PROBE BEFORE YOU ROUTE: LLM SELECTION VIA TEST-TIME PROBING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) vary substantially in capability and inference cost, making it important to select an appropriate model for each query. Conventional LLM routers select a model before any candidate model generation, potentially lowering routing performance because they overlook informative signals in the generated tokens. We therefore propose **Probe-then-Route**, which first probes each candidate LLM by allowing it to generate a small number of output tokens for the query and then selects one model based on the query together with the generated tokens. To reduce the additional inference cost of probing every candidate, we further introduce a variant, dubbed **Selective Probe-then-Route**, that first selects a candidate subset, probes only those candidates, and routes among them using the same trained router without retraining. We evaluate both methods using eight heterogeneous LLMs on 12 benchmarks spanning six domains, with one in-distribution (ID) and one out-of-distribution (OOD) benchmark per domain. Probing all eight candidates with up to eight output tokens per candidate improves average ID accuracy by 2.73% over the strongest baseline and outperforms all baselines on three of six ID benchmarks. **Probe-then-Route** also generalizes effectively out of distribution, achieving the highest accuracy on two of six OOD benchmarks, including one tie, and remaining competitive on the others. With **Selective Probe-then-Route**, additional inference cost can be reduced by 56.4% on ID benchmarks while retaining 83.3% of **Probe-then-Route**'s accuracy gain over the strongest baseline and still outperforming all baselines across three ID benchmarks.

1 INTRODUCTION

LLMs differ in capability and inference cost (Wang et al., 2025). Their relative performance varies across queries: a model that performs well on one query may be outperformed by another model on a different query (Lu et al., 2024). Moreover, multiple models may achieve comparable response quality on the same query while incurring different inference costs (Ong et al., 2025). Consequently, no single model is universally suitable for all queries when considering both response quality and inference cost (Hu et al., 2024). This motivates LLM routing, which selects an appropriate model for each query based on differences in model performance and cost (Wang et al., 2025).

Conventional LLM routers route queries before any candidate model generates tokens (Huang et al., 2025). This potentially lowers routing performance, as it overlooks informative signals in the generated tokens. Huang et al. (2025) provide empirical evidence for this limitation: a router given both the query and complete candidate responses substantially outperforms a router given only the query. However, generating complete responses from every candidate would incur substantial additional inference cost. This raises a natural question: *Can we make better routing decisions without generating complete candidate responses?*

To investigate this question, we propose **Probe-then-Route**. Given a query, we first probe each candidate LLM by allowing it to generate up to k output tokens, stopping early if its response is complete. The router then selects one model using the query together with the output tokens generated by all candidates. The selected model continues its response if unfinished. This allows the router to use observed response content without requiring complete responses from every candidate.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

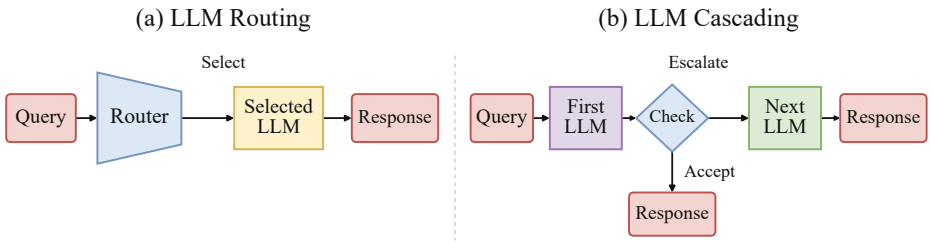

Figure 1: LLM routing and LLM cascading. (a) A router selects one candidate model before response generation. (b) A cascade evaluates a generated response and either accepts it or invokes the next model.

Even with a small k, probing every candidate incurs additional inference cost. To reduce this cost while preserving final-response accuracy, we extend Probe-then-Route with candidate selection before probing. This variant, **Selective Probe-then-Route**, uses query-based scores to select a candidate subset, probes only those candidates, and routes among them using the same trained router without retraining.

We evaluate both methods using eight heterogeneous LLMs on 12 benchmarks spanning six domains, with one in-distribution (ID) and one out-of-distribution (OOD) benchmark per domain. We study 13 values of k, doubling from $k = 1$ to $k = 4096$, and compare five alternative candidate-selection strategies for Selective Probe-then-Route. With $k = 8$, Probe-then-Route improves average ID accuracy by 2.73% over the strongest baseline. It outperforms all baselines on three of six ID benchmarks and achieves or ties the highest accuracy on two of six OOD benchmarks. On ID benchmarks, Selective Probe-then-Route reduces additional inference cost by 56.4% relative to Probe-then-Route while retaining 83.3% of its accuracy gain over the strongest baseline. It also outperforms all baselines on the same three ID benchmarks.

Our main contributions are as follows:

- We introduce **Probe-then-Route**, which uses the query and up to k output tokens from each candidate LLM to select the model that produces the final response. We further introduce **Selective Probe-then-Route**, a variant that restricts probing and final selection to a candidate subset while reusing the same trained router.
- We demonstrate that Probe-then-Route improves ID accuracy and generalizes effectively to OOD benchmarks. Selective Probe-then-Route retains most of the ID accuracy improvement while substantially reducing additional inference cost.
- We analyze the effects of k and candidate-selection strategies on accuracy and additional inference cost, and use component ablations to assess the predictor architecture.

2 Related Work

LLM Routing LLM routing selects a model for each query based on response quality and inference cost (Moslem & Kelleher, 2026). For binary routing, Hybrid LLM uses predicted query difficulty (Ding et al., 2024), while RouteLLM learns from preference data (Ong et al., 2025). For multi-model routing, RouterDC uses query–model representation similarity (Chen et al., 2024b), while Lookahead predicts candidate-response representations (Huang et al., 2025). These methods select a model before candidate response generation (Figure 1(a)). Our approach uses tokens generated by candidates for the query as additional routing signals.

LLM Cascading LLM cascading invokes models sequentially until a response is judged sufficiently good (Figure 1(b)). Like routing, it determines which model provides the final response, but makes this decision progressively using responses from models already invoked. FrugalGPT uses a learned quality scorer and stopping thresholds to decide when to stop (Chen et al., 2024a), while AutoMix uses self-verification confidence to decide whether to accept a response or invoke a larger model (Aggarwal et al., 2024).

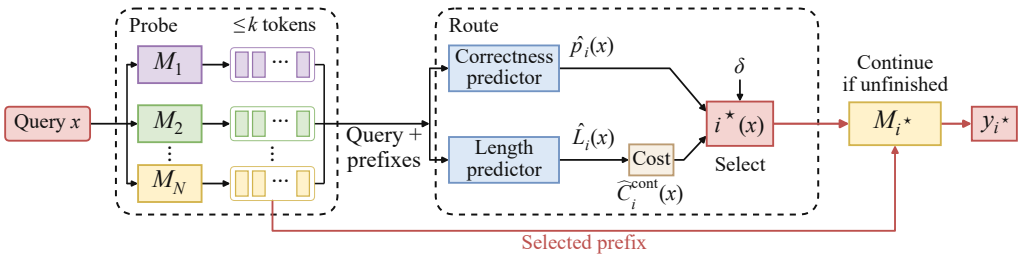

Figure 2: Probe-then-Route pipeline. Each candidate generates up to k tokens before routing. The illustrated selection is M_N, whose response is continued if unfinished.

3 Methodology

3.1 Problem Formulation

Let $\mathcal{X}$ denote the query space and $\mathcal{M} = \{M_1, \ldots, M_N\}$ a set of N candidate LLMs. Given a query $x \in \mathcal{X}$, each candidate model M_i can produce a response $y_i = M_i(x)$ with inference cost $c_i(x)$. The goal of LLM routing is to select a candidate model for each query, prioritizing response quality and favoring lower inference cost when response quality is similar.

3.2 Probe-then-Route

Probe-then-Route selects a candidate model using the query and response tokens generated by the candidate models. Figure 2 provides an overview of the routing pipeline. The method uses two predictors: a correctness predictor that estimates the probability that each candidate will produce a correct response, and a length predictor that estimates the total number of output tokens in that response. These predictions allow the router to prioritize response correctness and consider inference cost when candidates have similar predicted correctness. Figure 3 shows the architecture of both predictors.

Probing candidate models. Given a query x, we first allow each candidate model to generate up to k output tokens, stopping earlier if its response is complete. The query and the tokens generated by all candidates are then provided to both predictors. The value of k controls how much of each candidate response is available before model selection. A larger k potentially provides more response content but also increases probing cost.

Model selection and response continuation. Let $\hat{p}_i(x)$ denote candidate i's predicted probability of producing a correct response, and let $\widehat{C}_i^{\text{cont}}(x)$ denote its estimated continuation cost after probing. The router selects

$$i^\star(x) = \underset{i \in \{1,\ldots,N\}}{\arg\min}\ \widehat{C}_i^{\text{cont}}(x) \quad \text{subject to} \quad \hat{p}_i(x) \geq \max_{1 \leq j \leq N} \hat{p}_j(x) - \delta, \tag{1}$$

where $\delta \geq 0$ is a tolerance on predicted correctness. Predicted correctness determines which models are eligible, and estimated continuation cost determines the final choice among them. When $\delta = 0$, only candidates with the highest predicted correctness probability are eligible.

For each candidate, the length predictor estimates the total number of output tokens in its complete response. We subtract the number of tokens already generated from this estimate to obtain the estimated remaining token count, setting negative values to zero. For an unfinished candidate, continuation cost includes the input cost of providing the query and the already generated tokens, together with the output cost of generating the estimated remaining tokens. These costs use each model's input and output token prices. A candidate that has already completed its response has zero continuation cost. Since probing costs have already been incurred for all candidates, model selection

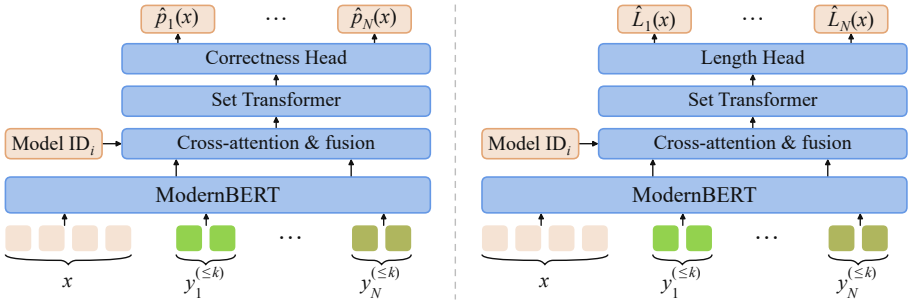

Figure 3: Architecture of the correctness predictor (left) and length predictor (right). Each predictor uses the query x, candidate response prefixes of up to k tokens, and model-ID embeddings to estimate correctness probabilities $\hat{p}_i(x)$ or total response lengths $\hat{L}_i(x)$.

depends only on their estimated continuation costs among candidates with similar predicted correctness. After selection, the chosen model continues from its generated tokens unless its response is already complete. Only its response is returned as the final response.

Predictor architecture. Both predictors use a frozen ModernBERT encoder (Warner et al., 2025) to encode the query and each candidate's generated tokens separately. We choose ModernBERT over BERT (Devlin et al., 2019) because its longer context window accommodates generated tokens at larger values of k. The two predictors share encoder representations but have separate trainable parameters. Within each predictor, cross-attention uses the generated-token representations as attention queries and the query representations as keys and values. A residual connection and layer normalization combine the attention output with the original generated-token representations to produce query-conditioned token representations.

Attention pooling summarizes the query and each candidate's conditioned token representations into fixed-length vectors. For each candidate, we concatenate its pooled token vector, the pooled query vector, and a learnable model-ID embedding. A multilayer perceptron (MLP) transforms this concatenated vector into a candidate representation. The model-ID embedding distinguishes candidates beyond their generated content. A Set Transformer (Lee et al., 2019) jointly processes the candidate representations, allowing each to incorporate information from the others. No positional embeddings are added across candidates. Finally, a prediction head produces one output for each candidate.

Correctness prediction. The correctness predictor estimates the probability that each candidate's complete response is correct, using the query and the tokens generated by all candidates during probing. Its output scores are divided by a temperature parameter and passed through a sigmoid function to obtain correctness. We train the correctness predictor using binary cross-entropy, with the correctness of each candidate's complete response as the training label.

Length prediction. The length predictor outputs $\hat{L}_i(x)$, an estimate of the total number of output tokens in candidate i's complete response. We predict response length rather than inference cost directly so that changes in model pricing can be incorporated into the cost calculation without retraining the predictor. We standardize the training targets using the mean and standard deviation of response lengths for each candidate model and train the predictor with mean squared error. We compute the loss only for responses unfinished after probing, since completed responses have known lengths.

3.3 Selective Probe-then-Route

Selective Probe-then-Route first selects a subset of candidate models, aiming to preserve final-response accuracy while reducing additional inference cost. Only the retained candidates are probed, and routing is performed among them. Figure 4 provides an overview of the pipeline.

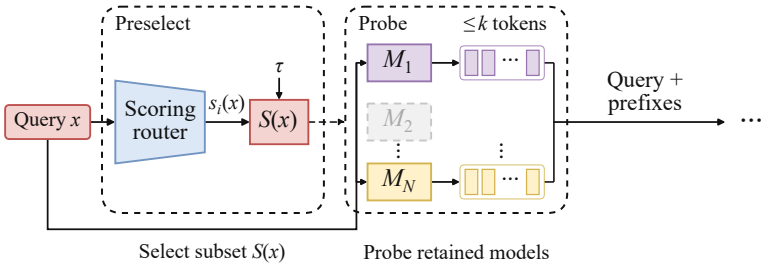

Figure 4: Selective Probe-then-Route with adaptive score-gap selection. A scoring router selects a candidate subset using threshold τ. Only retained candidates are probed and considered for final model selection. The selected response is continued if unfinished.

We use **adaptive score-gap selection** to determine this subset. A router scores all candidate models without requiring them to generate tokens, and we retain candidates whose scores are sufficiently close to the highest score. Let $s_i(x)$ denote the score of candidate i for query x, and let $s_{\max}(x)$ and $s_{\min}(x)$ denote the highest and lowest candidate scores, respectively. When $s_{\max}(x) > s_{\min}(x)$, we retain the subset

$$S(x) = \left\{ i : \frac{s_{\max}(x) - s_i(x)}{s_{\max}(x) - s_{\min}(x)} \leq \tau \right\},$$

where $\tau \in [0, 1]$ is the threshold on the normalized score gap. The highest-scoring candidate is always retained, and all candidates are retained when their scores are equal. Both the size and membership of the subset can vary across queries.

We reuse the trained correctness and length predictors without retraining. For candidates excluded from the subset, we replace the generated-token inputs with empty text while retaining their query and model-ID inputs. Final model selection is restricted to the retained candidates and uses the same decision rule as Probe-then-Route, based on predicted correctness and estimated continuation costs. The selected model then continues its response unless it has already finished. We describe and compare five alternative candidate-selection strategies in Appendix A.1.

4 Experiments

4.1 Experimental Setup

Models and data. We evaluate eight LLMs from four families: Qwen3 (Yang et al., 2025), Llama 3.1 (Grattafiori et al., 2024), Ministral 3 (Mistral AI, 2025), and Gemma 3 (Gemma Team, 2025). Model sizes range from 8B to 70B. The 12 benchmarks cover six domains, each with one ID and one OOD benchmark. We evaluate on the corresponding ID test partitions and on complete test splits or held-out subsets of the six OOD benchmarks. Model access, source splits, partitioning procedures, and evaluation-set sizes are provided in Appendix A.2.1 and A.2.2.

Baselines. We compare with all eight individual LLMs, random selection, EmbedLLM (Zhuang et al., 2025), MIRT (Song et al., 2025), GraphRouter (Feng et al., 2025), RadialRouter (Jin et al., 2025), CSCR (Shirkavand et al., 2025), Lookahead (Huang et al., 2025), and a cascading method FrugalGPT (Chen et al., 2024a). For GraphRouter and RadialRouter, we report three variants: balance (bal.), cost-first (cost), and performance-first (perf.). Random selection assigns equal probability to each candidate, and we report its exact expected metric values. We additionally report an oracle as an upper-bound reference on accuracy within the candidate pool. For each query, it selects the candidate with the lowest full-response inference cost among those with correct responses. If all responses are incorrect, it selects the lowest-cost candidate. Baseline descriptions are provided in Appendix A.2.3.

Evaluation metrics. We report accuracy for each benchmark and its unweighted average (Avg. acc.) across the six ID and six OOD benchmarks separately. Average selected-model cost (Sel.) measures the cost of generating the selected model's complete response alone, including input and

Table 1: ID results for our methods ($k = 8$), baselines, and an oracle upper bound. Costs are USD per 1,000 queries.

Method	MMLU	GPQA	MATH	MBPP	Logic Bench	Truthful QA	Avg. acc. ↑	Sel. cost ↓	Add. cost ↓	OSR ↑
Qwen3-8B	78.50	48.35	86.69	66.00	77.67	66.46	70.61	0.178	0.000	10.32
Llama-3.1-8B-Instruct	65.21	26.37	47.49	52.80	64.41	63.92	53.37	0.200	0.000	12.81
Ministral-3-8B-Instruct	77.13	41.76	90.69	64.20	79.92	67.09	70.13	0.156	0.000	22.50
Gemma-3-12B-IT	75.58	37.36	85.13	62.80	73.21	71.52	67.60	0.110	0.000	**37.77**
Qwen3-14B	81.45	51.65	88.87	**67.80**	77.94	69.62	72.89	0.355	0.000	1.61
Llama-3.1-70B-Instruct	81.59	50.55	64.98	46.60	71.12	68.99	63.97	0.510	0.000	1.61
Ministral-3-14B-Instruct	80.29	49.45	92.13	64.40	80.75	63.92	71.82	0.212	0.000	5.16
Gemma-3-27B-IT	79.94	45.05	88.78	64.60	76.35	73.42	71.36	0.151	0.000	8.29
Probe-then-Route	83.77	**62.64**	91.11	66.00	**82.78**	72.15	**76.41**	0.243	0.207	7.05
Selective Probe-then-Route	**83.80**	60.44	91.56	66.60	82.51	71.52	76.07	0.252	0.090	5.41
Random router	77.46	43.82	80.59	61.15	75.17	68.12	67.72	0.234	0.000	12.51
CSCR	79.76	45.05	88.71	64.60	76.40	**74.05**	71.43	0.130	0.000	10.02
EmbedLLM	83.59	47.25	90.09	66.40	79.70	72.78	73.30	0.255	0.000	6.92
GraphRouter (bal.)	79.23	53.85	84.04	**67.80**	77.67	69.62	72.04	0.244	0.000	3.01
GraphRouter (cost)	81.45	51.65	88.84	**67.80**	77.94	69.62	72.88	0.342	0.000	0.92
GraphRouter (perf.)	80.18	52.75	91.84	64.60	80.75	65.19	72.55	0.225	0.000	3.94
Lookahead	83.73	54.95	91.09	64.00	79.76	72.78	74.38	0.266	0.000	3.82
MIRT	80.08	49.45	91.71	64.40	80.64	66.46	72.12	0.209	0.000	8.56
RadialRouter (bal.)	81.52	51.65	91.40	**67.80**	77.94	69.62	73.32	0.289	0.000	1.51
RadialRouter (cost)	80.89	50.55	88.80	67.60	78.16	73.42	73.24	0.197	0.000	7.78
RadialRouter (perf.)	81.03	52.75	90.42	67.60	78.27	73.42	73.91	0.233	0.000	6.59
FrugalGPT	82.47	50.55	**92.16**	66.40	82.07	72.15	74.30	0.215	0.107	–
Oracle	95.68	82.42	97.53	80.40	94.99	93.67	90.78	0.092	0.000	100.00

output tokens. Average additional cost (Add.) measures the extra inference cost incurred relative to generating that response alone. For FrugalGPT, selected-model cost is the cost of generating the returned response, and additional cost is the sum of the generation costs of all other responses produced in the cascade. Total inference cost is the sum of selected-model and additional costs. Costs are averaged over queries and reported in USD per 1,000 queries. Following RouterArena (Lu et al., 2026), optimal selection ratio (OSR) measures how often the router selects the cheapest correct candidate among queries with at least one correct candidate. Metric details are provided in Appendix A.2.4.

Implementation details. We train the correctness and length predictors for 40 epochs using AdamW (Loshchilov & Hutter, 2019) and select checkpoints according to average ID validation accuracy. Detailed training settings and hardware information are provided in Appendix A.2.

4.2 Main Results

Tables 1 and 2 report results on the ID and OOD benchmarks, respectively. Probe-then-Route and Selective Probe-then-Route both use $k = 8$, a temperature of 12.9039, and $\delta = 0.001$. The selective results use adaptive score-gap selection with Lookahead and $\tau = 0.25$. Hyperparameter search

Table 2: OOD results for our methods ($k = 8$), baselines, and an oracle upper bound. Costs are USD per 1,000 queries.

Method	MMLU-CF	ARC-C	Math Bench-A	Human Eval	LogiQA-en	Simple QA	Avg. acc. ↑	Sel. cost ↓	Add. cost ↓	OSR ↑
Qwen3-8B	66.60	93.34	88.33	85.37	63.90	3.82	66.89	0.138	0.000	3.93
Llama-3.1-8B-Instruct	56.80	83.53	44.67	58.54	40.09	1.73	47.56	0.163	0.000	7.84
Ministral-3-8B-Instruct	67.75	93.26	91.33	85.98	60.06	5.20	67.26	0.120	0.000	19.60
Gemma-3-12B-IT	63.50	92.58	81.67	83.54	60.68	4.97	64.49	0.069	0.000	**47.60**
Qwen3-14B	**70.80**	93.86	**91.67**	86.59	68.05	4.62	69.26	0.276	0.000	1.30
Llama-3.1-70B-Instruct	66.20	89.08	70.33	76.83	52.07	**11.10**	60.94	0.360	0.000	4.89
Ministral-3-14B-Instruct	68.90	94.88	89.67	87.80	64.52	6.71	68.75	0.153	0.000	5.46
Gemma-3-27B-IT	67.65	93.86	87.33	81.71	61.75	8.67	66.83	0.088	0.000	9.41
Probe-then-Route	**70.80**	**94.97**	91.33	86.59	64.06	6.47	69.04	0.157	0.188	4.72
Selective Probe-then-Route	70.25	94.37	91.00	87.20	65.75	6.47	69.17	0.144	0.083	5.35
Random router	66.03	91.80	80.63	80.79	58.89	5.85	64.00	0.171	0.000	12.51
CSCR	67.10	93.43	87.33	81.71	61.75	7.28	66.43	0.072	0.000	5.95
EmbedLLM	69.10	94.37	89.67	87.80	59.75	4.74	67.57	0.154	0.000	6.38
GraphRouter (bal.)	69.45	93.77	**91.67**	86.59	64.52	4.62	68.44	0.167	0.000	3.54
GraphRouter (cost)	70.70	93.86	**91.67**	86.59	68.05	4.62	69.25	0.206	0.000	0.99
GraphRouter (perf.)	69.10	94.71	90.33	**88.41**	64.52	6.01	68.85	0.111	0.000	6.24
Lookahead	70.00	94.28	91.00	87.80	65.28	7.86	69.37	0.151	0.000	4.74
MIRT	68.55	94.45	89.67	87.80	64.52	6.01	68.50	0.110	0.000	8.40
RadialRouter (bal.)	**70.80**	93.86	90.67	87.80	**68.20**	4.74	69.35	0.190	0.000	1.35
RadialRouter (cost)	70.70	93.60	88.33	82.32	66.82	8.67	68.41	0.170	0.000	3.25
RadialRouter (perf.)	70.65	93.69	**91.67**	85.37	67.13	8.67	**69.53**	0.172	0.000	3.59
FrugalGPT	69.85	94.88	90.00	85.98	64.21	7.63	68.76	0.098	0.094	–
Oracle	87.40	98.89	97.67	96.95	90.17	23.93	82.50	0.051	0.000	100.00

details are provided in Appendix A.2.5. We examine the effects of k and candidate-selection strategy in the following sections.

In-distribution accuracy. As shown in Table 1, Probe-then-Route achieves the highest average accuracy of 76.41%, exceeding the strongest baseline, Lookahead, by 2.03 percentage points and the strongest individual LLM, Qwen3-14B, by 3.52 percentage points. Both Probe-then-Route and Selective Probe-then-Route outperform all individual LLMs and baselines on MMLU (Hendrycks et al., 2021a), GPQA (Rein et al., 2024), and LogicBench (Parmar et al., 2024). The largest improvement occurs on GPQA, where Probe-then-Route reaches 62.64% accuracy, exceeding the strongest baseline by 7.69 percentage points. Selective Probe-then-Route achieves an average accuracy of 76.07%, which also exceeds all baselines. These results show that initially generated tokens can help the router select models that produce correct responses. However, the gains vary across benchmarks: other methods or individual LLMs achieve higher accuracy on MATH (Hendrycks et al., 2021b), MBPP (Austin et al., 2021), and TruthfulQA (Lin et al., 2022).

Out-of-distribution accuracy. Table 2 shows that Probe-then-Route achieves an average OOD accuracy of 69.04%. This is 0.49 percentage points below the strongest routing baseline, RadialRouter in its performance-first setting, and 0.22 percentage points below the strongest individual

LLM, Qwen3-14B. Probe-then-Route achieves the highest accuracy on ARC-Challenge (Clark et al., 2018) and ties for the highest accuracy on MMLU-CF (Zhao et al., 2025). However, several baselines perform better on HumanEval (Chen et al., 2021), LogiQA-en (Liu et al., 2020) and SimpleQA (Wei et al., 2024). Selective Probe-then-Route achieves an average OOD accuracy of 69.17%, exceeding Probe-then-Route by 0.13 percentage points and reducing the gap to the strongest routing baseline to 0.36 percentage points. Overall, both proposed methods remain competitive on OOD benchmarks, but neither surpasses the strongest baseline in average OOD accuracy, unlike their results on ID benchmarks.

Inference cost and optimal selection. Relative to Probe-then-Route, Selective Probe-then-Route reduces additional inference cost by more than half on both ID and OOD benchmarks. On ID benchmarks, average additional cost decreases from \$0.207 to \$0.090 per 1,000 queries. Average accuracy decreases by 0.34 percentage points, retaining approximately 83.3% of Probe-then-Route's accuracy gain over the strongest routing baseline and remaining higher than all routing baselines. On OOD benchmarks, average additional cost decreases from \$0.188 to \$0.083 per 1,000 queries, while average accuracy increases by 0.13 percentage points. These changes correspond to total inference cost reductions of approximately 24.0% and 34.2% on ID and OOD benchmarks, respectively. Compared with FrugalGPT, Selective Probe-then-Route also achieves higher average accuracy and lower additional inference cost on both ID and OOD benchmarks, although its total inference cost is higher.

The effect on OSR is mixed: Selective Probe-then-Route decreases OSR from 7.05% to 5.41% on ID benchmarks and increases it from 4.72% to 5.35% on OOD benchmarks. Thus, the reported selective configuration reduces additional inference cost while retaining most of the ID accuracy gain and slightly improving OOD accuracy, but does not consistently improve selection of the cheapest correct candidate.

4.3 Effect of k

We examine how k affects the accuracy and additional inference cost of Probe-then-Route. We evaluate 13 values of k, doubling from $k = 1$ to $k = 4096$.

Accuracy. Figure 5 shows average accuracy for each value of k. On ID benchmarks, Probe-then-Route achieves its highest average accuracy of 76.41% at $k = 8$, followed by $k = 2$. Both settings outperform all evaluated settings with $k > 8$. To examine whether answers are already visible at small values of k, we analyze answer exposure in Appendix A.3. Accuracy fluctuates at larger values of k, with no consistent improvement as more tokens are observed. Accuracy fluctuates at larger values of k, with no consistent improvement as more tokens are observed. Nevertheless, average ID accuracy exceeds that of the strongest routing baseline at most evaluated values of k. On OOD benchmarks, the highest average accuracy occurs at $k = 1$, while $k = 8$ achieves 69.04%. Longer probes do not provide consistent improvements, and all evaluated settings remain below the strongest routing baseline in average OOD accuracy.

Additional inference cost. Figure 6 shows that average additional cost increases with k in both settings. Even at small values of k, additional inference cost remains substantial. At $k = 8$, it is \$0.207 per 1,000 queries on ID benchmarks and \$0.188 on OOD benchmarks. At $k = 4096$, these costs rise to approximately \$1.45 and \$0.78, respectively. Together, these results support using small values of k to obtain information from candidate responses while limiting additional inference cost.

4.4 Selective Probe-then-Route

We examine how the scoring router and cutoff τ affect Selective Probe-then-Route. We use EmbedLLM, MIRT, Lookahead, and three variants each of GraphRouter and RadialRouter, with six cutoffs: $\tau \in \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25\}$. We omit CSCR because its published procedure applies cost-aware scoring to four retrieved candidates (Shirkavand et al., 2025), whereas this comparison requires scores for all eight candidates.

For each value of k, we select the scoring router and cutoff with the highest average ID validation accuracy, breaking ties by lower average total inference cost. We then evaluate the selected

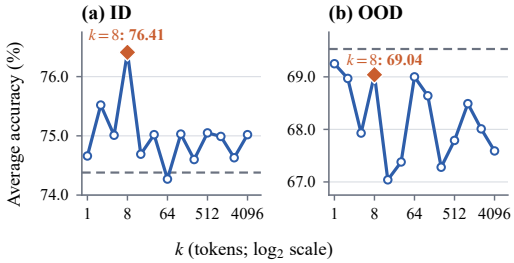

Figure 5: Average accuracy across six ID and six OOD benchmarks as a function of k. Dashed lines denote the strongest baseline.

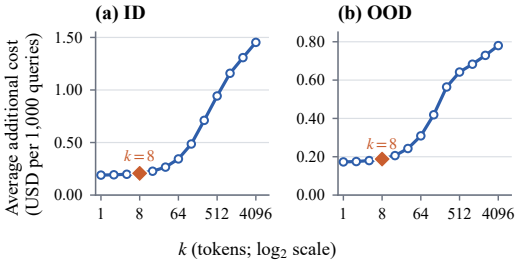

Figure 6: Average additional inference cost across six ID and six OOD benchmarks as a function of k.

configuration on both ID and OOD test sets. At $k = 8$, this procedure selects Lookahead with $\tau = 0.25$, achieving 76.07% average ID accuracy and 69.17% average OOD accuracy, with average additional costs of \$0.090 and \$0.083 per 1,000 queries, respectively. Table 3 reports the selected configurations and their test results across all 13 values of k. Comparisons with five alternative candidate-selection strategies are provided in Appendix A.1, and the complete validation sweep figures are provided in the supplementary material.

4.5 Ablation Studies

We examine the effects of generated-token inputs, model-ID embeddings, cross-attention, the Set Transformer, encoder choice, dropout, and balanced training. At $k = 8$, Probe-then-Route achieves 76.41% average ID accuracy, compared with 73.63% for the no-prefix variant. Removing model-ID embeddings lowers average ID accuracy at all 13 values of k. The benefit of cross-attention depends on k: removing it lowers ID accuracy at $k = 8$ but improves ID accuracy at every tested $k \geq 256$ and OOD accuracy at most of these values. Increasing dropout or the number of training entries per benchmark under balanced training does not consistently improve accuracy. Detailed settings and results for all ablations are provided in Appendix A.4.

5 Discussion

Several limitations remain. Both methods incur additional inference cost, even at small values of k. Accuracy gains are less consistent on OOD benchmarks, and configurations with higher ID accuracy do not necessarily achieve higher OOD accuracy. Finally, Selective Probe-then-Route reuses the trained router without jointly optimizing candidate selection and routing. Whether joint training can further improve the accuracy–cost trade-off remains an open question.

6 Conclusion

We introduced Probe-then-Route, which first probes each candidate LLM by allowing it to generate a small number of output tokens for the query and then selects one model based on the query together with the generated tokens. We further proposed Selective Probe-then-Route, which probes only a selected subset of candidates and selects among them without requiring router retraining. Experiments with eight candidate LLMs on 12 benchmarks show that small values of k can improve final-response accuracy. Increasing k does not consistently provide further accuracy gains. The reported selective configuration reduces additional inference cost by more than half on both ID and OOD benchmarks, while retaining most of the ID accuracy gain and slightly improving OOD accuracy relative to Probe-then-Route. These results highlight the importance of choosing k and the candidate subset when balancing final-response accuracy and additional inference cost.

AI USE STATEMENT

We used generative AI tools to assist with writing and polishing the manuscript, as well as literature retrieval and the discovery of related work. Other tasks requiring disclosure either did not involve generative AI tools or were not applicable to this work. We reviewed all AI-assisted work and take responsibility for the final content of this paper, including its text, claims, and references.

ETHICS STATEMENT

Our methods may inherit biases and harmful behaviors from the candidate LLMs and biases in the training data. Our experiments evaluate accuracy and inference cost but do not assess response safety or fairness. Deployment in sensitive applications would require additional evaluation of these aspects.

REPRODUCIBILITY STATEMENT

The main text describes Probe-then-Route and its selective variant. Benchmark split sizes are reported in Table 12, and baseline descriptions are provided in Appendix A.2.3. Training settings, hyperparameter search ranges, and selected inference configurations are provided in Appendix A.2.5. Candidate-selection strategies and ablation settings are detailed in Appendices A.1 and A.4, respectively. Supplementary material includes sweep plots and notebooks for both methods and the ablation studies.[1]

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

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

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

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.

[1] Anonymous supplementary material.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks Lakshmanan, and Ahmed H. Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, pp. 41348–41366, 2024.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A graph-based router for LLM selections. In *International Conference on Learning Representations*, 2025.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Daniela Gerz, Pei-Hao Su, Razvan Kusztos, Avishek Mondal, Michał Lis, Eshan Singhal, Nikola Mrkšić, Tsung-Hsien Wen, and Ivan Vulić. Multilingual and cross-lingual intent detection from spoken data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7468–7475. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.591.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Canbin Huang, Tianyuan Shi, Yuhua Zhu, Ruijun Chen, and Xiaojun Quan. Lookahead routing for large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

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: EMNLP*, 2025.

Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002. doi: 10.1162/106365602760234108.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.229.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 3622–3628. International Joint Conferences on Artificial Intelligence Organization, 2020. doi: 10.24963/ijcai.2020/501.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of NAACL-HLT*, pp. 1964–1974. Association for Computational Linguistics, 2024.

Yifan Lu, Rixin Liu, Jiayi Yuan, Xingqi Cui, Shenrun Zhang, Hongyi Liu, and Jiarong Xing. RouterArena: An open platform for comprehensive comparison of LLM routers. In *International Conference on Learning Representations*, 2026.

Mistral AI. Introducing Mistral 3. https://mistral.ai/news/mistral-3/, December 2025.

Yasmin Moslem and John D. Kelleher. Dynamic model routing and cascading for efficient LLM inference: A survey. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. Survey Certification.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14:265–294, 1978. doi: 10.1007/BF01588971.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, pp. 34433–34448, 2025.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. LogicBench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13679–13707. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.739.

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.

Reza Shirkavand, Shangqian Gao, Peiran Yu, and Heng Huang. Cost-aware contrastive routing for LLMs. In *Advances in Neural Information Processing Systems*, 2025.

Parshin Shojaee, Sai Sree Harsha, Dan Luo, Akash Maharaj, Tong Yu, and Yunyao Li. Federated retrieval augmented generation for multi-product question answering. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pp. 387–397, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. IRT-Router: Effective and interpretable multi-LLM routing via item response theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 10912–10922, 2025. doi: 10.18653/v1/2025.naacl-long.545.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

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.

Qihao Zhao, Yangyu Huang, Tengchao Lv, Lei Cui, Qinzheng Sun, Shaoguang Mao, Xin Zhang, Ying Xin, Qiufeng Yin, Scarlett Li, and Furu Wei. MMLU-CF: A contamination-free multi-task language understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13371–13391. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.656.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning compact representations of large language models. In *International Conference on Learning Representations*, 2025.

A Appendix

A.1 Candidate-Selection Strategies

We compare adaptive score-gap selection with five alternative candidate-selection strategies for Selective Probe-then-Route. All strategies reuse the trained correctness and length predictors without retraining and differ in how they determine which candidates to probe.

For each strategy and value of k, we select the configuration with the highest average ID validation accuracy, breaking ties by lower average total inference cost. For pruning strategies, we exclude configurations that remove no candidates on the ID validation set. We fix the selected hyperparameters before evaluating on the ID and OOD test sets. OSR is computed relative to the original pool of eight candidates. Complete validation sweep figures are provided in the supplementary material.

A.1.1 Adaptive Score-Gap Selection

We use the scoring routers and cutoff values described in Section 4.4. Table 3 reports the selected configurations and their ID and OOD test results across all 13 values of k.

A.1.2 Router-Based Top-m Selection

We use a router to rank candidate models without requiring them to generate output tokens, and retain its top m candidates before probing. The subset size is fixed, but its members can vary across queries. For each $m \in \{2, \ldots, 7\}$, we consider two criteria for choosing the router using the 2,302 ID validation queries. *Coverage* counts queries for which the retained subset contains at least one candidate that produces a correct response. *Total correctness* counts all correct candidate responses retained across queries. When routers tie under one criterion, we select the router with the higher value under the other criterion. Table 4 reports both criteria for each router and subset size. CSCR ranks only four retrieved candidates, so we evaluate it only for $m \leq 4$.

For each subset size, we first select one router under each criterion. For each value of k, we then evaluate the resulting router–subset-size combinations using Selective Probe-then-Route on the ID validation set. We select the final router and subset size with the highest average final-response accuracy, breaking ties by lower average total inference cost. Table 5 reports the corresponding ID and OOD test results.

A.1.3 Backward Elimination

This strategy is inspired by backward elimination in feature selection (Guyon & Elisseeff, 2003). Starting with all candidate models, we evaluate the removal of each remaining candidate on the ID validation set. At each step, we remove the candidate that yields the largest reduction in total inference cost, provided that the cumulative decrease in average final-response accuracy relative to

Table 3: Validation-selected configurations and corresponding ID and OOD test results for adaptive score-gap selection at each value of k. Accuracy and OSR are percentages; costs are USD per 1,000 queries.

(a) Selected configurations

k	Scoring router	τ
1	Lookahead	0.15
2	Lookahead	0.25
4	Lookahead	0.00
8	Lookahead	0.25
16	Lookahead	0.15
32	Lookahead	0.15
64	Lookahead	0.15
128	Lookahead	0.15
256	Lookahead	0.15
512	Lookahead	0.10
1024	Lookahead	0.25
2048	RadialRouter (cost-first)	0.25
4096	RadialRouter (performance-first)	0.25

(b) Test results

	ID				OOD			
k	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**
1	75.41	0.262	0.057	3.53	69.48	0.156	0.047	4.26
2	75.93	0.249	0.084	5.77	69.39	0.144	0.077	4.84
4	74.41	0.265	0.033	3.24	69.39	0.151	0.020	4.74
8	76.07	0.252	0.090	5.41	69.17	0.143	0.083	5.35
16	74.80	0.254	0.066	5.99	69.73	0.150	0.057	5.90
32	75.99	0.254	0.075	5.05	69.12	0.150	0.067	5.56
64	74.93	0.254	0.095	3.80	69.29	0.143	0.086	5.18
128	75.15	0.249	0.126	5.07	69.45	0.149	0.118	5.92
256	75.04	0.249	0.163	5.83	69.40	0.140	0.145	5.63
512	75.21	0.263	0.126	4.30	69.52	0.156	0.094	4.31
1024	76.64	0.228	0.459	6.72	69.84	0.152	0.286	4.48
2048	74.91	0.198	0.644	9.69	69.69	0.139	0.379	10.02
4096	75.38	0.194	0.445	8.34	69.27	0.158	0.185	4.86

the full candidate pool remains within a specified tolerance. The process stops when no eligible removal reduces cost or the minimum subset size of two is reached. We select the accuracy-loss tolerance separately for each value of k. The resulting subset is fixed for test evaluation. Table 6 reports the test results.

A.1.4 Greedy Coverage Selection

We construct a subset using candidate responses to training queries. A query is considered covered if at least one model in the subset produces a correct response. Starting with an empty subset, we repeatedly add the candidate that covers the largest number of previously uncovered queries until the desired subset size is reached. This follows the greedy approach to maximizing coverage, a special case of submodular maximization (Nemhauser et al., 1978). Ties are resolved by favoring the candidate with the lower combined cost of probing all training queries and generating the remaining output tokens for newly covered queries. We select the subset size using ID validation results separately for each value of k and keep the resulting subset fixed for test evaluation. Table 7 reports the test results.

Table 4: Validation counts for router-based top-m selection. Bold indicates the highest count for each subset size within each block, including ties. Dashes indicate that CSCR does not rank enough candidates.

Scoring router	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$
(a) Coverage						
CSCR	1,964	2,088	**2,144**	—	—	—
EmbedLLM	2,047	2,093	2,127	**2,144**	**2,156**	2,162
GraphRouter (balance)	1,969	2,059	2,097	2,120	2,155	**2,168**
GraphRouter (cost-first)	2,010	2,052	2,102	2,140	2,153	2,162
GraphRouter (performance-first)	2,013	2,052	2,100	2,123	2,155	2,164
Lookahead	**2,062**	**2,103**	2,124	2,138	2,153	2,161
MIRT	1,985	2,068	2,099	2,126	2,139	2,161
RadialRouter (balance)	2,015	2,062	2,113	2,133	2,155	2,164
RadialRouter (cost-first)	2,019	2,073	2,118	2,141	2,155	**2,168**
RadialRouter (performance-first)	2,010	2,077	2,108	2,135	2,154	2,163
(b) Total correctness						
CSCR	3,219	5,109	6,839	—	—	—
EmbedLLM	3,769	5,616	7,430	9,215	10,962	12,608
GraphRouter (balance)	3,361	5,165	6,911	8,674	10,468	12,297
GraphRouter (cost-first)	3,731	5,533	7,362	9,093	10,890	12,634
GraphRouter (performance-first)	3,736	5,537	7,363	9,159	10,892	12,639
Lookahead	**3,778**	**5,632**	**7,442**	**9,226**	10,967	12,634
MIRT	3,699	5,531	7,274	9,067	10,864	12,268
RadialRouter (balance)	3,743	5,545	7,379	9,189	10,994	12,593
RadialRouter (cost-first)	3,685	5,564	7,419	9,196	11,004	12,598
RadialRouter (performance-first)	3,728	5,568	7,343	9,190	**11,011**	**12,646**

A.1.5 Pareto Pruning

We compare candidate models using their accuracy and average full-response inference cost on the training data. Under standard Pareto dominance, a candidate is removed if another candidate has at least the same accuracy and no higher cost, with a strict improvement in at least one of these quantities.

We also consider approximate Pareto dominance, inspired by ϵ-dominance (Laumanns et al., 2002). Let a_i and c_i denote the accuracy and average full-response inference cost of candidate i, respectively. Candidate j approximately dominates candidate i if

$$\begin{aligned} (1+\epsilon)a_j \geq a_i, \qquad c_j \leq (1+\epsilon)c_i, \\ \text{and} \qquad \big(a_j > a_i \text{ or } c_j < c_i\big), \end{aligned} \tag{2}$$

where the same $\epsilon \geq 0$ controls the relative tolerance for accuracy and cost. We remove candidates that are approximately dominated by another candidate. When $\epsilon = 0$, this criterion reduces to standard Pareto dominance.

We consider both global and domain-specific pruning. Global pruning constructs one candidate subset using training data across all six domains. Domain-specific pruning constructs a separate subset using the training data from each domain. At inference time, a query-based domain predictor determines which subset to use. We evaluate $\epsilon \in \{0, 0.001, \ldots, 0.020\}$ and select the pruning variant and ϵ using ID validation results separately for each value of k. Table 8 reports the test results.

Domain predictor selection. We compare two ModernBERT-based domain predictors. Inspired by Shojaee et al. (2025), the first jointly trains the encoder and a linear classification head. Following the general feature-based classification setup of Gerz et al. (2021), the second freezes ModernBERT and trains an MLP domain classifier. For each predictor, we select the checkpoint with the highest micro accuracy on the ID validation set. Table 9 reports the results of the selected checkpoints. The

Table 5: Validation-selected configurations and corresponding ID and OOD test results for router-based top-m selection at each value of k. In (a), "Both" indicates that coverage and total correctness select the same scoring router. Accuracy and OSR are percentages; costs are USD per 1,000 queries.

(a) Selected configurations

k	Scoring router	m	Screening criterion
1	Lookahead	2	Both
2	Lookahead	4	Total correctness
4	EmbedLLM	5	Coverage
8	RadialRouter (cost-first)	7	Coverage
16	Lookahead	2	Both
32	Lookahead	2	Both
64	CSCR	4	Coverage
128	Lookahead	5	Total correctness
256	RadialRouter (performance-first)	6	Total correctness
512	EmbedLLM	5	Coverage
1024	Lookahead	2	Both
2048	Lookahead	2	Both
4096	CSCR	4	Coverage

(b) Test results

	ID				OOD			
k	Avg. acc.	Selected cost	Additional cost	OSR	Avg. acc.	Selected cost	Additional cost	OSR
1	74.98	0.258	0.056	3.78	69.44	0.156	0.040	4.45
2	75.67	0.249	0.102	5.43	68.58	0.145	0.095	5.06
4	75.21	0.168	0.110	15.62	67.71	0.099	0.111	13.85
8	76.13	0.228	0.161	7.21	68.44	0.146	0.160	5.54
16	75.17	0.249	0.065	5.56	69.02	0.151	0.048	5.59
32	75.42	0.246	0.074	4.83	68.62	0.147	0.057	5.42
64	74.21	0.227	0.233	5.25	68.61	0.116	0.209	6.50
128	75.26	0.217	0.306	8.75	68.91	0.142	0.286	7.56
256	74.73	0.186	0.463	18.72	68.40	0.124	0.440	13.44
512	75.33	0.188	0.545	16.54	67.74	0.122	0.401	12.14
1024	75.60	0.242	0.231	4.68	69.66	0.157	0.130	4.02
2048	74.87	0.239	0.251	4.95	69.51	0.150	0.144	5.97
4096	74.97	0.182	0.715	9.09	66.58	0.133	0.372	6.60

jointly trained predictor achieves higher micro and macro validation accuracy. We therefore use it throughout the domain-specific pruning experiments.

A.1.6 Selection-Frequency Pruning

We retain candidates according to how frequently Probe-then-Route selects them to produce the final response. In the offline variant, we compute selection frequencies on the complete ID validation set and remove candidates whose frequencies fall below a specified threshold. The resulting candidate subset is fixed for test evaluation. In the online variant, we first apply Probe-then-Route to an initial fraction of the query stream, compute selection frequencies, and apply the same pruning rule to subsequent queries.

For each value of k, we compare the offline and online variants on ID validation data, varying the frequency threshold and, for online pruning, the fraction of queries processed before pruning. We select one configuration and fix its hyperparameters for test evaluation. For online test evaluation, we mix ID and OOD queries into a single stream and average results over random seeds 0, 1, and 2. Pruning depends only on model-selection counts from earlier queries in the stream and uses

Table 6: Validation-selected configurations and corresponding ID and OOD test results for backward elimination at each probe length k. Multiple tolerances indicate tied configurations with identical test results. Accuracy and OSR are percentages; costs are USD per 1,000 queries.

(a) Selected configurations

k	**Accuracy-loss tolerance**
1	0.0000
2	0.0000
4	$\{0.0000, 0.0025, 0.0075, 0.0100\}$
8	0.0025
16	0.0075
32	$\{0.0000, 0.0025\}$
64	0.0000
128	$\{0.0025, 0.0050, 0.0075\}$
256	$\{0.0000, 0.0025, 0.0050\}$
512	$\{0.0000, 0.0025, 0.0050, 0.0075, 0.0100, 0.0125, 0.0150\}$
1024	$\{0.0100, 0.0125, 0.0150\}$
2048	$\{0.0000, 0.0025, 0.0050, 0.0075, 0.0100, 0.0125, 0.0150\}$
4096	$\{0.0000, 0.0025\}$

(b) Test results

	ID				**OOD**			
k	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**
1	73.20	0.207	0.031	9.55	68.81	0.151	0.027	3.61
2	74.06	0.212	0.030	4.02	69.31	0.099	0.026	5.25
4	72.53	0.163	0.035	17.42	68.02	0.085	0.032	13.70
8	75.14	0.210	0.093	7.13	68.84	0.109	0.084	6.98
16	73.05	0.166	0.041	26.95	67.78	0.085	0.038	13.56
32	74.81	0.207	0.095	11.50	68.22	0.122	0.088	8.19
64	73.83	0.260	0.218	2.80	68.81	0.151	0.189	4.09
128	73.40	0.181	0.072	7.58	68.86	0.091	0.064	7.44
256	72.90	0.188	0.085	24.85	69.08	0.092	0.063	13.99
512	70.71	0.146	0.094	36.50	67.37	0.071	0.055	34.99
1024	72.23	0.115	0.095	18.35	66.68	0.067	0.050	8.81
2048	71.60	0.177	0.163	24.10	68.58	0.093	0.069	12.57
4096	72.58	0.162	0.157	23.84	68.71	0.082	0.071	16.01

neither future queries nor response-correctness labels. Table 10 reports the test results of the selected configurations.

A.2 Experimental Setup Details

A.2.1 Candidate LLMs

We use eight open-weight LLMs from four model families, with model sizes ranging from 8B to 70B: (1) Qwen3-8B and Qwen3-14B, developed by the Qwen team; (2) Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct, the instruction-tuned models from Meta; (3) Ministral-3-8B-Instruct and Ministral-3-14B-Instruct, the instruction-tuned models from Mistral AI; and (4) Gemma-3-12B-IT and Gemma-3-27B-IT, the instruction-tuned models from Google DeepMind. This candidate pool includes different model families and sizes, allowing us to evaluate routing across heterogeneous LLMs.

Table 7: ID and OOD test results for greedy coverage selection. For each value of k, subset size m is selected on ID validation data. Accuracy and OSR are percentages; costs are USD per 1,000 queries.

		ID				OOD			
k	m	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**
1	6	74.55	0.256	0.170	6.13	69.28	0.161	0.154	4.07
2	6	75.62	0.250	0.172	5.36	68.97	0.143	0.156	4.65
4	7	74.38	0.176	0.184	15.81	68.50	0.101	0.167	10.67
8	4	75.58	0.272	0.165	3.28	68.20	0.166	0.149	4.45
16	6	73.73	0.216	0.199	25.29	67.25	0.151	0.182	9.20
32	7	74.80	0.226	0.247	10.96	67.44	0.137	0.226	7.58
64	5	74.49	0.235	0.281	4.54	69.00	0.140	0.250	4.79
128	7	74.67	0.218	0.454	8.14	68.91	0.143	0.390	6.24
256	7	74.55	0.214	0.659	14.72	67.80	0.135	0.515	6.43
512	3	74.48	0.283	0.475	1.95	68.29	0.178	0.301	1.35
1024	6	75.40	0.199	0.914	10.25	68.69	0.145	0.535	4.82
2048	7	74.07	0.198	1.169	13.59	68.53	0.142	0.660	7.10
4096	6	75.04	0.197	1.119	11.76	67.66	0.150	0.603	4.65

We access the Qwen3 models through Alibaba Cloud's Bailian API[2] and the remaining models through OpenRouter.[3] We prioritize official providers when available. Otherwise, we select providers that explicitly report 16-bit precision. Table 11 lists the providers and token prices used for cost calculation, as recorded on July 9, 2026. As of September 23, 2026, the provider recorded here as Weights & Biases is listed as CoreWeave on OpenRouter. Its Llama-3.1-70B-Instruct endpoint is no longer listed on OpenRouter as of that date.

A.2.2 Benchmarks and Data Splits

We use 12 benchmarks spanning six domains: general knowledge, science, mathematics, code generation, logical reasoning, and factual question answering. Each domain includes one in-distribution benchmark and one out-of-distribution benchmark. We train the routers on the training partitions of the six ID benchmarks and evaluate them on the corresponding ID test partitions and the evaluation sets specified below for the six OOD benchmarks. Table 12 lists the benchmarks and split sizes.

For the ID benchmarks, we retain the official training, validation, and test splits of MBPP. We partition the official MMLU test set, GPQA Main, and TruthfulQA into training, validation, and test sets with an approximate 7:1:2 ratio. We stratify MMLU and GPQA Main by subject and TruthfulQA by category to maintain similar proportions across the three partitions.

For MATH, we retain the official training set and divide the official test set into 10% validation and 90% test. For LogicBench, we use LogicBench(Aug) for training and divide LogicBench(Eval) into validation and test sets using the same ratio. We draw validation examples from these sets because candidate LLMs may have encountered the benchmark training examples during their own training. We stratify the MATH partition by problem type and difficulty level, and the LogicBench partition by logic type and reasoning pattern. We keep all questions associated with the same LogicBench context together in either the validation or test set.

For OOD evaluation, we use the official test sets of ARC-Challenge and LogiQA-en, all HumanEval tasks, and the complete English high-school and college application subsets of MathBench-A. For MMLU-CF and SimpleQA, we use the test portions of custom 7:1:2 partitions of the public MMLU-CF validation set and the released SimpleQA dataset, respectively. We partition MMLU-CF randomly and stratify SimpleQA by topic and answer type. For MMLU-CF, we use its public validation data because our evaluation requires access to individual queries and candidate responses. All custom partitions use a random seed of 42.

[2] https://bailian.console.aliyun.com/

[3] https://openrouter.ai/

Table 8: Validation-selected configurations and corresponding ID and OOD test results for Pareto pruning at each probe length k. Multiple ϵ values correspond to the same reported results. Accuracy and OSR are percentages; costs are USD per 1,000 queries.

(a) Selected configurations

k	**Pruning variant**	ϵ
1	Domain-specific	$\{0.000, 0.001, 0.002\}$
2	Domain-specific	$\{0.000, 0.001, 0.002\}$
4	Domain-specific	$\{0.000, 0.001, 0.002\}$
8	Domain-specific	$\{0.000, 0.001, 0.002\}$
16	Domain-specific	$\{0.000, 0.001, 0.002\}$
32	Global	$\{0.000, 0.001, \ldots, 0.007\}$
64	Global	$\{0.000, 0.001, \ldots, 0.007\}$
128	Global	$\{0.000, 0.001, \ldots, 0.007\}$
256	Domain-specific	$\{0.000, 0.001, 0.002\}$
512	Domain-specific	$\{0.000, 0.001, 0.002\}$
1024	Domain-specific	$\{0.000, 0.001, 0.002\}$
2048	Domain-specific	$\{0.000, 0.001, 0.002\}$
4096	Domain-specific	$\{0.000, 0.001, 0.002\}$

(b) Test results

	ID				**OOD**			
k	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**	**Avg. acc.**	**Selected cost**	**Additional cost**	**OSR**
1	75.38	0.244	0.079	6.95	68.58	0.130	0.074	4.21
2	75.12	0.241	0.080	7.76	69.02	0.124	0.075	5.30
4	74.09	0.160	0.082	21.22	67.66	0.088	0.077	19.99
8	75.33	0.241	0.086	7.67	68.66	0.128	0.081	6.67
16	74.30	0.197	0.094	26.95	67.65	0.114	0.090	13.82
32	75.09	0.209	0.079	11.53	68.48	0.124	0.072	8.33
64	73.90	0.216	0.113	7.28	69.36	0.132	0.101	5.32
128	73.79	0.190	0.177	8.65	69.19	0.119	0.153	6.53
256	73.82	0.191	0.275	25.34	68.31	0.106	0.253	17.05
512	74.32	0.165	0.352	33.26	67.49	0.097	0.284	25.45
1024	75.39	0.182	0.436	14.28	69.34	0.111	0.291	8.14
2048	74.43	0.185	0.500	15.57	68.91	0.116	0.304	11.99
4096	74.36	0.174	0.540	13.79	68.60	0.110	0.323	7.83

Table 9: Domain-prediction accuracy (%) on the ID validation set. Micro accuracy weights queries equally; macro accuracy averages recall across the six domains.

Predictor	**Micro accuracy**	**Macro accuracy**
Fine-tuned ModernBERT + linear head	**99.57**	**97.41**
Frozen ModernBERT + MLP	60.25	16.67

A.2.3 BASELINES

We compare our methods with the following baselines:

1. **Individual LLMs.** We evaluate each of the eight candidate LLMs and report each model's results on all ID and OOD benchmarks.
2. **Random Selection.** This baseline selects each of the eight candidate LLMs with equal probability for every query. We report exact expected metric values over all candidate selections.

Table 10: Validation-selected configurations and corresponding ID and OOD test results for selection-frequency pruning. Configurations are selected across the offline and online variants separately for each value of k. The pruning point is the percentage of the stream processed before online pruning; dashes indicate that it does not apply. Multiple cutoffs indicate tied configurations with identical test results. Online results are averaged over seeds 0, 1, and 2. Accuracy and OSR are percentages; costs are USD per 1,000 queries.

(a) Selected configurations

k	Variant	Frequency cutoff (%)	Pruning point (%)
1	Online	1.25	12.50
2	Online	1.25	12.50
4	Online	1.25	2.50
8	Online	1.25	12.50
16	Online	2.50	12.50
32	Online	1.25	2.50
64	Online	5.00	10.00
128	Online	1.25	10.00
256	Online	1.25	7.50
512	Offline	5.00	—
1024	Offline	$\{2.50, 3.75\}$	—
2048	Online	$\{3.75, 5.00\}$	10.00
4096	Online	2.50	15.00

(b) Test results

	ID				OOD			
k	Avg. acc.	Selected cost	Additional cost	OSR	Avg. acc.	Selected cost	Additional cost	OSR
1	74.47	0.256	0.161	6.46	69.19	0.160	0.146	4.14
2	75.66	0.243	0.163	7.23	69.00	0.140	0.148	5.07
4	74.89	0.165	0.181	20.85	68.00	0.095	0.164	17.81
8	76.12	0.243	0.183	7.22	68.29	0.147	0.166	5.55
16	74.59	0.201	0.218	28.61	67.18	0.134	0.199	16.74
32	74.97	0.222	0.255	12.31	67.43	0.133	0.233	9.27
64	74.44	0.236	0.252	4.80	69.06	0.140	0.223	4.94
128	74.70	0.216	0.414	9.78	68.80	0.143	0.355	8.88
256	74.49	0.206	0.689	18.21	67.38	0.123	0.546	14.62
512	74.72	0.191	0.785	9.07	67.76	0.125	0.535	8.14
1024	74.93	0.200	0.912	10.62	68.83	0.144	0.525	5.78
2048	74.34	0.194	1.058	14.69	68.32	0.137	0.578	10.83
4096	74.99	0.196	1.125	10.41	67.63	0.148	0.597	5.73

3. **EmbedLLM** (Zhuang et al., 2025). This method learns compact model representations from their performance on training queries and combines them with query representations to predict each candidate's response correctness.

4. **MIRT** (Song et al., 2025). This method uses multidimensional item response theory to model LLM abilities and query attributes, including difficulty, and predicts the probability that each candidate answers a query correctly.

5. **GraphRouter** (Feng et al., 2025). This method represents tasks, queries, and candidate LLMs in a graph and learns their relationships to select a model based on response performance and inference cost.

6. **RadialRouter** (Jin et al., 2025). This method uses a Transformer (Vaswani et al., 2017) with a radial attention structure to model interactions between the query and candidate LLM representations and produce model-selection scores.

Table 11: Providers and token prices recorded on July 9, 2026. Input and output prices are in USD per million tokens.

Model	Provider	Input	Output
Qwen3-8B	Alibaba Cloud	0.072	0.287
Qwen3-14B	Alibaba Cloud	0.144	0.574
Llama-3.1-8B-Instruct	Weights & Biases	0.220	0.220
Llama-3.1-70B-Instruct	Weights & Biases	0.800	0.800
Ministral-3-8B-Instruct	Mistral AI	0.150	0.150
Ministral-3-14B-Instruct	Mistral AI	0.200	0.200
Gemma-3-12B-IT	DeepInfra	0.050	0.150
Gemma-3-27B-IT	Novita	0.119	0.200

Table 12: Benchmark domains and numbers of instances in the splits used in our experiments. N/A indicates splits not used for router training or validation. MathBench-A includes the high-school and college subsets.

Domain	Setting	Benchmark	Train	Validation	Test
General knowledge	ID	MMLU	9,809	1,387	2,846
	OOD	MMLU-CF	N/A	N/A	2,000
Science	ID	GPQA Main	313	44	91
	OOD	ARC-Challenge	N/A	N/A	1,172
Mathematics	ID	MATH	7,500	500	4,500
	OOD	MathBench-A	N/A	N/A	300
Code generation	ID	MBPP	374	90	500
	OOD	HumanEval	N/A	N/A	164
Logical reasoning	ID	LogicBench	12,908	202	1,818
	OOD	LogiQA-en	N/A	N/A	651
Factual question answering	ID	TruthfulQA	553	79	158
	OOD	SimpleQA	N/A	N/A	865
	ID total		**31,457**	**2,302**	**9,913**
	OOD total		**N/A**	**N/A**	**5,152**

7. **CSCR** (Shirkavand et al., 2025). This method maps queries and candidate models into a shared representation space through cost-aware contrastive learning and selects a model using representation similarity and model cost.

8. **Lookahead** (Huang et al., 2025). This method predicts representations of potential candidate responses from the query and uses these representations to guide model selection. We use its ModernBERT-based implementation.

9. **FrugalGPT** (Chen et al., 2024a). This cascading method invokes LLMs sequentially and uses a learned response-quality scorer and stopping thresholds to decide whether to return a response or invoke another model. Its selected-model cost covers generation of the returned response, and its additional cost covers the preceding cascade stages.

All methods use the same pool of eight candidate LLMs. For GraphRouter and RadialRouter, we report the balance, cost-first, and performance-first variants. We additionally report an **oracle** as an upper-bound reference on accuracy within the original pool of eight candidates. For each query, it selects the candidate with the lowest full-response inference cost among those with correct responses. If all responses are incorrect, it selects the lowest-cost candidate. We charge only the selected response's inference cost and assign zero additional cost.

A.2.4 Evaluation Metrics

We evaluate the methods using accuracy, average selected-model cost, average additional cost, and OSR. Both cost metrics are averaged over queries and reported in USD per 1,000 queries.

Accuracy. We measure accuracy as the proportion of queries for which the selected model produces a correct response. We report accuracy for each benchmark and the unweighted average across the six ID and six OOD benchmarks separately.

Average selected-model cost. For each query, selected-model cost is the cost of generating the selected model's complete response alone, including its input and output tokens. We compute all inference costs using model-specific input and output token prices.

Average additional cost. For each query, additional cost measures the extra inference cost incurred relative to generating the selected model's complete response alone:

$$C_{\text{additional}}(q) = C_{\text{total}}(q) - C_{\text{selected}}(q), \tag{3}$$

where $C_{\text{total}}(q)$ includes probing candidate models and, when needed, continuing the selected model's response, and $C_{\text{selected}}(q)$ is the selected-model cost. Methods that select a model without probing have zero additional cost.

Optimal selection ratio. Following RouterArena (Lu et al., 2026), we report OSR, which measures how often the router selects the cheapest correct candidate among queries with at least one correct candidate:

$$\text{OSR} = \frac{\text{Number of queries where a cheapest correct candidate is selected}}{\text{Number of queries with at least one correct candidate}}. \tag{4}$$

Candidates are compared using the cost of generating their complete responses, excluding probing overhead. We use the same full pool of eight candidates as the reference for all methods, including Selective Probe-then-Route. We report OSR as a percentage.

A.2.5 Implementation Details

Each predictor uses a 12-head cross-attention layer and a one-layer Set Transformer with four attention heads. We train both predictors for 40 epochs using AdamW on an NVIDIA A100 80GB GPU. We use a batch size of 4, gradient accumulation over 4 steps, weight decay of 0.01, dropout of 0.1, and gradient clipping at 1.0. The learning rates are 3×10^{-4} for the correctness predictor and 1×10^{-3} for the length predictor. We use random seed 42. No learning-rate scheduler or early stopping is used. We select the checkpoint with the highest average ID validation accuracy.

For each probe length k, we grid-search 742 temperature–δ pairs. The temperature grid contains 41 logarithmically spaced values from 0.25 to 20, the anchor values $\{0.5, 0.75, 1, 1.5, 2, 2.5, 3, 4, 5, 8, 10, 20\}$, and one temperature fitted separately for that value of k. After removing duplicates, this gives 53 temperature values. We combine them with $\delta \in \{0, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.015, 0.02, 0.03, 0.05, 0.075, 0.10, 0.15, 0.20\}$. The reported settings are selected on the ID test set and are listed in Table 13. Selective Probe-then-Route reuses the temperature and δ selected for Probe-then-Route at the corresponding value of k, without retuning them.

A.3 Answer Exposure Analysis

We examine whether the accuracy achieved at small values of k can be associated with answers already appearing in the generated tokens. We analyze answer exposure on ID training data, ID test data, and OOD benchmarks.

Definition and setup. We consider an answer exposed when the observed tokens contain a complete answer that is not subsequently withdrawn or replaced. Merely mentioning an answer option or considering it tentatively does not qualify. Both correct and incorrect answers count. For coding benchmarks, exposure requires the complete final proposed implementation, including required

Table 13: Selected inference hyperparameters for each value of k. Probe-then-Route and Selective Probe-then-Route use the same temperature and δ.

k	Temperature	δ
1	5.993496	0.0010
2	16.064806	0.0010
4	12.903900	0.0075
8	12.903900	0.0010
16	0.311239	0
32	0.432341	0
64	3.866974	0.0025
128	2.004046	0.0025
256	1.038590	0.0010
512	0.930823	0.0300
1024	5.000000	0.0025
2048	4.814218	0.0025
4096	1.796100	0.0010

Table 14: Accuracy (%) of Probe-then-Route at $k = 2$ and $k = 8$. Avg. denotes the unweighted average across the six benchmarks in each setting. Bold indicates the higher displayed accuracy in each row, including ties.

Benchmark	$k = 2$	$k = 8$
ID benchmarks		
MMLU	**83.84**	83.77
GPQA	57.14	**62.64**
MATH	**91.69**	91.11
MBPP	65.20	**66.00**
LogicBench	81.85	**82.78**
TruthfulQA	**73.42**	72.15
Avg.	75.52	**76.41**
OOD benchmarks		
MMLU-CF	69.05	**70.80**
ARC-C	**94.97**	**94.97**
MathBench-A	89.67	**91.33**
HumanEval	85.98	**86.59**
LogiQA-en	**64.82**	64.06
SimpleQA	**9.36**	6.47
Avg.	68.97	**69.04**

supporting code, regardless of whether it passes the tests. For each benchmark and value of k, we report the percentage of available candidate responses whose answers are exposed within the first k output tokens, pooling responses from all eight candidate LLMs. Table 14 compares per-benchmark accuracy at $k = 2$ and $k = 8$.

Answer exposure at $k = 8$. Figures 7, 8, and 9 show sharp increases between $k = 4$ and $k = 8$ on several benchmarks. For MMLU, exposure increases from 2.04% to 68.47% on the training sample and from 2.17% to 68.81% on the test set. On OOD benchmarks, it increases from 2.19% to 71.31% on MMLU-CF, from 3.23% to 63.80% on ARC-C, and from 2.79% to 38.17% on LogiQA-en. GPQA also shows an increase, reaching 11.10% on training data and 13.32% on test data at $k = 8$. The pattern is benchmark-dependent: complete answers remain rarely exposed on mathematics and coding benchmarks at this value of k.

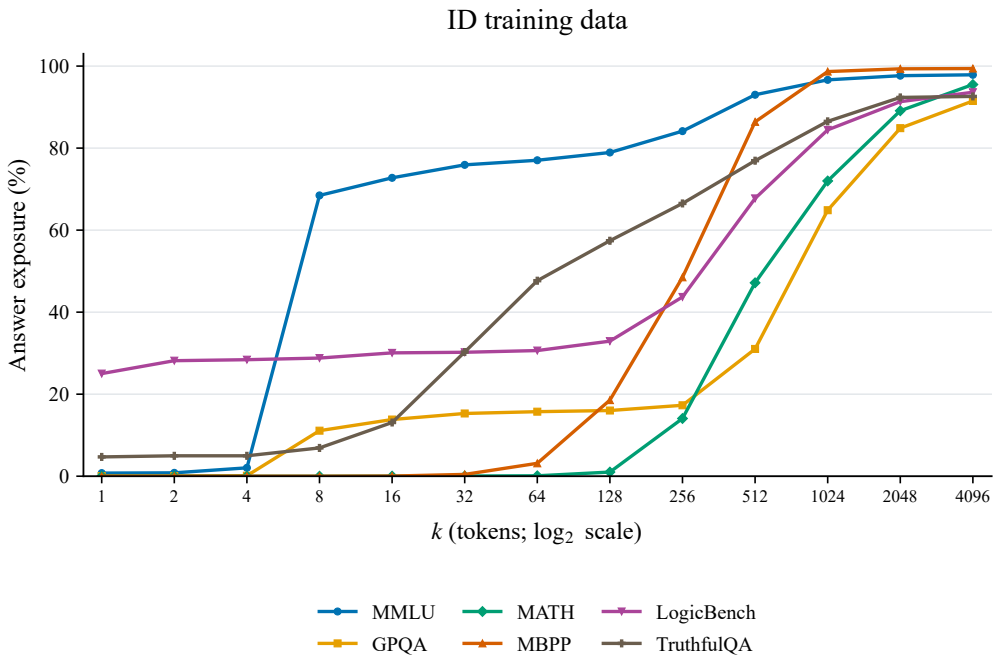

Figure 7: Answer exposure across values of k on ID training data, using stratified subsets of MMLU, MATH, and LogicBench and the full training splits of the other benchmarks. Rates are estimated over available responses from all eight candidate LLMs and include both correct and incorrect answers.

These observations provide a possible explanation for the strong average ID accuracy at $k = 8$. On several benchmarks, eight tokens allow the router to observe candidate answers and potentially compare them with the query and with one another. The similar increase on MMLU training and test data suggests that this source of routing information is available during both training and evaluation. However, answer exposure alone does not determine routing accuracy, since exposed answers can also be incorrect.

Routing with $k = 2$. At $k = 2$, LogicBench has substantially higher answer exposure than the other benchmarks: 28.17% on the training sample and 25.72% on the test set. Nevertheless, $k = 2$ is not a particularly strong setting for LogicBench. Its accuracy is 81.85%, compared with 82.45% at $k = 1$ and 82.78% at $k = 8$. Thus, the strong average ID accuracy at $k = 2$ is not explained simply by early answer exposure on LogicBench.

Conversely, $k = 2$ achieves 57.14% accuracy on GPQA and 91.69% on MATH, despite measured test-set answer exposure of 0% and less than 0.01%, respectively. These results suggest that useful routing signals may appear before a complete answer becomes visible. One hypothesis is that the opening tokens reflect how a candidate approaches the query, such as whether it begins with a direct answer or an explanation. Such patterns may correlate with final-response correctness even when the answer itself is not yet available.

Interpretation and limitations. The analysis suggests two possible sources of routing information: answers that become visible within the generated tokens, and earlier response patterns that may help predict correctness. The curves do not establish which signals the router actually uses or whether those signals cause the accuracy gains. We therefore use these results to motivate hypotheses about routing behavior rather than to establish a causal explanation.

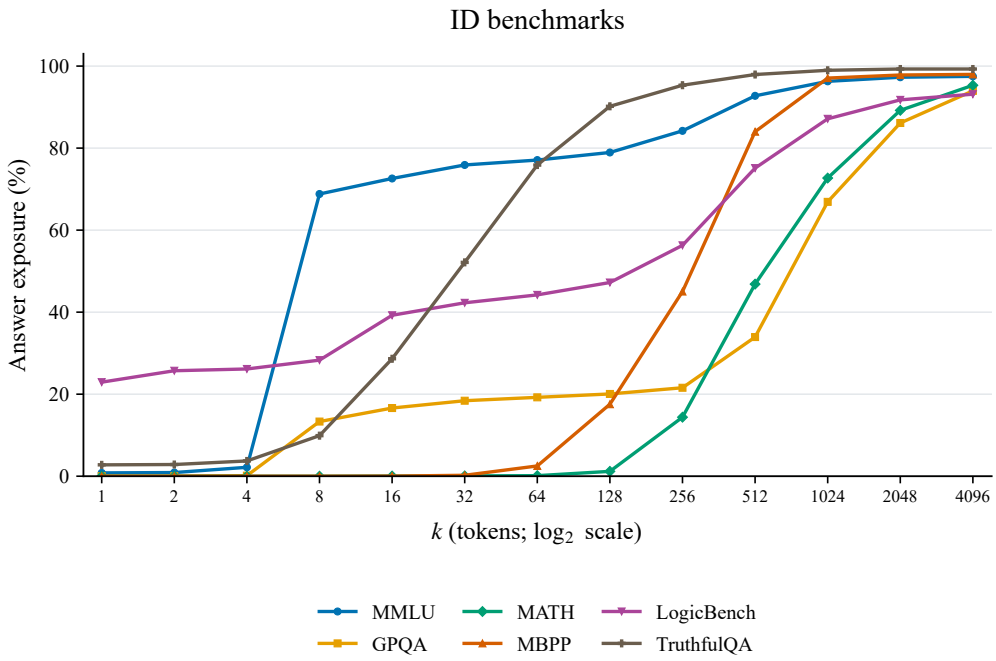

Figure 8: Answer exposure across values of k on ID test data. Rates are estimated over available responses from all eight candidate LLMs and include both correct and incorrect answers.

A.4 Additional Ablation Studies

We provide additional results on model components, dropout, and balanced training. For variants that use generated tokens, we evaluate 13 values of k, doubling from $k = 1$ to $k = 4096$. We report average accuracy across the six ID and six OOD benchmarks separately.

A.4.1 Model Components

We compare Probe-then-Route with variants that remove generated-token inputs, model-ID embeddings, cross-attention, or the Set Transformer. We also examine encoder choice by replacing ModernBERT with BGE-M3. The no-prefix variant removes cross-attention and generated-token pooling from both predictors. Each candidate's MLP receives only the pooled query vector and its model-ID embedding, with its input dimension adjusted accordingly. The Set Transformer and prediction head are retained. We train this variant without generated-token inputs and show its accuracy as a horizontal reference line because it does not depend on k.

Figure 10 shows the results. The no-prefix variant achieves average accuracies of 73.63% on ID benchmarks and 67.48% on OOD benchmarks. Probe-then-Route increases these values to 76.41% and 69.04%, respectively, at $k = 8$. Removing model-ID embeddings lowers average ID accuracy at all 13 values of k. At $k = 8$, removing cross-attention or the Set Transformer, or replacing ModernBERT with BGE-M3, also lowers average ID accuracy.

The effects of individual components vary across values of k and evaluation settings. Removing cross-attention improves average ID accuracy at every tested $k \geq 256$ and average OOD accuracy at all but $k = 1024$ within this range. Removing model-ID embeddings has mixed effects on OOD accuracy. Removing the Set Transformer can either improve or reduce average accuracy on both ID and OOD benchmarks, depending on k. Replacing ModernBERT with BGE-M3 lowers average ID accuracy at every tested value of k except $k = 1024$, while its effect on OOD accuracy varies with k. Thus, the advantages observed at $k = 8$ on ID benchmarks do not hold uniformly across all settings.

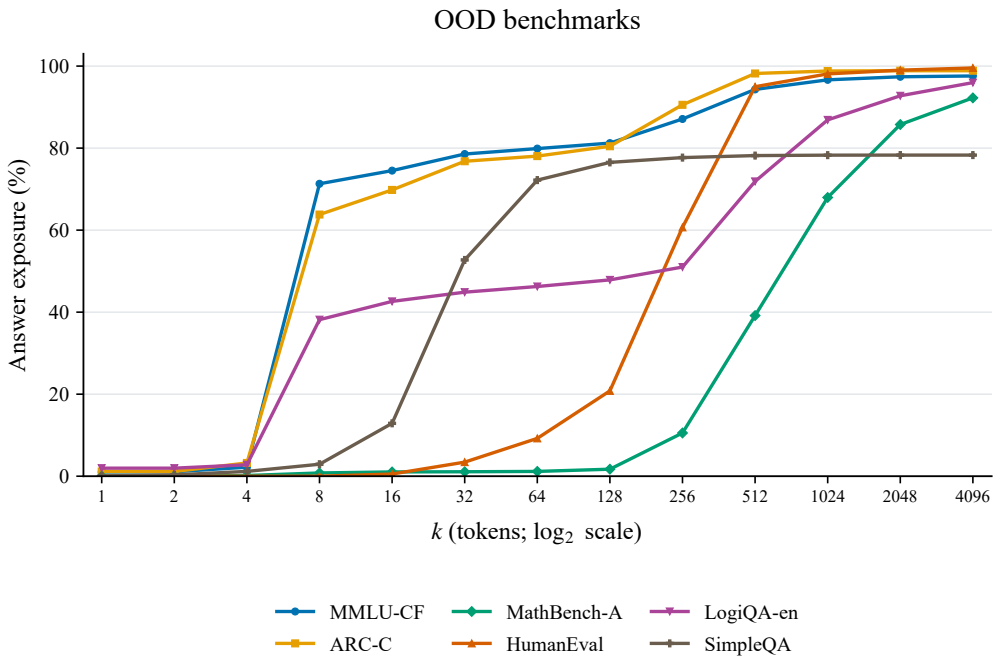

Figure 9: Answer exposure across values of k on OOD benchmarks. Rates are estimated over available responses from all eight candidate LLMs and include both correct and incorrect answers.

A.4.2 Dropout

We evaluate dropout rates of $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ across all 13 values of k. The reported Probe-then-Route configuration uses a dropout rate of 0.1. Figure 11 shows that this setting achieves the highest average ID accuracy among the tested rates at $k = 8$. However, other rates perform better at some values of k. Increasing dropout does not consistently improve accuracy. On ID benchmarks, rates of 0.4 and 0.5 often produce lower accuracy at larger values of k. On OOD benchmarks, the best-performing rate varies with k.

A.4.3 Balanced Training

The six ID benchmarks differ substantially in training-set size (Table 12). Consequently, combining all training data gives larger benchmarks a greater contribution to the training objective. Inspired by the dataset-resizing procedure in RadialRouter's implementation (Jin et al., 2025),[4] we examine training the correctness predictor with equal numbers of training entries from each benchmark. We vary the target number of entries per benchmark from 1,000 to 13,000 in increments of 1,000. For each benchmark, we repeat its training examples as needed and truncate the resulting sequence to the target size. The target size includes any repeated examples. The length predictor continues to use all ID training instances.

Figure 12 shows the results across training sizes and values of k. Increasing the number of entries per benchmark does not consistently improve accuracy on either ID or OOD benchmarks. The best-performing training size also varies with k.

[4] https://github.com/M1nasGera1s/RadialRouter/tree/1e4fb184a142619b7bb3611337a266ce286a328e

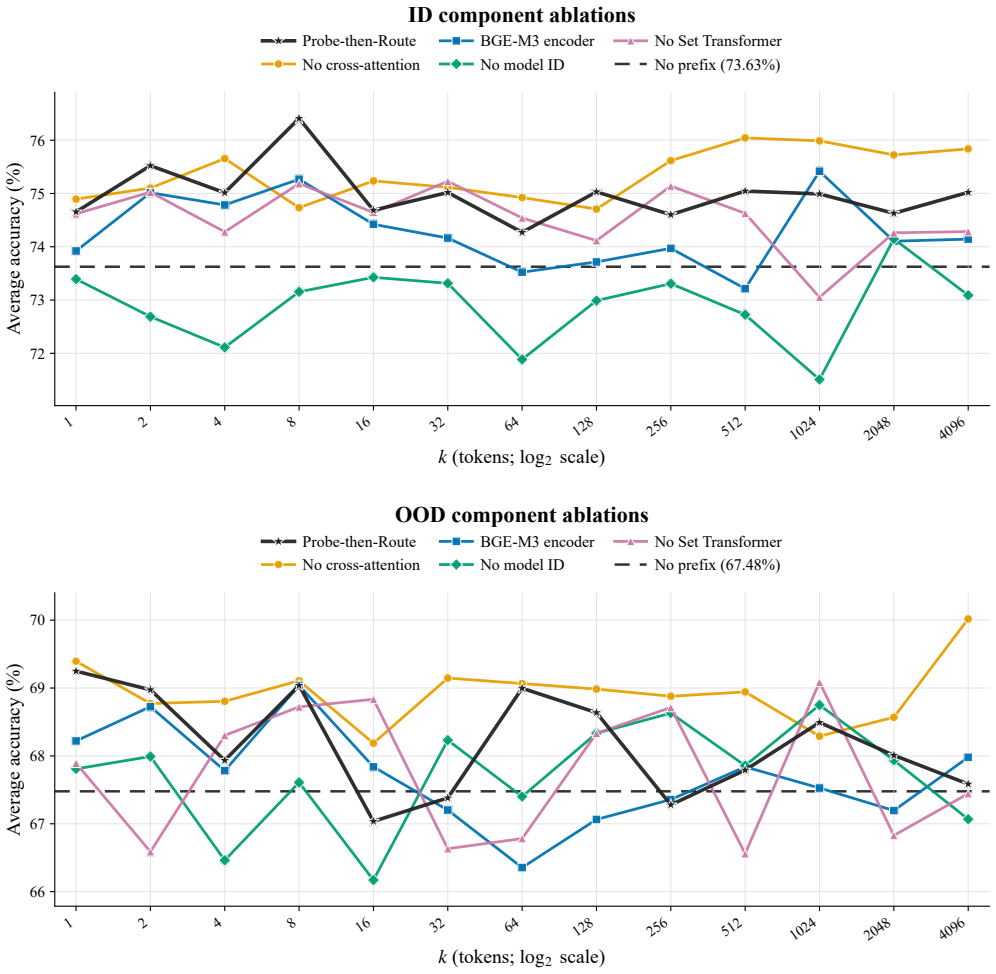

Figure 10: Component ablations on ID benchmarks (top) and OOD benchmarks (bottom). Dashed lines show the no-prefix variant, whose accuracy does not depend on k.

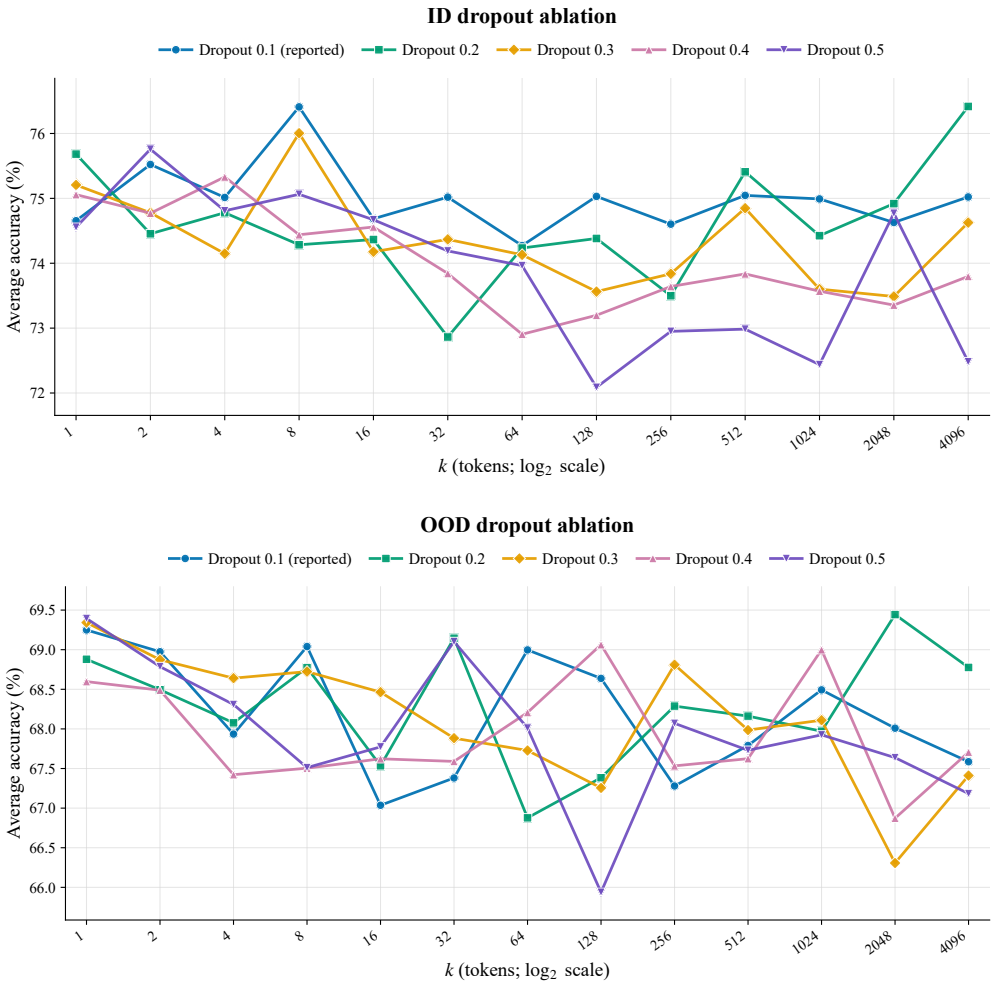

Figure 11: Effects of dropout on average ID accuracy (top) and OOD accuracy (bottom). Probe-then-Route uses a dropout rate of 0.1.

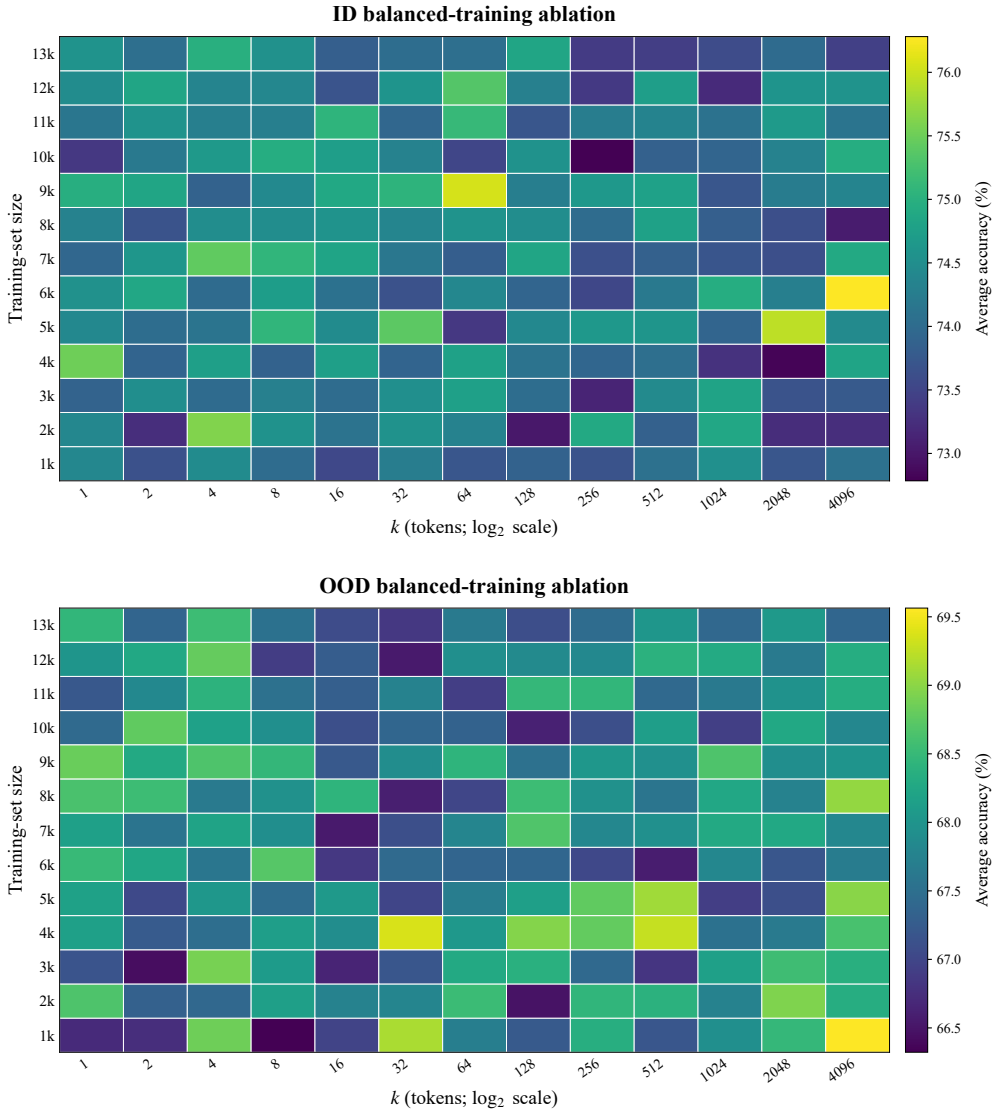

Figure 12: Balanced-training results on ID benchmarks (top) and OOD benchmarks (bottom). Training-set size denotes the number of entries per benchmark, including repeated examples. Colors indicate average accuracy.