a lightweight policy selector $\pi_\theta(a \mid \boldsymbol{\phi}(x))$. The state features include **(i)** the highest confidence score $s_{(1)}(x)$; **(ii)** the margin $\Delta_{12}(x) = s_{(1)}(x) - s_{(2)}(x)$, which characterizes local competition; and **(iii)** the score entropy $H_s(x) = -\sum_k p_k(x) \log p_k(x)$, which characterizes global uncertainty, where $p_k(x) = s_k(x) / \sum_\ell s_\ell(x)$.

To train the selector, we evaluate candidate actions in $\mathcal{A}_B$ on the training set and construct an oracle action according to the final answer of each action. For training query $x_i$ and action $a$, let $\mathbb{I}_{\text{corr}}(a; x_i)$ indicate whether the final answer produced by action $a$ is correct. The oracle action is defined as

$$a_i^* = \arg\max_{a \in \mathcal{A}_B} \big(\mathbb{I}_{\text{corr}}(a; x_i), -c(a; x_i)\big). \tag{13}$$

This definition first selects actions that produce the correct answer and, when correctness is tied, favors the lower-cost action. We then train $\pi_\theta$ using $a_i^*$ as supervision. Concretely, we standardize the state features and use a regularized logistic regression with class-balanced weights to mitigate the long-tailed distribution of oracle actions.

**Online Inference.** At test time, given query $x$, the system first estimates the model confidence scores $\mathbf{s}(x)$ following Section 4.1 and constructs the state features $\boldsymbol{\phi}(x)$. The policy selector $\pi_\theta$ then

predicts the budget action for the current query:

$$\hat{a} = \arg\max_{a \in \mathcal{A}_B} \pi_\theta(a \mid \boldsymbol{\phi}(x)). \tag{14}$$

According to the selected action $\hat{a}$, the system allocates model calls following $m_{\hat{a}}^{(j)}(x)$ and applies the corresponding aggregation rule to the obtained responses. The final output is written as

$$\hat{y} = \mathcal{F}_{\hat{a}}(x; \mathcal{M}, \mathbf{s}(x)). \tag{15}$$

Overall, routing and aggregation are jointly determined by the candidate ordering induced by $\mathbf{s}(x)$ and the budget action selected by $\pi_\theta$, forming a test-time compute allocation procedure driven by model confidence scores.

# 5 Experimental Evaluation

## 5.1 Experiment Settings

**Datasets.** We primarily evaluate all methods on LLMRouterBench [26], including 14 in-distribution (ID) benchmarks across 5 task categories: Mathematics [39, 33, 35], Code [21, 1], Logic [46, 54, 38], Knowledge [10, 44, 23, 51], and Affective [56, 43]. To assess out-of-distribution (OOD) generalization, we further use 5 OOD benchmarks corresponding to these task categories [36, 7, 45, 11, 30]. For each ID dataset, samples are split into train and test partitions with a 0.7:0.3 ratio, while all OOD datasets are used only for testing. Detailed dataset information is provided in Appendix B.

**Baselines.** We evaluate PARS under two settings: the *single-model setting* selects one model to answer each query and compares PARS (Single) with routing baselines, while the *multi-model setting* allows single-model or multi-model inference within a budget and compares the full PARS with multi-model baselines. Routing baselines include **RouterDC** [8], **GraphRouter** [14], **EmbedLLM** [63], and **Avengers** [61]; multi-model baselines include **Majority Voting** [6], **Self Consistency** [9], **Mixture-of-Agents (MoA)** [50], **Self-MoA** [28], **Symbolic-MoE** [5], and **JiSi** [47]. More baseline details are provided in Appendix D.

**Implementation Details.** All methods use the same 20-model open-source LLM pool with approximately 7B–9B parameters, covering general-purpose, reasoning-oriented, and domain-specialized models. We uniformly adopt `gte-Qwen2-7B-instruct` [31] as the query embedding model. Unless otherwise specified, the budget in the multi-model setting is set to 3 candidate outputs; for baselines that first generate multiple candidate responses and then perform synthesis, this budget applies to the candidate-response side, and one additional aggregator call is used when required by the original method. Candidate model information and further implementation details are provided in Appendices C and E.

## 5.2 Main Results

Tables 1 and 2 report ID and OOD results averaged over seven random seeds, respectively; seed-level standard deviations are provided in Appendix F.1.

**In-Distribution Results.** On ID benchmarks, PARS achieves the highest average accuracy under both the single-model and multi-model settings. PARS (Single) reaches 72.56% average accuracy, 1.02% above the strongest routing baseline, EmbedLLM (71.54%), showing that the confidence estimator more reliably assigns queries to suitable candidate models. PARS further reaches 75.19% average accuracy, exceeding the strongest multi-model baseline, JiSi (73.83%), by 1.36%. Although the best result on a few individual datasets can be influenced by domain-specific model capabilities or fixed collaboration patterns, the cross-domain average indicates that PARS more stably supports model selection and dynamic multi-model inference.

**Out-of-Distribution Results.** Table 2 shows that PARS retains stable gains on unseen benchmarks. PARS (Single) achieves 77.01% average accuracy, outperforming the strongest single-model baseline, Avengers (75.45%); PARS reaches 78.28%, also exceeding the strongest multi-model baseline, Self-MoA (77.72%), indicating stable cross-dataset generalization of the model confidence estimates. In leave-one-domain-out evaluation, where the target domain is entirely excluded from training, PARS still achieves the highest average accuracy and the smallest performance drop under both settings; detailed results are provided in Appendix F.2.

Table 1: Accuracy results on the in-distribution benchmarks. The best result in each group is **bolded**, and the second-best is underlined.

| Method | Mathematics | | | Code | | Logic | | | Knowledge | | | | Affective | | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME | M500 | MBen. | LCB. | MBPP | BBH | K&K. | KOR. | FQA. | GPQA | MQA. | MP. | Emory. | MELD | |
| *Open-Source LLMs (Top-5)* | | | | | | | | | | | | | | | |
| Qwen3-8B | 80.95 | 93.90 | 92.70 | 66.83 | 54.61 | 84.26 | 77.52 | 54.67 | 73.77 | 57.38 | 78.46 | 69.90 | 38.48 | 55.44 | **69.92** |
| Nemotron-9B | 74.60 | 93.52 | 95.24 | 64.89 | 42.91 | 86.42 | 71.02 | 58.32 | 70.93 | 52.86 | 75.13 | 70.52 | 38.69 | 46.87 | 67.28 |
| DS-R1-8B | 69.05 | 93.90 | 90.48 | 63.27 | 34.37 | 89.29 | 61.61 | 56.34 | 69.84 | 59.52 | 81.23 | 69.43 | 38.96 | 51.54 | 66.35 |
| GLM-Z1-9B | 65.87 | 93.90 | 97.78 | 60.75 | 63.73 | 83.86 | 46.58 | 54.21 | 68.59 | 55.00 | 70.08 | 68.14 | 34.52 | 46.29 | 64.95 |
| MiniCPM-8B | 70.63 | 92.95 | 93.02 | 62.96 | 35.35 | 88.98 | 60.73 | 29.30 | 66.12 | 50.95 | 65.26 | 69.05 | 34.86 | 53.01 | 62.37 |
| *Single-Model Methods* | | | | | | | | | | | | | | | |
| RouterDC | 72.22 | 91.33 | 53.33 | 59.94 | 62.80 | 64.51 | 72.51 | 59.73 | 73.98 | 36.67 | 69.63 | 56.00 | 31.58 | 43.78 | 60.57 |
| GraphRouter | 80.95 | 93.90 | 92.70 | 66.83 | 74.11 | 84.48 | 77.52 | 54.55 | 73.81 | 57.38 | 78.46 | 70.00 | 37.59 | 55.37 | 71.26 |
| EmbedLLM | 66.67 | 93.81 | 94.60 | 66.43 | 75.48 | 89.15 | 79.69 | 61.60 | 73.22 | 57.14 | 80.85 | 70.10 | 37.05 | 55.71 | 71.54 |
| Avengers | 69.05 | 93.14 | 91.43 | 66.20 | 75.18 | 89.37 | 78.54 | 60.61 | 73.48 | 56.90 | 81.23 | 69.67 | 36.50 | 55.41 | 71.19 |
| **PARS (Single)** | 78.57 | 94.29 | 93.02 | 66.61 | 75.48 | 90.74 | 80.23 | 65.07 | 74.14 | 54.52 | 79.88 | 70.43 | 36.91 | 55.95 | **72.56** |
| *Multi-Model Methods* | | | | | | | | | | | | | | | |
| Majority Voting | 86.11 | 93.67 | 92.22 | 72.40 | 73.21 | 89.20 | 78.91 | 65.20 | 73.98 | 57.50 | 79.84 | 73.17 | 38.76 | 58.11 | 73.73 |
| Self Consistency | 71.43 | 94.00 | 95.56 | 69.72 | 75.43 | 89.33 | 80.70 | 61.10 | 74.48 | 61.67 | 81.90 | 70.48 | 37.46 | 55.10 | 72.74 |
| MoA | 75.93 | 96.89 | 94.81 | 68.77 | 57.22 | 80.86 | 76.00 | 58.22 | 71.73 | 59.44 | 82.11 | 71.67 | 37.32 | 51.71 | 70.19 |
| Self-MoA | 83.33 | 96.67 | 96.30 | 70.45 | 55.75 | 74.49 | 78.03 | 46.40 | 73.20 | 56.67 | 80.54 | 70.11 | 36.36 | 53.33 | 69.40 |
| Symbolic-MoE | 83.33 | 96.29 | 95.56 | 70.12 | 58.51 | 85.14 | 65.81 | 61.83 | 73.56 | 58.33 | 82.27 | 70.86 | 37.87 | 55.02 | 71.04 |
| JiSi | 79.36 | 95.71 | 97.78 | 70.93 | 73.56 | 89.72 | 82.31 | 64.75 | 72.93 | 59.06 | 83.43 | 74.10 | 37.18 | 52.74 | 73.83 |
| **PARS** | 81.75 | 95.24 | 98.10 | 72.37 | 75.62 | 91.98 | 84.02 | 67.85 | 74.56 | 59.76 | 80.96 | 72.95 | 39.58 | 57.95 | **75.19** |

Table 2: Accuracy on the out-of-distribution benchmarks. Best in each group is **bolded**; second-best is underlined.

| Method | Mathematics | Code | Logic | Knowledge | Affective | Avg |
|---|---|---|---|---|---|---|
| | LMB. | HE. | WG. | ARC. | DD. | |
| *Single-Model Methods* | | | | | | |
| RouterDC | 70.25 | 67.07 | 71.03 | 89.76 | 50.68 | 69.76 |
| GraphRouter | 74.38 | 66.03 | 82.41 | 93.77 | 57.16 | 74.75 |
| EmbedLLM | 70.96 | 66.29 | 82.57 | 94.09 | 57.77 | 74.34 |
| Avengers | 70.84 | 73.87 | 82.00 | 94.21 | 56.32 | 75.45 |
| **PARS (Single)** | 73.55 | 77.13 | 82.20 | 93.60 | 58.56 | **77.01** |
| *Multi-Model Methods* | | | | | | |
| Majority Voting | 75.21 | 44.51 | 84.21 | 94.71 | 56.17 | 70.96 |
| Self Consistency | 75.21 | 62.20 | 83.27 | 94.28 | 59.10 | 74.81 |
| MoA | 76.03 | 76.63 | 83.69 | 94.82 | 55.71 | 77.38 |
| Self-MoA | 76.03 | 75.00 | 85.00 | 94.11 | 58.44 | 77.72 |
| Symbolic-MoE | 73.20 | 72.73 | 83.05 | 94.20 | 56.13 | 75.86 |
| JiSi | 74.38 | 75.09 | 83.76 | 94.51 | 58.20 | 77.19 |
| **PARS** | 76.03 | 78.96 | 83.43 | 94.62 | 58.35 | **78.28** |

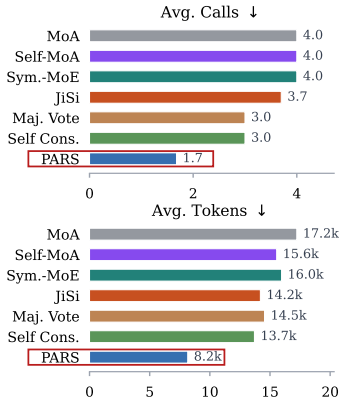


Figure 3: Inference cost.

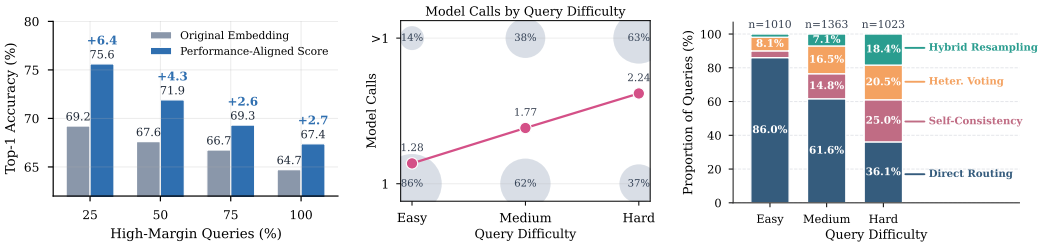
**Test-Time Efficiency.** Figure 3 shows that PARS uses 1.7 model calls on average, below all compared methods; even when quantified by tokens, the average cost is only 8.2k. Combined with the ID and OOD accuracy results, this indicates that PARS maintains strong performance without applying fixed multi-model collaboration to every query, yielding a more effective accuracy–cost trade-off. Uncached end-to-end evaluation further shows that PARS achieves the highest accuracy alongside the lowest latency and highest throughput; detailed results are provided in Appendix F.3.

### 5.3 Micro Experiments

**Score Estimator Ablation.** Table 3 shows that performance-aligned representation and structured smoothing provide stable gains under both the single-model and multi-model settings. Removing either component degrades performance, and falling back to original embedding leads to the largest drop, reducing PARS (Single) and PARS to 70.79% and 73.13%, respectively. These results indicate that performance-aligned representation and structured smoothing help select a more suitable top-ranked model and provide more reliable confidence scores for subsequent dynamic inference. Separate ablations of

Table 3: Accuracy ablation of score estimator components.

| Variant | PARS (Single) | PARS |
|---|---|---|
| Full score estimator | **72.56** | **75.19** |
| *w/o* decoupling | 72.03 | 74.32 |
| *w/o* structured smoothing | 71.76 | 73.96 |
| original embedding | 70.79 | 73.13 |




(a) Confidence Reliability (b) Model Call Allocation (c) Strategy Distribution

Figure 4: **(a)** Queries are ranked by the top-1/top-2 score margin; lower coverage keeps only higher-margin queries. **(b)** Model call allocation across query difficulty levels. **(c)** Strategy distribution over routing and multi-response actions under the same difficulty partition.

the two smoothing stages and a comparison of the selector with a threshold policy are provided in Appendix F.4.

**Confidence Margin Reliability.** We rank queries by the top-1/top-2 confidence margin and retain the highest-margin 25%, 50%, 75%, and all queries. If the margin reflects model-selection reliability, top-1 accuracy should increase as coverage decreases. Figure 4**(a)** shows that performance-aligned confidence scores consistently outperform original embedding across all coverage levels, with the high-margin 25% subset reaching 75.6% accuracy. This suggests that the margin does not merely follow the average accuracy improvement, but also provides a useful basis for distinguishing the inference needs of different queries.

**Difficulty-Aware Compute Allocation.** To examine dynamic inference behavior, we partition queries into Easy, Medium, and Hard groups according to the difficulty partition given in Appendix E. Figure 4**(b)** shows that the average number of model calls rises from 1.28 on Easy queries to 2.24 on Hard queries, indicating that the system allocates more computation to difficult samples. Figure 4**(c)** further shows that the share of Direct Routing drops from 86.0% to 36.1% as difficulty increases, while multi-response actions, including self-consistency sampling and heterogeneous voting, are used more often, showing a trend of allocating more compute as query difficulty grows.

**Budget Scaling.** Figure 5 compares PARS with major aggregation baselines under budgets from $B = 1$ to $B = 5$. As the budget increases, PARS improves from about 73% accuracy to about 76%, remains stable under larger budgets, and consistently outperforms other multi-model baselines. Meanwhile, its average number of model calls increases only from about 1 to slightly above 2; in contrast, fixed collaboration methods typically increase calls linearly with the budget, while methods with an aggregator require additional aggregation-related calls.

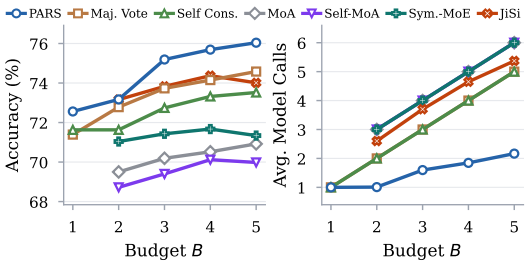



Figure 5: Accuracy and model calls across different inference budgets.

**Hyperparameter Sensitivity Analysis.** We further report sensitivity analysis for key hyperparameters in Figure 6. Overall, PARS remains stable under variations in diffusion scale, PLS components, and neighborhood size. We also analyze different training set sizes. Even with only partial training data, PARS maintains strong performance and further improves as more training data is used.

**Other Experiments.** Appendix further provides complementary analyses:
**(i) Embedding Robustness.** Table 14 shows that PARS remains stable across different embedding backbones, suggesting that the results do not rely on a specific encoder.
**(ii) Training-Side Scalability.** Table 15 shows that the costs of confidence-estimator construction and selector training increase only mildly as the model pool grows from 5 to 20 models.
**(iii) Routing Failure Rescue.** Figure 7 decomposes final outcomes into top-1 routing successes, routing failures rescued by multi-response actions, and remaining errors, showing that dynamic inference builds on strong routing and further rescues some failed cases.
**(iv) Dataset-Conditioned Behavior.** Figures 8 and 12 show dataset-level differences in strategy selection and model usage, indicating that PARS adjusts inference behavior by task characteristics.

## 6 Conclusion

This paper studies how to estimate query-level model confidence in a heterogeneous LLM pool and use it to support dynamic LLM routing and aggregation. We propose PARS, a model confidence estimation method for dynamic LLM routing and aggregation. It first decouples the representation space and applies structured smoothing to estimate confidence scores for candidate models on the query. Based on this confidence structure, PARS then adaptively selects single-model or multi-model inference strategies under a controlled test-time compute budget. Experiments show that PARS consistently outperforms representative baselines in both the single-model and multi-model settings, while maintaining lower inference cost and exhibiting stable out-of-distribution generalization.

**Limitations.** (i) *Cost characterization.* Model calls are used as an approximate proxy for test-time compute; future work can consider output token counts and model-specific prices for a finer-grained characterization of inference cost. (ii) *Action-space coverage.* The policy is defined over a finite action space covering common routing and aggregation patterns, leaving richer adaptive aggregation and iterative reasoning mechanisms as future extensions.

## Acknowledgments

This work was sponsored by the CCF-DAMENG Research Fund (No. CCF-DAMENG202605), the National Natural Science Foundation of China (No. 62472327), the Innovative Research Group Project of Hubei Province (No. 2024AFA017), and the New Cornerstone Science Foundation through the XPLORER PRIZE. This work was also supported by the WHU-Kingsoft Joint Lab and ModaLeap.

## 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. URL https://arxiv.org/abs/2108.07732.

[2] Lei Bai, Zhongrui Cai, Yuhang Cao, Maosong Cao, Weihan Cao, Chiyu Chen, Haojiong Chen, Kai Chen, Pengcheng Chen, Ying Chen, et al. Intern-s1: A scientific multimodal foundation model. *arXiv preprint arXiv:2508.15763*, 2025. URL https://arxiv.org/abs/2508.15763.

[3] Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

[4] Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024. URL https://arxiv.org/abs/2403.17297.

[5] Justin Chih-Yao Chen, Sukwon Yun, Elias Stengel-Eskin, Tianlong Chen, and Mohit Bansal. Symbolic mixture-of-experts: Adaptive skill-based routing for heterogeneous reasoning. *arXiv preprint arXiv:2503.05641*, 2025. URL https://arxiv.org/abs/2503.05641.

[6] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=m5106RRLgx.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

[8] Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=7RQvjayHrM.

[9] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*, 2024. URL https://openreview.net/forum?id=LjsjHF7nAN.

[10] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://aclanthology.org/2021.emnlp-main.300/.

[11] 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. URL https://arxiv.org/abs/1803.05457.

[12] DeepCogito. Cogito-v1-preview-llama-8b, 2025. URL https://huggingface.co/deepcogito/cogito-v1-preview-llama-8B.

[13] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=02f3mUtqnM.

[14] Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for LLM selections. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=eU39PDsZtT.

[15] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. URL https://arxiv.org/abs/2408.00118.

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

[17] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025. URL https://arxiv.org/abs/2506.04178.

[18] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 2025. URL https://www.nature.com/articles/s41586-025-09422-z.

[19] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

[20] IBM Granite Team. Granite-3.3-8b-instruct, 2025. URL https://huggingface.co/ibm-granite/granite-3.3-8b-instruct.

[21] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

[22] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://aclanthology.org/2023.acl-long.792/.

[23] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 2021. URL https://www.mdpi.com/2076-3417/11/14/6421.

[24] Ruihan Jin, Pengpeng Shao, Zhengqi Wen, Jinyang Wu, Mingkuan Feng, Shuai Zhang, and Jianhua Tao. RadialRouter: Structured representation for efficient and robust large language models routing. In *Findings of the Association for Computational Linguistics*, 2025. URL https://aclanthology.org/2025.findings-emnlp.787/.

[25] Dawei Li, Zhen Tan, Peijia Qian, Yifan Li, Kumar Chaudhary, Lijie Hu, and Jiayi Shen. Smoa: Improving multi-agent large language models with sparse mixture-of-agents. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2025.

[26] Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, et al. Llmrouterbench: A massive benchmark and unified framework for llm routing. *arXiv preprint arXiv:2601.07206*, 2026. URL https://arxiv.org/abs/2601.07206.

[27] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KfTf9vFvSn.

[28] Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking mixture-of-agents: Is mixing different large language models beneficial? *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=K6WwK8URlV.

[29] Yang Li. Rethinking predictive modeling for llm routing: When simple knn beats complex learned routers. *arXiv preprint arXiv:2505.12601*, 2025. URL https://arxiv.org/abs/2505.12601.

[30] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, 2017. URL https://aclanthology.org/I17-1099/.

[31] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023. URL https://arxiv.org/abs/2308.03281.

[32] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=iO4LZibEqW.

[33] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

[34] Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking LLMs with challenging tasks from real users in the wild. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MKEHCx25xp.

[35] Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark. In *Findings of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.findings-acl.411/.

[36] Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. Are your LLMs capable of stable reasoning? In *Findings of the Association for Computational Linguistics*, 2025. URL https://aclanthology.org/2025.findings-acl.905/.

[37] Zhaowei Liu, Xin Guo, Zhi Yang, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Mengping Li, Qi Qi, Zhiqiang Liu, Yiyang Han, et al. Fin-r1: A large language model for financial reasoning through reinforcement learning. *arXiv preprint arXiv:2503.16252*, 2025. URL https://arxiv.org/abs/2503.16252.

[38] Kaijing Ma, Xeron Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, Wenhao Huang, and Ge Zhang. KOR-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SVRRQ8goQo.

[39] Mathematical Association of America. American invitational mathematics examination (AIME), 2024. URL https://maa.org/maa-invitational-competitions/.

[40] MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, et al. Minicpm4: Ultra-efficient llms on end devices. *arXiv preprint arXiv:2506.07900*, 2025. URL https://arxiv.org/abs/2506.07900.

[41] NVIDIA. Nvidia-nemotron-nano-9b-v2, 2025. URL https://huggingface.co/nvidia/NVIDIA-Nemotron-Nano-9B-v2.

[42] 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 from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

[43] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. MELD: A multimodal multi-party dataset for emotion recognition in conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://aclanthology.org/P19-1050/.

[44] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

[45] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021. URL https://dl.acm.org/doi/abs/10.1145/3474381.

[46] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics*, 2023. URL https://aclanthology.org/2023.findings-acl.824/.

[47] Shengji Tang, Weihao Lin, Peng Ye, Jingqi Ye, Hao Li, Yiqun Zhang, Xiaosong Wang, Bo Zhang, Shuyue Hu, Tao Chen, et al. Beyond gemini-3-pro: Revisiting llm routing and aggregation at scale. *arXiv preprint arXiv:2601.01330*, 2026. URL https://arxiv.org/abs/2601.01330.

[48] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024. URL https://arxiv.org/abs/2406.12793.

[49] Ryan Teknium, Roger Jin, Jai Suphavadeeprasit, Dakota Mahan, Jeffrey Quesnelle, Joe Li, Chen Guang, Shannon Sands, and Karan Malhotra. Hermes 4 technical report. *arXiv preprint arXiv:2508.18255*, 2025. URL https://arxiv.org/abs/2508.18255.

[50] Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h0ZfDIrj7T.

[51] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=y10DM6R2r3.

[52] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sKYHBTAxVa.

[53] LLM Xiaomi, Bingquan Xia, Bowen Shen, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang, Huaqiu Liu, Jiebao Xiao, Jinhao Dong, et al. Mimo: Unlocking the reasoning potential of language model–from pretraining to posttraining. *arXiv preprint arXiv:2505.07608*, 2025. URL https://arxiv.org/abs/2505.07608.

[54] Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, 2025. URL https://aclanthology.org/2025.ijcnlp-long.148/.

[55] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

[56] Sayyed M Zahiri and Jinho D Choi. Emotion detection on tv show transcripts with sequence-based convolutional neural networks. *arXiv preprint arXiv:1708.04299*, 2017. URL https://arxiv.org/abs/1708.04299.

[57] Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=DWf4vroKWJ.

[58] Kaiyan Zhang, Sihang Zeng, Ermo Hua, Ning Ding, Zhang-Ren Chen, Zhiyuan Ma, Haoxin Li, Ganqu Cui, Biqing Qi, Xuekai Zhu, Xingtai Lv, Hu Jinfang, Zhiyuan Liu, and Bowen Zhou. Ultramedical: Building specialized generalists in biomedicine. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=pUcTrjRLOM.

[59] Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Capability instruction tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[60] Yifan Zhang, Xinkui Zhao, Zuxin Wang, Guanjie Cheng, Yueshen Xu, Shuiguang Deng, and Jianwei Yin. Lightrouter: Towards efficient llm collaboration with minimal overhead. *arXiv preprint arXiv:2505.16221*, 2025. URL https://arxiv.org/abs/2505.16221.

[61] Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Daling Wang, Zhen Wang, Xinrun Wang, et al. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants. *arXiv preprint arXiv:2505.19797*, 2025. URL https://arxiv.org/abs/2505.19797.

[62] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

[63] Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning compact representations of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Fs9EabmQrJ.

# Appendix

## Appendix Contents


**A Method Details** ..... **16**

A.1 Model Confidence Estimation via Decoupled Smoothing ..... 16

A.2 Budgeted Inference and Selector Learning ..... 18

**B Dataset Details** ..... **19**

**C Model Details** ..... **21**

C.1 Full Model Pool Results ..... 21

**D Baseline Details** ..... **21**

**E Implementation Details** ..... **24**

**F Additional Experiments** ..... **25**

F.1 Seed-Level Variability ..... 25

F.2 Cross-Domain Generalization ..... 25

F.3 End-to-End Inference Efficiency ..... 25

F.4 Additional Ablation Studies ..... 26

F.5 Hyperparameter Sensitivity Analysis ..... 26

F.6 Embedding Backbone Sensitivity ..... 27

F.7 Training-Side Cost under Pool Expansion ..... 27

F.8 Routing Failure Rescue Analysis ..... 27

F.9 Dataset-Level Strategy Distribution ..... 28

F.10 Oracle Strategy Prediction Analysis ..... 28

F.11 Oracle Reference Analysis ..... 29

F.12 Model Selection Diversity ..... 29

**G Broader Impacts** ..... **31**

**H Case Studies of Adaptive Action Selection** ..... **31**


# A Method Details

This appendix provides mathematical details for decoupled smoothing and the dynamic inference policy in Section 4.

## A.1 Model Confidence Estimation via Decoupled Smoothing

**Disagreement-weighted PLS alignment.** Using the disagreement weights $\omega_i$ defined in Section 4.1, let $\mathbf{W} = \mathrm{diag}(\sqrt{\omega_1}, \ldots, \sqrt{\omega_N})$, $\mathbf{H}_w = \mathbf{W}\mathbf{H}$, and $\mathbf{Y}_w = \mathbf{W}\mathbf{Y}$. The first PLS direction maximizes the squared covariance between the input and supervision projections:

$$(\mathbf{u}_1, \mathbf{v}_1) = \arg \max_{\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1} \mathrm{Cov}(\mathbf{H}_w\mathbf{u}, \mathbf{Y}_w\mathbf{v})^2. \tag{16}$$

**Proposition 1 (Supervised covariance alignment).** Let $\mathbf{C}_w = \mathbf{H}_w^\top \mathbf{Y}_w$ denote the weighted cross-covariance matrix, up to constants that do not affect the directions. For any unit input direction $\mathbf{u}$, the optimal supervision direction $\mathbf{v}$ satisfies

$$\max_{\|\mathbf{v}\|_2 = 1} (\mathbf{u}^\top \mathbf{C}_w \mathbf{v})^2 = \|\mathbf{C}_w^\top \mathbf{u}\|_2^2. \tag{17}$$

Thus, the input direction selected by PLS maximizes its joint covariance with all model-correctness dimensions.

***Proof.*** The Cauchy–Schwarz inequality gives $(\mathbf{u}^\top \mathbf{C}_w \mathbf{v})^2 \le \|\mathbf{C}_w^\top \mathbf{u}\|_2^2$. If $\mathbf{C}_w^\top \mathbf{u} \neq \mathbf{0}$, equality is attained with $\mathbf{v} \propto \mathbf{C}_w^\top \mathbf{u}$; otherwise, both sides are zero. Since $\mathbf{C}_w = \mathbf{H}^\top \mathbf{W}^2 \mathbf{Y}$, sample $i$'s cross-covariance term is weighted by $\omega_i$. □

Subsequent components follow standard PLS deflation. Let $g_{\mathrm{pls}}$ denote the fitted input-side transform, so that $\mathbf{T} = g_{\mathrm{pls}}(\mathbf{H})$. For a test query, $\mathbf{h}_x^{\mathrm{perf}} = g_{\mathrm{pls}}(\mathbf{h}_x)\mathbf{P}_x^\top$ gives the performance-aware correction in the main text without test labels.

**Propagation matrix construction.** For the sample graph, let $\mathcal{N}_{k_{\mathrm{nn}}}(i)$ denote the neighbors of training sample $i$ in the performance-aligned representation space. We first construct the directed adjacency matrix

$$A^0_{\mathrm{data},ij} = \mathbb{I}\left[j \in \mathcal{N}_{k_{\mathrm{nn}}}(i)\right]. \tag{18}$$

For the model graph, we build edges from the sample-smoothed correctness matrix. Let $\boldsymbol{\mu}$ be the column mean of $\mathbf{Y}^{(1)}$, and let $\bar{\mathbf{Y}}^{(1)} = \mathbf{Y}^{(1)} - \boldsymbol{\mu}$. The nonnegative correlation between models $u$ and $v$ is

$$C_{uv} = \max\left(\frac{\langle \bar{\mathbf{y}}_u^{(1)}, \bar{\mathbf{y}}_v^{(1)} \rangle}{\|\bar{\mathbf{y}}_u^{(1)}\|_2 \|\bar{\mathbf{y}}_v^{(1)}\|_2 + \varepsilon}, 0\right). \tag{19}$$

Let $\mathcal{E}_u$ denote the set of top-$k_{\mathrm{model}}$ correlated models of $u$ that also satisfy $C_{uv} > \eta$. The initial model adjacency matrix is

$$A^0_{\mathrm{model},uv} = C_{uv}\,\mathbb{I}\left[v \in \mathcal{E}_u\right]. \tag{20}$$

Both graphs undergo symmetrization, self-loop removal, and row normalization. For $g \in \{\mathrm{data}, \mathrm{model}\}$, let

$$\mathbf{A}_g = \frac{\mathbf{A}_g^0 + (\mathbf{A}_g^0)^\top}{2}, \qquad \mathrm{diag}(\mathbf{A}_g) = \mathbf{0}, \qquad \mathbf{S}_g = \mathbf{D}_g^{-1}\mathbf{A}_g, \tag{21}$$

where $\mathbf{D}_g$ is the row-degree matrix after self-loop removal; zero-degree rows remain zero.

**Fixed-point diffusion view. Assumption 1 (Non-expansive propagation).** The propagation matrix $\mathbf{S}$ is nonnegative and satisfies $\|\mathbf{S}\|_\infty \le 1$. This condition is ensured by row normalization; if a row has no valid neighbor, it can be treated as an all-zero row and still satisfies the condition.

**Proposition 2 (Well-posed diffusion).** Under Assumption 1, for any $\lambda \in (0, 1)$, the fixed-point equation

$$\mathbf{Z} = (1-\lambda)\mathbf{R} + \lambda \mathbf{S}\mathbf{Z} \tag{22}$$

has a unique solution $\mathbf{Z} = \mathcal{D}_{\lambda,\mathbf{S}}(\mathbf{R})$, where

$$\mathcal{D}_{\lambda,\mathbf{S}}(\mathbf{R}) = (1-\lambda)(\mathbf{I} - \lambda\mathbf{S})^{-1}\mathbf{R} = (1-\lambda)\sum_{t=0}^{\infty} \lambda^t \mathbf{S}^t \mathbf{R}. \tag{23}$$

***Proof.*** Since $\|\mathbf{S}\|_\infty \le 1$, we have $\rho(\lambda\mathbf{S}) \le \|\lambda\mathbf{S}\|_\infty < 1$. Therefore, $\mathbf{I} - \lambda\mathbf{S}$ is invertible and $(\mathbf{I} - \lambda\mathbf{S})^{-1} = \sum_{t=0}^{\infty} \lambda^t \mathbf{S}^t$. Substituting this inverse into the fixed-point equation gives the unique solution. □

This operator describes decay-weighted propagation over multi-hop graph neighborhoods. The two smoothing stages in the main text correspond to

$$\mathbf{Y}^{(1)} = \mathcal{D}_{\alpha,\mathbf{S}_{\mathrm{data}}}(\mathbf{Y}), \qquad \bar{\mathbf{Y}}^{(2)\top} = \mathcal{D}_{\beta,\mathbf{S}_{\mathrm{model}}}(\bar{\mathbf{Y}}^{(1)\top}). \tag{24}$$

The second expression uses transposition to express right-multiplication along the model dimension in this form. Restoring the column mean $\boldsymbol{\mu}$ and clipping the scores element-wise to $[0, 1]$ gives the final training score matrix $\mathbf{Y}^{(2)}$, satisfying the boundedness condition in Proposition 4.

**Proposition 3 (Stability).** Under Assumption 1, for any two input matrices $\mathbf{R}$ and $\mathbf{R}'$,

$$\left\|\mathcal{D}_{\lambda,\mathbf{S}}(\mathbf{R}) - \mathcal{D}_{\lambda,\mathbf{S}}(\mathbf{R}')\right\|_\infty \le \left\|\mathbf{R} - \mathbf{R}'\right\|_\infty. \tag{25}$$

***Proof.*** Using the Neumann series in Proposition 2 and $\|\mathbf{S}^t\|_\infty \le 1$,

$$\left\|\mathcal{D}_{\lambda,\mathbf{S}}(\mathbf{R}) - \mathcal{D}_{\lambda,\mathbf{S}}(\mathbf{R}')\right\|_\infty \le (1-\lambda)\sum_{t=0}^{\infty} \lambda^t \left\|\mathbf{R} - \mathbf{R}'\right\|_\infty = \left\|\mathbf{R} - \mathbf{R}'\right\|_\infty. \tag{26}$$

Thus, graph propagation does not amplify the maximum perturbation in the input supervision matrix. For sample-level smoothing, if the entries of $\mathbf{Y}$ lie in $[0, 1]$, then $\mathbf{Y}^{(1)}$ is a sub-convex combination of multi-hop neighborhood labels and remains in $[0, 1]$. Model-level smoothing satisfies the same non-expansive property on centered residuals. □

**Online model confidence as local kernel estimation.** **Assumption 2 (Local smoothness).** In the performance-aligned space, within the local neighborhood $\mathcal{N}_k(x)$ of a test query $x$, the conditional correctness of model $M_k$ is approximately smooth at the neighborhood scale; that is, neighboring queries have only small variation in conditional correctness.

Online confidence extraction in Section 4.1 can be viewed as local kernel regression. We use its neighborhood weights $w_i(x)$ and scores $s_k(x)$ to analyze the local estimate.

**Proposition 4 (Boundedness and local bias).** If $Y_{ik}^{(2)} \in [0, 1]$, $w_i(x) \geq 0$, and $\sum_i w_i(x) = 1$, then $s_k(x) \in [0, 1]$. Further, let $q_k(z) = \mathbb{E}[Y_k \mid z]$ denote the conditional correctness function of model $M_k$. If $\mathbb{E}[Y_{ik}^{(2)}] = q_k(x_i)$ within the neighborhood and $|q_k(x_i) - q_k(x)| \leq b_x$, then

$$|\mathbb{E}[s_k(x)] - q_k(x)| \leq b_x. \tag{27}$$

***Proof.*** Boundedness follows directly from convexity. For the bias term,

$$|\mathbb{E}[s_k(x)] - q_k(x)| = \left| \sum_i w_i(x) \big(q_k(x_i) - q_k(x)\big) \right| \leq \sum_i w_i(x) b_x = b_x. \tag{28}$$

□

**Proposition 5 (Variance of local model confidence estimation).** Under a conditional-independence approximation within the local neighborhood, if the conditional variance of $Y_{ik}^{(2)}$ is $\sigma_k^2(x)$, then the variance of the weighted estimate is approximately

$$\mathrm{Var}[s_k(x)] \approx \sigma_k^2(x) \sum_{x_i \in \mathcal{N}_k(x)} w_i(x)^2. \tag{29}$$

***Proof.*** Under the conditional-independence approximation, cross-covariance terms vanish, giving $\mathrm{Var}[s_k(x)] \approx \sum_i w_i(x)^2 \,\mathrm{Var}[Y_{ik}^{(2)}]$. Substituting the common conditional variance yields the result, with effective sample size $1/\sum_i w_i(x)^2$. □

### A.2 Budgeted Inference and Selector Learning

**Confidence feature interpretation.** The margin $\Delta_{12}(x) = s_{(1)}(x) - s_{(2)}(x)$ defined in Section 4.2 characterizes the local stability of the top-1 decision.

**Proposition 6 (Top-1 stability under score perturbation).** If the score perturbation $\boldsymbol{\epsilon}$ satisfies $\|\boldsymbol{\epsilon}\|_\infty < \Delta_{12}(x)/2$, then the top-1 model remains unchanged after perturbation.

***Proof.*** For any $j > 1$, the perturbed score gap between the top-1 model and model $M_{(j)}$ is at least

$$\big(s_{(1)}(x) - \|\boldsymbol{\epsilon}\|_\infty\big) - \big(s_{(j)}(x) + \|\boldsymbol{\epsilon}\|_\infty\big) \geq \Delta_{12}(x) - 2\|\boldsymbol{\epsilon}\|_\infty > 0. \tag{30}$$

Thus, $M_{(1)}$ still has the highest score. A larger margin therefore corresponds to a more stable direct-routing decision. □

For the normalized score distribution $\mathbf{p}(x)$, score entropy satisfies $0 \leq H_s(x) \leq \log L$, with the endpoints attained by a point mass and a uniform distribution, respectively. It captures dispersion over the model pool and complements the margin.

**Budget allocation and consensus resolution.** For a top-$K$ resampling action, the default configuration assigns the remaining $B - K$ calls to the highest-ranked retained model.

Let $\mathcal{B}_a(x)$ denote the normalized responses collected by action $a$, with vote count $b(y \mid x, a) = \sum_{(j,t) \in \mathcal{B}_a(x)} \mathbb{I}[\hat{y}_{j,t} = y]$. Ties are resolved first by the highest confidence among models supporting the answer, then by support from the primary model:

$$\hat{y}_a = \arg\max_y \left( b(y \mid x, a),\ \max_{(j,t):\hat{y}_{j,t}=y} s_{(j)}(x),\ \mathbb{I}\left[\exists t, \hat{y}_{1,t} = y\right] \right). \tag{31}$$

Table 4: Detailed information of the datasets used in this paper.

| Abbrev. | Dataset | Category | Size |
|---|---|---|---|
| | ***In-Distribution Datasets*** | | |
| AIME | AIME [39] | Mathematics | 60 |
| M500. | Math500 [33] | Mathematics | 500 |
| MBen. | MathBench [35] | Mathematics | 150 |
| LCB. | LiveCodeBench [21] | Code | 1,055 |
| MBPP | MBPP [1] | Code | 974 |
| BBH | BBH [46] | Logic | 1,080 |
| K&K. | Knights and Knaves [54] | Logic | 700 |
| KOR. | KORBench [38] | Logic | 1,250 |
| FQA. | FinQA [10] | Knowledge | 1,147 |
| GPQA | GPQA [44] | Knowledge | 198 |
| MQA. | MedQA [23] | Knowledge | 1,273 |
| MP. | MMLU-Pro [51] | Knowledge | 1,000 |
| Emory. | EmoryNLP [56] | Affective | 697 |
| MELD | MELD [43] | Affective | 1,232 |
| | ***Out-of-Distribution Datasets*** | | |
| LMB. | LiveMathBench [36] | Mathematics | 121 |
| HE. | HumanEval [7] | Code | 164 |
| WG. | WinoGrande [45] | Logic | 1,267 |
| ARC. | ARC Challenge [11] | Knowledge | 1,172 |
| DD. | DailyDialog [30] | Affective | 1,000 |

**Selector objective.** Let $n_a$ be the number of occurrences of action $a$ in the oracle labels defined in Section 4.2, with class-balanced weight $\lambda_a \propto 1/\max(n_a, 1)$. For training samples $\{(\boldsymbol{\phi}(x_i), a_i^*)\}$, the regularized logistic regression objective is

$$\min_{\theta} -\sum_i \lambda_{a_i^*} \log \pi_\theta(a_i^* \mid \boldsymbol{\phi}(x_i)) + \lambda_{\text{reg}} \|\theta\|_2^2. \tag{32}$$

The class-balanced weights reduce the dominance of frequent actions such as direct routing in the empirical risk, preserving supervision signals from long-tail multi-response actions.

## B Dataset Details

We evaluate on 19 benchmarks spanning five categories. Table 4 summarizes the benchmark configuration used in this paper. Following LLMRouterBench [26] and the corresponding public evaluation pipelines, the reported sizes refer to the number of benchmark instances actually used in our experiments after benchmark preprocessing, rather than the raw corpus size. For each in-distribution (ID) dataset, we randomly split instances into training and test partitions with a 0.7:0.3 ratio, while the out-of-distribution (OOD) datasets are used only for evaluation. For all benchmarks, we follow the official answer extraction and scoring protocol; in practice, this reduces to answer-level accuracy for most multiple-choice or short-answer datasets and execution-based correctness for code-generation datasets.

**AIME.** AIME is a high-difficulty mathematics benchmark built from American Invitational Mathematics Examination problems. It focuses on olympiad-style algebra, geometry, combinatorics, and number theory with short integer answers.

**Math500.** Math500 is a curated 500-problem subset of the MATH benchmark and is widely used for exact-answer mathematical reasoning. It provides a compact but challenging test bed for multi-step symbolic problem solving.

**MathBench.** MathBench is a hierarchical mathematics benchmark that covers multiple difficulty levels, from elementary arithmetic to advanced undergraduate topics. We use it to test whether routing generalizes across a broad range of mathematical subdomains.

**LiveCodeBench.** LiveCodeBench is a contamination-resistant code-generation benchmark collected from recent online programming contests. It evaluates functional correctness with hidden test cases and is intended to reduce benchmark leakage.

**MBPP.** MBPP, short for Mostly Basic Python Problems, contains entry-level Python programming tasks paired with unit tests. It is a standard benchmark for lightweight code-generation evaluation.

**BBH.** BBH, or BIG-Bench Hard, consists of 23 challenging reasoning tasks selected from BIG-Bench. It stresses multi-step symbolic, logical, and commonsense inference where earlier models significantly lagged human performance.

**Knights and Knaves.** Knights and Knaves is a logic-puzzle benchmark in which each character either always tells the truth or always lies. Solving each instance requires explicit reasoning over statements and hidden role assignments.

**KORBench.** KORBench evaluates knowledge-orthogonal reasoning, emphasizing rule following rather than retrieval of domain knowledge. Its tasks are designed to measure whether a model can apply newly given rules in a robust way.

**FinQA.** FinQA is a financial numerical-reasoning benchmark built from corporate reports and tables. Many questions require combining textual evidence with table values and performing multi-step arithmetic operations.

**GPQA.** GPQA is a graduate-level science QA benchmark designed to be Google-proof. It is deliberately difficult for both retrieval systems and non-expert annotators, making it a strong test of deep factual reasoning.

**MedQA.** MedQA is a medical multiple-choice benchmark derived from professional exam questions such as USMLE-style assessments. It targets clinical knowledge, diagnosis, and treatment reasoning.

**MMLU-Pro.** MMLU-Pro is a harder and more reasoning-centric extension of MMLU. Compared with the original MMLU, it uses more difficult questions and a more robust evaluation protocol across 14 knowledge domains.

**EmoryNLP.** EmoryNLP is an emotion-recognition benchmark built from *Friends* transcripts with seven utterance-level emotion labels. It tests whether a model can infer affective states from multi-party conversational context.

**MELD.** MELD is a multimodal conversational emotion benchmark that also originates from *Friends*. In addition to text, it includes aligned audio and visual cues, although in our setting the benchmark is used through its text evaluation protocol.

**LiveMathBench.** LiveMathBench is a contemporary mathematics benchmark assembled from recent competition and exam problems. Its main purpose is to reduce contamination and evaluate whether a routing method generalizes to newly appearing mathematical queries.

**HumanEval.** HumanEval is a handwritten Python function-synthesis benchmark with hidden unit tests. It remains one of the standard code-generation OOD evaluations because it measures executable correctness rather than string overlap.

**WinoGrande.** WinoGrande is a large-scale adversarial commonsense coreference benchmark derived from the Winograd-schema setting. It evaluates pronoun resolution and commonsense discrimination under carefully controlled lexical cues.

**ARC Challenge.** ARC Challenge is the difficult split of the AI2 Reasoning Challenge and contains grade-school science questions that simple retrieval baselines usually fail on. We use it as an OOD knowledge benchmark to test whether routing transfers beyond the training domains.

Table 5: Candidate model pool used in our experiments.

| Abbrev. | Model | Organization | Params |
|---|---|---|---|
| Qwen3-8B | Qwen3-8B [55] | Alibaba/Qwen | 8B |
| NVIDIA-Nemo | NVIDIA-Nemotron-Nano-9B-v2 [41] | NVIDIA | 9B |
| DS-Qwen3 | DeepSeek-R1-0528-Qwen3-8B [18] | DeepSeek | 8B |
| GLM-Z1 | GLM-Z1-9B-0414 [48] | Z.ai | 9B |
| MiniCPM | MiniCPM4.1-8B [40] | OpenBMB | 8B |
| Intern-S1-mini | Intern-S1-mini [2] | InternLM | 8B |
| DS-Qwen | DeepSeek-R1-Distill-Qwen-7B [18] | DeepSeek | 7B |
| Internlm3-it | internlm3-8b-instruct [4] | InternLM | 8B |
| Fin-R1 | Fin-R1 [37] | SUFE-AIFLM Lab | 7B |
| Qwen-Coder | Qwen2.5-Coder-7B-Instruct [19] | Alibaba/Qwen | 7B |
| Cogito-v1 | cogito-v1-preview-llama-8B [12] | DeepCogito | 8B |
| Gemma-2-it | gemma-2-9b-it [15] | Google | 9B |
| Llama-Nemo | Llama-3.1-Nemotron-Nano-8B-v1 [3] | NVIDIA | 8B |
| OpenThinker | OpenThinker3-7B [17] | OpenThoughts | 7B |
| Glm-4-chat | glm-4-9b-chat [48] | Z.ai | 9B |
| Llama-3.1-it | Llama-3.1-8B-Instruct [16] | Meta | 8B |
| Granite-3.3-it | granite-3.3-8b-instruct [20] | IBM | 8B |
| DH-Llama3-it | DeepHermes-3-Llama-3-8B-Preview [49] | Nous Research | 8B |
| UltraMedical | Llama-3.1-8B-UltraMedical [58] | Tsinghua C3I | 8B |
| MiMo-RL | MiMo-7B-RL-0530 [53] | Xiaomi MiMo | 7B |

**DailyDialog.** DailyDialog is a manually labeled multi-turn dialogue benchmark built from everyday human conversations. In our setting, we use it as an OOD affective benchmark to test whether routing and aggregation strategies transfer to conversational emotion understanding beyond the training domains.

# C Model Details

We use the same 20-model open-source candidate pool as LLMRouterBench [26]. The pool includes general-purpose instruction models, reasoning-tuned variants, and domain specialists in code, medicine, and finance. Table 5 summarizes the abbreviations, sources, releasing organizations, and approximate parameter scales. Following the benchmark convention, we report rounded parameter scales (7B/8B/9B) for compact presentation.

## C.1 Full Model Pool Results

Tables 6 and 7 report the average accuracy of all 20 candidate models on the ID and OOD benchmarks, respectively.

# D Baseline Details

Unless otherwise specified, all baselines are evaluated under the same protocol as our method, including the same 20 candidate models, the same data split, and the same OOD benchmarks. For voting- and self-consistency-based multi-model baselines, we set the available budget to three candidate outputs. For proposer–aggregator methods, we set the proposer-side budget to three and additionally retain one aggregation call to follow their natural inference pipeline. For route-and-aggregate methods that generate candidate responses before adaptive switching, we also set the number of candidate models to three; if aggregation is finally performed, one additional aggregator call is counted. When repeated sampling from the same model on the same query is required, we generate additional outputs ourselves with temperature 0.7 and top_p 1.0, since the released benchmark outputs in LLMRouterBench [26] provide only one default answer for each query–model pair. Unless otherwise noted, unspecified settings follow the original papers or official implementations. Table 8

Table 6: Single-model accuracy results of the 20-model pool on the in-distribution benchmarks.

| Model | Mathematics | | | Code | | Logic | | | Knowledge | | | | Affective | | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME | M500 | MBen. | LCB. | MBPP | BBH | K&K. | KOR. | FQA. | GPQA | MQA. | MP. | Emory. | MELD | |
| Qwen3-8B | 80.95 | 93.90 | 92.70 | 66.83 | 54.61 | 84.26 | 77.52 | 54.67 | 73.77 | 57.38 | 78.46 | 69.90 | 38.48 | 55.44 | 69.92 |
| NVIDIA-Nemo | 74.60 | 93.52 | 95.24 | 64.89 | 42.91 | 86.42 | 71.02 | 58.32 | 70.93 | 52.86 | 75.13 | 70.52 | 38.69 | 46.87 | 67.28 |
| DS-Qwen3 | 69.05 | 93.90 | 90.48 | 63.27 | 34.37 | 89.29 | 61.61 | 56.34 | 69.84 | 59.52 | 81.23 | 69.43 | 38.96 | 51.54 | 66.35 |
| GLM-Z1 | 65.87 | 93.90 | 97.78 | 60.75 | 63.73 | 83.86 | 46.58 | 54.21 | 68.59 | 55.00 | 70.08 | 68.14 | 34.52 | 46.29 | 64.95 |
| MiniCPM | 70.63 | 92.95 | 93.02 | 62.96 | 35.35 | 88.98 | 60.73 | 29.30 | 66.12 | 50.95 | 65.26 | 69.05 | 34.86 | 53.01 | 62.37 |
| Intern-S1-mini | 76.98 | 92.48 | 67.62 | 25.78 | 43.20 | 83.73 | 78.54 | 59.39 | 69.88 | 45.71 | 69.07 | 54.52 | 36.77 | 51.04 | 61.05 |
| DS-Qwen | 42.06 | 88.00 | 88.89 | 40.60 | 53.53 | 69.97 | 49.90 | 46.48 | 63.41 | 34.52 | 33.69 | 47.05 | 27.14 | 35.68 | 51.49 |
| Internlm3-it | 7.14 | 68.76 | 58.41 | 20.64 | 61.04 | 62.17 | 25.46 | 33.83 | 61.61 | 35.24 | 67.13 | 54.62 | 36.91 | 51.24 | 46.01 |
| Fin-R1 | 13.49 | 75.71 | 64.13 | 6.49 | 68.45 | 61.38 | 19.03 | 33.45 | 68.05 | 29.05 | 62.68 | 48.38 | 35.00 | 51.31 | 45.47 |
| Qwen-Coder | 1.59 | 66.57 | 68.57 | 27.31 | 75.04 | 58.02 | 20.85 | 33.68 | 62.91 | 31.90 | 48.54 | 44.62 | 37.53 | 52.90 | 45.00 |
| Cogito-v1 | 1.59 | 50.48 | 53.65 | 17.17 | 52.95 | 69.18 | 23.70 | 43.01 | 62.32 | 36.19 | 65.18 | 55.62 | 38.14 | 56.83 | 44.72 |
| Gemma-2-it | 1.59 | 47.62 | 55.24 | 17.08 | 62.65 | 59.83 | 9.68 | 34.90 | 64.54 | 30.95 | 63.31 | 52.57 | 39.58 | 53.59 | 42.37 |
| Llama-Nemo | 45.24 | 90.38 | 51.11 | 45.56 | 61.24 | 26.46 | 21.39 | 28.65 | 32.66 | 31.90 | 42.82 | 40.57 | 29.32 | 40.42 | 41.98 |
| OpenThinker | 41.27 | 86.00 | 62.86 | 45.43 | 11.95 | 44.00 | 17.94 | 11.24 | 61.28 | 30.24 | 51.80 | 41.57 | 23.51 | 33.36 | 40.17 |
| Glm-4-chat | 1.59 | 48.48 | 48.89 | 16.54 | 62.99 | 45.72 | 11.65 | 37.07 | 56.22 | 24.52 | 62.83 | 46.62 | 37.66 | 55.48 | 39.73 |
| Llama-3.1-it | 8.73 | 48.67 | 46.67 | 16.76 | 62.51 | 57.58 | 13.13 | 23.39 | 52.80 | 26.19 | 69.04 | 46.33 | 34.24 | 47.92 | 39.57 |
| Granite-3.3-it | 5.56 | 67.81 | 60.63 | 14.33 | 35.40 | 30.95 | 19.50 | 30.97 | 61.65 | 31.19 | 59.95 | 43.67 | 34.24 | 51.20 | 39.08 |
| DH-Llama3-it | 0.00 | 32.57 | 39.05 | 12.89 | 54.85 | 48.77 | 15.84 | 32.23 | 50.25 | 30.95 | 57.22 | 36.33 | 35.89 | 47.34 | 35.30 |
| UltraMedical | 0.00 | 42.67 | 24.44 | 13.47 | 54.02 | 38.10 | 15.50 | 13.41 | 54.30 | 26.90 | 68.96 | 40.00 | 31.72 | 42.01 | 33.25 |
| MiMo-RL | 0.00 | 26.67 | 8.57 | 6.26 | 56.80 | 30.03 | 3.66 | 30.40 | 15.79 | 3.81 | 29.77 | 18.95 | 29.12 | 40.66 | 21.46 |

Table 7: Single-model accuracy results of the 20-model pool on the out-of-distribution benchmarks.

| Model | Mathematics | Code | Logic | Knowledge | Affective | Avg |
|---|---|---|---|---|---|---|
| | LiveMathBench | HumanEval | Winogrande | ARC-C. | DailyDialog | |
| Qwen3-8B | 74.38 | 62.20 | 82.79 | 93.77 | 58.16 | 74.26 |
| GLM-Z1 | 73.55 | 60.37 | 80.35 | 91.98 | 49.88 | 71.22 |
| NVIDIA-Nemo | 70.25 | 48.17 | 78.61 | 93.26 | 45.85 | 67.23 |
| MiniCPM | 71.90 | 32.93 | 77.58 | 93.69 | 57.46 | 66.71 |
| Intern-S1-mini | 70.25 | 43.29 | 72.38 | 89.51 | 51.83 | 65.45 |
| DS-Qwen3 | 71.07 | 25.00 | 82.00 | 94.28 | 54.19 | 65.31 |
| Glm-4-chat | 19.83 | 74.39 | 76.16 | 91.04 | 56.60 | 63.61 |
| Fin-R1 | 39.67 | 77.44 | 73.09 | 89.08 | 36.29 | 63.11 |
| Llama-Nemo | 62.81 | 62.80 | 53.04 | 75.68 | 51.92 | 61.25 |
| Internlm3-it | 28.93 | 65.85 | 71.03 | 89.76 | 50.67 | 61.25 |
| Qwen-Coder | 31.40 | 77.44 | 61.40 | 82.17 | 49.28 | 60.34 |
| Gemma-2-it | 20.66 | 64.63 | 72.38 | 88.31 | 54.13 | 60.02 |
| Cogito-v1 | 17.36 | 69.51 | 64.72 | 88.14 | 54.97 | 58.94 |
| DS-Qwen | 61.16 | 45.12 | 55.41 | 81.91 | 37.69 | 56.26 |
| Granite-3.3-it | 31.40 | 51.22 | 69.53 | 84.39 | 41.80 | 55.67 |
| Llama-3.1-it | 19.01 | 61.59 | 60.46 | 85.58 | 42.84 | 53.89 |
| MiMo-RL | 12.40 | 52.44 | 63.22 | 77.13 | 60.53 | 53.14 |
| OpenThinker | 62.81 | 7.32 | 64.09 | 73.72 | 32.82 | 48.15 |
| DH-Llama3-it | 12.40 | 51.22 | 60.38 | 77.47 | 37.68 | 47.83 |
| UltraMedical | 12.40 | 56.10 | 57.06 | 79.52 | 25.71 | 46.16 |

summarizes the resulting default call costs under this budget cap, along with whether model choice or budget decision is query-adaptive before response generation.

**RouterDC.** We use the official RouterDC implementation [8] and retrain the router on the same data split as in our main experiments, replacing its encoder with `gte-Qwen2-7B-instruct`. Given the candidate model pool and training scale in our setting, we set the training batch size to 16. At test time, RouterDC selects a single routed model for each query, while the remaining hyperparameters follow the original implementation.

**GraphRouter.** We use the official GraphRouter implementation [14] and rebuild its task-query-model heterogeneous graph on the same 20-model pool used in this paper. Specifically, we encode queries, task descriptions, and model descriptions using `gte-Qwen2-7B-instruct`, and provide fixed textual descriptions for all datasets and candidate models to construct the contextual information for task and model nodes. We report the *Performance First* setting and train the model for 2,000 epochs. For queries with multiple tie-optimal models, we adopt tie-aware multi-hot supervision to cover all tie-optimal models. Beyond these changes, the remaining training and inference settings follow the original implementation.

Table 8: Default inference-cost convention for baselines and PARS under a budget cap of 3 candidate outputs. Costs are counted by model calls. Dynamic R/A denotes dynamic routing-aggregation. †JiSi invokes the aggregator only when adaptive switching keeps multiple candidate responses.

| Method | Type | Query-adaptive model choice | Pre-generation budget decision | Candidate outputs | Aggregator call | Calls |
|---|---|---|---|---|---|---|
| RouterDC | Single route | ✓ | × | 1 model | × | 1 |
| GraphRouter | | ✓ | × | 1 model | × | 1 |
| EmbedLLM | | ✓ | × | 1 model | × | 1 |
| Avengers | | ✓ | × | 1 model | × | 1 |
| PARS (Single) | | ✓ | × | 1 model | × | 1 |
| Majority Voting | Aggregation | × | × | 3 models | × | 3 |
| Self Consistency | | × | × | 3 samples | × | 3 |
| MoA | | × | × | 3 proposers | ✓ | 4 |
| Self-MoA | | × | × | 3 samples | ✓ | 4 |
| Symbolic-MoE | | ✓ | × | 3 models | ✓ | 4 |
| JiSi | Dynamic R/A | ✓ | × | 3 models | ✓† | 3 or 4 |
| PARS | | ✓ | ✓ | 1–3 models | × | 1–3 |

**EmbedLLM.** We follow the original EmbedLLM method [63] and train the router on the same data split and candidate pool as in our main experiments. To stay consistent with our main setup, we generate query embeddings using `gte-Qwen2-7B-instruct` and adjust the input dimension to 1024 accordingly. The remaining training configuration is left unchanged.

**Avengers.** We evaluate the original clustering-based routing method in Avengers [61] on the same candidate pool and data split as in this paper. This method does not train an additional parametric router; instead, it builds cluster-wise model rankings from the cluster structure of training queries and uses them for model selection at test time. To stay consistent with our main setup, we generate query embeddings using `gte-Qwen2-7B-instruct` and set the number of clusters to 30. The rest of the pipeline follows the original method.

**Majority Voting.** For Majority Voting [6], we first construct a fixed three-model candidate set from the training data. Specifically, for each in-distribution dataset, we select the top three models according to their average accuracy on the training split of that dataset; for datasets unseen during training, including all OOD benchmarks, we fall back to a global top-3 pool determined by each model's average performance over all training datasets. At test time, the final prediction is obtained by equal-weight majority voting over the outputs of these three models. Ties are resolved deterministically. This baseline therefore uses three fixed candidate responses for each query without query-level routing.

**Self Consistency.** For Self-Consistency [9], we first choose an anchor model for each in-distribution dataset based on training-set performance, and use a global anchor model for unseen datasets. We then aggregate up to three independently generated outputs from the anchor model and obtain the final prediction via normalized-answer voting. If fewer than two valid outputs are available for a query, we fall back to the single prediction of the anchor model. The remaining generation and voting settings follow the original implementation.

**Mixture-of-Agents (MoA).** We implement a single-layer MoA baseline [50] in our setting. The three proposer models are selected as the global top-3 models according to their average performance over all training datasets, and the aggregator is chosen as the best single model under the same criterion. For each query, the aggregator receives the three proposer responses together with a fixed synthesis instruction and produces the final answer. The aggregated output is then evaluated with the same benchmark evaluator as the other methods, while unspecified components remain unchanged.

**Self-MoA.** We further implement Self-MoA [28] as another proposer–aggregator baseline. Unlike MoA, Self-MoA does not use multiple heterogeneous proposer models; instead, it collects three independently generated outputs from the same anchor model and lets the aggregator synthesize these candidate answers. To keep model selection consistent with MoA, both the anchor model and

the aggregator are selected according to average model performance over all training datasets, i.e., the same global best model is used for both roles. This setting corresponds to three homogeneous proposer outputs plus one aggregation call.

**Symbolic-MoE.** We implement Symbolic-MoE [5] following its instance-level skill-based recruitment pipeline and adapt it to our 20-model pool and data split. Model skill profiles and aggregator selection are both built from the training split. For each query, the system recruits up to three candidate models according to its skill keywords, obtains their responses, and lets the selected aggregator synthesize these candidate outputs. Steps involving embedding matching uniformly use `gte-Qwen2-7B-instruct`, while the remaining adaptable hyperparameters follow the original method.

**JiSi.** We construct the JiSi [47] baseline from its route-and-aggregate workflow and rebuild its embedding bank using the training split in our setting, with both query and response embeddings generated by `gte-Qwen2-7B-instruct`. For each query, JiSi first selects the top-3 candidate models and generates their responses, and then performs response-aware rerouting and adaptive switching based on these responses. If multiple candidate responses remain after switching, the selected aggregator is called for synthesis. Because the response-aware stage depends on generated candidate responses, the top-3 candidate model calls are always counted as inference cost, and one aggregator call is counted only when aggregation is actually executed. The remaining adaptable hyperparameters and inference steps follow the original method.

# E Implementation Details

We further summarize the default implementation settings of PARS (Single) and PARS here, without repeating the dataset, model-pool, and baseline information already given above. Unless otherwise specified, all main results are obtained on the same 20-model pool as in the main text, and all query embeddings are produced by `gte-Qwen2-7B-instruct`.

**Environment.** All experiments were conducted on a server with two Intel Xeon Platinum 8375C CPUs, 503GB system memory, and four NVIDIA RTX A6000 GPUs, each with 48GB memory.

**Overall Protocol.** We follow LLMRouterBench evaluation protocol [26] and construct ID train/test splits with seven random seeds: 42, 233, 1024, 2026, 2027, 2060, and 3407. The in-distribution results in the main tables summarize the outcomes on 14 ID datasets, while the out-of-distribution results summarize the outcomes on 5 OOD datasets. We report averages across seeds.

**Performance-Aligned Confidence Estimator.** For PARS (Single), the default confidence estimator uses 50 nearest neighbors during retrieval, sets the diffusion scale to 5.0, and keeps 16 PLS components in the performance subspace. In the structured smoothing stage, we use a smoothing coefficient of 0.5 on the sample graph and a model-level smoothing coefficient of 0.35 on a top-6 model graph. Unless otherwise specified, this configuration is kept fixed in the single-model setting and also serves as the shared confidence-estimation backbone of PARS.

**Test-Time Compute Allocation.** For PARS, the default test-time compute budget is 3 model calls. We fit a selector on the training data to choose an action at test time based on the query state features. The action space includes Direct Routing, Homogeneous Self-Consistency, Heterogeneous Voting, and Hybrid Resampling over the top two routed candidates. When repeated sampling is required, the additional outputs are generated with temperature 0.7 and top_p 1.0. In the default resampling branch, two routed candidates are retained and the remaining budget is assigned preferentially to the currently highest-ranked model. The selector is implemented as a class-balanced logistic regression trained on standardized state features. For uncertainty-related features, the default entropy and margin weights are set to 0.3 and 0.7, respectively, and the maximum number of iterations is set to 2000.

**Difficulty Partition for Analysis Plots.** For the model-call and strategy-distribution analyses, we follow the difficulty partition used in prior work [47]. For each query, we first compute the proportion of models in the current candidate pool that answer it correctly, denoted by *correct_ratio*. We then partition queries into three groups: Easy if *correct_ratio* $> 0.7$, Medium if $0.3 \leq$ *correct_ratio* $\leq$

Table 9: Seed-level variability of average accuracy across ID and OOD benchmarks.

| Method | ID Avg. | OOD Avg. |
|---|---|---|
| *Single-Model Methods* | | |
| RouterDC | $60.57 \pm 0.00$ | $69.76 \pm 0.00$ |
| GraphRouter | $71.26 \pm 0.64$ | $74.75 \pm 0.56$ |
| EmbedLLM | $71.54 \pm 0.65$ | $74.34 \pm 0.58$ |
| Avengers | $71.19 \pm 0.79$ | $75.45 \pm 0.58$ |
| **PARS (Single)** | $\mathbf{72.56 \pm 0.63}$ | $\mathbf{77.01 \pm 0.19}$ |
| *Multi-Model Methods* | | |
| Majority Voting | $73.73 \pm 0.14$ | $70.96 \pm 0.00$ |
| Self Consistency | $72.74 \pm 0.83$ | $74.81 \pm 0.00$ |
| MoA | $70.19 \pm 0.25$ | $77.38 \pm 1.13$ |
| Self-MoA | $69.40 \pm 0.19$ | $77.72 \pm 0.00$ |
| Symbolic-MoE | $71.04 \pm 1.01$ | $75.86 \pm 0.66$ |
| JiSi | $73.83 \pm 0.70$ | $77.19 \pm 0.41$ |
| **PARS** | $\mathbf{75.19 \pm 0.47}$ | $\mathbf{78.28 \pm 0.13}$ |

Table 10: Leave-one-domain-out accuracy (%).

| Method | Math (LMB.) | Code (HE.) | Logic (WG.) | Knowledge (ARC.) | Affective (DD.) | Avg. ($\Delta$ vs. Full) |
|---|---|---|---|---|---|---|
| *Single-model setting* | | | | | | |
| EmbedLLM | 68.90 | 54.78 | 79.08 | 88.77 | 52.54 | 68.81 ($-5.53$) |
| Avengers | 69.31 | 56.21 | 78.00 | 90.53 | 53.00 | 69.41 ($-6.04$) |
| PARS (Single) | 71.31 | 64.92 | 81.39 | 92.06 | 55.79 | **73.09** ($-3.92$) |
| *Multi-model setting* | | | | | | |
| Symbolic-MoE | 70.55 | 59.21 | 80.56 | 92.16 | 56.06 | 71.71 ($-4.15$) |
| JiSi | 72.31 | 56.71 | 83.54 | 94.11 | 55.75 | 72.49 ($-4.70$) |
| PARS | 73.13 | 66.75 | 83.21 | 94.58 | 57.81 | **75.10** ($-3.18$) |

0.7, and Hard if *correct_ratio* $< 0.3$. This *correct_ratio* is used only for post-hoc analysis and is not provided as an input feature during online inference.

# F Additional Experiments

## F.1 Seed-Level Variability

Table 9 reports seed-level standard deviations for the main ID and OOD comparisons. Across random seeds, both PARS (Single) and the full PARS maintain stable advantages over their corresponding baselines.

## F.2 Cross-Domain Generalization

To evaluate generalization to unseen task domains, we conduct five-fold leave-one-domain-out (LODO) evaluation. In each fold, we exclude all training data from one domain and rebuild or train each method using the remaining four domains. Evaluation is performed on the excluded domain's corresponding OOD benchmark from Table 2. All methods use their main-experiment configurations.

Table 10 shows that PARS (Single) and PARS outperform the strongest baselines by 3.68% and 2.61% in the single-model and multi-model settings, respectively. Both also show the smallest declines in average accuracy relative to training on all five domains, indicating more stable performance under cross-domain distribution shift.

## F.3 End-to-End Inference Efficiency

We evaluate all methods without generation caching on a shared set of queries sampled equally from the five OOD benchmarks, using the same five-model deployment and generation settings.

Table 11: End-to-end inference cost and serving efficiency.

| Method | Accuracy (%) ↑ | Calls ↓ | Input / Output Tokens ↓ | Routing Overhead (s) | Latency P50/P95 (s) ↓ | Throughput (query/s) ↑ |
|---|---|---|---|---|---|---|
| Majority Voting | 71.46 | 3.00 | 567 / 10,915 | 0.00 | 27.45 / 419.52 | 0.1113 |
| Self Consistency | 73.52 | 3.00 | 563 / 10,983 | 0.00 | 24.24 / 135.43 | 0.0647 |
| MoA | 76.35 | 4.00 | 1,615 / 13,343 | 0.00 | 41.29 / 466.69 | 0.0784 |
| Self-MoA | 76.28 | 4.00 | 1,957 / 11,762 | 0.00 | 37.06 / 188.51 | 0.0586 |
| Symbolic-MoE | 76.12 | 4.00 | 2,485 / 14,569 | 1.86 | 41.88 / 458.27 | 0.0579 |
| JiSi | 77.04 | 3.98 | 1,440 / 11,082 | 1.56 | 39.55 / 460.68 | 0.0984 |
| PARS | **77.86** | **1.76** | **393 / 5,094** | 0.31 | **23.10 / 95.89** | **0.1392** |

Table 12: Accuracy (%) with different smoothing configurations.

| Variant | PARS (Single) | PARS |
|---|---|---|
| No smoothing | 71.76 | 73.96 |
| Inter-query only | 72.14 | 74.48 |
| Inter-model only | 72.08 | 74.39 |
| Full smoothing | **72.56** | **75.19** |

We report average calls, input/output tokens, and routing overhead per query. Token counts cover all online LLM calls, including the aggregator's candidate-response context. Routing overhead includes online embedding and routing operations, while end-to-end latency covers the entire inference process from query input to final response, including candidate generation and aggregation. Offline training and answer evaluation are excluded. Latency is measured on a separate fixed query subset with query-level concurrency set to one, preserving each method's candidate-call parallelism. Throughput is measured independently at a fixed query-level concurrency under the same hardware allocation; accuracy, calls, and tokens are computed from the same responses used for throughput measurement.

Table 11 shows that PARS exceeds the strongest baseline, JiSi, by 0.82% in accuracy, while achieving the lowest calls, input/output tokens, and P50/P95 latency, as well as the highest throughput. Its routing overhead averages 0.31 seconds per query and is included in the latency measurements. These results show that the accuracy–cost advantage also holds under end-to-end serving measurements.

### F.4 Additional Ablation Studies

**Smoothing Components.** Under the same experimental settings as Table 3, we evaluate variants retaining only inter-query or inter-model smoothing. Table 12 shows that both single-stage variants outperform no smoothing. Combining both smoothing stages improves PARS (Single) and PARS by 0.42% and 0.71% over the best single-stage variant, respectively, indicating that both stages contribute to inference accuracy.

**Selector Policy.** We compare the learned selector with a margin/entropy threshold policy using the same confidence estimator, action space, and budget. The threshold and fallback action are determined solely from the training split, with the threshold policy matched to the learned selector's average number of calls. Table 13 shows that, at the same average cost of 1.7 calls, the learned selector achieves 0.89% higher accuracy than the threshold policy.

### F.5 Hyperparameter Sensitivity Analysis

Under the default configuration, the sensitivity curves in Figure 6 show how Diffusion Scale $\gamma$, the number of PLS Components, the Neighborhood Size $k$, and the training set size affect accuracy. The curves indicate that, under both the single-model and multi-model settings, PARS remains stable under these setting variations, demonstrating favorable robustness. The training-set-size panel further shows that both variants remain effective even when the performance estimator is built from a reduced training subset.

Table 13: Accuracy and average calls with different inference policies.

| Policy | Accuracy (%) | Avg. Calls |
|---|---|---|
| Direct routing | 72.56 | 1.0 |
| Margin/entropy threshold | 74.30 | 1.7 |
| Learned selector | **75.19** | 1.7 |

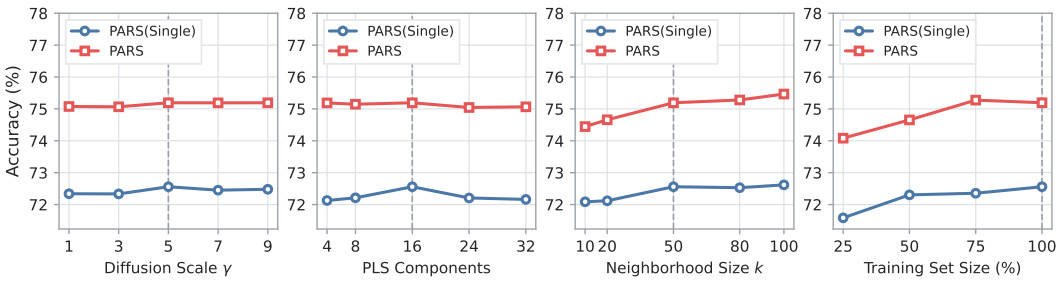



Figure 6: Hyperparameter sensitivity analysis under the default configuration. Each panel varies one factor while keeping the others fixed at their default values.

### F.6 Embedding Backbone Sensitivity

Across the three mainstream embedding backbones compared in Table 14, PARS maintains similar performance under different encoders and does not show clear dependence on any particular representation model. This suggests that PARS is not sensitive to the choice of embedding backbone and can maintain stable gains across different representation spaces.

Table 14: Accuracy comparison across embedding models.

| Embedding | PARS (Single) | PARS |
|---|---|---|
| BGE-M3 | 72.66 | 74.96 |
| GTE-Qwen2-7B | 72.56 | 75.19 |
| Qwen3-Embed-8B | 72.21 | 75.41 |

### F.7 Training-Side Cost under Pool Expansion

The main offline prerequisite of PARS is the model correctness matrix on the training split: each training query needs responses from the candidate models and a task-specific evaluation outcome indicating whether each model is correct. This prerequisite is not unique to PARS. Query-level routing and model-selection methods generally require the same kind of training-side performance signal to learn or estimate which model should be selected for a query. Therefore, Table 15 focuses on additional method-side cost after collecting the shared response-and-evaluation matrix.

Under the current benchmark setting with 20 similarly scaled candidate models, we construct nested pools with $L = 5/10/15/19/20$ and remeasure the cost of fresh confidence-estimator construction and selector training with five repeats per pool size. As summarized in Table 15, even when the pool is expanded from 5 to 20 models, confidence-estimator construction increases only from 6.43s to 7.09s, and selector training increases only from 6.04s to 6.39s. Compared with methods that involve heavier router training, such as fitting additional neural or graph-based parameters in RouterDC [8] and GraphRouter [14], PARS therefore has substantially lower additional training-side cost once the common response-and-evaluation matrix is available.

Table 15: Training-side cost under pool expansion (5-run mean).

| Pool $L$ | Estimator (s) | Selector (s) |
|---|---|---|
| 5 | 6.43 | 6.04 |
| 10 | 6.73 | 6.11 |
| 15 | 6.81 | 6.29 |
| 19 | 6.84 | 6.30 |
| 20 | 7.09 | 6.39 |

### F.8 Routing Failure Rescue Analysis

Figure 7 decomposes each dataset into queries already solved by top-1 routing, routing errors that can still be recovered by subsequent multi-response actions, and the remaining cases under the current action space. Overall, the main gain of PARS still comes from strong top-1 routing, while multi-response inference further improves performance by rescuing a subset of routing failures. After

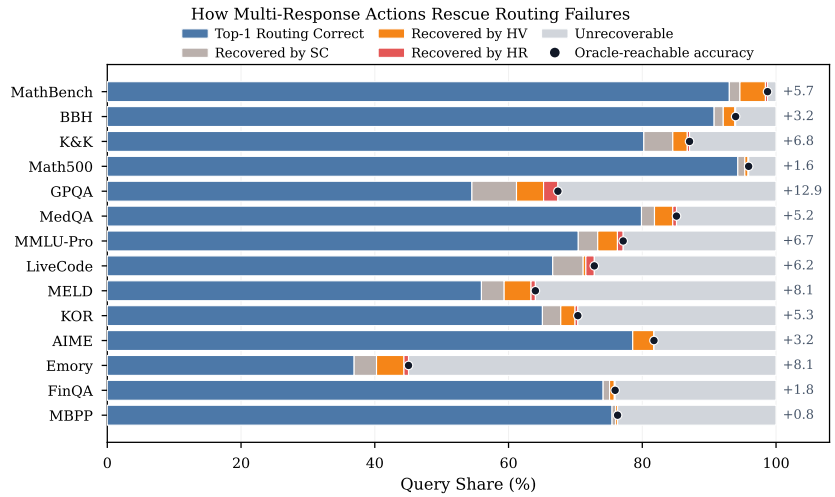



Figure 7: Routing failure rescue analysis across ID datasets. Each bar decomposes queries into top-1 routing correct cases, errors recovered by multi-response actions, and unrecoverable cases under the current action space.

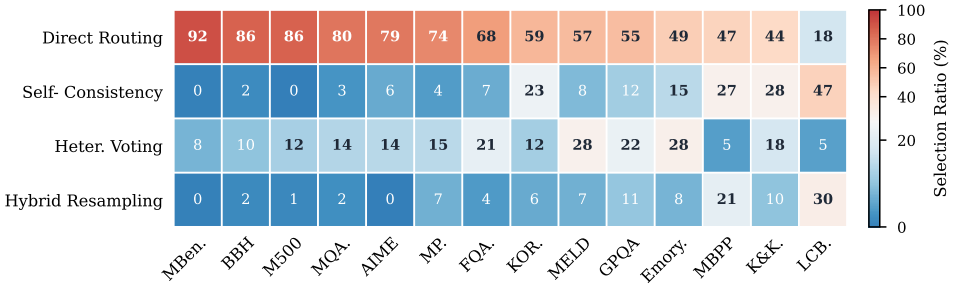



Figure 8: Dataset-level selection ratios of execution strategies. Each cell reports the rounded percentage of queries for which PARS selects the corresponding action on a dataset.

accounting for all actions, self-consistency and heterogeneous voting contribute comparably to failure recovery, while hybrid resampling also provides complementary rescue effects on some datasets. The pattern further varies across datasets: GPQA, MELD, and Emory show more visible rescue gains, whereas MBPP and Math500 leave relatively little room for additional recovery.

### F.9 Dataset-Level Strategy Distribution

Figure 8 gives a dataset-level view of the actions selected by PARS. The heatmap complements the difficulty-level analysis in Figure 4**(c)** by showing that strategy usage varies substantially across datasets, with some datasets dominated by Direct Routing and others relying more on multi-response actions.

### F.10 Oracle Strategy Prediction Analysis

From a category-level perspective, Figure 9 jointly shows the current action-space coverage and the selector's accuracy on oracle-solvable queries. Although the oracle-solvable fraction varies substantially across domains, reaching about 95% for Mathematics but only about 57% for Affective tasks, once the analysis is restricted to samples for which the current action space contains at least one feasible action, the lightweight selector maintains high accuracy in most categories and remains close to 90% even for Affective tasks.

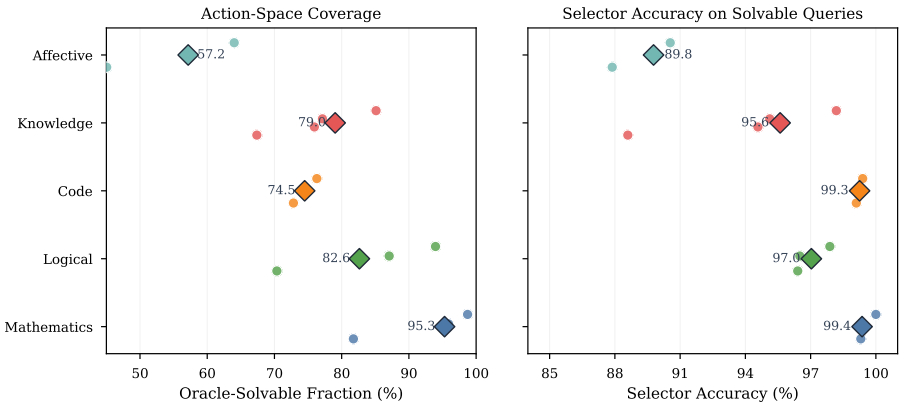



Figure 9: Category-level analysis of action-space coverage and selector accuracy under the current action space. Circles denote dataset-level results, and diamonds denote category means.

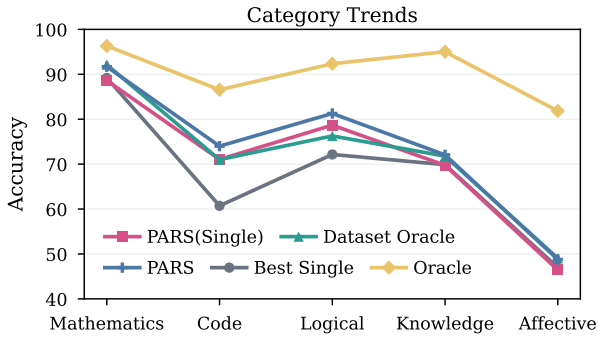



Figure 10: Category-level comparison of PARS (Single), PARS, Best Single, Dataset Oracle, and Oracle.

### F.11 Oracle Reference Analysis

Figure 10 compares PARS (Single), PARS, Best Single, Dataset Oracle, and Oracle across categories. Under the single-model setting, PARS (Single) is already overall close to Dataset Oracle, indicating that the method approaches the dataset-level optimum in single-model inference. In the multi-model setting, the full PARS further surpasses Dataset Oracle in category averages, showing that multi-model inference yields gains beyond single-model inference. At the same time, a visible gap remains between PARS and the absolute Oracle, suggesting that although the current method already exploits model complementarity effectively, there is still room for improvement toward the global theoretical optimum.

### F.12 Model Selection Diversity

Figure 11 and Figure 12 show how model usage is distributed across datasets under PARS (Single) and PARS, respectively. For readability, each pie explicitly shows the top-5 selected models on the corresponding dataset, while the remaining selected models are merged into "Other (merged)". The results reveal a clear dataset-conditioned specialization pattern: on some tasks, the confidence estimator consistently selects strong models that better match the capability requirements of the dataset, whereas on other tasks, model usage is more distributed, reflecting complementary strengths among candidate models. Moreover, compared with PARS (Single), the full PARS involves a richer set of model participants on several datasets because multi-response actions, such as heterogeneous voting and resampling, can jointly use multiple high-ranked candidates for the same query. Overall, this analysis suggests that PARS does not aim to call all models uniformly; instead, it builds on a strong confidence-estimation backbone and selects suitable models or model combinations according to dataset characteristics and query states.

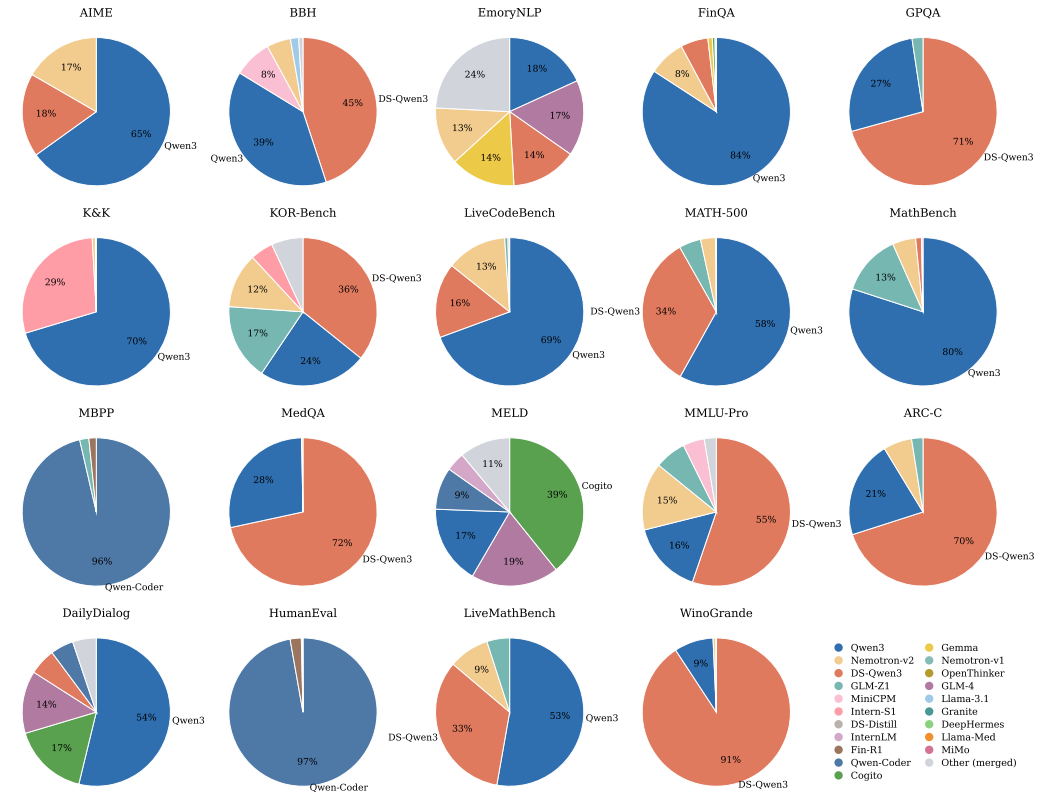



Figure 11: Dataset-level model usage distributions of PARS (Single).

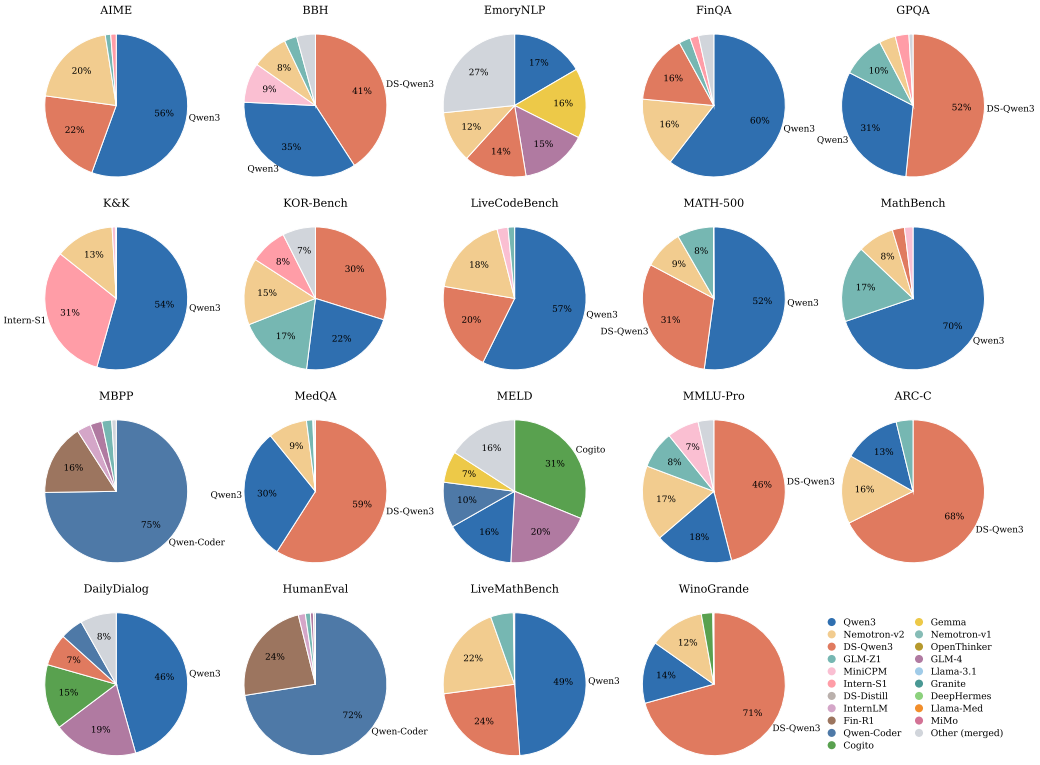



Figure 12: Dataset-level model usage distributions of PARS.

## G Broader Impacts

This work aims to improve the efficiency and accuracy of heterogeneous LLM systems. By adaptively choosing between single-model inference and multi-model collaboration for different queries, PARS may reduce unnecessary model calls, lower inference cost and resource consumption, and make better use of complementary model capabilities. At the same time, the method relies on outputs from candidate models and may inherit biases, unreliable answers, or safety risks from the underlying models. Practical deployments should therefore use appropriate safeguards, usage restrictions, and human oversight, especially in high-risk domains.

## H Case Studies of Adaptive Action Selection

We further present five representative cases to illustrate when the selected action prefers direct routing, when it invokes multi-response inference, and when multi-response inference can still fail under the current action space.

## Case 2: Direct Routing

**Dataset:** ARC Challenge
**Query:** Sugars are necessary for human cell function. Which of the following are human cells not capable of doing?
**Options:** (A) attaching sugars to protein molecules; (B) building larger molecules from sugars; (C) creating simple sugars from smaller molecules; (D) breaking down complex carbohydrates into simple sugars.
**Ground Truth:** C

---

### Top Routed Models

**Rank 1: DeepSeek-R1-0528-Qwen3-8B** **Score:** 0.907
**Response:** "First, the question is: "Sugars are necessary for human cell function. Which of the following are human cells not capable of doing?" The options are: A) attaching sugars to protein molecules B) building larger molecules from sugars C) creating simple ... and lysosomes. While human cells can perform most of these functions, option C may be the least accurate or incomplete, as not all simple sugars can be created from smaller molecules by human cells due to metabolic limitations. Answer: C"
**Final Answer:** C (correct) ✓

**Rank 2: Qwen3-8B** **Score:** 0.877
**Response:** "Okay, let's tackle this question. The question is about what human cells can't do regarding sugars. The options are A to D. Let me go through each one. First, the question states that sugars are necessary for human cell function. ... smaller molecules. If the answer is C, then that's the correct choice. But I'm not sure. Alternatively, maybe the answer is C. I think I'll go with C as the answer. So the answer is C. Answer: C Answer: C"
**Final Answer:** C (correct) ✓

**Rank 3: NVIDIA-Nemotron-Nano-9B-v2** **Score:** 0.864
**Response:** "Okay, let's tackle this question. The question is asking which of the options human cells are not capable of doing, given that sugars are necessary for their function. The options are A to D. First, I need to recall what ... precursors are not "smaller molecules" in the context of sugar synthesis. - **D)** Breaking down complex carbohydrates into simple sugars (digestion) is a standard cellular activity. The process described in **C** is not something human cells can do. Answer: C"
**Final Answer:** C (correct) ✓

---

### Action Outcomes

**1. Direct Routing** Selected
**Final Answer:** C **Result:** Correct **# Calls:** 1
**Note:** Routed to the top-1 model (DeepSeek-R1-0528-Qwen3-8B).

**2. Self-Consistency**
**Final Answer:** D **Result:** Wrong **# Calls:** 3
**Note:** Repeated DeepSeek-R1-0528-Qwen3-8B three times; the majority answer shifted away from C.

**3. Heterogeneous Voting**
**Final Answer:** C **Result:** Correct **# Calls:** 3
**Note:** All three routed candidates supported the same answer C.

**4. Hybrid Resampling**
**Final Answer:** C **Result:** Correct **# Calls:** 3
**Note:** Retained the strongest routed candidates, and the extra call preserved the same correct answer.

### Takeaway

In this example, the routed top-1 model is already reliable, and the other high-ranked candidates do not introduce meaningful disagreement. Direct routing is therefore sufficient, while extra sampling can even destabilize the final prediction.

## Case 3: Self-Consistency

**Dataset:** KORBench
**Query:** Ciphertext: "WFSF#ZCHX#KLHI#PZZO#NIEL#" Please provide the decrypted answer, encapsulated in double square brackets. For example, the format should be: [[decrypted answer]].
**Ground Truth:** [[WZKPNIZLCFSHHZELOIXF]]

---

### Top Routed Models

**Rank 1: DeepSeek-R1-0528-Qwen3-8B** **Score:** 0.612
**Response:** "First, the ciphertext is 'WFSF#ZCHX#KLHI#PZZO#NIEL#', and the model starts tracing the decryption procedure step by step. It reconstructs a candidate plaintext and then ... reads the grid back out into a final string, ultimately producing 'WZKPNLIZCFSHHZELOZHS'. That output looks internally consistent, but it is still the wrong decryption. [[WZKPNLIZCFSHHZELOZHS]]"
**Final Answer:** [[WZKPNLIZCFSHHZELOZHS]] (wrong) ✗

**Rank 2: Intern-S1-mini** **Score:** 0.477
**Response:** "The model tries to reverse the encryption rules and repeatedly revisits how the rows should be reconstructed. It begins a plausible derivation, but ... the reasoning never stabilizes into a complete decoded string. Instead it loops over the same constraints and fails to produce a valid final answer. NULL"
**Final Answer:** NULL (wrong) ✗

**Rank 3: Qwen3-8B** **Score:** 0.460
**Response:** "The model walks through the decryption rule, reconstructs intermediate rows, and then reads them back into a candidate plaintext. It follows the intended procedure, but ... the recovered string is still inconsistent with the correct decoding. It stops with a shorter wrong output rather than the gold plaintext. [[WFSFXHCZKLHIOZZPNIEL]]"
**Final Answer:** [[IDPCPXSPLL]] (wrong) ✗

---

### Action Outcomes

**1. Direct Routing**
**Final Answer:** [[WZKPNLIZCFSHHZELOZHS]] **Result:** Wrong **# Calls:** 1
**Note:** Routed to the highest-scoring candidate, which produced a coherent but incorrect decryption.

**2. Self-Consistency** Selected
**Final Answer:** [[WZKPNLIZLCFSHHZELOIXF]] **Result:** Correct **# Calls:** 3
**Note:** Repeating the top-1 route repaired the decoding error and recovered the correct plaintext.

**3. Heterogeneous Voting**
**Final Answer:** [[WZKPNLIZCFSHHZELOZHS]] **Result:** Wrong **# Calls:** 3
**Note:** Cross-model voting did not recover the correct decryption and stayed with the routed error.

**4. Hybrid Resampling**
**Final Answer:** [[WZKPNLIZCFSHHZELOZHS]] **Result:** Wrong **# Calls:** 3
**Note:** Local resampling stayed near the same mistaken route and did not fix the plaintext.

### Takeaway

In this example, all three routed models initially fail, but repeated sampling of the top-1 model corrects its own decoding path. Neither voting nor local resampling recovers the right plaintext, making this a clean self-consistency case.

## Case 1: Heterogeneous Voting

**Dataset:** ARC Challenge
**Query:** A red giant star differs from main sequence stars because it **Options:** (A) burns at a cooler temperature; (B) is always larger than other suns; (C) has greater density and gravity; (D) fuses hydrogen to make helium.
**Ground Truth:** A

---

### Top Routed Models

**Rank 1: DeepSeek-R1-0528-Qwen3-8B** **Score:** 0.830
**Response:** "First, the question asks how a red giant differs from a main sequence star, with four astronomy options. The model compares size, density, gravity, and fusion behavior, but ... it finally decides that being larger is the key distinguishing feature. That pushes it to option B, even though the scientifically correct contrast is the cooler surface temperature of red giants. Answer: B"
**Final Answer:** B (wrong) ✗

**Rank 2: NVIDIA-Nemotron-Nano-9B-v2** **Score:** 0.801
**Response:** "The model recalls how red giants form after a star expands and cools at the surface. It rules out density and hydrogen fusion as distinguishing properties, then ... it explicitly notes that the cooler surface temperature is the clearest physical difference from a main sequence star. It therefore selects option A. Answer: A"
**Final Answer:** A (correct) ✓

**Rank 3: Qwen3-8B** **Score:** 0.800
**Response:** "The model compares the four options against basic astronomy facts. It recognizes that red giants expand and cool, so their outer layers are cooler than those of main sequence stars. It also notes that ... option B overgeneralizes size, since not every red giant is larger than every other star. On that basis it chooses option A. Answer: A"
**Final Answer:** A (correct) ✓

---

### Action Outcomes

**1. Direct Routing**
**Final Answer:** B **Result:** Wrong **# Calls:** 1
**Note:** Routed to the highest-scoring candidate, which overemphasized star size and chose B.

**2. Self-Consistency**
**Final Answer:** B **Result:** Wrong **# Calls:** 3
**Note:** Repeating the top-1 route kept the same wrong reasoning and did not leave B.

**3. Heterogeneous Voting** Selected
**Final Answer:** A **Result:** Correct **# Calls:** 3
**Note:** The second- and third-ranked models independently supported A, so cross-model voting overturned the wrong top-1 answer.

**4. Hybrid Resampling**
**Final Answer:** B **Result:** Wrong **# Calls:** 3
**Note:** Extra budget stayed anchored on the wrong primary route and failed to recover A.

### Takeaway

In this example, the routed top-1 model is confidently wrong, and neither repeating it nor locally resampling around it fixes the error. The correction appears only when different routed models are allowed to vote, making this a clean heterogeneous-voting case.

## Case 4: Hybrid Resampling

**Dataset:** KORBench
**Query:** X X X X, X X X X, X X X X, 1 X X 3. Constraints: (2,3)>(3,3), (2,4)>(3,4), (3,2)<(3,3). Please provide each element in order from left to right, and from top to bottom, with each row separated by a comma and wrapped in double square brackets.
**Ground Truth:** [[2 3 1 4,3 1 4 2,4 2 3 1,1 4 2 3]]

---

### Top Routed Models

**Rank 1: DeepSeek-R1-0528-Qwen3-8B** **Score:** 0.135
**Response:** "First, the grid is 4x4, so the puzzle must satisfy Latin-square constraints plus the given inequalities. The model fills the rows and columns, then ... it commits to a final grid that looks internally structured but violates the true solution. It outputs [[2 3 1 4,1 2 4 3,4 1 3 2,3 4 2 1]], which is still wrong. [[2 3 1 4,1 2 4 3,4 1 3 2,3 4 2 1]]"
**Final Answer:** [[2314,1243,4132,3421]] (wrong) ✗

**Rank 2: Qwen3-8B** **Score:** 0.131
**Response:** "The model reconstructs the 4x4 grid step by step, checks the row and column uniqueness constraints, and then verifies the inequalities one by one. In the end ... it explicitly confirms that all constraints are satisfied and returns the correct grid [[2 3 1 4,3 1 4 2,4 2 3 1,1 4 2 3]]. [[2 3 1 4,3 1 4 2,4 2 3 1,1 4 2 3]]"
**Final Answer:** [[2314,3142,4231,1423]] (correct) ✓

**Rank 3: Intern-S1-mini** **Score:** 0.102
**Response:** "The model starts from the same 4x4 puzzle constraints and tries to propagate the inequalities through the grid. It begins a plausible search, but ... the reasoning repeatedly loops back over the same constraint lines and never stabilizes into a valid full grid. The final extraction is incomplete and therefore wrong. [[?,4,2,3]]"
**Final Answer:** [[?,4,2,3]] (wrong) ✗

---

### Action Outcomes

**1. Direct Routing**
**Final Answer:** [[2314,1243,4132,3421]] **Result:** Wrong **# Calls:** 1
**Note:** Routed to the highest-scoring candidate, which produced the wrong grid.

**2. Self-Consistency**
**Final Answer:** [[2314,3142,4231,1423]] **Result:** Correct **# Calls:** 3
**Note:** Repeating the top-1 route repaired the initial grid and recovered the correct answer.

**3. Heterogeneous Voting**
**Final Answer:** [[2314,1243,4132,3421]] **Result:** Wrong **# Calls:** 3
**Note:** Cross-model voting still stayed with the routed error instead of converging on the correct grid.

**4. Hybrid Resampling** Selected
**Final Answer:** [[2314,3142,4231,1423]] **Result:** Correct **# Calls:** 3
**Note:** The extra budget stayed within the top routed neighborhood and repaired the initial routing error without expanding to a broader vote.

### Takeaway

In this example, routing is wrong and heterogeneous voting still fails, but a local budget-controlled repair succeeds. Hybrid resampling is not the only successful multi-response strategy here, yet it is the action actually selected by the policy because it fixes the local error while staying within the routed candidate set.

## Case 5: Failure Case

**Dataset:** ARC Challenge
**Query:** Which of these do all cells release? **Options:** (A) sugar; (B) water; (C) energy; (D) oxygen.
**Ground Truth:** C

### Top Routed Models

**Rank 1: DeepSeek-R1-0528-Qwen3-8B** **Score:** 0.923
**Response:** "First, the question is: "Which of these do all cells release?" The options are A) sugar, B) water, C) energy, D) oxygen. I need to think about what all cells have in common in terms of what they release. Cells ... Only certain cells, like those in photosynthetic organisms, release oxygen during photosynthesis, while animal cells consume it. Based on this analysis, energy is the substance released by all cells, as it is a universal byproduct of cellular metabolism. Answer: C"
**Final Answer:** C (correct) ✓

**Rank 2: Qwen3-8B** **Score:** 0.895
**Response:** "Okay, let's tackle this question. The question is asking which of the options—sugar, water, energy, or oxygen—is something that all cells release. The options are A through D, and I need to pick the correct one. First, I should recall ... all cells are involved in water exchange, and water is a byproduct of cellular respiration. Even if some cells don't do cellular respiration, they might still release water through other processes. So the answer is B. Answer: B. Answer: B"
**Final Answer:** B (wrong) ✗

**Rank 3: GLM-Z1-9B-0414** **Score:** 0.875
**Response:** "Okay, let's tackle this question. The question is asking which of the options A to D all cells release. The options are sugar, water, energy, and oxygen. First, I need to recall what each of these substances does in a ... organisms, and sugar is not universally produced (e.g., animal cells consume it), water is consistently generated in various cellular activities. Energy (ATP) is stored rather than released as a waste product. Thus, the correct answer is B) water. Answer: B"
**Final Answer:** B (wrong) ✗

### Action Outcomes

**1. Direct Routing**
**Final Answer:** C **Result:** Correct **# Calls:** 1
**Note:** Routed to the top-1 model (DeepSeek-R1-0528-Qwen3-8B).

**2. Self-Consistency**
**Final Answer:** C **Result:** Correct **# Calls:** 3
**Note:** Repeated DeepSeek-R1-0528-Qwen3-8B three times; the majority answer remained C.

**3. Heterogeneous Voting** Selected
**Final Answer:** B **Result:** Wrong **# Calls:** 3
**Note:** The two lower-ranked candidates agreed on B and overruled the already-correct top-1 answer.

**4. Hybrid Resampling**
**Final Answer:** C **Result:** Correct **# Calls:** 3
**Note:** Retained the strongest routed candidates and avoided being dominated by the incorrect lower-ranked majority.

### Takeaway

Multi-response inference is not always a safe improvement. In this failure case, the selector over-triggers heterogeneous voting, and the incorrect majority from lower-ranked models overrides an already correct routed answer.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the contributions and scope of this paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We include the full set of assumptions and complete proofs for the theoretical statements in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the datasets, model pool, data splits, baselines, evaluation protocol, hyperparameters, and random seeds needed to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide a public repository in the abstract containing the code and data access instructions for the main experimental results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper specifies the experimental setup in the main text and appendix, including datasets, data splits, model pool, baselines, evaluation protocol, hyperparameters, and inference settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The main results are averaged over seven random seeds, and the appendix reports seed-level standard deviations for the main ID and OOD comparisons.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details are provided in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses public benchmarks and open-source models, and we have reviewed and followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts are discussed in Appendix G.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new high-risk models, scraped datasets, or other assets requiring special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly credit the creators and original owners of the assets used in the paper through citations and other appropriate means, and respect their licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper describes the use of LLMs as the candidate model pool for routing and aggregation experiments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.