DYNAMIC LLM ROUTING: WHEN TO ADMIT A NEW MODEL?

Anonymous authors
Paper under double-blind review

ABSTRACT

To leverage newly released Large Language Models (LLMs) cost-effectively, dynamic routing methods onboard new LLMs using observations of whether they answer the queries correctly (i.e., correctness observations). Such observations may mislead the routers to redirect queries from existing LLMs that would answer correctly to a new LLM that answers incorrectly. Consequently, this worsens the overall system accuracy. However, existing dynamic routing methods focus solely on router updating, leaving the deployment decision: *whether to admit a candidate LLM*, unaddressed. To bridge this gap, we propose the *Dual-Evidence Admission Filter (E^2-Filter)*, an admission decision method that can operate on top of existing dynamic routing methods. E^2-Filter combines *pool-history evidence* with *self-simulated evidence*. The former quantifies the reliability of predicted routing gains using residuals from pseudo-onboarding trials; the latter uses the candidate LLM's observations to assess whether redirecting queries to the candidate improves accuracy. These dual-channel evidences are then jointly used to determine whether to admit the candidate. Experiments across multiple benchmarks and routing methods demonstrate that E^2-Filter reduces harmful model admissions while retaining beneficial routing updates.

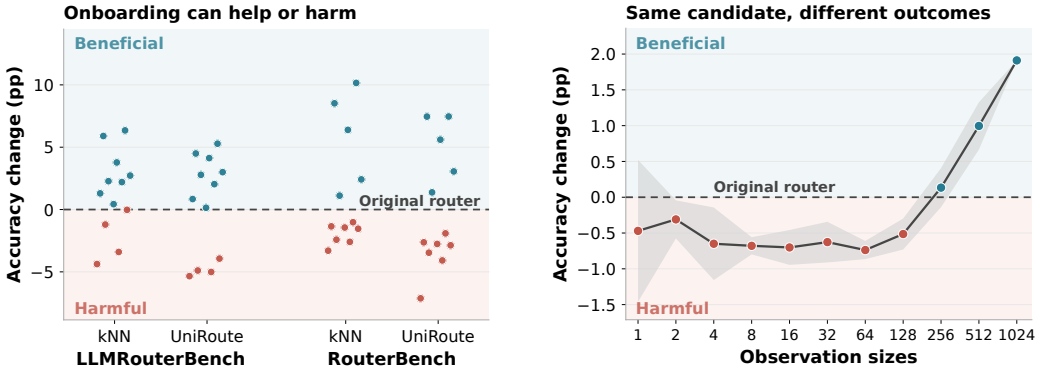
1 INTRODUCTION

Large language model (LLM) routing assigns input queries to suitable models to exploit their complementary strengths while balancing answer quality and inference costs (Ong et al., 2025; Chen et al., 2024; Jiang et al., 2023; Aggarwal et al., 2024; Dekoninck et al., 2024; Chen et al., 2023). To leverage newly released LLMs, routing methods need to be able to adapt to an expanding model pool. While repeatedly retraining the router for newly arrived models (i.e., LLMs) is computationally prohibitive, dynamic routing methods bypass this requirement by enabling unseen models to be incorporated without fully retraining the routers (Feng et al., 2025; Zhang et al., 2025; Shirkavand et al., 2025). Typically, these methods construct a candidate model's capability representation from correctness observations over a number of known queries, based on which the router predicts the candidate model's accuracy on unseen queries and dynamically selects the predicted best-performing model for each input query (Jitkrittum et al., 2026; Patel et al., 2026; Wang et al., 2026a).

Although incorporating a new model expands the router's available choices, it does not guarantee that the overall system becomes more accurate. With limited candidate observations, the updated router may redirect queries from existing models that would answer correctly to the candidate, which answers them incorrectly (Polo et al., 2024; Vivek et al., 2024). To investigate this empirically, we examine how incorporating a new model changes routing accuracy. Fig. 1(a) shows that, at a fixed observation size, onboarding can improve or degrade accuracy across different combinations of existing model pools and candidates. This phenomenon occurs with different onboarding methods (kNN and UniRoute-LearnedMap (Jitkrittum et al., 2026)) on two benchmarks (LLMRouterBench (Li et al., 2026) and RouterBench (Hu et al., 2024)), indicating that the issue is general rather than confined to a particular method or benchmark. Moreover, Fig. 1(b) shows that as observations increase, onboarding shifts from degrading accuracy to improving it—eventually driving deployed accuracy above the original router.

These observations motivate an explicit *new-model admission decision*: whether to deploy the updated router constructed from the available candidate observations or retain the original router. The key

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.



(a) Accuracy changes for kNN and UniRoute-LearnedMap on LLMRouterBench and RouterBench.

(b) Accuracy changes with observation size under a given benchmark, routing method, model pool, and candidate model.

Figure 1: Empirical motivation for new-model admission. All old model pools contain $|M_{\text{old}}| = 5$ models. Accuracy changes (percentage points) are measured relative to the original router and averaged over 10 support draws. (a) Each point represents an existing model pool paired with a candidate at a fixed budget ($k = 16$). The results show both beneficial and harmful onboarding across methods and benchmarks. (b) Onboarding Qwen2.5-Coder-7B-Instruct using kNN on LLMRouterBench with the original router held fixed. The update changes from harmful to beneficial as more candidate observations become available.

challenge is to assess the update's accuracy gain using the same limited candidate observations used to construct it, without access to correctness labels for the target queries. As shown in Fig. 1(a), the predicted gain reflects the scores that drive rerouting, but does not establish whether those reroutings will correct more errors than they introduce.

To address this challenge, we propose the *Dual-Evidence Admission Filter (E^2-Filter)*, a plug-in admission decision method that can operate on top of any existing dynamic routing method. E^2-Filter uses the available correctness observations from both existing models and the candidate to decide whether to admit the new model, aiming to capture the performance improvements while filtering out accuracy-degrading updates. E^2-Filter combines two complementary evidences into a unified admission score to guide deployment: *pool-history evidence* with *self-simulated evidence* (Sec. 3). To correct potential bias in routing gains resulting from limited candidate observations (Smith & Winkler, 2006; Cawley & Talbot, 2010), the pool-history component simulates pseudo-onboarding on existing models to compare their predicted and realized routing gains. These pseudo-onboarding residuals provide an empirical basis to adjust the predicted gain of admitting the new model. However, the new candidate may exhibit different correctness patterns from existing models (Polo et al., 2024), so historical residuals may not fully capture its gain-estimation bias. The self-simulated component leverages candidate observations to evaluate whether query reroutings yield a net accuracy gain by correcting more errors than they introduce. Recognizing that candidate-specific evidence alone can be sparse at small budgets, E^2-Filter integrates both evidence sources into a unified admission score to determine whether to deploy the updated router or maintain the existing one.

We evaluate E^2-Filter on three benchmarks with four different routing methods (Sec. 4). Our results show that E^2-Filter consistently improves aggregate routing gains over the admission baselines across multiple benchmarks and dynamic routing methods, reducing harmful deployments while retaining gains from beneficial updates. These findings highlight the importance of explicit admission decisions for reliably incorporating new models into existing routing systems.

2 Related Work and Problem Formulation

Related Work: Static and Dynamic LLM Routing. LLM routing selects an optimal LLM for each query to balance model utility, cost, and latency (Jiang et al., 2023; Aggarwal et al., 2024; Dekoninck et al., 2024; Somerstep et al., 2025; Zhuang et al., 2025). For simplicity, we focus on accuracy in this paper, as additional constraints can be incorporated via Lagrangian penalties without

loss of generality (Jitkrittum et al., 2026; Bhatti et al., 2026). Let $p(x|m)$ denote the probability that model m answers query x correctly. Under the accuracy-maximizing objective, an optimal routing method selects the model with the highest correctness probability for each query.

For a given model pool $\mathcal{M}_{\text{old}} = \{m_1, \ldots, m_M\}$, the correctness probabilities $p(x|m)$ are unknown for unseen queries. Each model m has historical observations $S_m = \{(x_i, y_m(x_i))\}_{i=1}^N$ on $\mathcal{Q}_{\text{train}} = \{x_i\}_{i=1}^N$, where $y_m(x) \in \{0, 1\}$ indicates whether m answers x correctly. A correctness predictor h_ϕ is fitted to the historical datasets $\{S_m\}_{m \in \mathcal{M}_{\text{old}}}$ to estimate these probabilities, with $h_\phi(x, m; S_m) \approx p(x|m)$. The resulting router selects the model with the highest estimated correctness probability:

$$\pi^{\text{old}}(x) = \arg\max_{m \in \mathcal{M}_{\text{old}}} h_\phi(x, m; S_m). \tag{1}$$

Dynamic onboarding incorporates a new candidate $m^\star \notin \mathcal{M}_{\text{old}}$ using a small observation set $S_{m^\star,k} = \{(x_j, y_{m^\star}(x_j))\}_{j=1}^k$ with $x_j \in \mathcal{Q}_{\text{train}}$ (typically $k < N$). To estimate $p(x|m^\star)$, these methods use the historical predictor h_ϕ and candidate observations $S_{m^\star,k}$ to obtain the correctness estimate $h_\phi(x, m^\star; S_{m^\star,k})$ via parameter adaptation (Yan et al., 2026; Wang et al., 2026b) or support-conditioned prediction (Jitkrittum et al., 2026; Patel et al., 2026; Wang et al., 2026a). Holding the old-model predictions fixed, the updated router is

$$\pi^{\text{new}}(x) = \begin{cases} m^\star, & \text{if } h_\phi(x, m^\star; S_{m^\star,k}) > \max\limits_{m \in \mathcal{M}_{\text{old}}} h_\phi(x, m; S_m), \\ \pi^{\text{old}}(x), & \text{otherwise.} \end{cases} \tag{2}$$

A detailed discussion of related work is provided in Appendix A.

Our Perspective: New-Model Admission Decision. Although dynamic onboarding expands the router's candidate choices, the updated router π^{new} does not necessarily yield higher routing accuracy than the original one π^{old}, as demonstrated in Fig. 1. This finding motivates an explicit *new-model admission decision*: determining whether to deploy the updated router or retain the original one. Specifically, both historical observations $\{S_m\}_{m \in \mathcal{M}_{\text{old}}}$ and candidate observations $S_{m^\star,k}$ are evaluated over queries sampled from the query distribution P_X. While the predictive model h_ϕ is trained on $\{S_m\}_{m \in \mathcal{M}_{\text{old}}}$ to form π^{old}, onboarding the candidate $m^\star$ incorporates its observation set $S_{m^\star,k}$ alongside h_ϕ to construct π^{new} as in Eq. (2). The admission method makes a binary decision: whether to deploy π^{new} or retain π^{old}. Prior to observing the true correctness outcomes on target queries $\mathcal{Q}_{\text{test}} \sim P_X$, our goal is to admit $m^\star$ if and only if doing so maximizes the expected accuracy gain over π^{old}. We detail our proposed admission framework in Sec. 3.

3 Dual-Evidence Admission Filter

A simple approach to new-model admission is to admit a candidate whenever its average predicted score advantage over the original router is positive (as in Predicted-Gain Admit in Sec. 4.1). However, limited candidate data can make predicted advantages misleading, yielding no actual routing improvements. Thus, the core challenge lies in evaluating candidate quality from limited observations.

Existing models provide richer historical observations, allowing us to simulate onboarding and assess errors in predicted routing gains. However, this history may not fully capture the new candidate's behavior. The candidate's own observations offer specific evidence about whether rerouting corrects or introduces errors, but are sparse. These complementary limitations motivate combining historical calibration with evidence from the candidate itself. We therefore propose the *Dual-Evidence Admission Filter (E^2-Filter)*, which integrates *pool-history evidence* and *self-simulated evidence* into a unified admission score. Algorithm 1 outlines the complete procedure of the proposed E^2-Filter. Specifically, it first calibrates historical gains via pseudo-onboarding (s_{hist}), then estimates candidate-specific fidelity through cross-fitting (s_{cand}), and finally fuses both into a unified decision score s to determine whether to admit the candidate.

3.1 Pool-History Evidence

When onboarding a new candidate $m^\star$, h_ϕ relies on the support set $S_{m^\star,k}$ to estimate correctness probabilities, potentially introducing estimation noise. To account for prediction bias, we calibrate

Algorithm 1 Dual-Evidence Admission Filter (E^2-Filter)

Input: Original router π^{old}, routing predictor h_ϕ, historical observations $\{S_m\}_{m\in\mathcal{M}_{\text{old}}}$ on $\mathcal{Q}_{\text{train}}$, candidate $m^\star$ with support set $S_{m^\star,k}$, unlabeled target queries $\mathcal{Q}_{\text{test}}$, gain threshold ϵ, admission threshold τ, trial count T, fold count J and fusion hyperparameter λ.

▷ *Pool-history evidence*

1: Compute the predicted gain $\widehat{g}_{\text{new}}$ on $\mathcal{Q}_{\text{test}}$ using Eq. (3).

2: Construct $\mathcal{R}_{\text{old}}$ through T pseudo-onboarding trials, from the predicted gain and realized gain using Eq. (4).

3: Compute s_{hist} from $\widehat{g}_{\text{new}}$ and $\mathcal{R}_{\text{old}}$ using Eq. (5).

▷ *Self-simulated evidence*

4: Count (r, c) through J-fold cross-fitting on $S_{m^\star,k}$, keeping old-model scores fixed.

5: Compute s_{cand} from (r, c) using Eq. (6).

▷ *Admission decision*

6: Combine s_{hist} and s_{cand} into s using Eq. (7).

Output: Deployed router π^{adm} selected using Eq. (8).

predicted gains using empirical residuals from pseudo-onboarding trials over existing models. Given an unlabeled target query batch $\mathcal{Q}_{\text{test}} \sim P_X$, the predicted routing gain of π^{new} over π^{old} is:

$$\widehat{g}_{\text{new}} = \frac{1}{|\mathcal{Q}_{\text{test}}|} \sum_{x\in\mathcal{Q}_{\text{test}}} \left[h_\phi(x, m^\star; S_{m^\star,k}) - \max_{m\in\mathcal{M}_{\text{old}}} h_\phi(x, m; S_m) \right]_+ . \tag{3}$$

Here, $[u]_+ = \max(u, 0)$, so only queries rerouted to the candidate contribute to the predicted gain.

To estimate the bias in $\widehat{g}_{\text{new}}$, we simulate T pseudo-onboarding trials on historical queries $\mathcal{Q}_{\text{train}}$. In each trial t, a model $m \in \mathcal{M}_{\text{old}}$ acts as a pseudo-candidate, while π^{old}_{-m} routes over the remaining pool $\mathcal{M}_{\text{old}} \setminus \{m\}$ using the unchanged historical scores h_ϕ. We sample a support set $S^t_{m,k} \subseteq S_m$ of size k and apply the same onboarding procedure used for the actual candidate, obtaining correctness estimates $h_\phi(x, m; S^t_{m,k})$. Using these estimates, we compute the predicted gain $\widehat{g}^t$ on $\mathcal{Q}_{\text{train}}$ following Eq. (3), with the maximum taken over $\mathcal{M}_{\text{old}} \setminus \{m\}$. The corresponding realized gain is

$$\delta^t = \frac{1}{|\mathcal{Q}_{\text{train}}|} \sum_{x\in\mathcal{Q}_{\text{train}}} \mathbf{1}\left\{ h_\phi(x, m; S^t_{m,k}) > \max_{m'\in\mathcal{M}_{\text{old}}\setminus\{m\}} h_\phi(x, m'; S_{m'}) \right\} \left[y_m(x) - y_{\pi^{\text{old}}_{-m}(x)}(x) \right] . \tag{4}$$

The corresponding estimation residual is $e^t = \delta^t - \widehat{g}^t$, yielding the historical residual bank $\mathcal{R}_{\text{old}} = \{e^t\}_{t=1}^T$. Finally, the pool-history evidence score s_{hist} measures the empirical probability that the bias-adjusted gain exceeds the gain threshold $\epsilon \geq 0$:

$$s_{\text{hist}} = \frac{1}{|\mathcal{R}_{\text{old}}|} \sum_{e^t\in\mathcal{R}_{\text{old}}} \mathbf{1}\{\widehat{g}_{\text{new}} + e^t > \epsilon\} . \tag{5}$$

3.2 Self-Simulated Evidence

Pool-history evidence estimates prediction bias across existing models, but may not fully capture that of candidate $m^\star$. We thus use candidate observations $S_{m^\star,k}$ to construct self-simulated evidence, estimating whether rerouting queries to $m^\star$ improves accuracy over the original router.

To evaluate candidate predictions on held-out observations, for $k > 1$, we partition $S_{m^\star,k}$ into $F = \min(J, k)$ folds $F_1, \ldots, F_F$ for cross-fitting (Chernozhukov et al., 2018). For each fold j, we apply the same onboarding procedure using $S_{m^\star,k} \setminus F_j$, while keeping the old-model predictions fixed. For a query x_i held out in fold j, we simulate rerouting it to $m^\star$ whenever $h_\phi(x_i, m^\star; S_{m^\star,k} \setminus F_j) > \max_{m\in\mathcal{M}_{\text{old}}} h_\phi(x_i, m; S_m)$. Across all folds, we count *beneficial reroutings* r, where the candidate answers correctly and the original router answers incorrectly, and *harmful reroutings* c, where the reverse holds. Queries with no rerouting or no change in correctness contribute to neither count.

Let θ denote the probability that a rerouting is beneficial, conditional on being either beneficial or harmful. To account for sparse counts, we model the beneficial rerouting count r among the $r + c$ discordant outcomes using a binomial likelihood, with a Jeffreys prior $\theta \sim \text{Beta}(1/2, 1/2)$ on the rescue probability (Jeffreys, 1946; Gelman et al., 1995). Hence, the self-simulated evidence score is the posterior probability that beneficial reroutings are more likely than harmful ones:

$$s_{\text{cand}} = \Pr\left(\theta > \tfrac{1}{2} \mid r, c\right) = 1 - I_{1/2}\left(r + \tfrac{1}{2}, c + \tfrac{1}{2}\right) , \tag{6}$$

where $I_x(a, b)$ is the regularized incomplete beta function. A score $s_{\text{cand}} > 1/2$ supports admission, while $r = c$ yields neutral evidence ($s_{\text{cand}} = 1/2$).

3.3 Admission Decision

Inspired by Bayesian log-odds evidence aggregation (Hinton, 2002; Friedman et al., 2000), E^2-Filter integrates s_{hist} and s_{cand} into an admission score s:

$$s = \text{logit}(s_{\text{hist}}) + \lambda \, \text{logit}(s_{\text{cand}}), \tag{7}$$

where $\text{logit}(u) = \log(u/(1-u))$.[1] The hyperparameter $\lambda > 0$ controls the relative contribution of self-simulated evidence. Given an admission threshold $\tau \in (0, 1)$, E^2-Filter resolves the admission problem by assigning the deployed router π^{adm} via:

$$\pi^{\text{adm}} = \begin{cases} \pi^{\text{new}}, & \text{if } s \geq \text{logit}(\tau), \\ \pi^{\text{old}}, & \text{otherwise.} \end{cases} \tag{8}$$

4 Experiments

We evaluate E^2-Filter on new-model admission across three benchmarks, addressing three questions: **(RQ1)** How does it compare with admission baselines across benchmarks, routing methods, and candidate observation budgets? **(RQ2)** Does it reject harmful updates while retaining beneficial ones? **(RQ3)** How do its evidence components and hyperparameters affect accuracy gains?

4.1 Experimental setup

Benchmarks and routing methods. We evaluate E^2-Filter on three benchmarks: RouterBench (Hu et al., 2024), LLMRouterBench (Li et al., 2026), and EmbedLLM (Zhuang et al., 2025) (details in App. B.1). We combine pool sizes $|\mathcal{M}_{\text{old}}| \in \{5, 9\}$ with four routing methods: kNN (Hu et al., 2024; Li, 2025), UniRoute-Cluster (clustering-based mapping), UniRoute-LearnedMap (parameterized mapping network) (Jitkrittum et al., 2026), and LOCUS (Patel et al., 2026) (details in App. B.2).

Evaluation protocol and implementation. For each benchmark, we evaluate every model as a candidate $m^\star$ and randomly sample old-model pools of the specified sizes from the remaining models. The original router uses correctness observations from all old models on 1,024 shared training queries and remains fixed across candidate support sizes. All admission methods compare the same original and updated routers using identical support sets, with results averaged over 10 independent support draws (details in App. B.3). We use official open-source implementations and a fixed E^2-Filter configuration across settings. Implementation details and code references are provided in App. B.5.

Baselines. We compare E^2-Filter with two deployable baselines and two hindsight references: (1) Always Admit, which always deploys the updated router. (2) Predicted-Gain Admit admits the candidate when its mean predicted score advantage over the best old model is positive, retaining negative margins. (3) Budget Threshold rejects for $k < k_0$ and admits otherwise, with k_0 chosen in hindsight to maximize AUDG within each setting. (4) Admission Oracle, which selects the router with higher test accuracy for each evaluation. The latter two serve as post-hoc offline references rather than deployable online methods. Details are provided in App. B.4.

Evaluation metrics. We evaluate admission decisions across candidate observation sizes k, since results at a single observation count may give an incomplete comparison (Fig. 2). We therefore propose *Area Under Deployment Gain (AUDG)* as our primary metric, summarizing accuracy gains over the original router as the normalized signed area along the logarithmic support-size axis:

$$\text{AUDG} = \frac{1}{\log_2 k_2 - \log_2 k_1} \int_{\log_2 k_1}^{\log_2 k_2} \left(\text{Acc}(\pi^{\text{adm}}) - \text{Acc}(\pi^{\text{old}})\right) \, \text{d}(\log_2 k). \tag{9}$$

where k_1 and k_2 denote the smallest and largest evaluated observation sizes and $\text{Acc}(\cdot)$ denotes test accuracy. We compute the area using trapezoidal integration over the evaluated $\log_2 k$ values and report AUDG in percentage points. Positive and negative accuracy changes contribute with their respective signs, while always retaining the original router contributes zero.

[1] Inputs are clipped away from 0 and 1 for numerical stability. At $k = 1$, admission uses only the raw pool-history score.

Table 1: AUDG (percentage points) results on model admission across three benchmarks and four routing methods. **Bold** indicates the best deployable admission policy within each setting. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

		Old Model Pool $\lvert\mathcal{M}_{\text{old}}\rvert = 5$					*Old Model Pool* $\lvert\mathcal{M}_{\text{old}}\rvert = 9$				
Benchmark	Routing Method	Always Admit	Predicted-Gain Admit	E^2-Filter **(Ours)**	Budget Threshold†	Admission Oracle†	Always Admit	Predicted-Gain Admit	E^2-Filter **(Ours)**	Budget Threshold †	Admission Oracle†
RouterBench	kNN	−0.403	−0.125	**+0.380**	+0.387	+0.945	−0.604	−0.187	**+0.365**	+0.236	+0.750
	UniRoute-Cluster	−1.122	−0.351	**+0.312**	+0.266	+0.833	−1.145	−0.563	**+0.243**	+0.236	+0.547
	UniRoute-LearnedMap	−0.762	−0.250	**+0.390**	+0.257	+0.875	−0.687	−0.507	**+0.263**	+0.154	+0.533
	LOCUS	−0.939	−0.427	**-0.015**	+0.000	+0.173	−0.469	−0.284	**+0.016**	+0.029	+0.130
LLMRouterBench	kNN	−1.028	−0.707	**+0.248**	+0.183	+0.514	−1.252	−1.017	**+0.072**	+0.063	+0.194
	UniRoute-Cluster	−1.644	−0.675	**+0.234**	+0.116	+0.494	−2.163	−0.999	**+0.047**	+0.015	+0.152
	UniRoute-LearnedMap	−1.581	−0.749	**+0.134**	+0.047	+0.422	−1.885	−1.026	**+0.012**	+0.004	+0.106
	LOCUS	−0.644	−0.011	**+0.015**	+0.000	+0.173	−0.494	−0.010	**+0.003**	+0.000	+0.057
EmbedLLM	kNN	−0.126	+0.089	**+0.342**	+0.232	+0.718	−0.616	−0.377	**+0.051**	+0.053	+0.341
	UniRoute-Cluster	−1.050	−0.175	**+0.308**	+0.134	+0.697	−1.826	−0.745	**+0.024**	+0.032	+0.291
	UniRoute-LearnedMap	−0.802	−0.191	**+0.269**	+0.121	+0.674	−1.430	−0.685	**+0.016**	+0.039	+0.292
	LOCUS	−0.456	−0.267	**+0.115**	+0.129	+0.452	−0.838	−0.537	**+0.037**	+0.043	+0.214
Average		−0.880	−0.320	**+0.228**	+0.156	+0.581	−1.118	−0.578	**+0.096**	+0.075	+0.301

Selected configuration: $(\epsilon, \lambda, \tau, J) = (0.0025, 1, 0.975, 5)$, chosen by the highest mean AUDG among 270 configurations on this evaluation set; this is post-hoc parameter selection.

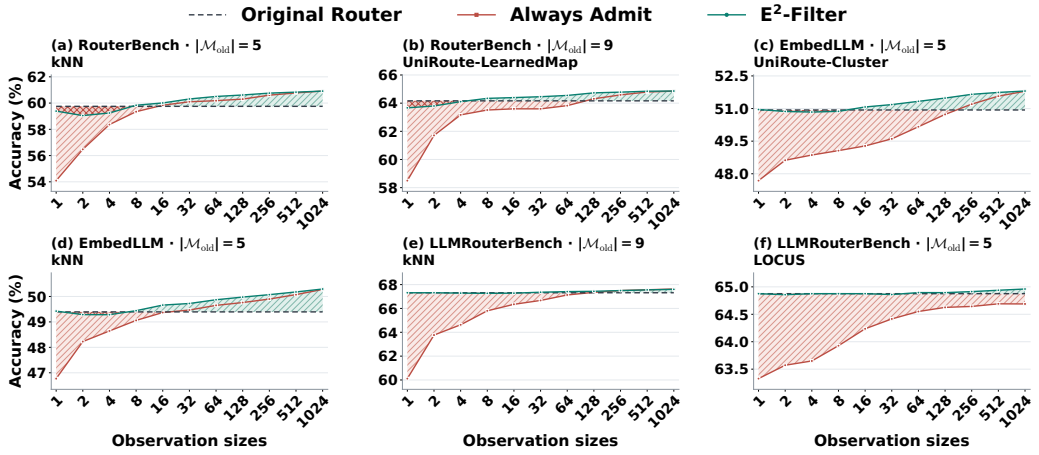


Figure 2: Routing accuracy versus candidate support size in six selected settings. We compare E^2-Filter, Always Admit, and the original router. Hatched regions indicate accuracy gains and losses relative to the original router.

4.2 Results on Multi-Benchmark New-Model Onboarding

Overall comparison. Tab. 1 shows that E^2-Filter achieves the highest AUDG among deployable admission methods across all 24 settings, spanning three benchmarks, four routing methods, and two pool sizes, consistently outperforming Always Admit and Predicted-Gain Admit. E^2-Filter also outperforms the hindsight Budget Threshold reference in most settings and on average for both pool sizes, as evidence from each candidate and old-model pool helps distinguish beneficial from harmful updates at the same support size. Fig. 2 illustrates the corresponding accuracy curves in six selected settings. At small observation sizes ($k \leq 8$), Always Admit reduces accuracy, while E^2-Filter keeps accuracy close to the original router by filtering harmful updates. As observations increase, E^2-Filter captures positive gains, including in the LOCUS setting (Fig. 2(f)), where Always Admit remains below the original router. These curves illustrate how limiting losses at small support sizes while retaining gains at larger sizes contributes to its higher AUDG. Complete accuracy tables and curves for all settings are provided in App. C.

Generalization and robustness.

Across benchmarks. E^2-Filter maintains positive aggregate gains on RouterBench, LLMRouterBench, and EmbedLLM. The improvements therefore extend across multiple benchmarks rather than being concentrated in a single evaluation suite. Absolute gains vary across benchmarks, reflecting both the

improvement opportunities offered by the candidate and the admission method's ability to identify beneficial updates. Admission Oracle provides a reference for interpreting these differences by measuring the gains available from choosing between the same original and updated router.

Across old-model pool sizes. E^2-Filter's gains decrease when the old-model pool expands from five to nine models. This pattern is consistent with less room for accuracy improvement from adding a single candidate to the larger pool. The comparison concerns incremental gains over the original router, so smaller AUDG values do not imply lower absolute routing accuracy. Despite the compressed oracle range, E^2-Filter continues to capture a substantial fraction of the remaining improvement margin. To assess admission quality across pool sizes, these gains should be considered alongside the corresponding oracle bounds and the remaining gaps to them.

Across routing methods. E^2-Filter obtains positive gains on average with kNN, UniRoute-Cluster, UniRoute-LearnedMap, and LOCUS, supporting its applicability across different dynamic routing methods. For kNN and UniRoute, Admission Oracle indicates greater potential gains, and E^2-Filter captures these opportunities. For LOCUS, the smaller oracle gain indicates less room for improvement through admission decisions, consistent with the smaller absolute gains achieved by E^2-Filter.

Evaluation with inference costs. As discussed in Sec. 2, inference cost can be incorporated into the routing objective through a Lagrangian penalty. Following Jitkrittum et al. (2026), we further evaluate accuracy–cost performance on EmbedLLM and RouterBench using the Area under the accuracy–cost curve. Cost-aware AUDG integrates the Area improvement over the original router across $\log_2 k$, with the same normalization as accuracy-based AUDG. We adapt E^2-Filter by recomputing its pool-history evidence from predicted Area gains and historical Area residuals, while retaining the original self-simulated evidence. Under this setup, E^2-Filter consistently outperforms admission baselines, highlighting its applicability when inference cost is considered alongside accuracy. The adaptation details, experiment settings, and complete results are deferred to App. D.

4.3 Admission Selectivity

The overall performance gains in Table 1 summarize each admission method's accuracy gains across observation sizes but do not explain *which* new model updates drive these gains. We therefore examine this question through the selection behavior of E^2-Filter across observation sizes, followed by comparisons at matched admission rates.

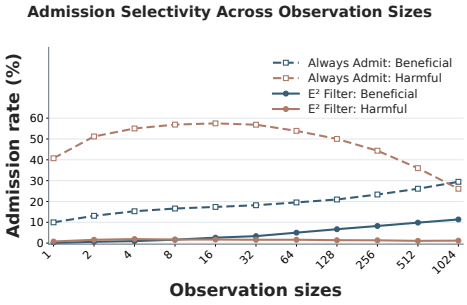


Figure 3: Admission selectivity across budgets. Dashed curves show beneficial and harmful deployments under Always Admit; solid curves show those admitted by E^2-Filter. Additional observations increase beneficial admissions while harmful admissions remain low.

Selectivity dynamics across support sizes. We classify a new model update as *beneficial* if the updated router achieves higher test accuracy than the original router, and as *harmful* if it achieves lower accuracy. Fig. 3 reports the proportions of all evaluations in which each method admits beneficial or harmful updates, including neutral updates in the denominator. Always Admit deploys harmful updates in more than 50% of evaluations at several small-to-intermediate observation sizes. In contrast, E^2-Filter keeps harmful admissions below 2% throughout the evaluated range. As the observation size increases, its beneficial admission rate rises from approximately zero to 11%, while its harmful admission rate remains low. These results suggest that additional observations allow E^2-Filter to admit more beneficial updates without a corresponding increase in harmful admissions.

Understanding the oracle gap. Despite its low harmful admission rate, E^2-Filter still rejects some beneficial updates, even at the largest observation sizes ($k = 1024$). These missed opportunities contribute to its gap to the hindsight Admission Oracle, which uses test outcomes to choose between the original and updated routers. The gap reflects both beneficial gains forgone through rejection and accuracy losses from admitted harmful updates. The selectivity curves show that E^2-Filter limits the

Table 2: AUDG (percentage points) and admission rates (%) at matched overall admission rates. All controls match E^2-Filter's candidate-balanced admission rate within each setting, support draw, and observation size, yielding a common integrated overall admission rate of 5.98%. Rates use all evaluations as the denominator. Metrics are integrated over $\log_2 k$ and averaged equally across 24 settings. Random selection and ranking ties are evaluated in expectation. Bold indicates the best result for each metric.

	Random	Pool-History Ranking	Self-Simulated Ranking	E^2-Filter
AUDG ↑	-0.015	$+0.138$	$+0.150$	**+0.162**
Beneficial Admission (%) ↑	1.89	4.22	4.13	**4.47**
Harmful Admission (%) ↓	3.71	1.73	1.81	**1.50**

Table 3: Ablation of pool-history evidence, self-simulated evidence, and their fusion across weights λ. AUDG is reported in percentage points and admission rates in percent, using all evaluations as the denominator. Metrics are integrated over $\log_2 k$ and averaged equally across 24 settings. The shaded row marks the selected configuration. Bold indicates the best value in each column with a directional arrow.

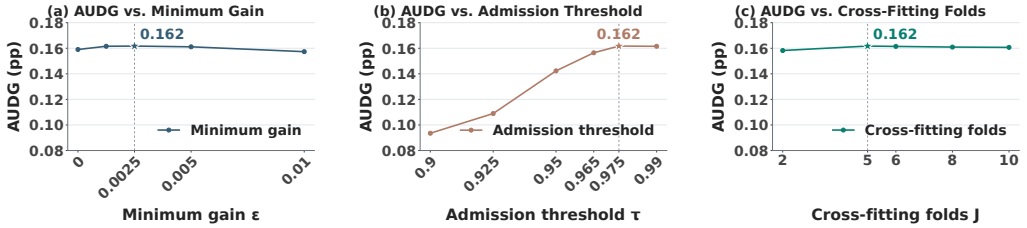
Evidence Configuration	AUDG ↑	Overall Admission (%)	Beneficial Admission (%) ↑	Harmful Admission (%) ↓
Pool-history evidence only	$+0.130$	3.94	3.06	0.87
Self-simulated evidence only	$+0.129$	2.32	2.15	**0.16**
$\lambda = 0.25$	$+0.143$	4.25	3.34	0.89
$\lambda = 0.5$	$+0.155$	4.83	3.76	1.05
$\lambda = 1$	**+0.162**	5.98	4.47	1.50
$\lambda = 2$	$+0.120$	8.74	5.65	3.04
$\lambda = 4$	$+0.092$	11.91	**6.76**	5.03

frequency of harmful admissions while leaving some beneficial updates unadmitted; the contribution of each to the accuracy gap also depends on the magnitude of the corresponding gains or losses.

Selection quality at matched admission rates. To further assess whether E^2-Filter's performance stems from deployment selection beyond controlling the overall admission rate, we compare it against random admission and single-evidence rankings, matching its candidate-balanced admission rate within each setting, support draw, and observation size. The results are shown in Table 2. Uniform random selection yields an AUDG near zero (-0.015 percentage points), whereas E^2-Filter achieves a positive AUDG of $+0.162$ percentage points. At matched admission rates, E^2-Filter achieves more than twice the beneficial admission rate while reducing the harmful admission rate to approximately 40% of that under random selection. This comparison shows that E^2-Filter's deployment gains reflect which updates it admits, beyond controlling the overall admission rate. Furthermore, ranking candidate updates solely by either pool-history or self-simulated evidence also produces positive AUDG, indicating that both sources carry useful signals for admission decisions. Combining these two evidence sources in E^2-Filter further improves AUDG while increasing beneficial admissions and reducing harmful admissions relative to either single-evidence ranking, supporting the value of dual-evidence combination in new model admission.

4.4 Ablation Study and Hyperparameter Analysis

Ablation of admission evidence. To test the contribution of each evidence source, we evaluate pool-history evidence and self-simulated evidence separately, and compare their combination under different weights λ. The results are shown in Table 3. Both evidence sources independently yield positive AUDG, with self-simulated evidence offering safer, more conservative filtering and pool-history evidence capturing broader update opportunities. Combining the two evidence sources with equal weights achieves the highest aggregate AUDG among the evaluated configurations, supporting their complementary roles in admission decisions. The weight λ controls the relative influence of self-simulated evidence on the admission score. While increasing λ boosts beneficial admissions, it also allows a higher proportion of harmful updates to pass through, causing AUDG to drop sharply at higher weights. This indicates that higher admission rates do not automatically translate into performance gains when harmful deployments go unconstrained. Comparing the top choices, $\lambda = 0.5$



(a) Minimum gain threshold ϵ. (b) Admission threshold τ. (c) Number of cross-fitting folds J.

Figure 4: Hyperparameter sensitivity of E^2-Filter measured by AUDG (percentage points)). Stars and dashed vertical lines mark the default values.

provides a conservative alternative with lower harmful admissions, whereas the selected $\lambda = 1$ achieves the highest aggregate AUDG among the displayed configurations.

Hyperparameter sensitivity. We systematically evaluate the sensitivity of E^2-Filter with respect to the minimum gain threshold ϵ, the admission threshold τ, and the number of cross-fitting folds J. First, AUDG varies modestly across the evaluated ϵ values, with $\epsilon = 0.0025$ achieving the peak AUDG of 0.162 pp (Fig. 4(a)). Second, AUDG increases with τ before plateauing near $\tau \in [0.975, 0.99]$ (Fig. 4(b)). Finally, varying the fold count J yields minimal performance fluctuations for $J \geq 5$ (Fig. 4(c)), suggesting limited additional benefit from more than five folds. The selected configuration uses $\epsilon = 0.0025$, $\tau = 0.975$, and $J = 5$.

5 Discussion, Future Work, and Conclusion

Active sampling and improved estimation. An admission method inherently relies on candidate observations to evaluate proposed routing updates. Consequently, if insufficient observation quantity or quality causes the methods to underestimate the updated router, achievable gains remain uncaptured—even when the new model is genuinely capable of producing correct answers. To overcome this limitation, future work could draw on active testing to deliberately sample query candidates that are maximally informative about potential routing gains (Kossen et al., 2021; Berrada et al., 2025). Combined with improved correctness predictors and prediction-assisted estimation frameworks (Kossen et al., 2022; Angelopoulos et al., 2023), such active sampling strategies can simultaneously refine the candidate router and provide richer, unbiased evidence for admission, ultimately preventing the systemic under-admission of capable models under adaptive sampling.

More realistic onboarding benchmarks. While holding out models from an existing pool enables controlled comparisons of admission rules, it fails to simulate the temporal arrival of models with novel capabilities. Indeed, some candidates offer minimal accuracy gains over a strong baseline pool, leaving virtually no headroom for beneficial admission even under oracle decisions (App. E). Future benchmarks should consider chronologically ordered model arrivals and explicitly quantify candidate-pool complementarity, addressing both redundant additions and models covering existing router failure modes. To maintain a focused analysis on routing updates, we intentionally scope this study to single-candidate onboarding at a time, directly reflecting the primary deployment paradigm where model candidates are evaluated sequentially. We leave joint, multi-candidate admission as future work, as multi-model interactions introduce non-trivial combinatorial evaluation challenges.

Conclusion. In summary, our findings highlight that expanding a model pool does not guarantee superior routing performance, making selective admission a consequential gatekeeper when candidate observations are limited. To navigate this uncertainty, we introduce E^2-Filter, a novel method that combines historical pool-onboarding experience with self-simulated candidate evidence—all without incurring extra evaluation costs. Extensive evaluations across three benchmarks, four routing predictors, and multiple pool scales demonstrate that E^2-Filter consistently outperforms unconditional admission in budget-integrated deployment gains. Ultimately, our work establishes selective admission as a practical and essential complement to standard router construction, providing a unified mechanism to retain beneficial model updates while preventing harmful deployments.

AI USE STATEMENT

In accordance with the ICLR Author Guidelines, we disclose that ChatGPT was utilized as an auxiliary tool during the preparation of this work. Specifically, AI assistance was used to help draft initial prose, polish academic language, write implementation and utility code, and refine script syntax for plotting experimental figures. All core scientific conceptualizations, mathematical derivations, algorithm designs, experimental logic, and final manuscript revisions were independently executed, manually verified, and approved by the authors, who maintain full responsibility for the integrity and accuracy of this work.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. AutoMix: Automatically mixing language models. In *NeurIPS*, 2024.

Anastasios N Angelopoulos, Stephen Bates, Clara Fannjiang, Michael I Jordan, and Tijana Zrnic. Prediction-powered inference. *Science*, 2023.

Gabrielle Berrada, Jannik Kossen, Freddie Bickford Smith, Muhammed Razzak, Yarin Gal, and Thomas Rainforth. Scaling up active testing to large language models. In *NeurIPS*, 2025.

Amit Singh Bhatti, Vishal Vaddina, and Dagnachew Birru. Proteus: Sla-aware routing via lagrangian rl for multi-llm serving systems. In *EuroMLSys*, 2026.

Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *JMLR*, 2010.

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

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 *NeurIPS*, 2024.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters, 2018.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for llms. *arXiv preprint arXiv:2410.10347*, 2024.

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 *ICLR*, 2024.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A graph-based router for LLM selections. In *ICLR*, 2025.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 2000.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. chapman and hall/cRc, 1995.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002.

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.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. *arXiv preprint arXiv:2503.10657*, 2025.

Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and physical sciences*, 1946.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In *ACL*, 2023.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, et al. Universal model routing for efficient llm inference. In *ICLR*, 2026.

Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active testing: Sample-efficient model evaluation. In *ICML*, 2021.

Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Thomas Rainforth. Active surrogate estimators: An active learning approach to label-efficient model evaluation. In *NeurIPS*, 2022.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. LLMRouterBench: A massive benchmark and unified framework for LLM routing. *arXiv preprint arXiv:2601.07206*, 2026.

Shaoang Li and Jian Li. Near-optimal online deployment and routing for streaming llms. In *ICLR*, 2026.

Yang Li. Rethinking predictive modeling for llm routing: When simple knn beats complex learned routers. *arXiv preprint arXiv:2505.12601*, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *ICLR*, 2025.

Shivam Patel, William Cocke, and Gauri Joshi. LOCUS: Low-dimensional model embeddings for efficient model exploration, comparison, and selection. *arXiv preprint arXiv:2601.21082*, 2026.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. *arXiv preprint arXiv:2402.14992*, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.

Marija Šakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *WSDM*, 2024.

Reza Shirkavand, Shangqian Gao, Peiran Yu, and Heng Huang. Cost-aware contrastive routing for llms. In *NeurIPS*, 2025.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

James E Smith and Robert L Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 2006.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. Carrot: A cost aware rate optimal router. *arXiv preprint arXiv:2502.03261*, 2025.

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 *ACL*, 2025.

Anchen Sun and Kaiqi Yang. Routeguard: Certifying routing gain in llm multi-agent systems when complementarity is not enough. *arXiv preprint arXiv:2608.07583*, 2026.

Clovis Varangot-Reille, Christophe Bouvard, and Antoine Gourru. Generalising LLM routing using past performance retrieval: A few-shot router is sufficient. In *EACL Student Research Workshop*, 2026.

Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. Anchor points: Benchmarking models with much fewer examples. In *EACL*, 2024.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Qiaosheng Zhang, and Shuyue Hu. ICL-router: In-context learned model representations for llm routing. In *AAAI*, 2026a.

Zijie Wang, Xinyu Yan, Che Wang, Zeng Zihao, Lei Xiao, and Wei Yang Bryan Lim. SEMIROUTER: Sparse-data enhanced routing for adaptive multi-LLM system. In *EACL*, 2026b.

Cheng Yan, Wuyang Zhang, Zhiyuan Ning, Fan Xu, Ziyang Tao, Lu Zhang, Bing Yin, and Yanyong Zhang. Breaking model lock-in: Cost-efficient zero-shot llm routing via a universal latent space. In *AAAI*, 2026.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. Arctic-embed 2.0: Multilingual retrieval without compromise. *arXiv preprint arXiv:2412.04506*, 2024.

Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Capability instruction tuning. In *AAAI*, 2025.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models. In *ICLR*, 2025.

A Related Work

LLM routing over static model pools. LLM routing selects a model for each query based on query-dependent model strengths, exploiting complementary capabilities to improve response quality and, in cost-aware settings, reduce inference cost (Shnitzer et al., 2023; Šakota et al., 2024). Hybrid LLM trains a query encoder to estimate query difficulty relative to a small–large model pair and routes queries between the two models accordingly (Ding et al., 2024). RouteLLM learns strong–weak model selection from the training data, including similarity-weighted ranking, matrix-factorization, and classifier-based routers (Ong et al., 2025). For multi-model routing, RouterDC (Chen et al., 2024) learns query and model representations through dual contrastive learning, while EmbedLLM (Zhuang et al., 2025) combines learned model representations with query representations to predict the probability that each model answers a given query correctly. IRT-Router instead models performance through model abilities and query difficulty and discrimination (Song et al., 2025). Overall, these methods focus on learning query–model relationships to support effective model selection within an established model pool.

LLM routing over dynamic model pools. In more realistic deployment scenarios, a router trained on the old model pool may need to incorporate newly released LLMs (Huang et al., 2025). A common strategy leverages knowledge from the existing router, together with observations from the new model, to predict its correctness on incoming queries. Some methods incorporate these observations through model-specific parameter updates. ZeroRouter (Yan et al., 2026) estimates a new model's latent abilities by optimizing against anchor-query outcomes, while SemiRouter (Wang et al., 2026b) trains a lightweight adapter with the existing router backbone frozen. Other methods accommodate new models without updating router parameters during onboarding. GraphRouter represents tasks, queries, and models in a heterogeneous graph and uses inductive edge prediction to support newly introduced models with limited interaction data (Feng et al., 2025). Model-SAT constructs model capability representations from aptitude-test results and conditions performance predictions on these representations and the input query (Zhang et al., 2025). UniRoute aggregates observations over query clusters defined by fixed clustering or a learned query-to-cluster mapping (Jitkrittum et al., 2026). LOCUS uses a Set Transformer to encode query–correctness pairs into a model representation, which is combined with the target query representation for correctness prediction (Patel et al., 2026). ICL-Router formulates correctness prediction as an in-context learning task, using an LLM conditioned on model-specific query–correctness demonstrations (Wang et al., 2026a). Retrieval-based methods estimate correctness from similar historical queries through kNN averaging or learned reweighting, as in ContextualRouter (Varangot-Reille et al., 2026).

Model admission and deployment in LLM routing.

Deploying an LLM routing system involves assessing the benefit of routing and deciding which models to include. RouteGuard estimates the benefit of routing by certifying a prespecified router's correctness gain over a fixed primary model (Sun & Yang, 2026). For evolving model pools, StageRoute jointly optimizes stage-wise model deployment and online routing under streaming arrivals and operational constraints (Li & Li, 2026). Our work focuses on whether a routing policy updated with a new candidate should replace a fixed incumbent policy, given only limited candidate observations. We assess the incremental routing gain of candidate inclusion using existing historical data and candidate support observations, without requiring additional model evaluations.

B Experimental and Implementation Details

B.1 Benchmarks

We evaluate E^2-Filter on three benchmarks, summarized in Table 4. Query splits and the pool of 1,024 training anchors are fixed across admission methods.

RouterBench. We retain the supplied query splits and recorded scores (Hu et al., 2024). For historical realized gains, self-simulated correctness counts, and test accuracy, source scores equal to 1 are mapped to binary correctness 1, and all other scores are mapped to 0. Candidate profiling uses fractional observations for all methods at $M = 5$ and for kNN and LOCUS at $M = 9$, while the UniRoute variants at $M = 9$ use binary observations. Both encodings are derived from the same evaluated responses and require no additional candidate evaluations.

Table 4: Benchmark statistics and onboarding episodes. Episode counts are reported per routing method and pool size, with three old-model pools per candidate. The 1,024 training anchors are included in the training split.

Dataset	Models	Train	Val	Test	Episodes
RouterBench	11	29,861	3,318	3,318	33
LLMRouterBench	20	1,024	5,224	5,232	60
EmbedLLM	112	29,673	3,000	3,000	336

LLMRouterBench. We use the performance-oriented pool of 20 models and a correctness subset drawn from 15 tasks covering reasoning, coding, mathematics, science, medicine, finance, and emotion classification (Li et al., 2026). We retain queries with matching content and observed binary outcomes across all models, removing incomplete records rather than imputing missing outcomes. We construct task-stratified splits by allocating 1,024 training queries approximately proportionally to task size and dividing the remaining queries approximately equally between validation and test sets within each task. All 1,024 training queries serve as anchors.

EmbedLLM. We retain the supplied binary outcomes, query splits, embeddings, and training-anchor pool for all 112 models (Zhuang et al., 2025). The benchmark covers knowledge, mathematics, medicine, commonsense, and reasoning tasks. Every model is evaluated as a candidate.

B.2 Routing Methods

We evaluate E^2-Filter with four routing methods: kNN, UniRoute-Cluster, UniRoute-LearnedMap, and LOCUS. All methods incorporate candidate support observations without updating predictor parameters during onboarding.

kNN. We retrieve neighboring queries by cosine similarity between `snowflake-arctic-embed-m-v2.0` (Yu et al., 2024) embeddings. Old-model predictions average observed scores over the 40 nearest training anchors, excluding the query itself during anchor-to-anchor retrieval. Candidate predictions average observed scores over the $\min(40, k)$ nearest support queries.

UniRoute variants. Both UniRoute variants (Jitkrittum et al., 2026) use cluster-based performance profiles and query embeddings from the Sentence Transformers (Reimers & Gurevych, 2019) checkpoint `all-mpnet-base-v2`. During onboarding, each observed cluster receives the mean candidate outcome over its support queries, while unobserved clusters receive the mean over all k support observations. UniRoute-Cluster predicts correctness using the profile value of the query's assigned cluster, whereas UniRoute-LearnedMap computes a weighted profile average through a learned query-to-cluster mapping.

We construct clusters using K-means++ with 10 initializations and at most 300 iterations. The learned mapping consists of input batch normalization, two 128-unit hidden layers with batch normalization and ReLU activations, and a K-way softmax output. It is trained on old-model observations using binary cross-entropy and Adam, with a learning rate of 0.005, a batch size of 64, and five epochs. For each variant, the cluster count $K \in \{3, \ldots, \lfloor N_{\text{val}}/50 \rfloor\}$ is selected to maximize old-model routing accuracy on the validation set, with ties favoring smaller K, where N_{val} is the number of validation queries. The selected clusters, mapping, and old-model profiles remain fixed during onboarding.

LOCUS. LOCUS (Patel et al., 2026) uses a Set Transformer encoder to map observed query–outcome pairs into a candidate embedding, which a decoder combines with MPNet query features to predict correctness. The encoder has two layers, four attention heads, 64 inducing points, and a latent dimension of 128. The MLP decoder has hidden width 64, and both modules use dropout of 0.1. Query features are the same 768-dimensional MPNet embeddings used by UniRoute.

We train LOCUS using binary cross-entropy and AdamW, with encoder and decoder learning rates of 2×10^{-4} and 8×10^{-4}, respectively, weight decay of 0.05, and a model batch size of 128. Training runs for at most 2,500 epochs, with early stopping after 500 epochs without improvement in validation routing accuracy; the best validation checkpoint is retained. Gaussian noise is applied to encoder and decoder query features, with its standard deviation decreasing from 0.10 to 0.05 according to a

cosine schedule over the first 500 epochs. The candidate is excluded from the model pools used for checkpoint training and validation. During onboarding, the frozen encoder constructs its embedding from the available support observations, leaving predictor parameters and old-model embeddings unchanged.

B.3 Experimental Setup

Onboarding episodes and model pools. For each candidate model $m^\star$, we randomly sample three old-model pools at each size $M \in \{5, 9\}$ from the remaining models, independently of candidate evaluation outcomes. These pools are shared across all four routing methods. Within each episode and routing method, all admission rules compare the same original router π^{old} and updated router π_k^{new}, constructed using identical candidate support observations. Exact prediction ties are resolved using a fixed model order with the candidate placed last, so ties between the candidate and an old model favor the old model.

Support sampling. We evaluate the observation budgets $\mathcal{K} = \{2^j : j = 0, \ldots, 10\}$ using ten randomly shuffled support orders. Support sets are nested prefixes of the shuffled 1,024-query training-anchor pool. Deterministic seeds derived from the episode identity and draw index ensure that support queries are shared across routing methods. Admission is assessed separately at each budget, without adaptive sampling or outcome-dependent stopping. At $k = 1024$, all draws contain the full anchor set, although their support orders may produce different cross-fitting folds. The combination of three benchmarks, four routing methods, and two old-model pool sizes yields 24 settings.

Information access. Routing predictors use old-model observations on training queries, with configurations and checkpoints selected using old-model validation outcomes. Deployable admission methods access only historical old-model observations, current candidate support observations, and unlabeled test query inputs. The protocol is therefore transductive with respect to query inputs. Method development included exploratory experiments on these benchmarks; the separation of decision inputs and evaluation labels applies to individual evaluation runs.

Result aggregation. Deployment gain is measured as the test accuracy difference between the deployed router and the original router, using their complete routing predictions. Within each setting, we average gains over the three old-model pools and ten support draws for each candidate, then average candidates equally. Each test query receives equal weight, without task-level macro-averaging. AUDG is computed by trapezoidal integration over $\log_2 k$, normalized by the evaluated log-budget range of 10. Negative deployment gains are retained in this calculation. Pool-size averages weight their 12 settings equally, and the overall average weights all 24 settings equally.

B.4 Admission Baselines

We compare E^2-Filter with two deployable baselines and two hindsight references.

Always Admit. This baseline unconditionally deploys the updated router constructed from the candidate support observations.

Predicted-Gain Admit. This baseline admits the candidate when its mean predicted score advantage over the best old model is strictly positive on the unlabeled target queries. The mean retains negative margins, whereas the gain proxy within E^2-Filter uses their positive parts.

Budget Threshold. This reference rejects candidates for $k < k_0$ and admits them otherwise. For each dataset–routing method–pool size setting, we select $k_0 \in \mathcal{K} \cup \{\infty\}$ in hindsight to maximize test AUDG, with ties favoring larger cutoffs. The selected cutoff is shared across candidates, old-model pools, and support draws. The choice $k_0 = \infty$ corresponds to rejecting all candidates.

Admission Oracle. This reference selects between the original and updated routers using their test accuracies, admitting only when the measured gain exceeds the numerical tolerance of 10^{-12}. Both Budget Threshold and Admission Oracle use test outcomes and are excluded from comparisons among deployable methods.

B.5 Implementation Details

Implementation. We implement the experiments in Python, using PyTorch for neural routing predictors and NumPy and SciPy for admission computations. Admission evaluation runs on CPUs using saved model outcomes, query embeddings, and predictor checkpoints. Historical residuals are computed offline and reused across candidate support draws. Our anonymous code is available at https://anonymous.4open.science/r/E2-Filter.

Historical evidence. For each evaluation episode and observation size k, we construct the residual bank using only the current old-model pool $\mathcal{M}_{\text{old}}$ and its historical observations on $\mathcal{Q}_{\text{train}}$, following Sec. 3.1. Each model $m \in \mathcal{M}_{\text{old}}$ serves as a pseudo-candidate against the remaining models $\mathcal{M}_{\text{old}} \setminus \{m\}$. We sample size-$k$ support sets and apply the same onboarding procedure used for the actual candidate. The original routing predictor is retained without retraining, and the remaining old-model scores stay fixed. We aggregate the resulting differences between realized and predicted routing gains into the residual bank for that episode and observation size, reusing it across candidate support draws.

Self-simulated evidence. For $k > 1$, we evaluate candidate support observations through cross-fitting, following Sec. 3.2. Old-model scores remain fixed throughout this procedure. The resulting rescue and corruption counts determine the self-simulated evidence score. Cross-fitting reuses the same k candidate observations and requires no additional candidate-model evaluations. Accepted candidates are deployed through the router constructed from all k observations.

Hyperparameter settings. Unless otherwise specified, we use $\epsilon = 0.0025$, $\lambda = 1$, $\tau = 0.975$, and $J = 5$ across all 24 settings. With $J = 5$ fixed, this shared configuration was selected retrospectively from 270 combinations of ϵ, λ, and τ by maximizing mean AUDG across the 24 settings on the existing test results, without a separate validation set. Both evidence scores are clipped to $[10^{-4}, 1 - 10^{-4}]$ before applying the logit transformation. At $k = 1$, admission uses only the raw pool-history score, with a threshold of 0.90. At every budget, admission additionally requires a nonempty historical bank and at least one strict predicted candidate switch relative to the frozen old-model scores.

C Detailed Experimental Results across Observation Sizes

This appendix extends the main comparison with complete accuracy curves and tables for all 24 settings, showing how admission filtering affects deployment performance as candidate observations increase. We report accuracy (%) at observation sizes $k \in \{2^j : j = 0, \ldots, 10\}$, following the evaluation and aggregation protocol in App. B.3. Admission methods and hindsight references follow the definitions in App. B.4.

Overall results. The complete results demonstrate the effectiveness of E^2-Filter in limiting accuracy degradation while retaining gains from beneficial model additions. At $k = 1$, Always Admit reduces accuracy relative to the original router in all 24 settings, whereas E^2-Filter largely preserves the original accuracy and achieves the highest displayed accuracy among deployable admission methods in every setting, including ties. As more observations become available, E^2-Filter retains opportunities for improvement, exceeding the original router in all 24 settings at $k = 1024$. This includes three LOCUS settings where Always Admit remains below the original router even with the full observation budget. These results show that filtering remains useful both when observations are sparse and when unconditional admission continues to be harmful despite additional observations. By limiting losses while preserving beneficial updates, E^2-Filter achieves the highest AUDG among deployable admission methods across all 24 settings (Tab. 1). Although individual methods can attain slightly higher accuracy at particular observation sizes, E^2-Filter provides the strongest overall deployment gains across the evaluated range.

Presentation of results. Original Router denotes the fixed accuracy of π^{old}. In the figures, red hatching indicates losses from Always Admit relative to this baseline, green hatching indicates gains from E^2-Filter, and darker cross-hatching indicates losses from E^2-Filter. The tables additionally report Predicted-Gain Admit and the two hindsight references, with the latter marked by † and shown in gray. Boldface and underlining identify the highest and second-highest distinct accuracies, respec-

tively, among Always Admit, Predicted-Gain Admit, and E^2-Filter, including ties after rounding to two decimal places. Green and red cell backgrounds indicate positive and negative gains over Original Router, computed before rounding; zero gains are unshaded, and hindsight references receive no gain coloring. Consequently, identical displayed accuracies can have different background colors. The shading and ranking annotations do not represent confidence intervals or statistical significance.

C.1 RouterBench

Fig. 5 presents the eight RouterBench settings, with corresponding accuracies in Tabs. 5 and 6.

The RouterBench results illustrate how filtering mitigates the substantial accuracy losses that can arise from onboarding with very limited observations. For kNN with nine old models at $k = 1$, Always Admit achieves 57.43% accuracy, compared with 63.47% for Original Router and 62.99% for E^2-Filter. In this setting, filtering reduces the accuracy loss from 6.04 to 0.48 percentage points. For LOCUS with nine old models at $k = 1$, E^2-Filter reaches 63.87%, matching the displayed accuracy of Original Router.

The benefit of selective admission also extends to larger observation sizes. For LOCUS with five old models at $k = 1024$, E^2-Filter reaches 59.93%, exceeding both Always Admit at 59.70% and Original Router at 59.88%. Thus, filtering retains beneficial updates within a setting where unconditional admission still produces a negative average gain, even after observing the full support set.

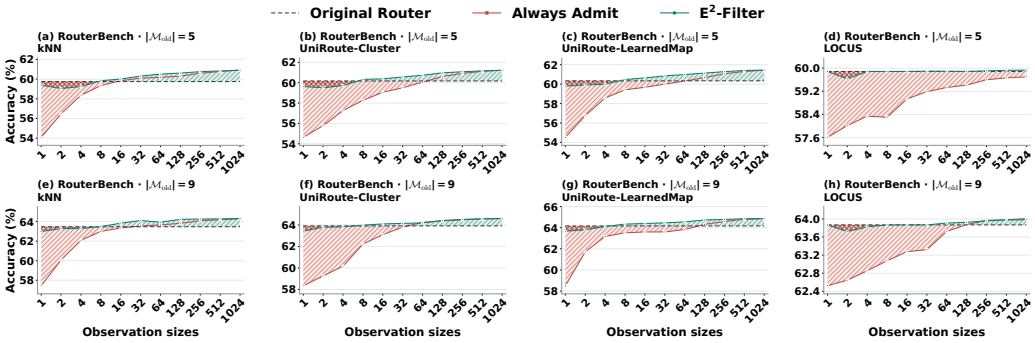


Figure 5: Accuracy (%) across observation sizes on RouterBench. The top and bottom rows use $|\mathcal{M}_{\text{old}}| = 5$ and 9, respectively; columns show kNN, UniRoute-Cluster, UniRoute-LearnedMap, and LOCUS. Each panel uses its own accuracy range. Hatched regions show accuracy differences relative to Original Router.

Table 5: Accuracy (%) across observation sizes on RouterBench with original model pool size $|\mathcal{M}_{\text{old}}| = 5$. Values use candidate-balanced pooling over ten support draws. Among Always Admit, Predicted-Gain Admit, and E^2-Filter **(Ours)**, **bold** marks the highest accuracy and underlining the second-highest distinct accuracy at each setting and observation size, including ties at two decimals. Original Router is excluded from this ranking. Green / red shading indicates an increase/decrease relative to Original Router using unrounded values; unshaded cells indicate equality. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

Model pool size $\lvert\mathcal{M}_{\text{old}}\rvert = 5$											
	Observation sizes										
Method	1	2	4	8	16	32	64	128	256	512	1024
kNN											
Original Router	59.75	59.75	59.75	59.75	59.75	59.75	59.75	59.75	59.75	59.75	59.75
Always Admit	54.09	56.47	58.36	59.34	59.84	60.10	60.18	60.30	60.60	60.77	**60.92**
Predicted-Gain Admit	54.51	56.69	58.69	59.70	**60.31**	**60.60**	**60.62**	**60.64**	60.68	60.70	60.77
E^2-Filter **(Ours)**	**59.39**	**59.05**	**59.24**	**59.84**	60.01	60.31	60.50	60.62	**60.76**	**60.83**	60.91
Budget Threshold†	59.75	59.75	59.75	59.75	59.84	60.10	60.18	60.30	60.60	60.77	60.92
Admission Oracle†	60.61	60.45	60.54	60.59	60.63	60.71	60.73	60.78	60.85	60.91	60.96
UniRoute-Cluster											
Original Router	60.17	60.17	60.17	60.17	60.17	60.17	60.17	60.17	60.17	60.17	60.17
Always Admit	54.64	55.78	57.26	58.27	59.06	59.50	60.06	60.61	60.92	61.12	**61.24**
Predicted-Gain Admit	54.78	56.72	58.50	59.91	**60.46**	**60.74**	**60.89**	**60.99**	61.01	61.07	61.08
E^2-Filter **(Ours)**	**59.63**	**59.50**	**59.72**	**60.32**	60.38	60.57	60.73	60.97	**61.08**	**61.17**	61.22
Budget Threshold†	60.17	60.17	60.17	60.17	60.17	60.17	60.17	60.61	60.92	61.12	61.24
Admission Oracle†	60.94	60.81	60.86	60.92	60.93	60.99	61.05	61.08	61.13	61.20	61.26
UniRoute-LearnedMap											
Original Router	60.34	60.34	60.34	60.34	60.34	60.34	60.34	60.34	60.34	60.34	60.34
Always Admit	54.53	56.76	58.57	59.42	59.67	60.00	60.31	60.68	61.06	61.30	61.43
Predicted-Gain Admit	54.89	57.14	59.28	60.18	60.56	60.79	**61.09**	**61.18**	61.26	61.29	61.29
E^2-Filter **(Ours)**	**59.80**	**59.91**	**59.99**	**60.47**	**60.64**	**60.84**	60.98	61.13	**61.28**	**61.40**	**61.44**
Budget Threshold†	60.34	60.34	60.34	60.34	60.34	60.34	60.34	60.68	61.06	61.30	61.43
Admission Oracle†	61.11	60.98	61.06	61.09	61.11	61.19	61.25	61.31	61.40	61.44	61.47
LOCUS											
Original Router	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88
Always Admit	57.61	58.02	58.34	58.30	58.93	59.19	59.33	59.41	59.60	59.66	59.70
Predicted-Gain Admit	58.38	58.77	59.13	59.07	59.56	59.62	59.76	59.74	59.89	59.89	59.88
E^2-Filter **(Ours)**	**59.88**	**59.63**	**59.88**	**59.88**	**59.88**	**59.89**	**59.89**	**59.89**	**59.91**	**59.92**	**59.93**
Budget Threshold†	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88	59.88
Admission Oracle†	60.00	59.99	60.03	60.02	60.11	60.06	60.07	60.07	60.07	60.09	60.11

Table 6: Accuracy (%) across observation sizes on RouterBench with original model pool size $|\mathcal{M}_{\text{old}}| = 9$. Values use candidate-balanced pooling over ten support draws. Among Always Admit, Predicted-Gain Admit, and E^2-Filter **(Ours)**, **bold** marks the highest accuracy and underlining the second-highest distinct accuracy at each setting and observation size, including ties at two decimals. Original Router is excluded from this ranking. Green / red shading indicates an increase/decrease relative to Original Router using unrounded values; unshaded cells indicate equality. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

Model pool size $\lvert\mathcal{M}_{\text{old}}\rvert = 9$											
	Observation sizes										
Method	1	2	4	8	16	32	64	128	256	512	1024
kNN											
Original Router	63.47	63.47	63.47	63.47	63.47	63.47	63.47	63.47	63.47	63.47	63.47
Always Admit	57.43	60.08	62.09	63.00	63.35	63.55	63.66	63.83	64.09	64.18	64.30
Predicted-Gain Admit	57.97	60.30	62.65	**63.68**	**63.98**	**64.20**	**64.16**	64.21	**64.27**	**64.28**	**64.31**
E^2-Filter **(Ours)**	**62.99**	**63.28**	**63.29**	63.50	63.84	64.11	63.94	**64.23**	**64.27**	**64.28**	**64.31**
Budget Threshold†	63.47	63.47	63.47	63.47	63.47	63.55	63.66	63.83	64.09	64.18	64.30
Admission Oracle†	64.22	64.13	64.17	64.21	64.21	64.24	64.22	64.23	64.27	64.29	64.32
UniRoute-Cluster											
Original Router	63.91	63.91	63.91	63.91	63.91	63.91	63.91	63.91	63.91	63.91	63.91
Always Admit	58.34	59.25	60.16	62.22	63.10	63.80	**64.24**	64.38	64.48	64.56	**64.59**
Predicted-Gain Admit	58.34	61.82	63.06	63.68	64.01	64.01	64.03	63.93	63.91	63.91	63.91
E^2-Filter **(Ours)**	**63.42**	**63.79**	**63.82**	**63.98**	**64.10**	**64.14**	64.19	**64.41**	**64.51**	**64.58**	**64.59**
Budget Threshold†	63.91	63.91	63.91	63.91	63.91	63.91	64.24	64.38	64.48	64.56	64.59
Admission Oracle†	64.56	64.36	64.36	64.37	64.43	64.45	64.45	64.48	64.51	64.58	64.59
UniRoute-LearnedMap											
Original Router	64.16	64.16	64.16	64.16	64.16	64.16	64.16	64.16	64.16	64.16	64.16
Always Admit	58.50	61.71	63.17	63.51	63.60	63.60	63.82	64.31	64.59	64.79	**64.86**
Predicted-Gain Admit	58.50	62.02	63.89	64.14	64.26	64.16	64.21	64.21	64.19	64.16	64.16
E^2-Filter **(Ours)**	**63.67**	**63.80**	**64.11**	**64.34**	**64.40**	**64.46**	**64.55**	**64.73**	**64.78**	**64.85**	**64.86**
Budget Threshold†	64.16	64.16	64.16	64.16	64.16	64.16	64.16	64.31	64.59	64.79	64.86
Admission Oracle†	64.79	64.64	64.61	64.60	64.62	64.64	64.67	64.74	64.80	64.85	64.86
LOCUS											
Original Router	63.87	63.87	63.87	63.87	63.87	63.87	63.87	63.87	63.87	63.87	63.87
Always Admit	62.52	62.65	62.86	63.08	63.28	63.32	63.73	63.87	**63.97**	**63.98**	**64.01**
Predicted-Gain Admit	63.15	63.20	63.46	63.44	63.53	63.47	63.73	63.81	63.84	63.85	63.87
E^2-Filter **(Ours)**	**63.87**	**63.72**	**63.82**	**63.87**	**63.87**	**63.87**	**63.91**	**63.92**	63.96	**63.98**	63.98
Budget Threshold†	63.87	63.87	63.87	63.87	63.87	63.87	63.87	63.87	63.97	63.98	64.01
Admission Oracle†	63.97	63.98	63.97	63.98	63.99	63.99	64.01	64.03	64.02	64.01	64.01

C.2 LLMROUTERBENCH

Fig. 6 presents the eight LLMRouterBench settings, with corresponding accuracies in Tabs. 7 and 8.

On LLMRouterBench, admission filtering remains useful beyond the smallest observation sizes. Always Admit stays below the original router in all eight settings at $k = 64$. For UniRoute-Cluster with nine old models at this budget, it reaches 66.73%, compared with the original accuracy of 68.06%, while E^2-Filter achieves 68.12%. This contrast shows how filtering can preserve deployment accuracy while candidate observations remain insufficient for reliable unconditional admission.

The LOCUS results further demonstrate that collecting more observations does not by itself ensure a beneficial routing update. Even at $k = 1024$, Always Admit remains below the original router with both pool sizes, whereas E^2-Filter obtains positive gains of approximately 0.08 and 0.01 percentage points for pools of five and nine models, respectively. These results support an explicit admission decision in addition to constructing the updated router: candidate-specific evidence helps retain beneficial updates when unconditional deployment would reduce average accuracy.

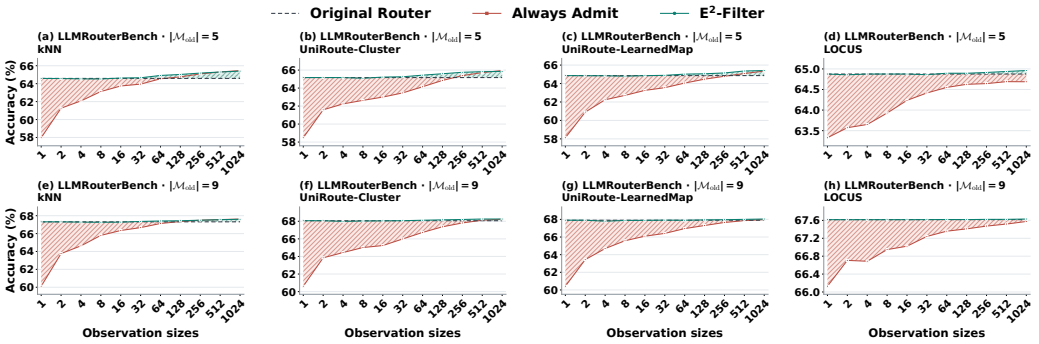


Figure 6: Accuracy (%) across observation sizes on LLMRouterBench. The top and bottom rows use $|\mathcal{M}_{\text{old}}| = 5$ and 9, respectively; columns show kNN, UniRoute-Cluster, UniRoute-LearnedMap, and LOCUS. Each panel uses its own accuracy range. Hatched regions show accuracy differences relative to Original Router.

Table 7: Accuracy (%) across observation sizes on LLMRouterBench with original model pool size $|\mathcal{M}_{\text{old}}| = 5$. Values use candidate-balanced pooling over ten support draws. Among Always Admit, Predicted-Gain Admit, and E^2-Filter **(Ours)**, **bold** marks the highest accuracy and underlining the second-highest distinct accuracy at each setting and observation size, including ties at two decimals. Original Router is excluded from this ranking. Green / red shading indicates an increase/decrease relative to Original Router using unrounded values; unshaded cells indicate equality. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

Model pool size $\lvert\mathcal{M}_{\text{old}}\rvert = 5$											
	Observation sizes										
Method	1	2	4	8	16	32	64	128	256	512	1024
kNN											
Original Router	64.59	64.59	64.59	64.59	64.59	64.59	64.59	64.59	64.59	64.59	64.59
Always Admit	57.97	61.27	62.08	63.12	63.74	63.95	64.54	64.76	65.09	**65.30**	**65.46**
Predicted-Gain Admit	57.97	61.73	62.46	63.82	64.46	**64.72**	64.90	64.95	64.99	65.18	65.17
E^2-Filter **(Ours)**	**64.59**	**64.57**	**64.53**	**64.52**	**64.63**	64.66	**64.91**	**65.03**	**65.18**	**65.30**	65.40
Budget Threshold†	64.59	64.59	64.59	64.59	64.59	64.59	64.59	64.76	65.09	65.30	65.46
Admission Oracle†	64.91	64.93	64.95	64.96	65.01	65.04	65.08	65.13	65.26	65.40	65.50
UniRoute-Cluster											
Original Router	65.17	65.17	65.17	65.17	65.17	65.17	65.17	65.17	65.17	65.17	65.17
Always Admit	58.48	61.58	62.24	62.64	63.00	63.46	64.15	64.85	65.39	65.72	**65.95**
Predicted-Gain Admit	58.48	62.25	63.44	64.62	65.05	**65.28**	65.42	65.51	65.61	65.70	65.63
E^2-Filter **(Ours)**	**65.17**	**65.17**	**65.17**	**65.09**	**65.22**	65.26	**65.44**	**65.59**	**65.75**	**65.82**	65.85
Budget Threshold†	65.17	65.17	65.17	65.17	65.17	65.17	65.17	65.17	65.39	65.72	65.95
Admission Oracle†	65.44	65.47	65.49	65.52	65.59	65.63	65.67	65.74	65.86	65.93	66.00
UniRoute-LearnedMap											
Original Router	64.86	64.86	64.86	64.86	64.86	64.86	64.86	64.86	64.86	64.86	64.86
Always Admit	58.13	60.90	62.24	62.72	63.26	63.55	64.05	64.47	64.79	65.08	65.35
Predicted-Gain Admit	58.13	61.93	63.23	64.22	64.61	64.86	64.99	**65.06**	65.14	65.29	**65.44**
E^2-Filter **(Ours)**	**64.86**	**64.85**	**64.84**	**64.80**	**64.85**	**64.89**	**65.02**	**65.06**	**65.15**	**65.36**	65.41
Budget Threshold†	64.86	64.86	64.86	64.86	64.86	64.86	64.86	64.86	64.86	65.08	65.35
Admission Oracle†	65.17	65.21	65.21	65.23	65.25	65.24	65.26	65.28	65.36	65.44	65.51
LOCUS											
Original Router	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88
Always Admit	63.33	63.57	63.65	63.93	64.24	64.42	64.55	64.63	64.64	64.69	64.69
Predicted-Gain Admit	**64.88**	**64.86**	**64.88**	64.86	64.86	**64.86**	64.88	64.87	64.86	64.86	64.88
E^2-Filter **(Ours)**	**64.88**	**64.86**	**64.88**	**64.88**	**64.88**	**64.86**	**64.90**	**64.89**	**64.91**	**64.94**	**64.96**
Budget Threshold†	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88	64.88
Admission Oracle†	65.00	64.99	65.00	65.02	65.03	65.06	65.09	65.09	65.08	65.10	65.08

Table 8: Accuracy (%) across observation sizes on LLMRouterBench with original model pool size $|\mathcal{M}_{\text{old}}| = 9$. Values use candidate-balanced pooling over ten support draws. Among Always Admit, Predicted-Gain Admit, and E^2-Filter **(Ours)**, **bold** marks the highest accuracy and underlining the second-highest distinct accuracy at each setting and observation size, including ties at two decimals. Original Router is excluded from this ranking. Green / red shading indicates an increase/decrease relative to Original Router using unrounded values; unshaded cells indicate equality. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

Model pool size $\lvert\mathcal{M}_{\text{old}}\rvert = 9$											
	Observation sizes										
Method	1	2	4	8	16	32	64	128	256	512	1024
kNN											
Original Router	67.32	67.32	67.32	67.32	67.32	67.32	67.32	67.32	67.32	67.32	67.32
Always Admit	60.12	63.77	64.63	65.80	66.35	66.67	67.13	67.35	**67.50**	**67.57**	**67.63**
Predicted-Gain Admit	60.12	64.12	64.90	66.36	66.97	67.25	67.39	67.39	67.38	67.42	67.51
E^2-Filter **(Ours)**	**67.32**	**67.32**	**67.28**	**67.27**	**67.28**	**67.36**	**67.41**	**67.44**	**67.50**	67.56	67.60
Budget Threshold†	67.32	67.32	67.32	67.32	67.32	67.32	67.32	67.35	67.50	67.57	67.63
Admission Oracle†	67.44	67.44	67.47	67.46	67.47	67.49	67.48	67.51	67.58	67.63	67.67
UniRoute-Cluster											
Original Router	68.06	68.06	68.06	68.06	68.06	68.06	68.06	68.06	68.06	68.06	68.06
Always Admit	60.62	63.85	64.44	65.01	65.24	65.99	66.73	67.37	67.81	68.12	68.23
Predicted-Gain Admit	60.62	64.73	65.98	67.12	67.73	68.02	**68.12**	68.15	68.19	68.19	68.18
E^2-Filter **(Ours)**	**68.06**	**68.06**	**67.99**	**68.03**	**68.05**	**68.07**	**68.12**	**68.16**	**68.20**	**68.26**	**68.25**
Budget Threshold†	68.06	68.06	68.06	68.06	68.06	68.06	68.06	68.06	68.06	68.12	68.23
Admission Oracle†	68.13	68.14	68.15	68.16	68.18	68.20	68.23	68.25	68.30	68.32	68.30
UniRoute-LearnedMap											
Original Router	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.90
Always Admit	60.42	63.44	64.69	65.58	66.10	66.41	66.96	67.31	67.65	67.86	67.97
Predicted-Gain Admit	60.42	64.57	65.80	67.06	67.50	67.84	**67.90**	**67.95**	67.96	67.99	68.02
E^2-Filter **(Ours)**	**67.90**	**67.90**	**67.79**	**67.88**	**67.89**	**67.91**	**67.90**	67.94	**67.97**	**68.00**	**68.04**
Budget Threshold†	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.90	67.97
Admission Oracle†	67.99	67.99	68.00	68.00	68.01	68.00	68.00	68.00	68.02	68.05	68.07
LOCUS											
Original Router	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62
Always Admit	66.13	66.71	66.69	66.95	67.02	67.24	67.36	67.41	67.47	67.52	67.58
Predicted-Gain Admit	67.57	67.59	67.58	67.61	67.61	**67.62**	**67.62**	**67.62**	**67.62**	**67.62**	67.62
E^2-Filter **(Ours)**	**67.62**	**67.62**	**67.62**	**67.62**	**67.62**	**67.62**	**67.62**	**67.62**	**67.62**	**67.62**	**67.63**
Budget Threshold†	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62	67.62
Admission Oracle†	67.65	67.65	67.66	67.65	67.67	67.67	67.68	67.68	67.70	67.70	67.72

C.3 EMBEDLLM

Fig. 7 presents the eight EmbedLLM settings, with corresponding accuracies in Tabs. 9 and 10.

The EmbedLLM results show that filtering retains the benefits of candidate onboarding as the observation budget increases. At $k = 1024$, E^2-Filter achieves the highest displayed accuracy among deployable admission methods in five of the eight settings, including ties. For UniRoute-Cluster with five old models, it reaches 51.81%, compared with 51.74% for Predicted-Gain Admit and 51.79% for Always Admit. At $k = 128$, Predicted-Gain Admit is slightly ahead of E^2-Filter (51.54% versus 51.48%), but both improve upon the original accuracy of 50.93%, while Always Admit remains below it at 50.73%. This trajectory illustrates how filtering can recover positive gains before unconditional admission becomes beneficial and preserve gains at larger observation sizes.

The magnitude of the observed gains also varies with pool size. At $k = 1024$, the gain from E^2-Filter over the original router is smaller with nine old models than with five for all four routing methods. For UniRoute-LearnedMap, the improvement decreases from approximately 0.87 to 0.39 percentage points. Despite these smaller incremental gains in the larger-pool settings, E^2-Filter retains positive improvements across both pool sizes.

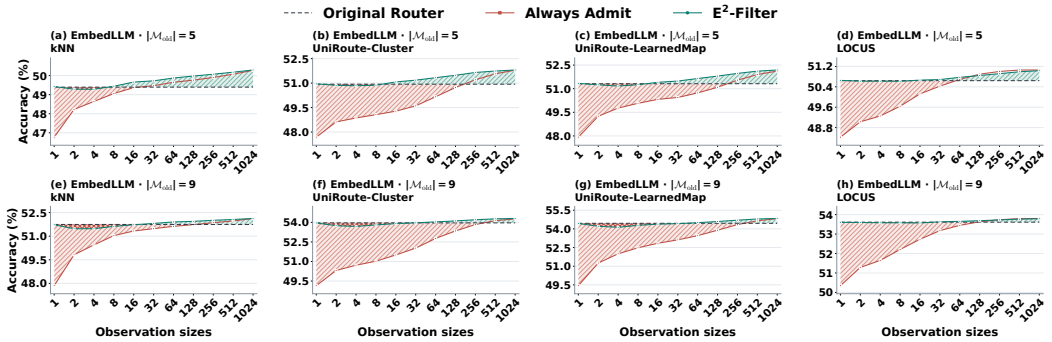


Figure 7: Accuracy (%) across observation sizes on EmbedLLM. The top and bottom rows use $|\mathcal{M}_{\text{old}}| = 5$ and 9, respectively; columns show kNN, UniRoute-Cluster, UniRoute-LearnedMap, and LOCUS. Each panel uses its own accuracy range. Hatched regions show accuracy differences relative to Original Router.

Table 9: Accuracy (%) across observation sizes on EmbedLLM with original model pool size $|\mathcal{M}_{\text{old}}| = 5$. Values use candidate-balanced pooling over ten support draws. Among Always Admit, Predicted-Gain Admit, and E^2-Filter **(Ours)**, **bold** marks the highest accuracy and underlining the second-highest distinct accuracy at each setting and observation size, including ties at two decimals. Original Router is excluded from this ranking. Green / red shading indicates an increase/decrease relative to Original Router using unrounded values; unshaded cells indicate equality. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

Model pool size $\lvert\mathcal{M}_{\text{old}}\rvert = 5$											
	Observation sizes										
Method	1	2	4	8	16	32	64	128	256	512	1024
kNN											
Original Router	49.39	49.39	49.39	49.39	49.39	49.39	49.39	49.39	49.39	49.39	49.39
Always Admit	46.77	48.23	48.64	49.06	49.37	49.47	49.65	49.76	49.90	50.07	50.28
Predicted-Gain Admit	46.77	48.67	48.94	49.29	49.65	**49.78**	**49.91**	49.97	50.02	50.09	50.22
E^2-Filter **(Ours)**	**49.42**	**49.29**	**49.28**	**49.44**	**49.66**	49.72	49.87	**49.98**	**50.07**	**50.18**	**50.29**
Budget Threshold†	49.39	49.39	49.39	49.39	49.39	49.47	49.65	49.76	49.90	50.07	50.28
Admission Oracle†	50.02	50.02	50.02	50.04	50.07	50.07	50.11	50.14	50.18	50.27	50.38
UniRoute-Cluster											
Original Router	50.93	50.93	50.93	50.93	50.93	50.93	50.93	50.93	50.93	50.93	50.93
Always Admit	47.69	48.62	48.86	49.06	49.28	49.61	50.16	50.73	51.20	51.57	51.79
Predicted-Gain Admit	47.69	49.53	49.83	50.29	50.81	51.14	**51.39**	**51.54**	**51.66**	51.70	51.74
E^2-Filter **(Ours)**	**50.96**	**50.86**	**50.84**	**50.87**	**51.07**	**51.18**	51.33	51.48	51.65	**51.74**	**51.81**
Budget Threshold†	50.93	50.93	50.93	50.93	50.93	50.93	50.93	50.93	51.20	51.57	51.79
Admission Oracle†	51.43	51.46	51.49	51.51	51.54	51.60	51.67	51.73	51.80	51.85	51.88
UniRoute-LearnedMap											
Original Router	51.32	51.32	51.32	51.32	51.32	51.32	51.32	51.32	51.32	51.32	51.32
Always Admit	47.92	49.25	49.77	50.07	50.32	50.44	50.73	51.09	51.54	51.90	52.12
Predicted-Gain Admit	47.92	49.99	50.40	50.76	51.23	51.40	51.62	**51.80**	**51.98**	52.06	52.09
E^2-Filter **(Ours)**	**51.34**	**51.25**	**51.17**	**51.25**	**51.41**	**51.49**	**51.65**	**51.80**	51.97	**52.11**	**52.19**
Budget Threshold†	51.32	51.32	51.32	51.32	51.32	51.32	51.32	51.32	51.54	51.90	52.12
Admission Oracle†	51.80	51.83	51.85	51.87	51.91	51.94	52.02	52.08	52.15	52.22	52.28
LOCUS											
Original Router	50.64	50.64	50.64	50.64	50.64	50.64	50.64	50.64	50.64	50.64	50.64
Always Admit	48.42	49.03	49.26	49.64	50.12	50.43	50.69	**50.90**	**51.00**	**51.06**	**51.06**
Predicted-Gain Admit	49.11	49.72	49.81	50.11	50.34	50.56	50.69	50.79	50.86	50.87	50.89
E^2-Filter **(Ours)**	**50.65**	**50.62**	**50.62**	**50.62**	**50.66**	**50.69**	**50.77**	50.85	50.92	50.99	51.03
Budget Threshold†	50.64	50.64	50.64	50.64	50.64	50.64	50.69	50.90	51.00	51.06	51.06
Admission Oracle†	50.93	50.98	51.01	51.03	51.07	51.11	51.13	51.18	51.19	51.20	51.20

Table 10: Accuracy (%) across observation sizes on EmbedLLM with original model pool size $|\mathcal{M}_{\text{old}}| = 9$. Values use candidate-balanced pooling over ten support draws. Among Always Admit, Predicted-Gain Admit, and E^2-Filter **(Ours)**, **bold** marks the highest accuracy and underlining the second-highest distinct accuracy at each setting and observation size, including ties at two decimals. Original Router is excluded from this ranking. Green / red shading indicates an increase/decrease relative to Original Router using unrounded values; unshaded cells indicate equality. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

Model pool size $\lvert\mathcal{M}_{\text{old}}\rvert = 9$											
	Observation sizes										
Method	1	2	4	8	16	32	64	128	256	512	1024
kNN											
Original Router	51.74	51.74	51.74	51.74	51.74	51.74	51.74	51.74	51.74	51.74	51.74
Always Admit	47.86	49.82	50.45	51.04	51.33	51.48	51.62	51.73	51.86	51.97	**52.13**
Predicted-Gain Admit	47.86	50.33	50.78	51.32	51.68	**51.87**	**51.92**	**51.94**	51.93	51.95	52.01
E^2-Filter **(Ours)**	**51.73**	**51.47**	**51.47**	**51.62**	**51.72**	51.81	51.90	**51.94**	**52.01**	**52.06**	52.12
Budget Threshold†	51.74	51.74	51.74	51.74	51.74	51.74	51.74	51.74	51.86	51.97	52.13
Admission Oracle†	52.10	52.04	52.04	52.06	52.06	52.07	52.08	52.10	52.12	52.13	52.20
UniRoute-Cluster											
Original Router	53.96	53.96	53.96	53.96	53.96	53.96	53.96	53.96	53.96	53.96	53.96
Always Admit	49.15	50.31	50.71	51.02	51.50	52.03	52.77	53.34	53.83	54.13	54.27
Predicted-Gain Admit	49.15	51.61	52.09	52.85	53.47	53.82	**54.07**	**54.16**	54.19	54.22	54.23
E^2-Filter **(Ours)**	**53.95**	**53.74**	**53.69**	**53.80**	**53.91**	**53.96**	54.04	54.11	**54.20**	**54.27**	**54.30**
Budget Threshold†	53.96	53.96	53.96	53.96	53.96	53.96	53.96	53.96	53.96	54.13	54.27
Admission Oracle†	54.19	54.17	54.17	54.19	54.22	54.24	54.27	54.30	54.33	54.36	54.35
UniRoute-LearnedMap											
Original Router	54.45	54.45	54.45	54.45	54.45	54.45	54.45	54.45	54.45	54.45	54.45
Always Admit	49.44	51.26	52.00	52.48	52.84	53.12	53.45	53.89	54.34	54.65	54.82
Predicted-Gain Admit	49.44	52.38	52.92	53.51	53.99	54.26	54.44	**54.59**	54.68	54.74	54.74
E^2-Filter **(Ours)**	**54.44**	**54.21**	**54.12**	**54.28**	**54.38**	**54.42**	**54.50**	**54.59**	**54.70**	**54.79**	**54.84**
Budget Threshold†	54.45	54.45	54.45	54.45	54.45	54.45	54.45	54.45	54.45	54.65	54.82
Admission Oracle†	54.65	54.65	54.65	54.68	54.69	54.71	54.75	54.79	54.83	54.88	54.90
LOCUS											
Original Router	53.61	53.61	53.61	53.61	53.61	53.61	53.61	53.61	53.61	53.61	53.61
Always Admit	50.35	51.29	51.64	52.21	52.73	53.18	53.45	53.63	**53.74**	**53.80**	**53.80**
Predicted-Gain Admit	51.21	52.05	52.35	52.73	53.09	53.38	53.54	53.65	53.72	53.75	53.76
E^2-Filter **(Ours)**	**53.61**	**53.60**	**53.58**	**53.58**	**53.57**	**53.63**	**53.65**	**53.69**	53.73	53.77	53.79
Budget Threshold†	53.61	53.61	53.61	53.61	53.61	53.61	53.61	53.63	53.74	53.80	53.80
Admission Oracle†	53.77	53.77	53.77	53.81	53.80	53.82	53.84	53.86	53.87	53.89	53.90

D COST-AWARE MODEL ADMISSION

As discussed in Sec. 2, inference cost can be incorporated into routing through a Lagrangian penalty. We extend our evaluation to this setting and examine whether E^2-Filter improves the cost–quality trade-off when admitting a new model. Following Jitkrittum et al. (2026), we summarize this trade-off using the normalized area under the accuracy–cost curve (*Area*). The adaptation expresses predicted gains and pool-history residuals in terms of Area, while retaining the self-simulated evidence and admission rule from Sec. 3.

D.1 COST-AWARE ROUTING AND EVALUATION

Let $c_m(x)$ denote the inference cost of model m on query x. For a cost penalty $\eta \geq 0$, the original router becomes

$$\pi_\eta^{\text{old}}(x) = \arg\max_{m \in \mathcal{M}_{\text{old}}} \left[h_\phi(x, m; S_m) - \eta \frac{c_m(x) - c_{\min}}{c_{\max} - c_{\min}} \right], \tag{10}$$

where $c_{\min}$ and $c_{\max}$ are the minimum and maximum mean model costs in $\mathcal{M}_{\text{old}} \cup \{m^\star\}$ for the current episode. The updated router π_η^{new} additionally considers $m^\star$ using $h_\phi(x, m^\star; S_{m^\star,k})$, with the old-model predictions held fixed. Both routers use the same cost normalization, and ties between the candidate and the original choice retain the original choice.

Varying η yields a set of accuracy–cost operating points for each router. For $v \in \{\text{old}, \text{new}\}$, let $Q^v(B)$ denote the highest mean correctness among the evaluated policies $\{\pi_\eta^v\}$ whose mean cost does not exceed B. We define

$$A^v = \frac{1}{c_{\max} - c_{\min}} \int_{c_{\min}}^{c_{\max}} Q^v(B)\, \mathrm{d}B. \tag{11}$$

Below the cheapest evaluated operating point, we extend its quality to $c_{\min}$. The common integration interval makes A^{new} and A^{old} comparable within an episode. Their difference measures the integrated change in routing quality across cost budgets.

D.2 ADAPTING DUAL-EVIDENCE ADMISSION FILTER

Pool-history evidence. We first compute predicted Area values $\widehat{A}^{\text{old}}$ and $\widehat{A}^{\text{new}}$ by replacing the correctness outcomes of the selected models with their corresponding predictions from h_ϕ. The predicted routing gain in Eq. (3) is then replaced by

$$\widehat{g}^A_{\text{new}} = \widehat{A}^{\text{new}} - \widehat{A}^{\text{old}}. \tag{12}$$

This calculation uses the target inputs $\mathcal{Q}_{\text{test}}$ and cost information, without accessing their correctness labels.

We calibrate this prediction using the pseudo-onboarding procedure in Sec. 3.1. In each trial t, an existing model serves as a pseudo-candidate and is onboarded using a size-k support set. On $\mathcal{Q}_{\text{train}}$, we compute its predicted Area gain $\widehat{g}^{t,A}$ and realized Area gain $\delta^{t,A}$, obtaining

$$e^{t,A} = \delta^{t,A} - \widehat{g}^{t,A}, \qquad \mathcal{R}^A_{\text{old}} = \{e^{t,A}\}_{t=1}^T. \tag{13}$$

Each trial uses a common cost interval for its original and updated routers, as in Eq. (11). The resulting pool-history evidence is

$$s^A_{\text{hist}} = \frac{1}{|\mathcal{R}^A_{\text{old}}|} \sum_{e^{t,A} \in \mathcal{R}^A_{\text{old}}} \mathbf{1}\left\{\widehat{g}^A_{\text{new}} + e^{t,A} > \epsilon\right\}. \tag{14}$$

Thus, both the predicted benefit and its calibration residuals measure the same cost–quality objective.

Self-simulated evidence and admission. We retain s_{cand} from Eq. (6), computed from beneficial and harmful reroutings using cross-fitted candidate observations $S_{m^\star,k}$. This component continues to assess correctness gains under accuracy-based rerouting; it is not an estimate of Area improvement. For $k > 1$, the combined admission score is

$$s^A = \text{logit}(s^A_{\text{hist}}) + \lambda\, \text{logit}(s_{\text{cand}}). \tag{15}$$

Following Eq. (8), we deploy the updated routing family if $s^A \geq \text{logit}(\tau)$ and otherwise retain the original one. We use the same hyperparameters as in the accuracy experiments: $\lambda = 1$, $\tau = 0.975$, and $\epsilon = 0.0025$, clipping logit inputs to $[10^{-4}, 1 - 10^{-4}]$. Admission also requires a nonempty residual bank and at least one predicted candidate rerouting under the accuracy-based scores. At $k = 1$, we use the historical fallback $s^A_{\text{hist}} \geq 0.90$. The adaptation requires no additional candidate correctness observations beyond $S_{m^\star,k}$.

D.3 Experiments

Setup. We evaluate on RouterBench (Hu et al., 2024) and EmbedLLM (Zhuang et al., 2025), using kNN, UniRoute-Cluster, UniRoute-LearnedMap, and LOCUS with $|\mathcal{M}_{\text{old}}| \in \{5, 9\}$, yielding 16 settings. We retain the original episode construction, data splits, and frozen old-model predictors. Candidate support sizes are $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$, with ten support draws and nested support sets within each draw. RouterBench provides query-level response costs in USD; for EmbedLLM, we use model parameter count as a proxy for inference cost.[2] We omit LLMRouterBench because nine of its 20 models have zero recorded cost for all test responses.

For each router, we evaluate 75 fixed cost penalties, including zero and a penalty enforcing the cheapest-model endpoint, and numerically integrate Area over 1,001 uniformly spaced cost budgets. For each episode and support size k, we construct the residual bank through pseudo-onboarding trials within the current old-model pool. Each old model serves as a pseudo-candidate against the remaining models, with the original routing predictor held fixed. Historical residuals are computed as the differences between realized and predicted Area gains on the training anchors. Predictions on support queries are cross-fitted; at $k = 1$, the sole support query is omitted from this evaluation. Admission uses historical old-model observations, candidate support observations, unlabeled target queries, and cost information. Correctness labels on $\mathcal{Q}_{\text{test}}$ are reserved for outcome evaluation and the hindsight references below.

Baselines and metric. We compare E^2-Filter with Always Admit and Predicted-Gain Admit, where the latter admits if $\widehat{g}^A_{\text{new}} > 0$. We also report Budget Threshold, which selects a support-size threshold in hindsight to maximize AUDG within each setting, and Admission Oracle, which selects the router with higher Area for each episode, support draw, and support size. These two references use held-out outcomes and are excluded from the deployable-method ranking.

Cost-aware AUDG follows the same definition as accuracy-based AUDG, replacing accuracy gain with $A^{\text{new}} - A^{\text{old}}$. At each k, the deployed gain equals this difference if the candidate is admitted and zero otherwise. We first average over episodes and support draws within each candidate, then equally over candidates. We integrate the resulting mean gain over $\log_2 k$ using the trapezoidal rule, normalize by the log-support span $\log_2(1024) - \log_2(1) = 10$, and multiply by 100. The resulting AUDG is expressed in percentage points of Area and retains negative gains. Overall averages assign equal weight to the eight settings at each old-pool size.

Results. Table 11 shows that E^2-Filter outperforms Always Admit and Predicted-Gain Admit in all 16 settings, with positive AUDG in 11. For five old models, its average AUDG is $+0.154$, compared with -0.463 for Always Admit and -0.440 for Predicted-Gain Admit. For nine old models, the corresponding values are $+0.053$, -0.674, and -0.673. Predicted-Gain Admit remains negative in every setting, indicating that expressing predicted gains in cost-aware units alone does not resolve errors in admission decisions.

On RouterBench, E^2-Filter achieves positive AUDG for three of the four routing predictors at both pool sizes. For example, with kNN it obtains $+0.255$ and $+0.183$ for five and nine old models, respectively, whereas Always Admit obtains -0.328 and -0.538. The positive gains also extend to both UniRoute variants, while LOCUS retains negative AUDG values of -0.036 and -0.009, respectively. On EmbedLLM, E^2-Filter achieves positive AUDG for all four predictors with five old models. With nine, kNN, UniRoute-Cluster, and UniRoute-LearnedMap retain small negative AUDG values of -0.001, -0.017, and -0.025, respectively, substantially reducing the losses of

[2] One retired EmbedLLM model is assigned an unverified estimate of 34B parameters. The EmbedLLM results should therefore be interpreted with respect to this specified model-size proxy.

Table 11: Cost-aware budgeted model admission across two benchmarks. Cells report signed AUDG of UniRoute-style cost–quality Area relative to the old router, in percentage points of Area (higher is better). **Bold** indicates the best deployable admission policy within each setting. † Budget Threshold and Admission Oracle, serving as hindsight references, are excluded from this comparison.

		Old Model Pool $\lvert\mathcal{M}_{\text{old}}\rvert = 5$					*Old Model Pool* $\lvert\mathcal{M}_{\text{old}}\rvert = 9$				
Dataset	Routing Predictor	Always Admit	Predicted-Gain Admit	E^2-Filter (Ours)	Budget Threshold†	Admission Oracle†	Always Admit	Predicted-Gain Admit	E^2-Filter (Ours)	Budget Threshold†	Admission Oracle†
RouterBench	kNN	−0.328	−0.312	**+0.255**	+0.340	+0.877	−0.538	−0.538	**+0.183**	+0.179	+0.612
	UniRoute-Cluster	−0.812	−0.801	**+0.253**	+0.275	+0.818	−0.789	−0.788	**+0.112**	+0.278	+0.561
	UniRoute-LearnedMap	−0.621	−0.605	**+0.248**	+0.236	+0.809	−0.541	−0.542	**+0.152**	+0.152	+0.480
	LOCUS	−0.582	−0.553	**-0.036**	+0.041	+0.224	−0.464	−0.464	**-0.009**	+0.061	+0.157
EmbedLLM	kNN	−0.097	−0.071	**+0.189**	+0.177	+0.511	−0.463	−0.460	**-0.001**	+0.052	+0.254
	UniRoute-Cluster	−0.563	−0.525	**+0.134**	+0.129	+0.510	−1.158	−1.153	**-0.017**	+0.044	+0.238
	UniRoute-LearnedMap	−0.587	−0.569	**+0.091**	+0.095	+0.454	−1.029	−1.028	**-0.025**	+0.043	+0.233
	LOCUS	−0.111	−0.081	**+0.095**	+0.141	+0.408	−0.414	−0.408	**+0.033**	+0.066	+0.219
Overall Average		−0.463	−0.440	**+0.154**	+0.179	+0.576	−0.674	−0.673	**+0.053**	+0.109	+0.344

RouterBench uses recorded response costs in USD; EmbedLLM uses parameter count as a cost proxy. Retaining the old router gives zero AUDG. Overall Average weights the eight settings within each pool size equally. †Hindsight references use test outcomes and are excluded from the ranking. Shared configuration: $(\epsilon, \lambda, \tau, J) = (0.0025, 1, 0.975, 5)$, selected by a 270-configuration scan of the existing accuracy evaluation; not selected separately for cost-aware results.

-0.463, -1.158, and -1.029 under Always Admit. Positive Admission Oracle values in these settings indicate remaining opportunities for beneficial admission.

These results extend the evidence for E^2-Filter to cost–quality objectives under both recorded monetary costs and a model-size proxy, without retuning the admission thresholds. The improvement concerns the integrated accuracy–cost frontier and does not imply an improvement at every cost budget.

E Admission Headroom and Pool Size

Holding out an existing model provides a controlled onboarding setting, but does not guarantee that admitting it offers a substantial accuracy gain. This appendix examines three aspects of this limitation: the admission gains available in the main evaluation, their variation across larger old-model pools, and the additional query coverage provided by a candidate in the full model pool. The first two analyses concern the constructed routing policies; the third measures candidate complementarity independently of a learned router.

Measuring admission headroom. For a fixed benchmark, routing method, and old-pool size, Admission Oracle selects the more accurate of the original and updated routers for each candidate, old-model pool, support draw, and observation size. Let $A^{+}_{c,e,d}(k)$ and $A^{0}_{c,e}$ denote their test accuracies on the $[0, 1]$ scale, where c indexes the candidate, e the old-model pool, and d the support draw. For evaluated support sizes $1 = k_1 < \cdots < k_L = 1024$, define

$$h_c(k) = 100\,\mathbb{E}_{e,d}\left[\left(A^{+}_{c,e,d}(k) - A^{0}_{c,e}\right)_{+}\right],$$
$$H_c = \frac{1}{10}\sum_{\ell=1}^{L-1} \frac{h_c(k_\ell) + h_c(k_{\ell+1})}{2} \log_2\left(\frac{k_{\ell+1}}{k_\ell}\right). \tag{16}$$

Here $(u)_+ = \max(u, 0)$, and the expectation is the empirical average over pools and support draws. The positive part is applied before averaging. Thus, H_c is the candidate's oracle AUDG in percentage points (pp), using the same normalized trapezoidal integration as the main metric. It measures the gain available from choosing between the two constructed routers, rather than the gain achievable by an optimal routing predictor. All oracle quantities are computed retrospectively from test outcomes. We use $H_c \le 0.1$ pp to describe low-headroom candidates; this cutoff is descriptive, not a statistical significance threshold.

Headroom in the main evaluation. Table 12 summarizes the available gains across the three benchmarks and four routing methods. Mean oracle AUDG decreases from $M = 5$ to $M = 9$ on every benchmark. Moreover, at $M = 9$, most candidates have low headroom: 90.9% on RouterBench for every routing method, 75.0–90.0% on LLMRouterBench, and 75.0–83.0% on EmbedLLM. Positive average oracle gains therefore coexist with limited improvement opportunities for many individual candidates. These cases remain useful for assessing the prevention of harmful updates, whereas candidates with greater headroom test whether an admission method retains beneficial updates.

Table 12: Admission headroom in the main evaluation. Mean oracle AUDG averages candidates and the four routing methods equally. The final column reports the range across methods of the percentage of candidates with $H_c \leq 0.1$ pp at $M = 9$. These ranges describe variation across methods, not confidence intervals.

	Mean oracle AUDG (pp)		Low-headroom candidates (%)
Benchmark	$M = 5$	$M = 9$	$M = 9$
RouterBench	0.707	0.490	90.9
LLMRouterBench	0.401	0.127	75.0–90.0
EmbedLLM	0.635	0.285	75.0–83.0

Headroom with larger old-model pools. Table 13 extends the analysis on EmbedLLM to $M \in \{5, 9, 22, 56, 111\}$ using kNN, UniRoute-Cluster, and LOCUS. This diagnostic uses its own five- and nine-model controls and should be interpreted separately from the main evaluation. The support sizes are $k \in \{1, 4, 8, 16, 32, 64, 128, 192, 256, 512, 1024\}$. For $M \in \{5, 9, 22, 56\}$, we use five support draws, with one stored full-support result at $k = 1024$. Pools of five and nine models use three contexts per candidate; pools of 22 and 56 use one. At $M = 111$, each candidate is evaluated against the other 111 models. Oracle AUDG is computed on this support-size grid using Eq. 16.

Across these evaluated settings, mean oracle AUDG decreases from 0.6211 pp at $M = 5$ to 0.0397 pp at $M = 56$ and 0.0065 pp at $M = 111$. Over $M = 5$ to $M = 56$, the mean share of low-headroom candidates also increases from 72.0% to 95.8%. This trend indicates that the larger sampled pools leave fewer opportunities for beneficial admission under the evaluated routers. Because context sampling differs across pool sizes, these results characterize the evaluated configurations rather than isolate the effect of pool size alone.

Table 13: Admission headroom across old-pool sizes on EmbedLLM under the additional diagnostic protocol. Oracle AUDG is averaged equally over evaluated candidates within each method, then equally over kNN, UniRoute-Cluster, and LOCUS. All rows report evaluated results.

Old-pool size M	Mean oracle AUDG (pp)
5	0.6211
9	0.2878
22	0.1405
56	0.0397
111	0.0065

Candidate complementarity in the full pool. The preceding analyses depend on learned routing predictions. To examine overlap in observed correctness independently of a routing predictor, we hold out each EmbedLLM model and use the other 111 models as incumbents. Let $\mathcal{M}$ be the full set of 112 models. The candidate's marginal contribution to per-query oracle coverage is

$$U_c = \frac{100}{|\mathcal{Q}_{\text{test}}|} \sum_{x \in \mathcal{Q}_{\text{test}}} \left[\max_{m \in \mathcal{M}} y_m(x) - \max_{m \in \mathcal{M} \setminus \{c\}} y_m(x) \right]. \tag{17}$$

For binary correctness, U_c is the percentage of test queries answered correctly by candidate c and by none of the other 111 models.

Table 14 reports this additional coverage on the 3,000 test queries. Of the 112 candidates, 81 add no coverage and 110 add at most 0.1 pp. The mean is 0.0247 pp and the median is zero, indicating substantial correctness overlap within the full pool. Nevertheless, the maximum of 1.3667 pp shows that some candidates provide additional coverage. This quantity is independent of support size and differs from oracle AUDG. In particular, $U_c = 0$ does not imply that admitting c cannot improve a learned router: the candidate may answer queries correctly even when the original router fails to select a correct incumbent. Thus, U_c is not an upper bound on admission gains over that router.

Table 14: Additional oracle coverage when each EmbedLLM candidate is added to the other 111 models. The categories partition all 112 candidates according to U_c, the fraction of test queries answered correctly only by that candidate, expressed in percentage points. The 0.1 pp cutoff is descriptive.

Additional coverage	Candidates	Share (%)
None: $U_c = 0$	81	72.3
Small: $0 < U_c \leq 0.1$ pp	29	25.9
Above 0.1 pp: $U_c > 0.1$ pp	2	1.8
Total	112	100.0
Mean: 0.0247 pp	Median: 0 pp	Maximum: 1.3667 pp

Choice of pool sizes and evaluation implications. Our main evaluation uses $M \in \{5, 9\}$ to compare admission under two levels of competition while keeping pool sizes consistent across benchmarks. These settings contain positive oracle headroom alongside many low-headroom candidates, allowing evaluation of both beneficial-update retention and harmful-update rejection. They also permit variation in incumbent composition on RouterBench: holding out one of its 11 models leaves ten eligible incumbents, giving $\binom{10}{9} = 10$ possible nine-model pools but only one ten-model pool. The three sampled nine-model contexts contain two or three distinct compositions per candidate.

Together, these analyses show why held-out-model evaluations should report available admission headroom alongside realized gains. Small deployment gains can reflect limited opportunities under the constructed routers as well as limitations of the admission method. Future benchmarks should include candidates with measurable complementarity and chronological model arrivals, while retaining low-headroom cases to test the rejection of harmful updates.