

# Embedding Geometry Shapes LLM Routing: Tracing Failure Modes and Improving Generalization

Leonardo Torti<sup>1, \*</sup> Emanuele Rimoldi<sup>1,\*</sup> Tomer Galanti<sup>2</sup> Tomaso Poggio<sup>1</sup>  
Pierfrancesco Beneventano<sup>1</sup>

## Abstract

With the growing number of LLMs available, selecting which one to use for a prompt has become a key question. A large part of the routing literature relies on semantically embedding prompts into a vector space and then applying a lightweight classifier to the embedded representations. These methods perform robustly in-distribution but they struggle to generalize to unseen tasks. By studying the geometry of semantic embeddings, we trace a clear failure mode: features that determine the neighbors of unseen prompts are an unreliable proxy for LLM selection. We therefore replace the semantic embedding with the following map: we run a partial forward pass of Qwen2.5-7B-Instruct on the prompt and use the hidden state of its last token at a deep layer as the representation. When applying kNN and other simple classifiers to this new representation, we obtain a clear result in generalization: on average across cost penalties, their performance improves. Moreover, kNN and another simple classifier that we propose outperform every evaluated semantic-embedding baseline. Our two simple methods also beat a recent SoTA hidden-states based router on RouterBench.

## 1 Introduction

A large number of LLMs are now available and post-training your own open-weight model is becoming increasingly accessible. This raises questions about which **fundamental principles** should govern LLM selection, commonly called LLM routing, when addressing a specific task. We tackle the following problem:

*Given an evaluation suite, construct a router that selects the optimal model  
for the tasks seen at deployment time.*

The evaluation suite records the reward (e.g., accuracy) and cost of each available model on each query in a given set. This set of queries is usually made of different task families (Section 2). Using this data, the task is to train a classifier, called a router, that takes a prompt as input and selects which model to use.

We focus on compute-aware routing: methods that do not rely on reasoning. Most leading compute-aware approaches first map each prompt to a vector space using a semantic text encoder, then train a classifier on these representations to select a model.

We study their in-distribution (ID) performance, where training queries are representative of the deployment distribution, and out-of-distribution (OOD) performance, where that distribution changes. In-distribution, popular compute-aware methods appear robust, and many achieve similar performance, with simple methods such as kNN often matching more complex approaches [1, 2]. Out of distribution, the differences between methods become more pronounced, and most routers struggle.

**Our objective is to unveil:** (i) *the mechanism underlying the failure modes of these methods;*  
(ii) *what causes these more or less pronounced differences;* and  
(iii) *how to leverage those to improve the current methods.*

\*Main contributors. Correspondence to {pierb, leotorti}@mit.edu.

<sup>1</sup>Massachusetts Institute of Technology.

<sup>2</sup>Texas A&M University.

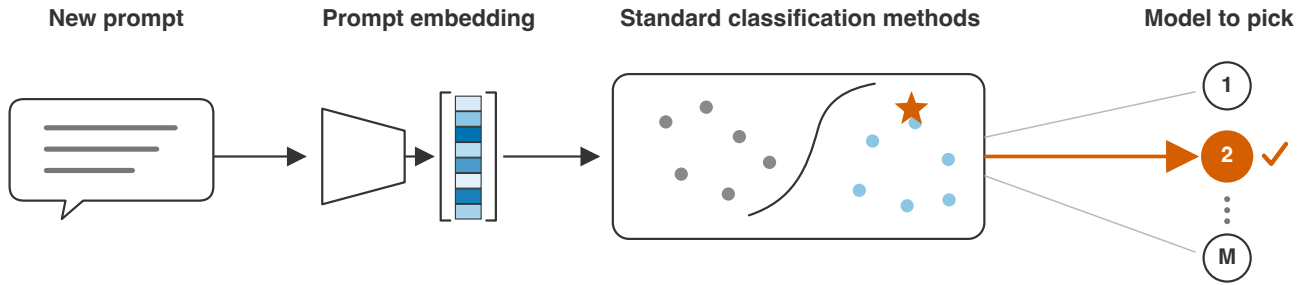


Figure 1: Compute-aware routing: a prompt is mapped to a representation, on which a standard classifier selects the model to use.

Through the geometry of query representations, we ask whether features shaping locality also inform model selection. Semantic proximity can retrieve queries that provide unreliable guidance on unseen tasks. This motivates replacing semantic embeddings with frozen LLM hidden states while keeping routing rules fixed. The resulting gains let simple routers compete with substantially more elaborate pipelines, and our geometric analysis links them to changes in the features shaping locality.

In summary, our contributions are:

- **ID: explaining why simple routers remain competitive.** Our geometric analysis shows that semantic embeddings are strongly organized by family and topic. This organization explains why simple neighborhood methods and rules that select a single model for each family or finer group can match more complex routers.
- **OOD: tracing the failure mode to misleading neighborhoods.** We show that queries from an unseen family can concentrate near a seen family close in topic or prompt format, but whose model-performance information poorly guides selection for those queries; correspondingly, semantic kNN does no better than always selecting the best single model. With the semantic encoder fixed, we introduce Family Adaptation: a multinomial logistic classifier weights seen families per query, and the router combines their average per-model rewards and costs. Bypassing potentially misleading nearest-family neighborhoods, this simple rule outperforms all evaluated external routers using GTE on RouterBench and all except Ridge on SPROUT.
- **Improving generalization through a change of representation.** Building on our analysis of semantic-embedding failure modes and prior work proposing routing methods with LLM hidden states, we replace semantic embeddings with a frozen LLM’s last-token state from a single internal layer, keeping routing algorithms unchanged. This improves held-out-family routing without further adaptation: kNN-100 and Family Adaptation each outperform every evaluated external semantic-embedding baseline in mean utility across cost penalties on RouterBench and SPROUT. Beyond these popular and strong baselines, we compare our simple, interpretable pipelines with SharedTrunkNet [3], a recent state-of-the-art router based on LLM hidden states. We use the same Qwen2.5-7B-Instruct encoder for all methods to isolate pipeline differences. Family Adaptation clearly outperforms SharedTrunkNet on RouterBench and trails it only slightly on SPROUT; even kNN-100 outperforms it on RouterBench and remains close behind on SPROUT.
- **Understanding the geometric change.** Keeping routing rules fixed lets us attribute the gains to the representation change. Compared with semantic embeddings, hidden-state embeddings much more often retrieve neighbors from training families sharing the unseen query’s answer format. This suggests a possible explanation for semantic routers’ strong ID performance: answer format is often shared within a family, so neighbors from that family share it even when other features shape locality. Holding out the query’s family breaks this association, exposing whether those features retrieve examples useful for routing. Still, greater answer-format agreement does not fully explain the gains.

## 2 Background and Notation

**The routing problem.** Let  $\mathcal{X}$  denote the space of prompts, which we also call queries, and let  $\mathcal{M} = \{m_1, \dots, m_M\}$  be a fixed pool of candidate models. We evaluate the routing problem on a benchmark  $\mathcal{S} = \{x_i\}_{i=1}^n \subset \mathcal{X}$ , which is a set of prompts. Routing benchmarks, including those studied here, are essentially compositions of several individual benchmarks, such as MMLU and GSM8K. We call each constituent benchmark a family and denote its queries by  $\mathcal{S}_j$ , so that  $\mathcal{S} = \bigcup_{j=1}^J \mathcal{S}_j$ , where  $J$  is the number of families. For every query  $x \in \mathcal{S}$ , we have a reward  $r(x, m) \in [0, 1]$ , measuring how well  $m$  solves  $x$ , and the cost  $c(x, m) \in \mathbb{R}_{\geq 0}$  of the response in US dollars. Binary accuracy is the special case  $r(x, m) \in \{0, 1\}$ . For a cost penalty  $\lambda \geq 0$ , the utility of answering  $x$  with  $m$  is

$$u_\lambda(x, m) = r(x, m) - \lambda c(x, m).$$

A router is a map  $\pi_\lambda : \mathcal{X} \rightarrow \mathcal{M}$  that, given a prompt, selects a model to answer it before observing any candidate’s response. The routing problem is to construct, from the training data composed of query-model pairs and their measured utilities, a router whose selections maximize the utility on the prompts encountered at deployment. Routers that predict rewards and costs select

$$\pi_\lambda(x) \in \arg \max_{m \in \mathcal{M}} [\hat{r}(x, m) - \lambda \hat{c}(x, m)],$$

where  $\hat{r}$  and  $\hat{c}$  are estimates obtained from training data. For a held-out set of prompts  $\mathcal{T}$ , we evaluate the empirical utility

$$\hat{U}_\lambda(\pi_\lambda; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} u_\lambda(x, \pi_\lambda(x)),$$

computed from the recorded reward and cost of each selected model, without generating new responses. We also report mean utility averaged equally across the 13 cost penalties.

**Embeddings.** A semantic embedding is a vector  $\mathcal{F}(x) \in \mathbb{R}^d$  produced by a frozen text encoder  $\mathcal{F} : \mathcal{X} \rightarrow \mathbb{R}^d$  trained to map texts with similar meaning to nearby vectors. The classification algorithm, e.g.,  $k$ -nearest neighbors, logistic regression, an MLP, or clustering, then maps  $\mathcal{F}(x)$  to a model in  $\mathcal{M}$ . Our geometric study uses GTE-Qwen2-7B-instruct [4, 5] as the main semantic encoder; additional results for all-MiniLM-L12-v2 [6] appear in Appendix B.3. In Section 4, we replace  $\mathcal{F}$  with the embedding map  $\mathcal{H} : \mathcal{X} \rightarrow \mathbb{R}^d$ , derived from Qwen2.5-7B-Instruct [7] hidden states and defined there in detail.

We will compare prompts through the cosine similarity of their representations, based on their normalized representation  $z(x) = \mathcal{F}(x)/\|\mathcal{F}(x)\|_2$ , and analogously for  $\mathcal{H}$ . The neighborhood  $\mathcal{N}_k(x; \mathcal{R})$  of a prompt  $x$  is the set of the  $k$  most similar prompts in a reference pool  $\mathcal{R}$ , excluding  $x$  itself. Routing uses training prompts as the reference pool for neighborhoods; geometric diagnostics specify their pools separately.

**Datasets and evaluation protocols.** We use two leading routing benchmarks. RouterBench [8] consists of 34,771 prompts from seven source datasets (MMLU, HellaSwag, GSM8K, ARC, WinoGrande, MBPP and MT-Bench), with recorded rewards and dollar costs for 11 models. SPROUT [9] consists of 43,887 prompts from six source datasets (GPQA, MATH, MMLU-Pro, MuSR, OpenHermes and RAGBench), with rewards computed for 13 models; we compute its costs from the released token counts and fixed model prices. These datasets also support finer subdivisions of families, defined by topic labels or other native labels. We will refer to these subdivisions as cells, or topics when the cell represents a topic. Families without finer labels form a single cell. Appendix A gives full descriptions, label provenance and preprocessing. For in-distribution evaluation, we use five family-stratified 70/30 train-test splits and average the scores across splits. For OOD evaluation, we hold out each family in turn, train on the remaining families, and pool the held-out-family predictions across prompts. Compared methods share the same outer splits; router hyperparameters are fixed or selected using training data only, with method-specific validation procedures detailed in Appendix A.

### 3 On Semantic Embeddings

#### 3.1 Related Work

Simple routers can match more complex methods when routing in distribution. Li [1] demonstrates the competitiveness of kNN, while Lu et al. [2] show that many different routing methods achieve nearly identical ID performance, see also [10]. Lee [11] further highlights the predictive value of task type. These findings motivate our geometric analysis, which identifies the features shaping semantic neighborhoods and explains why simple routing rules remain competitive in distribution. This understanding then provides the starting point for investigating what fails when queries come from unseen families.

Generalization to unseen tasks presents additional challenges. Shnitzer et al. [12] study routing to new tasks through model-correctness prediction. RouteLLM [13] learns from preference data and evaluates generalization to other benchmarks and model pairs. ProxRouter [14] improves robustness to outlier tasks by weighting model-performance estimates according to proximity and uncertainty; its KM-Prox variant is included in our comparisons. Our analysis examines the geometric mechanism behind generalization failures. While keeping the semantic embedding fixed, our diagnosis motivates our Family Adaptation method. Related aggregation strategies appear in UniRoute [15], which combines cluster-level model-performance estimates, and TRouter [16], which combines task-specific reward and cost predictions using an MLP. Family Adaptation is also inspired by distribution-weighted combinations in multiple-source adaptation [17, 18].

#### 3.2 In-Distribution Performance Matches Family-Cell Lookup

kNN matches or outperforms many evaluated routers, consistent with Lu et al. [2]. Even family and family-cell lookup remain competitive. To route a query, family lookup selects the model with the highest mean training utility within that query’s family; family-cell lookup applies the same rule within its cell. Both use the query’s known group label. See Figure 2. We further test whether kNN improves utility when its choices differ from family-cell lookup, finding small mean gains on both benchmarks (Appendix C.1).

#### 3.3 How Geometry Shapes ID Routing

GTE neighborhoods strongly preserve family identity, helping explain why neighborhood routing and group lookup achieve similar in-distribution performance (Section 3.2). We report results for GTE-Qwen2-7B-Instruct on RouterBench and SPROUT. Appendix B.3 repeats the family and subject geometry tests with the semantic encoder all-MiniLM-L12-v2. We see that 95.5% of the 100 nearest training examples to a prompt share its family on RouterBench and 97.3% on SPROUT. Within families, subjects (or in general cells) provide a further level of organization (Figure 3); tests can be found in Appendix B.3.

Many other query properties could shape locality. As one example, we examine difficulty, measured by the fraction of models achieving reward at least 0.5, with higher fractions indicating easier queries. Within

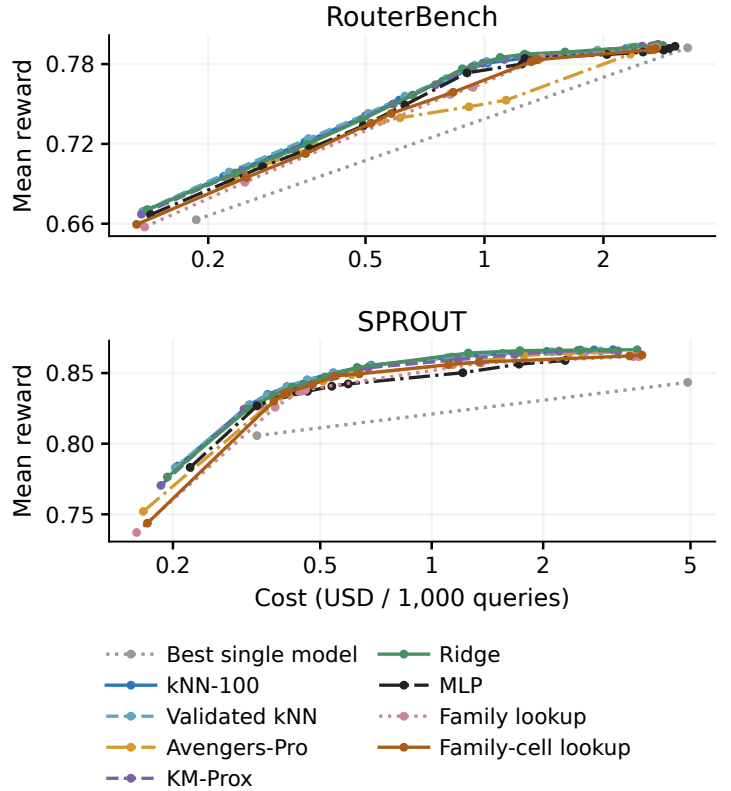


Figure 2: In-distribution reward–cost frontiers with GTE. Neighborhood methods and group lookup remain competitive with more elaborate routers.

each family-cell, neighbors have difficulty closer to that of the query than randomly selected examples do. The difference is small on RouterBench and larger on SPROUT (Appendix B.2).

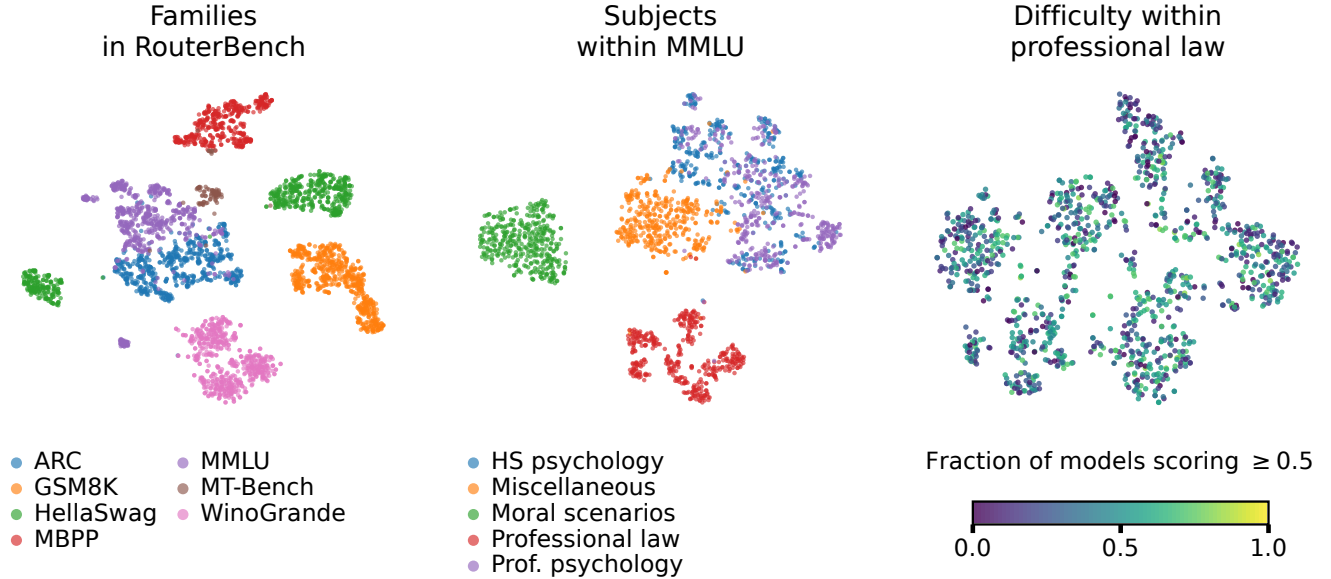


Figure 3: GTE geometry on RouterBench: families, subjects within MMLU, and difficulty within professional law. Each panel uses a separate t-SNE projection.

### 3.4 Held-Out Families Expose Misleading Semantic Neighborhoods

We saw that in-distribution routing can rely strongly on family structure, particularly for locality-based methods. We now examine what happens when the query’s family is absent from training. Removing all other members of a query’s family redirects its neighborhood to essentially another single family. On average, a single training family supplies 78.4% of each held-out query’s 100 nearest neighbors on RouterBench and 84.1% on SPROUT (Appendix B.4). Figure 4 shows which training families are retrieved.

Concentration alone does not establish a routing failure. We therefore test whether the families attracting neighbors provide useful evidence for LLM selection. For each held-out family and cost penalty, we predict model utilities using neighbors from each source family separately. We then ask whether the family that best predicts utility differences between candidate models is also the one supplying the most neighbors. These families rarely coincide (Appendix B.6).

Consistent with this mismatch, GTE kNN-100 achieves mean utility across cost penalties of 0.6615 on RouterBench and 0.7529 on SPROUT, compared with 0.6618 and 0.7575 obtained by simply always choosing the best single model selected from source training data separately for each penalty and outer split. Together, these results show that GTE neighborhood structure does not reliably identify useful model-selection evidence when the query’s family is absent from training. The representation comparison in Section 4 tests whether this failure is specific to semantic embeddings.

### 3.5 A Simple Classifier: Family Adaptation

On both benchmarks, Family Adaptation improves mean held-out-family utility over GTE kNN-100 while keeping the embedding fixed. Inspired by multiple-source domain adaptation [17], it combines performance information from several training families. We train a multinomial logistic classifier on the observed training-family labels and use its output probabilities  $p(f | x)$  as weights over those families for an unseen

query  $x$ . For each model, we combine the families’ mean training utilities:

$$\hat{u}_\lambda(x, m) = \sum_f p(f \mid x) (\bar{r}_{fm} - \lambda \bar{c}_{fm}).$$

Here,  $\bar{r}_{fm}$  and  $\bar{c}_{fm}$  are model  $m$ ’s mean reward and cost on training queries in family  $f$ . The router selects the model with the largest estimate. The classifier has balanced class weights and fixed regularization; the reward and cost averages use only training queries. Test family labels are not required.

This rule retains coarse performance information while allowing several families to contribute. Using GTE, it raises mean held-out utility over the penalty grid from 0.6615 to 0.7006 on RouterBench and from 0.7529 to 0.7603 on SPROUT relative to kNN-100 (Appendix D).

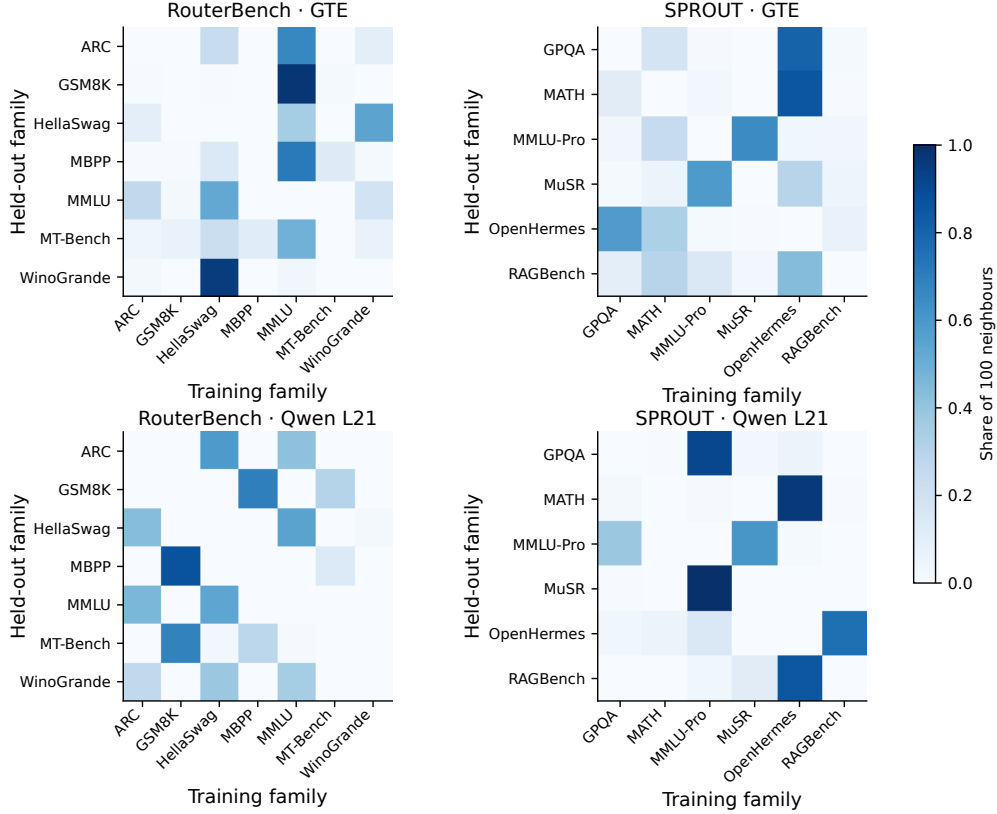


Figure 4: Changing the representation changes which training families supply held-out queries’ neighbors. Each matrix reports the family composition of the 100 nearest training queries under held-out-family evaluation.

## 4 Hidden-State Embeddings

Let  $x$  be a prompt. We now consider the embedding  $\mathcal{H} : \mathcal{X} \rightarrow \mathbb{R}^{3584}$ , where  $\mathcal{H}(x)$  is the hidden state of the last input token at layer 21 of Qwen2.5-7B-Instruct. This representation can be computed by running the forward pass only through layer 21, without generating an answer. L21 was fixed following exploratory RouterBench analysis; Appendix E shows results’ robustness across layers.

### 4.1 Related Work

Recent work has proposed methods based on LLM hidden states for correctness prediction and routing. Moreno Cencerrado et al. [19] predict correctness from question-only activations, while Lugoloobi et al. [20]

use pre-generation success predictions for model selection. Chen et al. [21] use activation statistics to classify prompt intent. Wu et al. [22] introduce ProbeDirichlet for escalation from a small model to a larger one using LLM hidden states. Varshney et al. [3] use a separate encoder’s prefill activations and an ensemble of neural predictors to estimate correctness across candidate models. Pichlmeier et al. [23] introduce Funnel Pooling, a learned aggregation of hidden states for LoRA-expert routing, trained on question–answer pairs. For queries in different languages, they observe that standard pooling can yield clusters organized more by language and format than by topic, and propose a pooling that produces more topic-aligned clusters. We build on this line of work from a different angle: rather than designing tailored predictors, we show that simple classifiers on hidden states can be competitive with more complex routers, and try to explain why.

## 4.2 Changing the Embedding Improves Routing

Replacing GTE with L21 raises mean held-out-family utility for every evaluated routing rule on both benchmarks except Avengers-Pro on RouterBench (Table 1). We compare the representations while keeping the queries, candidate models, routing rules and evaluation splits fixed.

Family structure persists: the training family supplying the most neighbors to each held-out query accounts, on average, for 72.7% of its 100 nearest neighbors on RouterBench and 92.0% on SPROUT, compared with 78.4% and 84.1% under  $\mathcal{F}$  (Appendix B.4).

We will see in the following paragraphs that the retrieved families are not only different but also a better proxy for routing on held-out families. A brief check in Appendix B.7 also shows that L21 predicts pairwise model-utility orderings more accurately than GTE.

Figure 5 compares identical routing rules across the two representations, while Table 1 reports mean utility across all cost penalties. kNN-100 improves at nine of thirteen penalties on RouterBench and all thirteen on SPROUT. Its mean utility rises from 0.6615 to 0.7007 and from 0.7529 to 0.7731, respectively. Results by held-out family, with equal-query and equal-family averages, and paired confidence intervals, are reported in Appendix D.3.

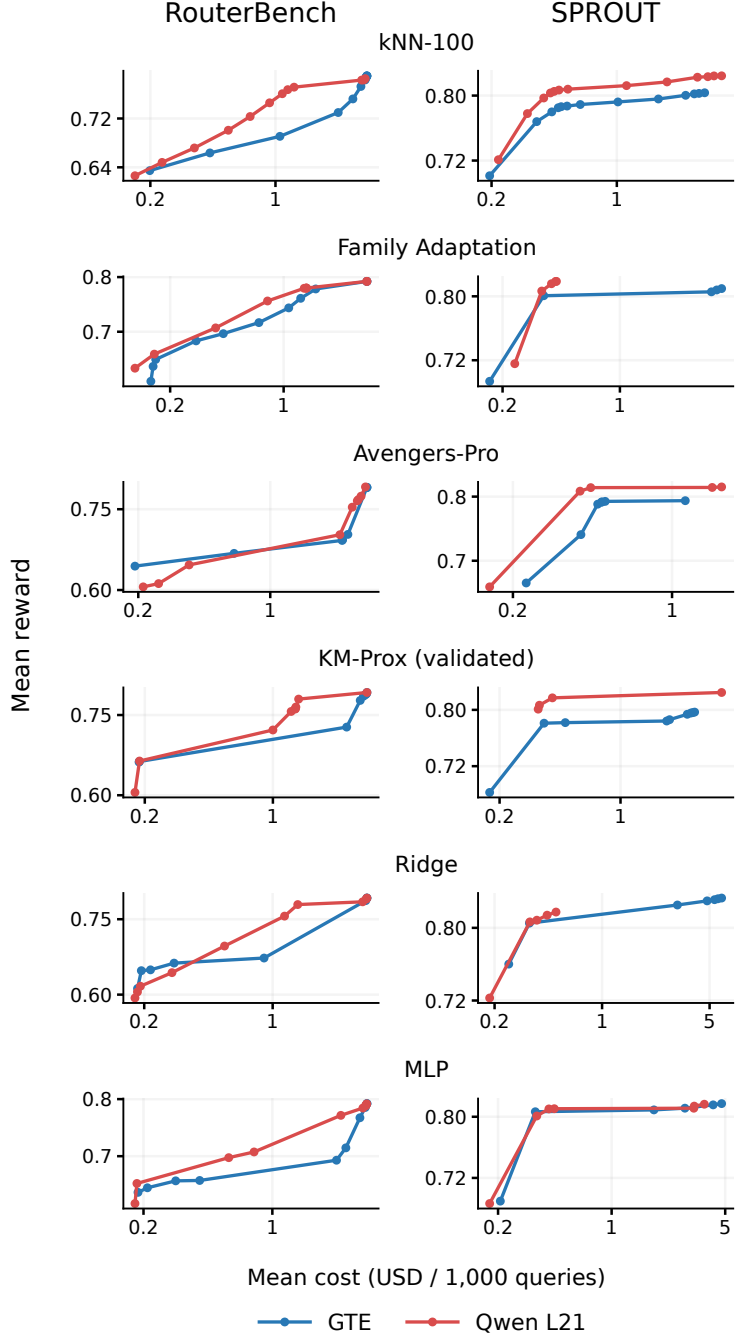


Figure 5: Matched routing rules under held-out-family evaluation: GTE in blue and L21 in red. The representation change improves mean utility for most evaluated rules; Table 1 gives the averages and Appendix D reports every penalty.

To put this improvement in perspective, the GTE baselines (excluding our introduced Family Adaptation) span mean utilities from 0.6566 to 0.6796 on RouterBench and 0.7439 to 0.7674 on SPROUT across the thirteen cost penalties, a spread of 0.0230 and 0.0235, respectively. The gain of kNN with the embedding change is 0.0391 and 0.0202, respectively.

The improvement extends beyond kNN: mean utility increases for every evaluated routing rule on both benchmarks, except Avengers-Pro on RouterBench. This suggests that hidden states provide more exploitable patterns for linear methods such as ridge regression and nonlinear methods such as MLP, as well as more informative neighbors. We emphasize kNN because it is the simplest method we measured and the most geometrically interpretable.

Table 1: L21 raises mean utility for every evaluated routing rule on both benchmarks except Avengers-Pro on RouterBench. Values average all 13 cost penalties under held-out-family evaluation; bold marks the higher value within each encoder pair. Full results with 95% confidence intervals are reported in Appendix D.

| Method              | RouterBench   |               | SPROUT |               |
|---------------------|---------------|---------------|--------|---------------|
|                     | GTE           | L21           | GTE    | L21           |
| Best single model   | 0.6618        |               | 0.7575 |               |
| kNN-100             | 0.6615        | <b>0.7007</b> | 0.7529 | <b>0.7731</b> |
| Validated kNN       | 0.6566        | <b>0.7034</b> | 0.7534 | <b>0.7739</b> |
| Family Adaptation   | 0.7006        | <b>0.7157</b> | 0.7603 | <b>0.7767</b> |
| Avengers-Pro        | <b>0.6662</b> | 0.6581        | 0.7439 | <b>0.7714</b> |
| KM-Prox (validated) | 0.6764        | <b>0.6984</b> | 0.7483 | <b>0.7764</b> |
| Ridge               | 0.6796        | <b>0.6902</b> | 0.7674 | <b>0.7712</b> |
| MLP                 | 0.6755        | <b>0.6918</b> | 0.7564 | <b>0.7667</b> |

The L21 Family Adaptation achieves the highest observed mean utility among the methods in Table 1 on both benchmarks. L21 kNN-100 also exceeds every evaluated external GTE baseline in mean utility on both benchmarks (Table 18).

Finally, we compare L21 kNN-100 and L21 Family Adaptation with the SharedTrunkNet router of Varshney et al. [3], using a common frozen Qwen2.5-7B-Instruct encoder. We retain its architecture and its own feature preprocessing and layer selection, while adapting feature selection and network training to our utility objective. We follow the validation procedure as closely as possible (Appendix A.3).

Table 2: Common-encoder comparison with utility-adapted SharedTrunkNet. The simple L21 routers have higher mean utility on RouterBench; SharedTrunkNet has higher mean utility on SPROUT. Values average all 13 cost penalties under held-out-family evaluation; bold marks the highest mean. Full results with 95% confidence intervals are reported in Appendix D.2.

| Method                | RouterBench   | SPROUT        |
|-----------------------|---------------|---------------|
| L21 kNN-100           | 0.7007        | 0.7731        |
| L21 Family Adaptation | <b>0.7157</b> | 0.7767        |
| SharedTrunkNet        | 0.6949        | <b>0.7829</b> |

Our methods achieve these results with minimal training: kNN-100 requires no fitted prediction model, while Family Adaptation fits only a linear classifier.

Furthermore, we note that most gains from the methods above come for  $\lambda > 0$ . This hints at the possibility, which we conjecture, that the gains come from a better representation of cost by the embedding space. We will address this later in Section 4.4.

### 4.3 What Is the Difference Between Hidden-State and Semantic Embeddings?

We now investigate which possible features could be responsible for the generalization improvement achieved with hidden-state embeddings. Here, answer format denotes whether a task is multiple-choice or free-form. When retrieving neighbors for a query from a held-out family, the share of neighbors matching the query’s answer format increases: from 76.4% to 99.9% on RouterBench and from 58.9% to 90.2% on SPROUT (Appendix B.5). It is also consistent with the objective of the two encoders: GTE is fine-tuned to produce embeddings for text similarity and retrieval, whereas Qwen2.5-Instruct is trained to generate responses to instructions [4, 7]. This finding also exposes a potential cause for why kNN and other methods work well in-distribution with semantic embeddings: within the same family, the answer format does not change in most benchmarks. Thus, family is a good proxy for answer format. When the query’s family is held out, the failure mode appears: semantic proximity no longer reliably retrieves useful evidence for model selection.

Restricting GTE retrieval to queries sharing the test query’s answer format recovers most of L21’s mean utility gain on RouterBench, but much less on SPROUT. We predict reward and cost from the 100 nearest eligible training queries, with the test query’s family excluded. This control changes both format agreement and the available source-family composition; it does not isolate answer format as the sole cause.

Table 3: Restricting GTE retrieval by answer format recovers most of L21’s mean utility gain on RouterBench and less on SPROUT. The control also changes source-family composition. Values average held-out-family kNN-100 utility across 13 cost penalties; full results and 95% intervals appear in Appendix F.2.

| Embedding / retrieval   | RouterBench   | SPROUT        |
|-------------------------|---------------|---------------|
| GTE                     | 0.6615        | 0.7529        |
| GTE, same answer format | 0.6974        | 0.7584        |
| L21                     | <b>0.7007</b> | <b>0.7731</b> |

L21 therefore provides useful routing information not recovered by this format-restricted GTE control. The additional query properties associated with the remaining gap remain to be identified.

### 4.4 Reward and Cost Contributions Differ Across Benchmarks

We now examine whether L21’s routing gains come from changes in reward estimates, cost estimates, or both. Prior work shows that prefill activations contain information predictive of answer correctness [19, 3]. For each held-out query and candidate model, we predict reward and cost separately by averaging the recorded outcomes of its 100 nearest training queries.

The prediction gains differ across benchmarks: L21 improves reward prediction on both, while cost prediction improves on RouterBench and slightly worsens on SPROUT (Table 4).

To separate these contributions, we replace GTE’s reward or cost estimates with L21’s in kNN-100 and compare mean utility across the thirteen cost penalties (Table 5). Prediction error and routing utility need not move together: on SPROUT, L21 cost estimates have slightly higher mean squared error but yield a small utility gain when paired with GTE reward estimates.

Table 4: Reduction in mean squared error from GTE to L21 (%). Negative values indicate higher error. Full results and 95% intervals appear in Appendix F.1.

| Prediction | RouterBench | SPROUT |
|------------|-------------|--------|
| Reward     | 13.6        | 22.2   |
| Cost       | 59.0        | −2.6   |

Table 5: Changing cost estimates yields the larger utility gain on RouterBench; changing reward estimates yields the larger gain on SPROUT. Full results and 95% intervals appear in Appendix F.3.

| Reward | Cost | RouterBench   | SPROUT        |
|--------|------|---------------|---------------|
| GTE    | GTE  | 0.6615        | 0.7529        |
| L21    | GTE  | 0.6754        | 0.7709        |
| GTE    | L21  | 0.6965        | 0.7560        |
| L21    | L21  | <b>0.7007</b> | <b>0.7731</b> |

## 5 Conclusion

We study how the geometry of prompt representations shapes compute-aware LLM routing. On RouterBench and SPROUT, family and topic structure help explain why simple routers remain competitive in distribution. Holding out a family exposes a failure mode: semantic proximity can retrieve examples whose model-performance information transfers poorly.

These findings give a practical response to the introductory question of constructing a router from an evaluation suite. Family Adaptation combines performance evidence across training families, while replacing semantic embeddings with frozen LLM hidden states improves mean held-out-family utility for most evaluated routing rules. This improves routing utility on the evaluated held-out families without requiring a more elaborate routing rule.

We thus identify a concrete failure mode, use it to motivate effective routing changes, and partially explain the differences between representations. The answer-format control recovers part of the representation gain, while reward and cost controls reveal benchmark-dependent sources of improvement. A complete explanation of the gains and a general characterization of optimal routing remain open.

## Limitations

At  $\lambda = 0$ , no evaluated router improves on the best-single-model baseline in our out-of-distribution comparison. Beating this baseline requires identifying queries on which the best model fails but models with lower average performance succeed. Since LLMs are stochastic response generators, cases where the best model fails but another succeeds may depend on the particular responses sampled. Collecting multiple responses for each model–query pair would help assess whether these routing opportunities are consistent across generations. This limitation is due to compute/cost constraints, and it is shared by the literature.

Our routing comparison also uses a single pair of semantic and hidden-state encoders, and the observed gains are demonstrated for the encoders considered. Whether these results extend to other encoder sizes remains unclear.

Finally, the format-restricted GTE control leaves part of L21’s gain unexplained, especially on SPROUT. Identifying the additional query properties associated with these gains remains future work.

## AI Use Statement

We used generative AI tools to assist with implementing and debugging methods, preparing and checking datasets, designing and refining experiments, refining hypothesis, and analyzing results. We also used these tools for literature search, preparing figures and tables, drafting and editing text, and formatting references. The authors take responsibility for the final content, methods, and conclusions.

## Acknowledgment

This work was supported by the Center for Brains, Minds and Machines (CBMM, NSF STC award CCF-1231216); and the McGovern Institute for Brain Research at MIT.

## References

- [1] Yang Li. Rethinking predictive modeling for LLM routing: When simple kNN beats complex learned routers. *arXiv preprint arXiv:2505.12601*, 2025. URL <https://arxiv.org/abs/2505.12601>. [Link].
- [2] Yifan Lu, Qiyue Zhang, Shenrun Zhang, Zhibo Yu, Zhuang Wang, Hanjie Chen, and Jiarong Xing. The routing plateau: Understanding the accuracy limits of LLM routers. *arXiv preprint arXiv:2606.07587*, 2026. URL <https://arxiv.org/abs/2606.07587>. [Link].
- [3] Tanay Varshney, Annie Surla, Michelle Xu, Gomathy Venkata Krishnan, Maximilian Jeblick, David Austin, Neal Vaidya, and Davide Onofrio. LLM router: Rethinking routing with prefill activations. *arXiv preprint arXiv:2603.20895*, 2026. URL <https://arxiv.org/abs/2603.20895>. [Link].
- [4] Alibaba NLP. gte-Qwen2-7B-instruct model card, 2024. [Link].
- [5] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023. URL <https://arxiv.org/abs/2308.03281>. [Link].
- [6] Sentence-Transformers. all-MiniLM-L12-v2 model card. Hugging Face, n.d. URL <https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2>. Accessed October 5, 2026. [Link].
- [7] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. [Link].
- [8] 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. In *ICML 2024 Workshop on Agentic Markets (AgenticMarkets)*, 2024. arXiv:2403.12031. [Link].
- [9] 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. [Link].
- [10] Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Bqing 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. URL <https://arxiv.org/abs/2601.07206>. [Link].
- [11] Janghoon Lee. Most of the LLM routing gap is task type. *arXiv preprint arXiv:2608.23023*, 2026. URL <https://arxiv.org/abs/2608.23023>. [Link].
- [12] 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. URL <https://arxiv.org/abs/2309.15789>. [Link].
- [13] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations (ICLR)*, 2025. URL [https://proceedings.iclr.cc/paper\\_files/paper/2025/file/5503a7c69d48a2f86fc00b3dc09de686-Paper-Conference.pdf](https://proceedings.iclr.cc/paper_files/paper/2025/file/5503a7c69d48a2f86fc00b3dc09de686-Paper-Conference.pdf). [Link].
- [14] Shivam Patel, Neharika Jali, Ankur Mallick, and Gauri Joshi. ProxRouter: Proximity-weighted LLM query routing for improved robustness to outliers. *arXiv preprint arXiv:2510.09852*, 2025. [Link].
- [15] Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient LLM inference. *arXiv preprint arXiv:2502.08773*, 2025. URL <https://arxiv.org/abs/2502.08773>. [Link].

- [16] Hui Liu, Bin Zou, Kecheng Chen, Jie Liu, Wenya Wang, and Haoliang Li. Task-aware LLM routing with multi-level task-profile-guided data synthesis for cold-start scenarios. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22047–22076. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1007. URL <https://aclanthology.org/2026.acl-long.1007/>. [Link].
- [17] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems*, volume 21, pages 1041–1048, 2008. URL [https://papers.nips.cc/paper\\_files/paper/2008/hash/0e65972dce68dad4d52d063967f0a705-Abstract.html](https://papers.nips.cc/paper_files/paper/2008/hash/0e65972dce68dad4d52d063967f0a705-Abstract.html). [Link].
- [18] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL [https://papers.neurips.cc/paper\\_files/paper/2018/hash/2e2079d63348233d91cad1fa9b1361e9-Abstract.html](https://papers.neurips.cc/paper_files/paper/2018/hash/2e2079d63348233d91cad1fa9b1361e9-Abstract.html). [Link].
- [19] Iván Vicente Moreno Cencerrado, Arnau Padrés Masdemont, Anton Gonzalvez Hawthorne, David Demitri Africa, and Lorenzo Pacchiardi. No answer needed: Predicting LLM answer accuracy from question-only linear probes. *arXiv preprint arXiv:2509.10625*, 2025. URL <https://arxiv.org/abs/2509.10625>. [Link].
- [20] William Lugoloobi, Thomas Foster, William Bankes, and Chris Russell. LLMs encode their failures: Predicting success from pre-generation activations. *arXiv preprint arXiv:2602.09924*, 2026. URL <https://arxiv.org/abs/2602.09924>. [Link].
- [21] Nan Chen, Zhouhao Yang, and Soufiane Hayou. Training-free versus training-based intent classification in LLMs: Accuracy, robustness, and failure modes. *arXiv preprint arXiv:2608.02415*, 2026. URL <https://arxiv.org/abs/2608.02415>. [Link].
- [22] Wanxing Wu, He Zhu, Yixia Li, Lei Yang, Jiehui Zhao, Hongru Wang, Jian Yang, Benyou Wang, Bingyi Jing, and Guanhua Chen. Towards fair and comprehensive evaluation of routers in collaborative LLM systems. *arXiv preprint arXiv:2602.11877*, 2026. URL <https://arxiv.org/abs/2602.11877>. [Link].
- [23] Josef Pichlmeier, Sebastian Nicolas Mueller, Jakob Sturm, Josef Dräxl, and Andre Luckow. What the router sees matters: Funnel pooling for fast, content driven expert routing. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 895–909. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.eacl-srw.66. URL <https://aclanthology.org/2026.eacl-srw.66/>. [Link].
- [24] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. URL <https://arxiv.org/abs/2009.03300>. [Link].
- [25] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1472. URL <https://aclanthology.org/P19-1472/>. [Link].
- [26] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL <https://arxiv.org/abs/2110.14168>. [Link].

- [27] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL <https://arxiv.org/abs/1803.05457>. [Link].
- [28] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019. URL <https://arxiv.org/abs/1907.10641>. [Link].
- [29] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL <https://arxiv.org/abs/2108.07732>. [Link].
- [30] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*, 2023. URL <https://arxiv.org/abs/2306.05685>. [Link].
- [31] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL <https://arxiv.org/abs/2311.12022>. [Link].
- [32] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL <https://arxiv.org/abs/2103.03874>. [Link].
- [33] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024. URL <https://arxiv.org/abs/2406.01574>. [Link].
- [34] Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. *arXiv preprint arXiv:2310.16049*, 2023. URL <https://arxiv.org/abs/2310.16049>. [Link].
- [35] Teknium. OpenHermes 2.5: An open dataset of synthetic data for generalist LLM assistants, 2023. URL <https://huggingface.co/datasets/teknium/OpenHermes-2.5>. [Link].
- [36] Robert Friel, Masha Belyi, and Atindriyo Sanyal. RAGBench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2024. URL <https://arxiv.org/abs/2407.11005>. [Link].
- [37] Yiqun Zhang, Hao Li, Jianhao Chen, Hangfan Zhang, Peng Ye, Lei Bai, and Shuyue Hu. Beyond GPT-5: Making LLMs cheaper and better via performance-efficiency optimized routing. *arXiv preprint arXiv:2508.12631*, 2025. [Link].
- [38] Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552. URL <https://doi.org/10.1214/aos/1176344552>. [Link].
- [39] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL <https://jmlr.org/papers/v9/vandermaaten08a.html>. [Link].

- [40] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(95):2837–2854, 2010. URL <https://jmlr.org/papers/v11/vinh10a.html>. [Link].
- [41] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985. doi: 10.1007/BF01908075. URL <https://doi.org/10.1007/BF01908075>. [Link].
- [42] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL <https://aclanthology.org/D19-1410/>. [Link].
- [43] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc., 2020. URL [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf). [Link].

# Guide to the Appendices

Evaluation and preprocessing are documented in [Protocol Details](#); uncertainty estimates are specified under [Confidence Intervals](#).

- **ID competitiveness:** family and subject structure, decisions beyond group lookup, and lookup with predicted group labels.
- **OOD failure and Family Adaptation:** neighborhood concentration, model-performance transfer, and full routing results.
- **Representation gains:** results across all penalties, the SharedTrunkNet protocol, and layer choice and robustness.
- **Sources of improvement:** answer-format agreement, followed by the format-restriction and reward-cost controls.

## A Protocol Details

### A.1 Data and evaluation protocol

**Data.** RouterBench combines MMLU [24], HellaSwag [25], GSM8K [26], ARC [27], WinoGrande [28], MBPP [29], and MT-Bench [30]. SPROUT combines GPQA [31], MATH [32], MMLU-Pro [33], MuSR [34], OpenHermes [35], and RAGBench [36]. We use the RouterBench zero-shot release, specifically `routerbench_0shot.pkl`, and SPROUT at fixed dataset revisions.<sup>1</sup>

RouterBench is restricted to the seven families listed above. Its released prompt parts are joined with two newline characters; SPROUT prompts are used as released. A query is eligible only if its prompt is nonempty and every candidate model has a finite reward in  $[0, 1]$  and a finite, nonnegative cost. We exclude queries exceeding 8,192 tokens under either main encoder’s tokenizer, counting GTE’s final EOS token. Exact duplicate prompt strings are removed across each benchmark, keeping the first eligible occurrence. Inputs are never truncated. We use both benchmarks’ released rewards without rescaling. RouterBench costs use the released values. For SPROUT, the cost in USD per query is

$$c(x, m) = \frac{n_{\text{in}}(x, m)p_{\text{in}}(m) + n_{\text{out}}(x, m)p_{\text{out}}(m)}{10^6},$$

where  $n_{\text{in}}$  and  $n_{\text{out}}$  are the released input and output token counts, and  $p_{\text{in}}$  and  $p_{\text{out}}$  are the fixed prices in USD per million tokens listed in Table 6. These prices come from the CARROT release.<sup>2</sup>

---

<sup>1</sup>Dataset snapshots: [RouterBench \(revision 7840214\)](#) and [SPROUT \(revision d54e3cd\)](#).

<sup>2</sup><https://huggingface.co/CARROT-LLM-Routing>

Table 6: Fixed SPROUT prices in USD per million tokens.

| Model                      | Input | Output |
|----------------------------|-------|--------|
| Claude-3.5-Sonnet-v1       | 3.00  | 15.00  |
| Titan Text Premier v1      | 0.50  | 1.50   |
| GPT-4o                     | 2.50  | 10.00  |
| GPT-4o-mini                | 0.15  | 0.60   |
| Granite-3-2B Instruct      | 0.10  | 0.10   |
| Granite-3-8B Instruct      | 0.20  | 0.20   |
| Llama-3.1-8B Instruct      | 0.20  | 0.20   |
| Llama-3.1-70B Instruct     | 0.90  | 0.90   |
| Llama-3.2-1B Instruct      | 0.06  | 0.06   |
| Llama-3.2-3B Instruct      | 0.06  | 0.06   |
| Llama-3.3-70B Instruct     | 0.90  | 0.90   |
| Llama-3-405B Instruct      | 3.50  | 3.50   |
| Mixtral-8x7B-Instruct-v0.1 | 0.60  | 0.60   |

The retained datasets contain 34,771 RouterBench queries and 43,887 SPROUT queries. RouterBench uses 11 candidate models: WizardLM-13B-V1.2, Claude Instant v1, Claude v1, Claude v2, GPT-3.5-Turbo-1106, GPT-4-1106-preview, Code Llama Instruct 34B, Llama-2-70B-Chat, Mistral-7B-Chat, Mixtral-8x7B-Chat, and Yi-34B-Chat. SPROUT uses 13: Claude-3.5-Sonnet-v1, Titan Text Premier v1, GPT-4o, GPT-4o-mini, Granite-3-2B and 3-8B Instruct, Llama-3.1-8B and 70B Instruct, Llama-3.2-1B and 3B Instruct, Llama-3.3-70B Instruct, Llama-3-405B Instruct, and Mixtral-8x7B-Instruct-v0.1.

Family-cells use MMLU subjects and MT-Bench evaluation labels on RouterBench. On SPROUT, MATH subjects are recovered by matching question text after whitespace normalization; MMLU-Pro subjects are recovered using the released question indices with a text check. MuSR and RAGBench cells use released dataset identifiers. All other queries use the whole family as their cell, including OpenHermes and queries without a matched subject. Task and source identifiers are not interpreted as semantic subjects.

**Splits and selection.** SPROUT’s released training, validation and test partitions are pooled before preprocessing and constructing the evaluation splits below. Five in-distribution splits reserve 30% of each family for testing; 20% of the outer training set is used for validation where required. Held-out evaluation has seven outer folds for RouterBench and six for SPROUT. Within each fold, validation holds out each remaining training family in turn, except for MLP and SharedTrunkNet, whose procedures are specified in Appendices A.2 and A.3. Outside these exceptions, hyperparameters maximize mean routing utility over equally weighted validation queries. The test family is excluded from router-hyperparameter selection. The historical choice of the fixed L21 representation is described separately in Appendix E.

**Encoder inputs.** Both main encoders receive the prompts without an added instruction prefix or chat template. GTE uses its published bidirectional model and the final-layer state at the EOS token, appending EOS if it is not already present. Qwen2.5-7B-Instruct receives no added special tokens or EOS; L21 is the last nonpadding input token’s state after transformer block 21, corresponding to `hidden_states[21]`. Both encoders run in evaluation mode with bfloat16 weights. Extracted vectors are converted to float32 and unit-normalized before routing or geometric analysis; SharedTrunkNet uses the preprocessing specified in Appendix A.3.

### Cost penalties.

RouterBench: {0, 1, 2, 5, 10, 20, 30, 50, 75, 100, 150, 250, 500},  
 SPROUT: {0, 0.5, 1, 2, 5, 10, 25, 50, 75, 100, 150, 250, 500}.

Mean utility across penalties is the arithmetic average of these thirteen evaluated values; it depends on this grid.

**Simple routing rules.** kNN estimates each model’s reward and cost by uniformly averaging its recorded outcomes over the nearest training queries by cosine similarity, then selects the model with the highest estimated utility. kNN-100 fixes  $k = 100$ . Validated kNN selects  $k \in \{5, 10, 20, 40, 100, 200\}$  per penalty. Family Adaptation uses multinomial logistic regression with balanced class weights, inverse regularization strength  $C = 1$  and at most 3,000 iterations, without coordinate standardization. Ridge predicts per-model utility with an unpenalized intercept and selects its regularization strength from  $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$  by validation routing utility, separately for each penalty. It uses the same normalized vectors, without PCA. MLP training and validation are specified in Appendix A.2.

We implement Avengers-Pro following Zhang et al. [37] and KM-Prox following Patel et al. [14], using the utility objective and source-only validation protocol above. KM-Prox uses inverse temperature 20 in its fixed version and selects from  $\{0, 1, 5, 10, 20, 50, 100\}$  in its validated version.

**Aggregation and curves.** In-distribution results average five split-level query means. Held-out results pool all test queries with equal query weights, so larger families contribute more to the mean. A displayed frontier retains only measured points that no other point of the same method dominates in both reward and cost. No interpolation is used for utility comparisons. The tables retain all penalties, including dominated points. Cost axes use a symmetric logarithmic scale. Routing confidence intervals are defined below.

**Confidence intervals.** Where reported, routing confidence intervals are pointwise 95% percentile bootstrap intervals [38] from 2,000 paired samples of test queries. In OOD evaluation, queries are resampled within each held-out family, preserving family sizes. In in-distribution evaluation, we resample unique test queries and keep their repeated appearances across splits together. Each appearance retains its weight in the original five-split mean, with weights renormalized in each sample. The same samples are used across methods and penalties; mean-utility intervals are computed after averaging the thirteen penalties within each sample. Intervals are conditional on the evaluated families, splits and fitted routers, and do not include retraining variability or uncertainty over new families. The lookup-disagreement analysis uses 1,000 paired samples, as specified in Appendix C.1.

## A.2 MLP training and validation

For each cost penalty, MLP fits a separate scalar regressor for each candidate model to its recorded utility  $r(x, m) - \lambda c(x, m)$ . It selects the model with the largest predicted utility. Inputs are the unit-normalized GTE or L21 embeddings, without PCA or coordinate standardization; output values are unrestricted. Table 7 gives the training settings.

Table 7: MLP candidate grid and fixed training settings.

| Setting                     | Values                                                         |
|-----------------------------|----------------------------------------------------------------|
| Hidden layers               | (100, 100) or (100, 100, 100), with ReLU                       |
| L2 regularization $\alpha$  | $10^{-4}, 10^{-2}, 1, 10, 100$                                 |
| Training objective          | Squared error with L2 weight regularization                    |
| Optimizer                   | Adam; learning rate $10^{-3}$                                  |
| Adam parameters             | $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$           |
| Batch size / maximum epochs | 200 / 200                                                      |
| Early stopping              | Enabled; internal validation fraction 0.1                      |
| Stopping criterion          | Validation $R^2$ ; tolerance $10^{-4}$ , patience parameter 10 |

The two architectures and five regularization values give ten candidate configurations at each of thirteen penalties: 130 candidate settings per outer evaluation. A configuration is shared across the candidate-model regressors, but their weights are trained separately. We select the configuration by the realized utility of its routing decisions on validation queries, separately for each benchmark, encoder, outer split and penalty.

For ID evaluation, selection uses the 20% validation split of the outer training data. For held-out-family evaluation, it uses two predetermined source families: each is held out in turn while fitting on the other source families, and validation utilities are pooled with equal query weights. The two validation families are fixed before fitting and are identical for GTE and L21. The outer test family is excluded throughout.

After selection, only the winning configuration is refitted on the outer training set. Every fit, including the final refit, reserves its own random 10% of fitting queries for early stopping and restores the weights with the highest validation  $R^2$ ; this internal subset is separate from the queries used to select the routing configuration. The same procedure is used for both encoders.

### A.3 SharedTrunkNet training and validation

**Comparison and inputs.** We adapt the released SharedTrunkNet pipeline of Varshney et al. [3] to predict utility  $u_\lambda(x, m) = r(x, m) - \lambda c(x, m)$ . All compared methods use the same raw prompts, frozen Qwen2.5-7B-Instruct checkpoint, candidate models, recorded outcomes, outer splits and penalty grids. kNN-100 and Family Adaptation use unit-normalized L21 states as in the main text. SharedTrunkNet retains its own feature preprocessing and network. The comparison evaluates these routing pipelines with a common encoder and utility objective; it does not reproduce the original paper’s model pools or absolute scores.

**Features and layer selection.** For each layer  $\ell \in \{15, \dots, 28\}$ , we use the last input token’s hidden state. We standardize its coordinates and fit an unwhitened PCA with 100 components on the outer training queries only. These transforms remain fixed during feature selection. For each candidate model and penalty, a Ridge probe with fixed  $\alpha = 1$  and an intercept predicts utility in five shuffled folds of the outer training queries. We apply the released ternary layer-search rule to negative mean validation MSE. This selects a layer separately for every model and penalty; it is a heuristic search, not an exhaustive maximization. The selected 100-dimensional blocks are concatenated in candidate-model order, giving  $100M$  inputs for  $M$  candidate models. Blocks are repeated when models select the same layer. The five folds split training queries, not whole families; the outer test family is excluded from all preprocessing, selection and fitting.

**Network and utility objective.** We use the released shared network: hidden widths 256 and 128, ReLU activations, dropout 0.3 and 0.2, and  $M$  linear outputs. Training uses Adam with learning rate  $10^{-3}$ , weight decay  $10^{-4}$ , batches of 512, at most 150 epochs and early stopping with patience 15 and improvement threshold  $10^{-4}$ . Each of ten independently initialized networks reserves a random 15% of the outer training queries for early stopping; the five networks with lowest validation MSE are retained. Their raw outputs are averaged, and the model with the largest predicted utility is selected. We train a separate ensemble for each penalty. The utility adaptation replaces logistic-regression/AUC feature selection with Ridge/MSE, and binary cross-entropy with utility MSE in the network. Outputs are not passed through a sigmoid, and cost is not subtracted again. The architecture, optimizer, stopping rule and ten-to-five ensemble procedure are retained.

**Alternative selection procedures.** The fixed-L21 control repeats the L21 PCA block for every candidate model. The common-layer control selects one layer for all models and penalties by mean validation routing utility over the thirteen penalties. It trains one screening network per layer and penalty using the two source-family holdouts defined for MLP in Appendix A.2, fitting preprocessing within each training fold. Both controls use the same final shared architecture and ten-to-five ensemble rule, with the common layer’s block repeated  $M$  times. Thus, these controls compare selection procedures and feature assembly.

**Other reproducibility information.** ID outer splits use seeds 0–4 and their inner validation splits use 10000–10004, respectively. MLP uses seed 1234 for initialization, internal early-stopping splits and

batch shuffling; its two source-family validation folds are fixed before fitting, identically for both encoders.<sup>3</sup> Avengers-Pro jointly selects the number of clusters from  $\{32, 64, 128\}$ , retrieved clusters from  $\{1, 3\}$  and reward-cost mixing weight from  $\{0, 0.01, \dots, 1\}$  at each penalty, with cluster-weighting parameter  $\beta = 9$ . KM-Prox uses 32 clusters. Both use ordinary k-means with ten initializations and seed 42. SharedTrunkNet uses seed 42 for PCA, probe-validation folds and common-layer screening, and seeds 42–51 for ensemble members. Geometric sampling, PCA, t-SNE and k-means use seed 42; t-SNE uses PCA initialization, automatic learning rate and 1,000 iterations. Routing and cost-reward bootstrap analyses use seed 20261005; lookup-disagreement intervals use seed 42. The software environment uses Python 3.11.16, PyTorch 2.5.1, Transformers 4.44.2, NumPy 1.26.4 and scikit-learn 1.5.2. Encoder revisions are GTE a8d08b3, Qwen a09a354 and MiniLM a50ef00.

## A.4 Computing infrastructure and asset documentation

**Computing infrastructure.** Experiments ran on a shared Linux cluster managed by Slurm, with AMD EPYC and Intel Xeon CPUs. Recorded GPU hardware includes NVIDIA A100 80GB and H100 80GB devices. Neural-router workers requested one GPU and four allocated CPUs, with Slurm host-memory requests of 32G for MLP and 24G for SharedTrunkNet. CPU baseline jobs requested eight allocated CPUs and 48G host memory; PCA preprocessing used sixteen allocated CPUs. These are job allocations, not minimum hardware requirements. Software versions are specified above.

**Licenses and access.** The GTE, Qwen and MiniLM checkpoints and the reused NVIDIA router code use Apache 2.0; RouterBench software uses MIT. The RouterBench and SPROUT dataset cards do not specify a separate dataset license. Software licenses do not establish the terms of all underlying prompts or generated responses.

The constituent dataset cards declare MIT for MMLU, HellaSwag, GSM8K and MMLU-Pro; CC BY-SA 4.0 for ARC; and CC BY 4.0 for MBPP, GPQA, MuSR and RAGBench. The MATH and WinoGrande repositories publish MIT and Apache 2.0 software licenses, respectively. We do not infer additional question-level permissions from these, from the license of a separate MT-Bench annotation release, or from the availability of OpenHermes, whose card does not specify a license. The Avengers-Pro README declares MIT, although its linked license file was unavailable; we did not identify a repository-wide license in the consulted LLMRouterBench release. Applicable upstream terms remain in force. The anonymous package will preserve required third-party notices and provide download and preparation instructions rather than redistribute benchmark text or pretrained weights. This also respects GPQA’s request not to publish question examples online.

**Consent information.** We reuse released benchmarks and collect no new data from human participants. The releases do not provide consent information for every record. OpenHermes lists LMSYS-Chat-1M among its sources; the LMSYS documentation reports consent through the collection website’s terms of use. This does not establish the consent status of every OpenHermes example in SPROUT, whose individual upstream sources were not audited here.

**Potentially sensitive content.** The reused text may contain personally identifying or offensive content. For example, LMSYS reports efforts to remove identifying information but also retains unsafe conversations. We do not claim that every benchmark example is free of such content or that we performed a comprehensive content audit. The package will link to the original data releases instead of republishing their raw prompts and responses.

<sup>3</sup>For each held-out family, the two MLP validation families are: RouterBench: ARC or GSM8K  $\rightarrow$  HellaSwag, MMLU; HellaSwag  $\rightarrow$  MT-Bench, GSM8K; MBPP  $\rightarrow$  WinoGrande, HellaSwag; MMLU  $\rightarrow$  WinoGrande, MBPP; MT-Bench  $\rightarrow$  HellaSwag, MBPP; WinoGrande  $\rightarrow$  ARC, MMLU. SPROUT: GPQA  $\rightarrow$  MMLU-Pro, OpenHermes; MATH  $\rightarrow$  MMLU-Pro, MuSR; MMLU-Pro  $\rightarrow$  GPQA, RAGBench; MuSR  $\rightarrow$  OpenHermes, RAGBench; OpenHermes  $\rightarrow$  MuSR, MMLU-Pro; RAGBench  $\rightarrow$  OpenHermes, MATH.

## B Geometric Measurements

### B.1 Neighborhood measures and visualization

Neighbor comparisons use original unit-normalized embeddings. Same-family share is the fraction of retrieved examples from the query’s family; its random comparison is the fraction of eligible reference queries from that family, averaged over queries. Largest-family share is the largest fraction of a query’s neighbors belonging to any one family, averaged over queries.

The three-panel visualization uses a fixed t-SNE projection [39] of each displayed set, after PCA to at most 50 dimensions. It samples at most 600 queries per family, at most 400 per displayed MMLU subject, and at most 1,200 in professional law. The middle panel uses the five largest MMLU subjects by query count; the last panel uses professional law, the largest subject. Each panel uses perplexity 30 and a fixed random state. These are descriptive projections of the dataset, not estimates of test performance.

### B.2 Difficulty within family-cells

For the difficulty comparison, we use the solve fraction: the fraction of candidate models achieving reward at least 0.5 on a query. Higher solve fractions indicate easier queries. We use GTE embeddings and the five ID splits. For each test query, we retrieve its ten nearest training queries from the same family-cell and average their absolute differences in solve fraction from the query. The random comparison averages this difference over all training queries in the same cell. Cells with fewer than ten eligible training queries are excluded. Averaging both quantities equally over eligible test-query appearances across the five splits gives  $D_{\text{neighbors}}$  and  $D_{\text{random}}$ . Table 8 reports these differences and their relative reduction,  $100(1 - D_{\text{neighbors}}/D_{\text{random}})\%$ .

Table 8: Mean absolute difference in solve fraction within family-cells.

| Benchmark   | Neighbors | Random selection | Relative reduction |
|-------------|-----------|------------------|--------------------|
| RouterBench | 0.2463    | 0.2565           | 4.01%              |
| SPROUT      | 0.1861    | 0.2467           | 24.58%             |

These results measure similarity in observed difficulty, not the utility of a routing decision.

On SPROUT, the additional similarity may partly reflect finer groups not identified by the available labels. If these groups differ in average solve fraction and are separated in the embedding space, neighbors can have similar difficulty without the space being organized directly by difficulty.

### B.3 Family and subject structure

For each test query, we measure the share of its 100 nearest training queries belonging to its family. Table 9 reports full-data GTE results, averaged over queries and the five ID splits. Random selection samples uniformly from all training queries.

Table 9: Full-data GTE same-family share among 100 neighbors.

| Benchmark   | Nearest neighbors | Random selection |
|-------------|-------------------|------------------|
| RouterBench | 95.51%            | 29.55%           |
| SPROUT      | 97.30%            | 31.88%           |

We test subject organization in the original normalized embedding space, separately within each family with at least two known subjects. On each of the five ID splits, ordinary k-means is fitted to training embeddings with one cluster per observed training subject and ten initializations. Test queries are assigned to the nearest fitted center. Subject labels determine the number of clusters and are used to

score the assignments; they do not enter the clustering objective. We report normalized mutual information (NMI; larger values indicate greater agreement) [40] and adjusted Rand index (ARI; adjusted for chance agreement) [41], averaged over the five splits. Missing subject labels are excluded. Table 10 covers all eligible families with native subject labels: MMLU on RouterBench, and MATH and MMLU-Pro on SPROUT. Native task or source identifiers are not treated as subjects.

We repeat the locality and subject-recovery tests with `sentence-transformers/all-MiniLM-L12-v2` [6, 42, 43], using attention-mask mean pooling. Prompts exceeding its 512-token limit, including special tokens, are excluded rather than truncated. Both encoders use the same retained queries and splits: 34,659 RouterBench queries and 41,324 SPROUT queries. All MuSR prompts exceed this limit, leaving five SPROUT families.

Table 10: Recovery of native subjects within families on test queries. Matched columns use the same MiniLM-compatible queries and splits.

| Benchmark   | Family   | Subjects | Full-data GTE |       | Matched subset NMI |        |
|-------------|----------|----------|---------------|-------|--------------------|--------|
|             |          |          | NMI           | ARI   | GTE                | MiniLM |
| RouterBench | MMLU     | 57       | 0.722         | 0.418 | 0.726              | 0.640  |
| SPROUT      | MATH     | 7        | 0.327         | 0.246 | 0.334              | 0.296  |
| SPROUT      | MMLU-Pro | 14       | 0.286         | 0.172 | 0.280              | 0.505  |

For each test query, we measure the same-family share among its 100 nearest training examples by cosine similarity, then average over queries and the five splits. Both encoders retain strong family locality (Table 11); subject recovery, measured by the k-means procedure above, varies by family (Table 10).

Table 11: Same-family share among 100 nearest training queries, on the matched subset. Random selection samples uniformly from all training queries.

| Benchmark   | GTE    | MiniLM | Random selection |
|-------------|--------|--------|------------------|
| RouterBench | 95.48% | 90.43% | 29.49%           |
| SPROUT      | 97.33% | 90.74% | 34.53%           |

## B.4 Family concentration in held-out neighborhoods

For each query, we exclude its entire family and retrieve the 100 nearest remaining queries by cosine similarity. We measure the largest fraction of these neighbors belonging to any one training family, then average this fraction equally over all queries. GTE and L21 use the same queries and eligible training pools.

Table 12: Mean largest-family share among 100 neighbors, with the query’s family excluded.

| Benchmark   | GTE   | L21   |
|-------------|-------|-------|
| RouterBench | 78.4% | 72.7% |
| SPROUT      | 84.1% | 92.0% |

## B.5 Answer-format agreement in held-out neighborhoods

We test whether changing the encoder changes the answer formats represented among a query’s nearest neighbors. We use two groups, assigned from the benchmark’s task families. On RouterBench, ARC, HellaSwag, MMLU and WinoGrande are multiple-choice; GSM8K, MBPP and MT-Bench are free-form. On SPROUT, GPQA, MMLU-Pro and MuSR are multiple-choice; MATH, OpenHermes and RAGBench are free-form. These are coarse task labels, not classifications of the models’ generated answers.

For each query, we exclude its entire family and retrieve the 100 nearest remaining queries by cosine similarity. We measure the fraction whose answer-format label matches the query’s. We use identical queries and eligible neighbors for GTE and L21, and average over all held-out queries with equal weights. The random-selection column is the expected matching fraction when sampling uniformly from the same eligible queries. The analysis covers all seven RouterBench families (34,771 queries) and all six SPROUT families (43,887 queries).

Table 13: Percentage of held-out neighbors sharing the query’s answer-format label.

| Benchmark   | Randomly selected queries | GTE   | L21   |
|-------------|---------------------------|-------|-------|
| RouterBench | 51.2%                     | 76.4% | 99.9% |
| SPROUT      | 38.9%                     | 58.9% | 90.2% |

L21 neighborhoods follow this answer-format grouping much more closely, even when the query’s own family is unavailable. This measurement establishes a change in neighborhood composition; its contribution to routing utility is a separate question.

## B.6 Geometric Proximity and Model Performance

We test whether the family supplying the most neighbors also best predicts utility differences between candidate models. For each held-out family and each remaining training family, we retrieve the nearest 100 queries within that training family using GTE cosine similarity. If the family has fewer than 100 queries, we use all of them; this applies to MT-Bench, which has 80 queries. We average each candidate model’s recorded rewards and costs over these neighbors, giving predicted utility  $\hat{u}_\lambda(x, m) = \hat{r}(x, m) - \lambda \hat{c}(x, m)$  at every prescribed cost penalty.

We center both predicted and observed utilities across the  $M$  candidate models, separately for each query:

$$\tilde{u}_\lambda(x, m) = u_\lambda(x, m) - \frac{1}{M} \sum_{j=1}^M u_\lambda(x, m_j).$$

For each training family, we measure the mean squared error between its predicted and observed centered utilities, averaged over models and queries in the held-out family. This tests relative model preferences. The family with the lowest error is identified retrospectively for this diagnostic.

We then rank the training families by their shares of the unrestricted 100-neighbor set, averaged over queries in the held-out family. Rank 1 means the largest share. Table 14 reports the retrieval rank of the lowest-error family and how often that family ranks first. Exact ties receive average ranks; results are averaged over tied lowest-error families. Mean ranks weight held-out families equally at each penalty. The final row averages these ranks equally across the thirteen penalties.

Table 14: Retrieval rank of the family with the lowest relative-utility prediction error. Top-1 counts held-out families for which it is also the most retrieved. Each benchmark uses its prescribed penalty grid.

| $\lambda$    | RouterBench (six sources) |       | SPROUT (five sources) |       |
|--------------|---------------------------|-------|-----------------------|-------|
|              | Mean rank                 | Top-1 | Mean rank             | Top-1 |
| 0            | 3.14                      | 0/7   | 3.00                  | 2/6   |
| 0.5          | n/a                       | n/a   | 3.00                  | 2/6   |
| 1            | 3.14                      | 0/7   | 3.00                  | 2/6   |
| 2            | 3.14                      | 0/7   | 3.00                  | 2/6   |
| 5            | 3.36                      | 0/7   | 3.00                  | 2/6   |
| 10           | 3.36                      | 0/7   | 3.00                  | 2/6   |
| 20           | 2.71                      | 1/7   | n/a                   | n/a   |
| 25           | n/a                       | n/a   | 3.33                  | 1/6   |
| 30           | 3.29                      | 1/7   | n/a                   | n/a   |
| 50           | 3.29                      | 1/7   | 3.33                  | 1/6   |
| 75           | 3.29                      | 1/7   | 3.33                  | 1/6   |
| 100          | 3.29                      | 1/7   | 3.67                  | 1/6   |
| 150          | 3.29                      | 1/7   | 3.67                  | 1/6   |
| 250          | 3.29                      | 1/7   | 3.17                  | 2/6   |
| 500          | 3.57                      | 0/7   | 3.17                  | 2/6   |
| Mean / total | 3.24                      | 7/91  | 3.21                  | 21/78 |

## B.7 Predicting Model Preferences

For every held-out query, we average recorded rewards and costs over its 100 nearest training examples. At each penalty, we compare the sign of each predicted pairwise utility difference with the recorded difference. Recorded ties are excluded; predicted ties receive half credit. Accuracy is first averaged within each query, then over queries with at least one non-tied pair. This separates the accuracy of model comparisons from the utility of the final selected model.

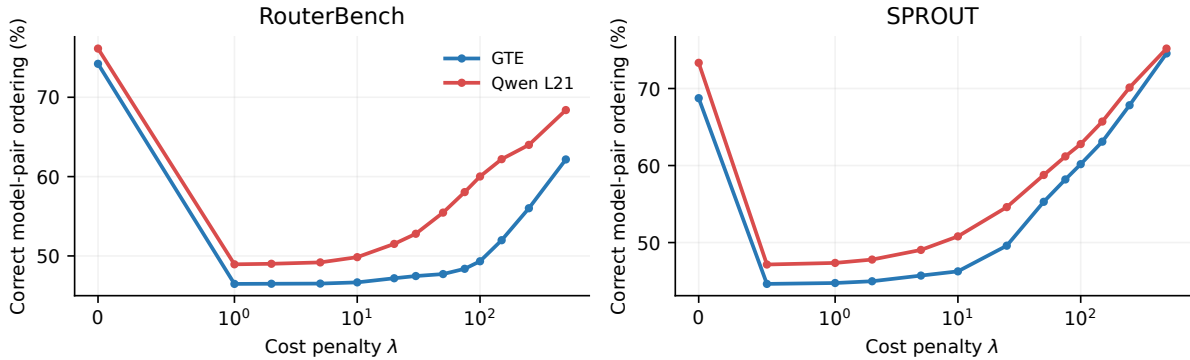


Figure 6: Correct utility ordering across candidate-model pairs, evaluated at every penalty. At  $\lambda = 0$ , recorded reward ties change the set of evaluated pairs.

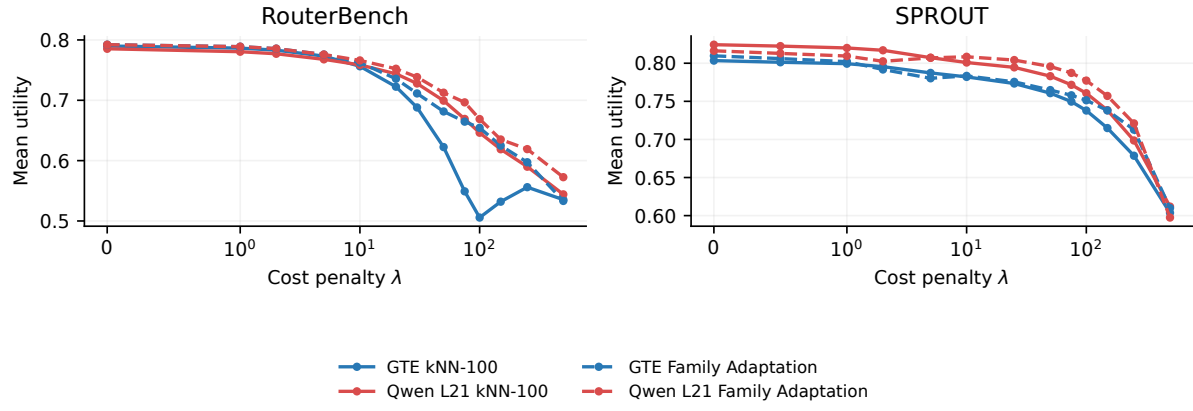


Figure 7: Held-out utility at every penalty for kNN-100 and Family Adaptation. Unlike reward-cost frontiers, these curves retain every evaluated policy.

## C ID Routing and Group Lookups

### C.1 Query-Specific Decisions Beyond Group Lookup

**Question and comparison.** Similar overall scores do not tell us whether kNN makes useful choices beyond group lookup. We compare GTE kNN-100 with family lookup and family-cell lookup on the same five ID splits of each benchmark, at all thirteen cost penalties. Family lookup selects the model with the highest mean training utility within the query’s family. Family-cell lookup uses the available native labels within that family; where no finer label exists, the cell is the family. A cell absent from training falls back to family lookup. kNN-100 instead selects the model with the highest mean utility among the query’s 100 nearest training examples. All choices use training outcomes only.

**What we measure.** For each test query, we record whether kNN and the lookup choose different models. We then subtract the recorded utility of the lookup’s choice from that of kNN’s choice. We report the fraction of different choices, the mean utility gain on those choices, and the gain over all test queries. Agreeing choices contribute zero, so the overall gain equals the disagreement fraction multiplied by the gain on disagreements. We average queries equally within each split, then average the five splits equally. Table 15 also averages the thirteen penalties equally; its gain on disagreements is the overall mean gain divided by the mean disagreement fraction.

**Uncertainty.** The 95% intervals use 1,000 paired bootstrap samples of unique test queries, keeping each query’s repeated test appearances and all its penalty-specific results together. These intervals describe uncertainty over test queries for the fixed splits and fitted routers; they do not include retraining variability. Per-penalty intervals are pointwise.

Table 15: GTE kNN-100 compared with group lookup across penalties, with 95% paired intervals. Positive gains favor kNN.

| Dataset     | Lookup      | Different choices        | Gain on disagreements         | Overall gain                  |
|-------------|-------------|--------------------------|-------------------------------|-------------------------------|
| RouterBench | Family      | 30.04%<br>[29.80, 30.29] | +0.0233<br>[+0.0178, +0.0290] | +0.0070<br>[+0.0054, +0.0087] |
| RouterBench | Family-cell | 28.78%<br>[28.54, 29.03] | +0.0186<br>[+0.0131, +0.0245] | +0.0054<br>[+0.0038, +0.0071] |
| SPROUT      | Family      | 45.50%<br>[45.17, 45.79] | +0.0199<br>[+0.0166, +0.0233] | +0.0090<br>[+0.0076, +0.0106] |
| SPROUT      | Family-cell | 47.75%<br>[47.44, 48.07] | +0.0135<br>[+0.0101, +0.0170] | +0.0065<br>[+0.0048, +0.0081] |

**Results.** The additional gain is small on both benchmarks. Relative to family-cell lookup, kNN improves mean utility by 0.0054 on RouterBench and 0.0065 on SPROUT, despite choosing a different model in 28.78% and 47.75% of cases, respectively. Thus, its query-specific decisions contain some useful information beyond a fixed choice per group, but this adds only a small average improvement. The effect varies with the cost penalty (Table 16): on RouterBench the intervals include zero at  $\lambda = 0, 1, 2$ , while SPROUT has positive pointwise intervals throughout and a larger gain of 0.0228 at  $\lambda = 500$ .

Table 16: kNN-100 utility gains at every penalty, with 95% paired intervals.

| $\lambda$          | Against family lookup<br>Overall gain | Against family-cell lookup |                               |                               |
|--------------------|---------------------------------------|----------------------------|-------------------------------|-------------------------------|
|                    |                                       | Different<br>choices       | Gain on<br>disagreements      | Overall gain                  |
| <b>RouterBench</b> |                                       |                            |                               |                               |
| 0                  | 0.0000<br>[−0.0018, +0.0019]          | 23.55%<br>[23.06, 24.03]   | +0.0038<br>[−0.0039, +0.0120] | +0.0009<br>[−0.0009, +0.0029] |
| 1                  | +0.0004<br>[−0.0015, +0.0023]         | 24.92%<br>[24.42, 25.42]   | +0.0057<br>[−0.0018, +0.0136] | +0.0014<br>[−0.0005, +0.0034] |
| 2                  | +0.0004<br>[−0.0015, +0.0022]         | 25.51%<br>[25.03, 25.99]   | +0.0062<br>[−0.0012, +0.0140] | +0.0016<br>[−0.0003, +0.0036] |
| 5                  | +0.0013<br>[−0.0008, +0.0033]         | 29.27%<br>[28.77, 29.79]   | +0.0078<br>[+0.0007, +0.0149] | +0.0023<br>[+0.0002, +0.0043] |
| 10                 | +0.0015<br>[−0.0007, +0.0038]         | 22.00%<br>[21.54, 22.47]   | +0.0115<br>[+0.0013, +0.0222] | +0.0025<br>[+0.0003, +0.0049] |
| 20                 | +0.0036<br>[+0.0013, +0.0060]         | 21.08%<br>[20.62, 21.53]   | +0.0202<br>[+0.0095, +0.0316] | +0.0043<br>[+0.0020, +0.0067] |
| 30                 | +0.0061<br>[+0.0035, +0.0088]         | 24.78%<br>[24.30, 25.25]   | +0.0263<br>[+0.0158, +0.0365] | +0.0065<br>[+0.0039, +0.0091] |
| 50                 | +0.0122<br>[+0.0098, +0.0149]         | 25.38%<br>[24.96, 25.84]   | +0.0424<br>[+0.0330, +0.0521] | +0.0108<br>[+0.0083, +0.0132] |
| 75                 | +0.0114<br>[+0.0082, +0.0146]         | 27.53%<br>[27.04, 28.03]   | +0.0289<br>[+0.0186, +0.0392] | +0.0080<br>[+0.0051, +0.0108] |
| 100                | +0.0158<br>[+0.0122, +0.0191]         | 33.10%<br>[32.57, 33.63]   | +0.0216<br>[+0.0125, +0.0310] | +0.0071<br>[+0.0041, +0.0103] |
| 150                | +0.0148<br>[+0.0109, +0.0186]         | 39.11%<br>[38.56, 39.62]   | +0.0226<br>[+0.0139, +0.0316] | +0.0088<br>[+0.0055, +0.0124] |
| 250                | +0.0116<br>[+0.0081, +0.0152]         | 43.54%<br>[42.98, 44.07]   | +0.0196<br>[+0.0121, +0.0274] | +0.0085<br>[+0.0052, +0.0120] |
| 500                | +0.0119<br>[+0.0085, +0.0154]         | 34.39%<br>[33.83, 34.92]   | +0.0198<br>[+0.0108, +0.0287] | +0.0068<br>[+0.0037, +0.0099] |
| <b>SPROUT</b>      |                                       |                            |                               |                               |
| 0                  | +0.0051<br>[+0.0027, +0.0073]         | 55.37%<br>[54.88, 55.85]   | +0.0070<br>[+0.0026, +0.0110] | +0.0039<br>[+0.0015, +0.0060] |
| 0.5                | +0.0054<br>[+0.0032, +0.0077]         | 57.35%<br>[56.86, 57.83]   | +0.0078<br>[+0.0037, +0.0115] | +0.0045<br>[+0.0021, +0.0066] |
| 1                  | +0.0055<br>[+0.0032, +0.0079]         | 58.86%<br>[58.38, 59.34]   | +0.0077<br>[+0.0036, +0.0114] | +0.0045<br>[+0.0021, +0.0067] |
| 2                  | +0.0066<br>[+0.0043, +0.0089]         | 58.08%<br>[57.58, 58.56]   | +0.0102<br>[+0.0063, +0.0142] | +0.0059<br>[+0.0037, +0.0082] |
| 5                  | +0.0057<br>[+0.0031, +0.0082]         | 38.24%<br>[37.80, 38.70]   | +0.0132<br>[+0.0071, +0.0201] | +0.0051<br>[+0.0027, +0.0077] |
| 10                 | +0.0064<br>[+0.0038, +0.0089]         | 35.35%<br>[34.88, 35.81]   | +0.0143<br>[+0.0076, +0.0210] | +0.0051<br>[+0.0027, +0.0075] |
| 25                 | +0.0116<br>[+0.0093, +0.0142]         | 33.62%<br>[33.17, 34.12]   | +0.0148<br>[+0.0088, +0.0211] | +0.0050<br>[+0.0029, +0.0072] |
| 50                 | +0.0079<br>[+0.0057, +0.0102]         | 35.87%<br>[35.36, 36.35]   | +0.0072<br>[+0.0011, +0.0135] | +0.0026<br>[+0.0004, +0.0049] |
| 75                 | +0.0068<br>[+0.0047, +0.0091]         | 40.91%<br>[40.44, 41.41]   | +0.0105<br>[+0.0053, +0.0164] | +0.0043<br>[+0.0022, +0.0067] |
| 100                | +0.0070<br>[+0.0049, +0.0093]         | 45.56%<br>[45.11, 46.07]   | +0.0092<br>[+0.0042, +0.0143] | +0.0042<br>[+0.0019, +0.0065] |
| 150                | +0.0100<br>[+0.0077, +0.0124]         | 50.74%<br>[50.28, 51.21]   | +0.0111<br>[+0.0066, +0.0157] | +0.0056<br>[+0.0033, +0.0079] |
| 250                | +0.0157<br>[+0.0135, +0.0181]         | 53.11%<br>[52.62, 53.58]   | +0.0200<br>[+0.0155, +0.0246] | +0.0106<br>[+0.0082, +0.0130] |
| 500                | +0.0238<br>[+0.0205, +0.0272]         | 57.74%<br>[57.27, 58.23]   | +0.0395<br>[+0.0343, +0.0448] | +0.0228<br>[+0.0198, +0.0258] |

## C.2 Lookup with predicted group labels

This check replaces the known family or cell label with a label predicted from the query embedding. On each of the five ID splits, we fit separate multinomial logistic classifiers for family and native-cell labels using GTE embeddings. Coordinate standardization is fitted on training queries only. Both classifiers use fixed  $C = 1$ , unweighted classes and the L-BFGS solver, with tolerance  $10^{-4}$  and at most 3,000 iterations; there is no hyperparameter search. We then select the model with the highest mean training utility in the predicted group. All nonempty training cells are retained, matching the known-label lookup. No test group label is supplied to the predicted-label router.

We evaluate both benchmarks at all thirteen penalties, reporting label accuracy and routing utility alongside the corresponding known-label lookup. Each result is averaged over the five splits.

Table 17 summarizes all thirteen penalties. Predicting labels preserves almost all of the known-label lookup’s mean utility on both benchmarks. Cell labels are less accurately predicted than family labels, but their mean utility loss remains below 0.0006.

Table 17: Lookup with predicted labels, with 95% intervals. Utilities average five splits and thirteen penalties;  $\Delta$  is predicted-label minus known-label utility.

| Benchmark   | Label  | Accuracy                 | Known-label utility           | Predicted-label utility       | $\Delta$                         |
|-------------|--------|--------------------------|-------------------------------|-------------------------------|----------------------------------|
| RouterBench | Family | 99.97%<br>[99.95, 99.99] | 0.72124<br>[0.71720, 0.72523] | 0.72117<br>[0.71714, 0.72520] | −0.00007<br>[−0.00018, +0.00001] |
| RouterBench | Cell   | 93.85%<br>[93.56, 94.14] | 0.72288<br>[0.71868, 0.72713] | 0.72232<br>[0.71819, 0.72646] | −0.00056<br>[−0.00095, −0.00020] |
| SPROUT      | Family | 99.91%<br>[99.88, 99.94] | 0.80966<br>[0.80609, 0.81320] | 0.80955<br>[0.80597, 0.81312] | −0.00012<br>[−0.00026, 0.00000]  |
| SPROUT      | Cell   | 91.68%<br>[91.38, 91.96] | 0.81225<br>[0.80881, 0.81584] | 0.81182<br>[0.80843, 0.81533] | −0.00042<br>[−0.00100, +0.00020] |

## D Routing Results Across All Penalties

### D.1 Overall routing results

ID results average five splits; OOD results pool queries from all held-out families. Native-label lookups are reported for ID evaluation. The Mean column averages all thirteen penalties and is repeated in each part of the full tables. The realized oracle selects, for each test query and cost penalty, the model with the highest recorded utility.

In what follows, the tables report:

- **Table 18:** mean OOD utility across all thirteen cost penalties for the GTE methods, the best single model, and our two L21 routers.
- **Tables 19–21:** RouterBench ID results at every cost penalty.
- **Tables 22–24:** RouterBench OOD results at every cost penalty.
- **Tables 25–27:** SPROUT ID results at every cost penalty.
- **Tables 28–30:** SPROUT OOD results at every cost penalty.
- **Table 31:** mean OOD utility for SharedTrunkNet with fixed L21, a common selected layer, or per-model selected layers, alongside our L21 routers.
- **Table 32:** paired utility differences between each L21 router and SharedTrunkNet with per-model selected layers.
- **Tables 33–34:** the same comparison at every cost penalty on RouterBench and SPROUT, respectively.
- **Tables 35–36:** the change from GTE to L21 for kNN-100, broken down by held-out family on RouterBench and SPROUT, with both query-weighted and family-weighted averages.

All tables include 95% confidence intervals.

Table 18: Mean held-out utility across thirteen cost penalties, with 95% intervals. Bold marks the highest mean among the listed methods.

| Method                     | RouterBench                       | SPROUT                            |
|----------------------------|-----------------------------------|-----------------------------------|
| <i>GTE embeddings</i>      |                                   |                                   |
| Best single model          | 0.6618<br>[0.6586, 0.6652]        | 0.7575<br>[0.7544, 0.7604]        |
| kNN-100                    | 0.6615<br>[0.6583, 0.6651]        | 0.7529<br>[0.7498, 0.7559]        |
| Validated kNN              | 0.6566<br>[0.6534, 0.6601]        | 0.7534<br>[0.7504, 0.7564]        |
| Avengers-Pro               | 0.6662<br>[0.6631, 0.6698]        | 0.7439<br>[0.7408, 0.7470]        |
| KM-Prox (fixed)            | 0.6764<br>[0.6732, 0.6797]        | 0.7460<br>[0.7429, 0.7491]        |
| KM-Prox (validated)        | 0.6764<br>[0.6732, 0.6798]        | 0.7483<br>[0.7453, 0.7513]        |
| Ridge                      | 0.6796<br>[0.6765, 0.6829]        | 0.7674<br>[0.7645, 0.7703]        |
| MLP                        | 0.6755<br>[0.6722, 0.6789]        | 0.7564<br>[0.7536, 0.7592]        |
| Family Adaptation          | 0.7006<br>[0.6973, 0.7041]        | 0.7603<br>[0.7573, 0.7632]        |
| <i>Qwen L21 embeddings</i> |                                   |                                   |
| kNN-100                    | 0.7007<br>[0.6972, 0.7043]        | 0.7731<br>[0.7701, 0.7760]        |
| Family Adaptation          | <b>0.7157</b><br>[0.7125, 0.7191] | <b>0.7767</b><br>[0.7737, 0.7797] |

Table 19: RouterBench in-distribution utility with 95% intervals, part 1 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 0                          | 1                          | 2                          | 5                          | 10                         | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7913<br>[0.7869, 0.7957] | 0.7888<br>[0.7844, 0.7932] | 0.7863<br>[0.7819, 0.7907] | 0.7788<br>[0.7744, 0.7833] | 0.7664<br>[0.7618, 0.7709] | 0.7176<br>[0.7134, 0.7219] |
| GTE / Best single model        | 0.7922<br>[0.7879, 0.7967] | 0.7890<br>[0.7846, 0.7934] | 0.7857<br>[0.7813, 0.7902] | 0.7759<br>[0.7715, 0.7805] | 0.7596<br>[0.7550, 0.7641] | 0.6994<br>[0.6951, 0.7035] |
| GTE / Family lookup            | 0.7925<br>[0.7880, 0.7969] | 0.7895<br>[0.7850, 0.7939] | 0.7867<br>[0.7822, 0.7912] | 0.7785<br>[0.7739, 0.7830] | 0.7708<br>[0.7661, 0.7753] | 0.7212<br>[0.7172, 0.7252] |
| GTE / Family-cell lookup       | 0.7916<br>[0.7870, 0.7960] | 0.7884<br>[0.7839, 0.7929] | 0.7855<br>[0.7810, 0.7900] | 0.7775<br>[0.7730, 0.7820] | 0.7697<br>[0.7650, 0.7742] | 0.7229<br>[0.7187, 0.7271] |
| GTE / KM-Prox                  | 0.7936<br>[0.7892, 0.7980] | 0.7906<br>[0.7861, 0.7950] | 0.7876<br>[0.7831, 0.7920] | 0.7799<br>[0.7754, 0.7843] | 0.7733<br>[0.7687, 0.7779] | 0.7287<br>[0.7246, 0.7329] |
| GTE / KM-Prox (validated)      | 0.7949<br>[0.7904, 0.7993] | 0.7914<br>[0.7870, 0.7960] | 0.7884<br>[0.7840, 0.7928] | 0.7807<br>[0.7762, 0.7852] | 0.7735<br>[0.7690, 0.7780] | 0.7289<br>[0.7248, 0.7330] |
| GTE / MLP                      | 0.7934<br>[0.7889, 0.7978] | 0.7876<br>[0.7832, 0.7919] | 0.7859<br>[0.7816, 0.7904] | 0.7764<br>[0.7721, 0.7809] | 0.7668<br>[0.7623, 0.7710] | 0.7245<br>[0.7205, 0.7285] |
| GTE / Realized oracle          | 0.9194<br>[0.9170, 0.9219] | 0.9192<br>[0.9167, 0.9216] | 0.9189<br>[0.9165, 0.9214] | 0.9182<br>[0.9158, 0.9207] | 0.9170<br>[0.9146, 0.9195] | 0.9033<br>[0.9008, 0.9059] |
| GTE / Ridge                    | 0.7941<br>[0.7898, 0.7984] | 0.7913<br>[0.7871, 0.7956] | 0.7884<br>[0.7841, 0.7928] | 0.7806<br>[0.7761, 0.7850] | 0.7731<br>[0.7685, 0.7776] | 0.7290<br>[0.7249, 0.7332] |
| GTE / Family Adaptation        | 0.7925<br>[0.7881, 0.7969] | 0.7898<br>[0.7853, 0.7942] | 0.7867<br>[0.7822, 0.7912] | 0.7785<br>[0.7740, 0.7830] | 0.7707<br>[0.7661, 0.7752] | 0.7211<br>[0.7171, 0.7251] |
| GTE / Validated kNN            | 0.7935<br>[0.7890, 0.7978] | 0.7905<br>[0.7861, 0.7948] | 0.7876<br>[0.7831, 0.7919] | 0.7807<br>[0.7762, 0.7851] | 0.7736<br>[0.7690, 0.7782] | 0.7294<br>[0.7252, 0.7336] |
| GTE / kNN-100                  | 0.7925<br>[0.7882, 0.7969] | 0.7899<br>[0.7855, 0.7942] | 0.7871<br>[0.7828, 0.7915] | 0.7798<br>[0.7753, 0.7841] | 0.7722<br>[0.7676, 0.7767] | 0.7282<br>[0.7240, 0.7325] |
| Qwen L21 / Avengers-Pro        | 0.7928<br>[0.7885, 0.7971] | 0.7903<br>[0.7859, 0.7946] | 0.7873<br>[0.7829, 0.7917] | 0.7796<br>[0.7751, 0.7839] | 0.7669<br>[0.7624, 0.7713] | 0.7173<br>[0.7130, 0.7216] |
| Qwen L21 / KM-Prox             | 0.7925<br>[0.7880, 0.7969] | 0.7898<br>[0.7853, 0.7942] | 0.7870<br>[0.7825, 0.7915] | 0.7787<br>[0.7742, 0.7832] | 0.7705<br>[0.7659, 0.7751] | 0.7252<br>[0.7212, 0.7292] |
| Qwen L21 / KM-Prox (validated) | 0.7928<br>[0.7884, 0.7972] | 0.7898<br>[0.7854, 0.7943] | 0.7871<br>[0.7827, 0.7916] | 0.7790<br>[0.7746, 0.7835] | 0.7729<br>[0.7684, 0.7773] | 0.7301<br>[0.7260, 0.7341] |
| Qwen L21 / MLP                 | 0.7906<br>[0.7863, 0.7948] | 0.7864<br>[0.7820, 0.7906] | 0.7853<br>[0.7809, 0.7895] | 0.7783<br>[0.7738, 0.7827] | 0.7668<br>[0.7623, 0.7712] | 0.7299<br>[0.7256, 0.7339] |
| Qwen L21 / Ridge               | 0.7965<br>[0.7921, 0.8009] | 0.7935<br>[0.7891, 0.7979] | 0.7906<br>[0.7862, 0.7950] | 0.7832<br>[0.7788, 0.7876] | 0.7750<br>[0.7704, 0.7794] | 0.7358<br>[0.7317, 0.7400] |
| Qwen L21 / Family Adaptation   | 0.7926<br>[0.7881, 0.7971] | 0.7898<br>[0.7853, 0.7942] | 0.7867<br>[0.7822, 0.7912] | 0.7785<br>[0.7739, 0.7830] | 0.7707<br>[0.7661, 0.7753] | 0.7217<br>[0.7177, 0.7258] |
| Qwen L21 / Validated kNN       | 0.7962<br>[0.7917, 0.8005] | 0.7935<br>[0.7891, 0.7979] | 0.7907<br>[0.7862, 0.7950] | 0.7836<br>[0.7790, 0.7880] | 0.7762<br>[0.7716, 0.7807] | 0.7360<br>[0.7318, 0.7402] |
| Qwen L21 / kNN-100             | 0.7952<br>[0.7908, 0.7995] | 0.7921<br>[0.7876, 0.7964] | 0.7898<br>[0.7853, 0.7941] | 0.7827<br>[0.7781, 0.7872] | 0.7750<br>[0.7704, 0.7795] | 0.7352<br>[0.7309, 0.7394] |

Table 20: RouterBench in-distribution utility with 95% intervals, part 2 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 20                         | 30                         | 50                         | 75                         | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7407<br>[0.7360, 0.7453] | 0.7188<br>[0.7139, 0.7237] | 0.7022<br>[0.6974, 0.7071] | 0.6937<br>[0.6887, 0.6987] | 0.7176<br>[0.7134, 0.7219] |
| GTE / Best single model        | 0.7269<br>[0.7221, 0.7316] | 0.6942<br>[0.6893, 0.6991] | 0.6537<br>[0.6482, 0.6591] | 0.6490<br>[0.6435, 0.6545] | 0.6994<br>[0.6951, 0.7035] |
| GTE / Family lookup            | 0.7568<br>[0.7522, 0.7614] | 0.7429<br>[0.7384, 0.7474] | 0.7157<br>[0.7109, 0.7204] | 0.6958<br>[0.6908, 0.7006] | 0.7212<br>[0.7172, 0.7252] |
| GTE / Family-cell lookup       | 0.7561<br>[0.7515, 0.7608] | 0.7424<br>[0.7378, 0.7470] | 0.7172<br>[0.7123, 0.7219] | 0.6993<br>[0.6941, 0.7043] | 0.7229<br>[0.7187, 0.7271] |
| GTE / KM-Prox                  | 0.7615<br>[0.7570, 0.7659] | 0.7508<br>[0.7461, 0.7554] | 0.7301<br>[0.7253, 0.7346] | 0.7082<br>[0.7033, 0.7128] | 0.7287<br>[0.7246, 0.7329] |
| GTE / KM-Prox (validated)      | 0.7610<br>[0.7565, 0.7655] | 0.7510<br>[0.7463, 0.7555] | 0.7313<br>[0.7266, 0.7359] | 0.7080<br>[0.7032, 0.7126] | 0.7289<br>[0.7248, 0.7330] |
| GTE / MLP                      | 0.7551<br>[0.7505, 0.7597] | 0.7462<br>[0.7417, 0.7508] | 0.7282<br>[0.7236, 0.7329] | 0.7021<br>[0.6975, 0.7067] | 0.7245<br>[0.7205, 0.7285] |
| GTE / Realized oracle          | 0.9147<br>[0.9123, 0.9172] | 0.9125<br>[0.9100, 0.9150] | 0.9084<br>[0.9059, 0.9109] | 0.9040<br>[0.9015, 0.9066] | 0.9033<br>[0.9008, 0.9059] |
| GTE / Ridge                    | 0.7622<br>[0.7575, 0.7667] | 0.7521<br>[0.7475, 0.7567] | 0.7325<br>[0.7277, 0.7370] | 0.7073<br>[0.7026, 0.7120] | 0.7290<br>[0.7249, 0.7332] |
| GTE / Family Adaptation        | 0.7568<br>[0.7522, 0.7613] | 0.7427<br>[0.7381, 0.7473] | 0.7189<br>[0.7141, 0.7236] | 0.6949<br>[0.6899, 0.6996] | 0.7211<br>[0.7171, 0.7251] |
| GTE / Validated kNN            | 0.7617<br>[0.7570, 0.7662] | 0.7508<br>[0.7461, 0.7553] | 0.7290<br>[0.7244, 0.7336] | 0.7088<br>[0.7041, 0.7135] | 0.7294<br>[0.7252, 0.7336] |
| GTE / kNN-100                  | 0.7604<br>[0.7558, 0.7650] | 0.7489<br>[0.7442, 0.7535] | 0.7279<br>[0.7232, 0.7327] | 0.7072<br>[0.7024, 0.7121] | 0.7282<br>[0.7240, 0.7325] |
| Qwen L21 / Avengers-Pro        | 0.7363<br>[0.7317, 0.7407] | 0.7213<br>[0.7163, 0.7260] | 0.7002<br>[0.6951, 0.7050] | 0.6772<br>[0.6719, 0.6821] | 0.7173<br>[0.7130, 0.7216] |
| Qwen L21 / KM-Prox             | 0.7572<br>[0.7527, 0.7617] | 0.7498<br>[0.7452, 0.7543] | 0.7303<br>[0.7255, 0.7347] | 0.7066<br>[0.7017, 0.7112] | 0.7252<br>[0.7212, 0.7292] |
| Qwen L21 / KM-Prox (validated) | 0.7615<br>[0.7567, 0.7659] | 0.7508<br>[0.7461, 0.7552] | 0.7302<br>[0.7254, 0.7347] | 0.7109<br>[0.7063, 0.7156] | 0.7301<br>[0.7260, 0.7341] |
| Qwen L21 / MLP                 | 0.7558<br>[0.7512, 0.7602] | 0.7457<br>[0.7412, 0.7501] | 0.7291<br>[0.7243, 0.7336] | 0.7141<br>[0.7095, 0.7188] | 0.7299<br>[0.7256, 0.7339] |
| Qwen L21 / Ridge               | 0.7648<br>[0.7601, 0.7692] | 0.7552<br>[0.7505, 0.7596] | 0.7367<br>[0.7320, 0.7411] | 0.7188<br>[0.7141, 0.7234] | 0.7358<br>[0.7317, 0.7400] |
| Qwen L21 / Family Adaptation   | 0.7568<br>[0.7522, 0.7613] | 0.7429<br>[0.7383, 0.7474] | 0.7219<br>[0.7173, 0.7264] | 0.6958<br>[0.6909, 0.7006] | 0.7217<br>[0.7177, 0.7258] |
| Qwen L21 / Validated kNN       | 0.7645<br>[0.7597, 0.7690] | 0.7541<br>[0.7493, 0.7586] | 0.7361<br>[0.7312, 0.7406] | 0.7178<br>[0.7130, 0.7225] | 0.7360<br>[0.7318, 0.7402] |
| Qwen L21 / kNN-100             | 0.7640<br>[0.7592, 0.7684] | 0.7532<br>[0.7483, 0.7577] | 0.7352<br>[0.7303, 0.7399] | 0.7166<br>[0.7116, 0.7214] | 0.7352<br>[0.7309, 0.7394] |

Table 21: RouterBench in-distribution utility with 95% intervals, part 3 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 100                        | 150                        | 250                        | 500                        | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.6824<br>[0.6773, 0.6875] | 0.6620<br>[0.6571, 0.6672] | 0.6376<br>[0.6325, 0.6427] | 0.5794<br>[0.5740, 0.5850] | 0.7176<br>[0.7134, 0.7219] |
| GTE / Best single model        | 0.6443<br>[0.6388, 0.6498] | 0.6350<br>[0.6295, 0.6405] | 0.6164<br>[0.6108, 0.6220] | 0.5697<br>[0.5641, 0.5754] | 0.6994<br>[0.6951, 0.7035] |
| GTE / Family lookup            | 0.6753<br>[0.6704, 0.6802] | 0.6539<br>[0.6486, 0.6593] | 0.6292<br>[0.6238, 0.6348] | 0.5885<br>[0.5831, 0.5941] | 0.7212<br>[0.7172, 0.7252] |
| GTE / Family-cell lookup       | 0.6840<br>[0.6788, 0.6890] | 0.6598<br>[0.6547, 0.6650] | 0.6323<br>[0.6269, 0.6376] | 0.5936<br>[0.5879, 0.5992] | 0.7229<br>[0.7187, 0.7271] |
| GTE / KM-Prox                  | 0.6915<br>[0.6868, 0.6965] | 0.6660<br>[0.6609, 0.6712] | 0.6399<br>[0.6346, 0.6452] | 0.6002<br>[0.5949, 0.6057] | 0.7287<br>[0.7246, 0.7329] |
| GTE / KM-Prox (validated)      | 0.6912<br>[0.6864, 0.6960] | 0.6659<br>[0.6610, 0.6708] | 0.6391<br>[0.6339, 0.6441] | 0.5994<br>[0.5941, 0.6050] | 0.7289<br>[0.7248, 0.7330] |
| GTE / MLP                      | 0.6842<br>[0.6797, 0.6891] | 0.6623<br>[0.6575, 0.6671] | 0.6343<br>[0.6293, 0.6392] | 0.5953<br>[0.5904, 0.6006] | 0.7245<br>[0.7205, 0.7285] |
| GTE / Realized oracle          | 0.9000<br>[0.8974, 0.9026] | 0.8923<br>[0.8897, 0.8950] | 0.8772<br>[0.8744, 0.8800] | 0.8414<br>[0.8384, 0.8445] | 0.9033<br>[0.9008, 0.9059] |
| GTE / Ridge                    | 0.6893<br>[0.6846, 0.6943] | 0.6663<br>[0.6614, 0.6713] | 0.6395<br>[0.6342, 0.6447] | 0.6005<br>[0.5952, 0.6061] | 0.7290<br>[0.7249, 0.7332] |
| GTE / Family Adaptation        | 0.6716<br>[0.6665, 0.6766] | 0.6539<br>[0.6486, 0.6594] | 0.6293<br>[0.6238, 0.6348] | 0.5884<br>[0.5830, 0.5939] | 0.7211<br>[0.7171, 0.7251] |
| GTE / Validated kNN            | 0.6922<br>[0.6874, 0.6969] | 0.6702<br>[0.6653, 0.6752] | 0.6424<br>[0.6375, 0.6475] | 0.6010<br>[0.5959, 0.6065] | 0.7294<br>[0.7252, 0.7336] |
| GTE / kNN-100                  | 0.6911<br>[0.6862, 0.6962] | 0.6687<br>[0.6637, 0.6736] | 0.6408<br>[0.6359, 0.6461] | 0.6004<br>[0.5952, 0.6060] | 0.7282<br>[0.7240, 0.7325] |
| Qwen L21 / Avengers-Pro        | 0.6721<br>[0.6671, 0.6771] | 0.6658<br>[0.6609, 0.6709] | 0.6457<br>[0.6407, 0.6509] | 0.5897<br>[0.5846, 0.5952] | 0.7173<br>[0.7130, 0.7216] |
| Qwen L21 / KM-Prox             | 0.6865<br>[0.6816, 0.6912] | 0.6575<br>[0.6522, 0.6628] | 0.6310<br>[0.6256, 0.6365] | 0.5904<br>[0.5849, 0.5960] | 0.7252<br>[0.7212, 0.7292] |
| Qwen L21 / KM-Prox (validated) | 0.6953<br>[0.6902, 0.7002] | 0.6690<br>[0.6640, 0.6741] | 0.6441<br>[0.6390, 0.6495] | 0.6074<br>[0.6022, 0.6129] | 0.7301<br>[0.7260, 0.7341] |
| Qwen L21 / MLP                 | 0.6994<br>[0.6946, 0.7041] | 0.6807<br>[0.6757, 0.6855] | 0.6505<br>[0.6456, 0.6556] | 0.6055<br>[0.6006, 0.6107] | 0.7299<br>[0.7256, 0.7339] |
| Qwen L21 / Ridge               | 0.7045<br>[0.6998, 0.7093] | 0.6826<br>[0.6777, 0.6877] | 0.6530<br>[0.6481, 0.6583] | 0.6113<br>[0.6062, 0.6168] | 0.7358<br>[0.7317, 0.7400] |
| Qwen L21 / Family Adaptation   | 0.6755<br>[0.6705, 0.6804] | 0.6539<br>[0.6485, 0.6593] | 0.6292<br>[0.6238, 0.6347] | 0.5884<br>[0.5830, 0.5939] | 0.7217<br>[0.7177, 0.7258] |
| Qwen L21 / Validated kNN       | 0.7046<br>[0.6997, 0.7095] | 0.6836<br>[0.6787, 0.6886] | 0.6530<br>[0.6481, 0.6580] | 0.6141<br>[0.6089, 0.6194] | 0.7360<br>[0.7318, 0.7402] |
| Qwen L21 / kNN-100             | 0.7038<br>[0.6988, 0.7087] | 0.6827<br>[0.6778, 0.6875] | 0.6534<br>[0.6484, 0.6588] | 0.6140<br>[0.6088, 0.6194] | 0.7352<br>[0.7309, 0.7394] |

Table 22: RouterBench out-of-distribution utility with 95% intervals, part 1 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 0                          | 1                          | 2                          | 5                          | 10                         | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7905<br>[0.7871, 0.7942] | 0.7873<br>[0.7839, 0.7910] | 0.7841<br>[0.7807, 0.7878] | 0.7743<br>[0.7709, 0.7780] | 0.7581<br>[0.7547, 0.7618] | 0.6662<br>[0.6631, 0.6698] |
| GTE / Best single model        | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7758<br>[0.7724, 0.7795] | 0.7596<br>[0.7561, 0.7632] | 0.6618<br>[0.6586, 0.6652] |
| GTE / KM-Prox                  | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7758<br>[0.7724, 0.7795] | 0.7595<br>[0.7560, 0.7632] | 0.6764<br>[0.6732, 0.6797] |
| GTE / KM-Prox (validated)      | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7758<br>[0.7724, 0.7795] | 0.7576<br>[0.7542, 0.7613] | 0.6764<br>[0.6732, 0.6798] |
| GTE / MLP                      | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7758<br>[0.7724, 0.7795] | 0.7594<br>[0.7559, 0.7631] | 0.6755<br>[0.6722, 0.6789] |
| GTE / Realized oracle          | 0.9189<br>[0.9171, 0.9206] | 0.9186<br>[0.9169, 0.9203] | 0.9184<br>[0.9167, 0.9201] | 0.9177<br>[0.9160, 0.9194] | 0.9166<br>[0.9148, 0.9182] | 0.9030<br>[0.9012, 0.9047] |
| GTE / Ridge                    | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7758<br>[0.7724, 0.7795] | 0.7596<br>[0.7561, 0.7632] | 0.6796<br>[0.6765, 0.6829] |
| GTE / Family Adaptation        | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7758<br>[0.7724, 0.7795] | 0.7625<br>[0.7588, 0.7664] | 0.7006<br>[0.6973, 0.7041] |
| GTE / Validated kNN            | 0.7913<br>[0.7879, 0.7950] | 0.7879<br>[0.7845, 0.7916] | 0.7847<br>[0.7812, 0.7884] | 0.7746<br>[0.7712, 0.7783] | 0.7581<br>[0.7547, 0.7618] | 0.6566<br>[0.6534, 0.6601] |
| GTE / kNN-100                  | 0.7900<br>[0.7866, 0.7937] | 0.7864<br>[0.7831, 0.7902] | 0.7831<br>[0.7798, 0.7869] | 0.7732<br>[0.7699, 0.7770] | 0.7563<br>[0.7529, 0.7601] | 0.6615<br>[0.6583, 0.6651] |
| Qwen L21 / Avengers-Pro        | 0.7914<br>[0.7880, 0.7951] | 0.7882<br>[0.7848, 0.7919] | 0.7850<br>[0.7816, 0.7887] | 0.7754<br>[0.7720, 0.7791] | 0.7439<br>[0.7404, 0.7478] | 0.6581<br>[0.6548, 0.6615] |
| Qwen L21 / KM-Prox             | 0.7921<br>[0.7886, 0.7957] | 0.7887<br>[0.7853, 0.7924] | 0.7854<br>[0.7819, 0.7891] | 0.7754<br>[0.7719, 0.7791] | 0.7597<br>[0.7562, 0.7634] | 0.7028<br>[0.6996, 0.7063] |
| Qwen L21 / KM-Prox (validated) | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7758<br>[0.7724, 0.7795] | 0.7660<br>[0.7623, 0.7697] | 0.6984<br>[0.6954, 0.7018] |
| Qwen L21 / MLP                 | 0.7888<br>[0.7854, 0.7925] | 0.7855<br>[0.7821, 0.7892] | 0.7820<br>[0.7786, 0.7857] | 0.7733<br>[0.7698, 0.7771] | 0.7593<br>[0.7558, 0.7630] | 0.6918<br>[0.6887, 0.6952] |
| Qwen L21 / Ridge               | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7739<br>[0.7705, 0.7776] | 0.7538<br>[0.7502, 0.7575] | 0.6902<br>[0.6871, 0.6936] |
| Qwen L21 / Family Adaptation   | 0.7921<br>[0.7887, 0.7958] | 0.7889<br>[0.7854, 0.7926] | 0.7856<br>[0.7822, 0.7893] | 0.7759<br>[0.7724, 0.7796] | 0.7661<br>[0.7624, 0.7698] | 0.7157<br>[0.7125, 0.7191] |
| Qwen L21 / Validated kNN       | 0.7910<br>[0.7875, 0.7946] | 0.7869<br>[0.7833, 0.7906] | 0.7833<br>[0.7798, 0.7870] | 0.7713<br>[0.7677, 0.7751] | 0.7625<br>[0.7588, 0.7663] | 0.7034<br>[0.7001, 0.7068] |
| Qwen L21 / kNN-100             | 0.7854<br>[0.7818, 0.7891] | 0.7805<br>[0.7769, 0.7843] | 0.7771<br>[0.7734, 0.7809] | 0.7678<br>[0.7642, 0.7716] | 0.7587<br>[0.7549, 0.7626] | 0.7007<br>[0.6972, 0.7043] |

Table 23: RouterBench out-of-distribution utility with 95% intervals, part 2 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 20                         | 30                         | 50                         | 75                         | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7250<br>[0.7216, 0.7288] | 0.6259<br>[0.6219, 0.6303] | 0.5724<br>[0.5684, 0.5769] | 0.5119<br>[0.5077, 0.5163] | 0.6662<br>[0.6631, 0.6698] |
| GTE / Best single model        | 0.7270<br>[0.7235, 0.7306] | 0.6484<br>[0.6445, 0.6526] | 0.5445<br>[0.5399, 0.5489] | 0.4959<br>[0.4911, 0.5002] | 0.6618<br>[0.6586, 0.6652] |
| GTE / KM-Prox                  | 0.7267<br>[0.7232, 0.7304] | 0.6935<br>[0.6901, 0.6972] | 0.6267<br>[0.6233, 0.6305] | 0.5481<br>[0.5443, 0.5520] | 0.6764<br>[0.6732, 0.6797] |
| GTE / KM-Prox (validated)      | 0.7243<br>[0.7208, 0.7281] | 0.6923<br>[0.6888, 0.6961] | 0.6292<br>[0.6256, 0.6330] | 0.5527<br>[0.5489, 0.5567] | 0.6764<br>[0.6732, 0.6798] |
| GTE / MLP                      | 0.7216<br>[0.7180, 0.7254] | 0.6778<br>[0.6741, 0.6816] | 0.5894<br>[0.5853, 0.5936] | 0.5258<br>[0.5212, 0.5302] | 0.6755<br>[0.6722, 0.6789] |
| GTE / Realized oracle          | 0.9142<br>[0.9125, 0.9159] | 0.9120<br>[0.9103, 0.9138] | 0.9080<br>[0.9063, 0.9098] | 0.9037<br>[0.9019, 0.9054] | 0.9030<br>[0.9012, 0.9047] |
| GTE / Ridge                    | 0.7269<br>[0.7234, 0.7306] | 0.6907<br>[0.6873, 0.6944] | 0.5392<br>[0.5347, 0.5438] | 0.6054<br>[0.6009, 0.6099] | 0.6796<br>[0.6765, 0.6829] |
| GTE / Family Adaptation        | 0.7357<br>[0.7319, 0.7397] | 0.7114<br>[0.7074, 0.7154] | 0.6815<br>[0.6773, 0.6858] | 0.6646<br>[0.6603, 0.6690] | 0.7006<br>[0.6973, 0.7041] |
| GTE / Validated kNN            | 0.7248<br>[0.7214, 0.7285] | 0.6907<br>[0.6872, 0.6944] | 0.6021<br>[0.5983, 0.6061] | 0.5310<br>[0.5270, 0.5353] | 0.6566<br>[0.6534, 0.6601] |
| GTE / kNN-100                  | 0.7225<br>[0.7192, 0.7263] | 0.6881<br>[0.6846, 0.6919] | 0.6225<br>[0.6188, 0.6264] | 0.5491<br>[0.5453, 0.5535] | 0.6615<br>[0.6583, 0.6651] |
| Qwen L21 / Avengers-Pro        | 0.7127<br>[0.7091, 0.7166] | 0.6800<br>[0.6764, 0.6841] | 0.6210<br>[0.6173, 0.6251] | 0.5499<br>[0.5460, 0.5543] | 0.6581<br>[0.6548, 0.6615] |
| Qwen L21 / KM-Prox             | 0.7287<br>[0.7251, 0.7324] | 0.6991<br>[0.6955, 0.7028] | 0.6805<br>[0.6768, 0.6844] | 0.6681<br>[0.6644, 0.6722] | 0.7028<br>[0.6996, 0.7063] |
| Qwen L21 / KM-Prox (validated) | 0.7349<br>[0.7314, 0.7388] | 0.7249<br>[0.7214, 0.7288] | 0.6953<br>[0.6916, 0.6991] | 0.6623<br>[0.6586, 0.6662] | 0.6984<br>[0.6954, 0.7018] |
| Qwen L21 / MLP                 | 0.7224<br>[0.7189, 0.7262] | 0.7009<br>[0.6974, 0.7049] | 0.6676<br>[0.6634, 0.6720] | 0.6539<br>[0.6496, 0.6584] | 0.6918<br>[0.6887, 0.6952] |
| Qwen L21 / Ridge               | 0.7195<br>[0.7158, 0.7233] | 0.7381<br>[0.7344, 0.7419] | 0.6982<br>[0.6944, 0.7021] | 0.6556<br>[0.6514, 0.6601] | 0.6902<br>[0.6871, 0.6936] |
| Qwen L21 / Family Adaptation   | 0.7523<br>[0.7486, 0.7560] | 0.7385<br>[0.7348, 0.7423] | 0.7126<br>[0.7090, 0.7164] | 0.6966<br>[0.6927, 0.7005] | 0.7157<br>[0.7125, 0.7191] |
| Qwen L21 / Validated kNN       | 0.7467<br>[0.7430, 0.7506] | 0.7323<br>[0.7285, 0.7363] | 0.7042<br>[0.7004, 0.7081] | 0.6722<br>[0.6681, 0.6764] | 0.7034<br>[0.7001, 0.7068] |
| Qwen L21 / kNN-100             | 0.7439<br>[0.7401, 0.7478] | 0.7279<br>[0.7240, 0.7319] | 0.6994<br>[0.6954, 0.7037] | 0.6691<br>[0.6648, 0.6735] | 0.7007<br>[0.6972, 0.7043] |

Table 24: RouterBench out-of-distribution utility with 95% intervals, part 3 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 100                        | 150                        | 250                        | 500                        | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.6037<br>[0.5995, 0.6083] | 0.5994<br>[0.5948, 0.6041] | 0.5800<br>[0.5754, 0.5846] | 0.5483<br>[0.5437, 0.5529] | 0.6662<br>[0.6631, 0.6698] |
| GTE / Best single model        | 0.5983<br>[0.5936, 0.6028] | 0.5884<br>[0.5836, 0.5928] | 0.6167<br>[0.6124, 0.6212] | 0.4822<br>[0.4776, 0.4870] | 0.6618<br>[0.6586, 0.6652] |
| GTE / KM-Prox                  | 0.4796<br>[0.4755, 0.4837] | 0.6351<br>[0.6308, 0.6397] | 0.6139<br>[0.6094, 0.6185] | 0.5669<br>[0.5624, 0.5715] | 0.6764<br>[0.6732, 0.6797] |
| GTE / KM-Prox (validated)      | 0.4747<br>[0.4706, 0.4790] | 0.6347<br>[0.6302, 0.6391] | 0.6163<br>[0.6119, 0.6208] | 0.5690<br>[0.5646, 0.5735] | 0.6764<br>[0.6732, 0.6798] |
| GTE / MLP                      | 0.6171<br>[0.6126, 0.6218] | 0.6120<br>[0.6075, 0.6167] | 0.5921<br>[0.5876, 0.5966] | 0.5432<br>[0.5385, 0.5479] | 0.6755<br>[0.6722, 0.6789] |
| GTE / Realized oracle          | 0.8997<br>[0.8979, 0.9015] | 0.8921<br>[0.8903, 0.8939] | 0.8771<br>[0.8751, 0.8789] | 0.8414<br>[0.8393, 0.8435] | 0.9030<br>[0.9012, 0.9047] |
| GTE / Ridge                    | 0.6336<br>[0.6292, 0.6382] | 0.6168<br>[0.6123, 0.6214] | 0.5994<br>[0.5949, 0.6040] | 0.5202<br>[0.5155, 0.5250] | 0.6796<br>[0.6765, 0.6829] |
| GTE / Family Adaptation        | 0.6544<br>[0.6498, 0.6590] | 0.6248<br>[0.6204, 0.6294] | 0.5973<br>[0.5929, 0.6020] | 0.5332<br>[0.5286, 0.5381] | 0.7006<br>[0.6973, 0.7041] |
| GTE / Validated kNN            | 0.4683<br>[0.4638, 0.4730] | 0.5252<br>[0.5201, 0.5304] | 0.5537<br>[0.5489, 0.5582] | 0.5433<br>[0.5388, 0.5480] | 0.6566<br>[0.6534, 0.6601] |
| GTE / kNN-100                  | 0.5057<br>[0.5013, 0.5102] | 0.5320<br>[0.5270, 0.5370] | 0.5560<br>[0.5513, 0.5605] | 0.5353<br>[0.5307, 0.5398] | 0.6615<br>[0.6583, 0.6651] |
| Qwen L21 / Avengers-Pro        | 0.4688<br>[0.4646, 0.4735] | 0.5908<br>[0.5864, 0.5954] | 0.5479<br>[0.5434, 0.5525] | 0.4997<br>[0.4952, 0.5044] | 0.6581<br>[0.6548, 0.6615] |
| Qwen L21 / KM-Prox             | 0.6354<br>[0.6317, 0.6393] | 0.6367<br>[0.6323, 0.6413] | 0.6168<br>[0.6124, 0.6213] | 0.5695<br>[0.5649, 0.5740] | 0.7028<br>[0.6996, 0.7063] |
| Qwen L21 / KM-Prox (validated) | 0.6220<br>[0.6180, 0.6261] | 0.5978<br>[0.5932, 0.6024] | 0.6172<br>[0.6128, 0.6217] | 0.5169<br>[0.5122, 0.5215] | 0.6984<br>[0.6954, 0.7018] |
| Qwen L21 / MLP                 | 0.6114<br>[0.6067, 0.6159] | 0.6148<br>[0.6103, 0.6193] | 0.6062<br>[0.6017, 0.6107] | 0.5272<br>[0.5226, 0.5318] | 0.6918<br>[0.6887, 0.6952] |
| Qwen L21 / Ridge               | 0.6154<br>[0.6112, 0.6199] | 0.5881<br>[0.5834, 0.5928] | 0.5594<br>[0.5547, 0.5642] | 0.5043<br>[0.4997, 0.5092] | 0.6902<br>[0.6871, 0.6936] |
| Qwen L21 / Family Adaptation   | 0.6689<br>[0.6647, 0.6733] | 0.6350<br>[0.6307, 0.6395] | 0.6191<br>[0.6147, 0.6235] | 0.5726<br>[0.5683, 0.5771] | 0.7157<br>[0.7125, 0.7191] |
| Qwen L21 / Validated kNN       | 0.6462<br>[0.6420, 0.6504] | 0.6178<br>[0.6134, 0.6223] | 0.5890<br>[0.5846, 0.5934] | 0.5405<br>[0.5360, 0.5451] | 0.7034<br>[0.7001, 0.7068] |
| Qwen L21 / kNN-100             | 0.6462<br>[0.6418, 0.6508] | 0.6189<br>[0.6143, 0.6235] | 0.5899<br>[0.5852, 0.5946] | 0.5441<br>[0.5394, 0.5487] | 0.7007<br>[0.6972, 0.7043] |

Table 25: SPROUT in-distribution utility with 95% intervals, part 1 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 0                          | 0.5                        | 1                          | 2                          | 5                          | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.8648<br>[0.8612, 0.8683] | 0.8632<br>[0.8596, 0.8667] | 0.8618<br>[0.8582, 0.8654] | 0.8593<br>[0.8558, 0.8628] | 0.8536<br>[0.8500, 0.8572] | 0.8138<br>[0.8104, 0.8173] |
| GTE / Best single model        | 0.8434<br>[0.8395, 0.8471] | 0.8409<br>[0.8370, 0.8446] | 0.8385<br>[0.8346, 0.8422] | 0.8336<br>[0.8296, 0.8373] | 0.8188<br>[0.8148, 0.8227] | 0.7868<br>[0.7831, 0.7908] |
| GTE / Family lookup            | 0.8614<br>[0.8577, 0.8650] | 0.8596<br>[0.8559, 0.8632] | 0.8580<br>[0.8544, 0.8616] | 0.8546<br>[0.8509, 0.8583] | 0.8504<br>[0.8464, 0.8543] | 0.8097<br>[0.8061, 0.8132] |
| GTE / Family-cell lookup       | 0.8626<br>[0.8589, 0.8662] | 0.8605<br>[0.8568, 0.8641] | 0.8591<br>[0.8554, 0.8627] | 0.8553<br>[0.8515, 0.8591] | 0.8510<br>[0.8473, 0.8549] | 0.8122<br>[0.8088, 0.8158] |
| GTE / KM-Prox                  | 0.8635<br>[0.8599, 0.8670] | 0.8620<br>[0.8585, 0.8655] | 0.8604<br>[0.8567, 0.8638] | 0.8584<br>[0.8549, 0.8618] | 0.8532<br>[0.8495, 0.8569] | 0.8148<br>[0.8114, 0.8183] |
| GTE / KM-Prox (validated)      | 0.8651<br>[0.8616, 0.8685] | 0.8642<br>[0.8606, 0.8676] | 0.8625<br>[0.8590, 0.8659] | 0.8611<br>[0.8575, 0.8646] | 0.8548<br>[0.8512, 0.8585] | 0.8166<br>[0.8132, 0.8200] |
| GTE / MLP                      | 0.8577<br>[0.8543, 0.8611] | 0.8563<br>[0.8529, 0.8597] | 0.8563<br>[0.8528, 0.8597] | 0.8543<br>[0.8508, 0.8578] | 0.8477<br>[0.8441, 0.8512] | 0.8105<br>[0.8073, 0.8139] |
| GTE / Realized oracle          | 0.9770<br>[0.9755, 0.9784] | 0.9768<br>[0.9753, 0.9782] | 0.9766<br>[0.9751, 0.9780] | 0.9762<br>[0.9747, 0.9776] | 0.9750<br>[0.9736, 0.9765] | 0.9605<br>[0.9590, 0.9620] |
| GTE / Ridge                    | 0.8665<br>[0.8630, 0.8702] | 0.8645<br>[0.8610, 0.8682] | 0.8627<br>[0.8591, 0.8663] | 0.8613<br>[0.8578, 0.8650] | 0.8573<br>[0.8537, 0.8609] | 0.8179<br>[0.8147, 0.8214] |
| GTE / Family Adaptation        | 0.8611<br>[0.8575, 0.8647] | 0.8593<br>[0.8557, 0.8630] | 0.8575<br>[0.8538, 0.8611] | 0.8543<br>[0.8505, 0.8579] | 0.8500<br>[0.8460, 0.8539] | 0.8093<br>[0.8057, 0.8128] |
| GTE / Validated kNN            | 0.8654<br>[0.8620, 0.8689] | 0.8642<br>[0.8607, 0.8676] | 0.8626<br>[0.8591, 0.8660] | 0.8609<br>[0.8574, 0.8643] | 0.8562<br>[0.8526, 0.8598] | 0.8183<br>[0.8150, 0.8217] |
| GTE / kNN-100                  | 0.8665<br>[0.8630, 0.8701] | 0.8650<br>[0.8616, 0.8685] | 0.8636<br>[0.8602, 0.8672] | 0.8612<br>[0.8578, 0.8648] | 0.8561<br>[0.8525, 0.8596] | 0.8187<br>[0.8154, 0.8222] |
| Qwen L21 / Avengers-Pro        | 0.8657<br>[0.8621, 0.8691] | 0.8640<br>[0.8604, 0.8675] | 0.8626<br>[0.8591, 0.8661] | 0.8612<br>[0.8577, 0.8648] | 0.8558<br>[0.8522, 0.8594] | 0.8169<br>[0.8135, 0.8203] |
| Qwen L21 / KM-Prox             | 0.8625<br>[0.8590, 0.8661] | 0.8611<br>[0.8575, 0.8647] | 0.8594<br>[0.8559, 0.8630] | 0.8568<br>[0.8531, 0.8605] | 0.8516<br>[0.8478, 0.8554] | 0.8142<br>[0.8106, 0.8176] |
| Qwen L21 / KM-Prox (validated) | 0.8669<br>[0.8635, 0.8703] | 0.8653<br>[0.8617, 0.8688] | 0.8639<br>[0.8603, 0.8674] | 0.8625<br>[0.8588, 0.8660] | 0.8566<br>[0.8529, 0.8601] | 0.8189<br>[0.8155, 0.8223] |
| Qwen L21 / MLP                 | 0.8627<br>[0.8592, 0.8661] | 0.8603<br>[0.8569, 0.8637] | 0.8590<br>[0.8557, 0.8623] | 0.8553<br>[0.8519, 0.8588] | 0.8466<br>[0.8431, 0.8500] | 0.8128<br>[0.8096, 0.8162] |
| Qwen L21 / Ridge               | 0.8686<br>[0.8652, 0.8721] | 0.8668<br>[0.8633, 0.8703] | 0.8657<br>[0.8621, 0.8693] | 0.8634<br>[0.8598, 0.8669] | 0.8572<br>[0.8536, 0.8608] | 0.8209<br>[0.8177, 0.8244] |
| Qwen L21 / Family Adaptation   | 0.8613<br>[0.8577, 0.8649] | 0.8596<br>[0.8559, 0.8632] | 0.8579<br>[0.8542, 0.8615] | 0.8546<br>[0.8509, 0.8583] | 0.8502<br>[0.8463, 0.8542] | 0.8096<br>[0.8060, 0.8131] |
| Qwen L21 / Validated kNN       | 0.8685<br>[0.8650, 0.8718] | 0.8669<br>[0.8633, 0.8702] | 0.8654<br>[0.8619, 0.8688] | 0.8629<br>[0.8594, 0.8662] | 0.8576<br>[0.8540, 0.8609] | 0.8212<br>[0.8178, 0.8246] |
| Qwen L21 / kNN-100             | 0.8676<br>[0.8640, 0.8711] | 0.8659<br>[0.8623, 0.8694] | 0.8645<br>[0.8609, 0.8680] | 0.8623<br>[0.8587, 0.8657] | 0.8573<br>[0.8538, 0.8609] | 0.8206<br>[0.8172, 0.8240] |

Table 26: SPROUT in-distribution utility with 95% intervals, part 2 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 10                         | 25                         | 50                         | 75                         | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.8446<br>[0.8410, 0.8482] | 0.8337<br>[0.8298, 0.8375] | 0.8188<br>[0.8150, 0.8228] | 0.8068<br>[0.8030, 0.8108] | 0.8138<br>[0.8104, 0.8173] |
| GTE / Best single model        | 0.8015<br>[0.7975, 0.8059] | 0.7973<br>[0.7929, 0.8018] | 0.7889<br>[0.7845, 0.7934] | 0.7804<br>[0.7760, 0.7850] | 0.7868<br>[0.7831, 0.7908] |
| GTE / Family lookup            | 0.8436<br>[0.8395, 0.8476] | 0.8269<br>[0.8229, 0.8310] | 0.8152<br>[0.8112, 0.8194] | 0.8037<br>[0.7997, 0.8079] | 0.8097<br>[0.8061, 0.8132] |
| GTE / Family-cell lookup       | 0.8448<br>[0.8411, 0.8488] | 0.8335<br>[0.8296, 0.8374] | 0.8205<br>[0.8165, 0.8245] | 0.8063<br>[0.8024, 0.8104] | 0.8122<br>[0.8088, 0.8158] |
| GTE / KM-Prox                  | 0.8466<br>[0.8429, 0.8504] | 0.8342<br>[0.8304, 0.8382] | 0.8166<br>[0.8126, 0.8207] | 0.8056<br>[0.8016, 0.8097] | 0.8148<br>[0.8114, 0.8183] |
| GTE / KM-Prox (validated)      | 0.8477<br>[0.8440, 0.8515] | 0.8370<br>[0.8332, 0.8408] | 0.8204<br>[0.8165, 0.8243] | 0.8066<br>[0.8027, 0.8108] | 0.8166<br>[0.8132, 0.8200] |
| GTE / MLP                      | 0.8381<br>[0.8345, 0.8417] | 0.8274<br>[0.8237, 0.8313] | 0.8138<br>[0.8102, 0.8175] | 0.8025<br>[0.7988, 0.8063] | 0.8105<br>[0.8073, 0.8139] |
| GTE / Realized oracle          | 0.9733<br>[0.9719, 0.9748] | 0.9696<br>[0.9682, 0.9711] | 0.9646<br>[0.9632, 0.9661] | 0.9599<br>[0.9584, 0.9615] | 0.9605<br>[0.9590, 0.9620] |
| GTE / Ridge                    | 0.8516<br>[0.8481, 0.8553] | 0.8383<br>[0.8345, 0.8421] | 0.8214<br>[0.8177, 0.8254] | 0.8080<br>[0.8043, 0.8121] | 0.8179<br>[0.8147, 0.8214] |
| GTE / Family Adaptation        | 0.8432<br>[0.8391, 0.8472] | 0.8265<br>[0.8224, 0.8307] | 0.8149<br>[0.8109, 0.8190] | 0.8034<br>[0.7994, 0.8076] | 0.8093<br>[0.8057, 0.8128] |
| GTE / Validated kNN            | 0.8499<br>[0.8464, 0.8536] | 0.8381<br>[0.8345, 0.8419] | 0.8226<br>[0.8188, 0.8264] | 0.8104<br>[0.8066, 0.8144] | 0.8183<br>[0.8150, 0.8217] |
| GTE / kNN-100                  | 0.8499<br>[0.8465, 0.8537] | 0.8385<br>[0.8348, 0.8422] | 0.8231<br>[0.8193, 0.8269] | 0.8106<br>[0.8069, 0.8146] | 0.8187<br>[0.8154, 0.8222] |
| Qwen L21 / Avengers-Pro        | 0.8491<br>[0.8454, 0.8526] | 0.8385<br>[0.8347, 0.8424] | 0.8228<br>[0.8190, 0.8268] | 0.8103<br>[0.8064, 0.8142] | 0.8169<br>[0.8135, 0.8203] |
| Qwen L21 / KM-Prox             | 0.8460<br>[0.8421, 0.8499] | 0.8336<br>[0.8296, 0.8375] | 0.8164<br>[0.8125, 0.8206] | 0.8051<br>[0.8012, 0.8094] | 0.8142<br>[0.8106, 0.8176] |
| Qwen L21 / KM-Prox (validated) | 0.8506<br>[0.8468, 0.8543] | 0.8383<br>[0.8345, 0.8420] | 0.8215<br>[0.8177, 0.8254] | 0.8094<br>[0.8055, 0.8134] | 0.8189<br>[0.8155, 0.8223] |
| Qwen L21 / MLP                 | 0.8365<br>[0.8330, 0.8402] | 0.8364<br>[0.8327, 0.8403] | 0.8161<br>[0.8125, 0.8199] | 0.8047<br>[0.8010, 0.8088] | 0.8128<br>[0.8096, 0.8162] |
| Qwen L21 / Ridge               | 0.8517<br>[0.8481, 0.8553] | 0.8405<br>[0.8368, 0.8442] | 0.8257<br>[0.8219, 0.8295] | 0.8128<br>[0.8090, 0.8167] | 0.8209<br>[0.8177, 0.8244] |
| Qwen L21 / Family Adaptation   | 0.8435<br>[0.8394, 0.8475] | 0.8268<br>[0.8228, 0.8310] | 0.8152<br>[0.8111, 0.8193] | 0.8037<br>[0.7997, 0.8079] | 0.8096<br>[0.8060, 0.8131] |
| Qwen L21 / Validated kNN       | 0.8526<br>[0.8490, 0.8562] | 0.8399<br>[0.8363, 0.8437] | 0.8241<br>[0.8204, 0.8279] | 0.8120<br>[0.8083, 0.8161] | 0.8212<br>[0.8178, 0.8246] |
| Qwen L21 / kNN-100             | 0.8516<br>[0.8482, 0.8551] | 0.8390<br>[0.8354, 0.8427] | 0.8232<br>[0.8194, 0.8271] | 0.8112<br>[0.8073, 0.8152] | 0.8206<br>[0.8172, 0.8240] |

Table 27: SPROUT in-distribution utility with 95% intervals, part 3 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 100                        | 150                        | 250                        | 500                        | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7943<br>[0.7904, 0.7983] | 0.7731<br>[0.7693, 0.7773] | 0.7367<br>[0.7325, 0.7411] | 0.6689<br>[0.6643, 0.6738] | 0.8138<br>[0.8104, 0.8173] |
| GTE / Best single model        | 0.7720<br>[0.7676, 0.7766] | 0.7551<br>[0.7507, 0.7598] | 0.7214<br>[0.7168, 0.7262] | 0.6371<br>[0.6321, 0.6423] | 0.7868<br>[0.7831, 0.7908] |
| GTE / Family lookup            | 0.7927<br>[0.7886, 0.7970] | 0.7709<br>[0.7667, 0.7753] | 0.7313<br>[0.7270, 0.7358] | 0.6574<br>[0.6524, 0.6624] | 0.8097<br>[0.8061, 0.8132] |
| GTE / Family-cell lookup       | 0.7955<br>[0.7915, 0.7998] | 0.7752<br>[0.7711, 0.7795] | 0.7364<br>[0.7322, 0.7408] | 0.6584<br>[0.6536, 0.6635] | 0.8122<br>[0.8088, 0.8158] |
| GTE / KM-Prox                  | 0.7948<br>[0.7908, 0.7989] | 0.7764<br>[0.7724, 0.7807] | 0.7434<br>[0.7391, 0.7478] | 0.6780<br>[0.6735, 0.6832] | 0.8148<br>[0.8114, 0.8183] |
| GTE / KM-Prox (validated)      | 0.7951<br>[0.7911, 0.7993] | 0.7766<br>[0.7726, 0.7809] | 0.7466<br>[0.7424, 0.7510] | 0.6777<br>[0.6731, 0.6826] | 0.8166<br>[0.8132, 0.8200] |
| GTE / MLP                      | 0.7937<br>[0.7899, 0.7977] | 0.7746<br>[0.7706, 0.7787] | 0.7424<br>[0.7383, 0.7467] | 0.6719<br>[0.6675, 0.6767] | 0.8105<br>[0.8073, 0.8139] |
| GTE / Realized oracle          | 0.9554<br>[0.9538, 0.9570] | 0.9467<br>[0.9451, 0.9484] | 0.9319<br>[0.9302, 0.9338] | 0.9030<br>[0.9010, 0.9051] | 0.9605<br>[0.9590, 0.9620] |
| GTE / Ridge                    | 0.7982<br>[0.7945, 0.8022] | 0.7783<br>[0.7744, 0.7824] | 0.7447<br>[0.7406, 0.7490] | 0.6798<br>[0.6752, 0.6848] | 0.8179<br>[0.8147, 0.8214] |
| GTE / Family Adaptation        | 0.7923<br>[0.7882, 0.7966] | 0.7704<br>[0.7662, 0.7747] | 0.7311<br>[0.7266, 0.7356] | 0.6571<br>[0.6521, 0.6622] | 0.8093<br>[0.8057, 0.8128] |
| GTE / Validated kNN            | 0.7993<br>[0.7955, 0.8033] | 0.7800<br>[0.7760, 0.7840] | 0.7471<br>[0.7429, 0.7513] | 0.6816<br>[0.6770, 0.6865] | 0.8183<br>[0.8150, 0.8217] |
| GTE / kNN-100                  | 0.7997<br>[0.7959, 0.8037] | 0.7808<br>[0.7769, 0.7849] | 0.7471<br>[0.7430, 0.7512] | 0.6812<br>[0.6766, 0.6860] | 0.8187<br>[0.8154, 0.8222] |
| Qwen L21 / Avengers-Pro        | 0.7984<br>[0.7946, 0.8025] | 0.7776<br>[0.7737, 0.7817] | 0.7380<br>[0.7337, 0.7423] | 0.6751<br>[0.6704, 0.6800] | 0.8169<br>[0.8135, 0.8203] |
| Qwen L21 / KM-Prox             | 0.7941<br>[0.7902, 0.7984] | 0.7754<br>[0.7713, 0.7797] | 0.7432<br>[0.7390, 0.7477] | 0.6788<br>[0.6741, 0.6840] | 0.8142<br>[0.8106, 0.8176] |
| Qwen L21 / KM-Prox (validated) | 0.7990<br>[0.7951, 0.8031] | 0.7807<br>[0.7765, 0.7849] | 0.7490<br>[0.7446, 0.7533] | 0.6824<br>[0.6777, 0.6874] | 0.8189<br>[0.8155, 0.8223] |
| Qwen L21 / MLP                 | 0.7937<br>[0.7901, 0.7977] | 0.7741<br>[0.7702, 0.7782] | 0.7438<br>[0.7399, 0.7480] | 0.6777<br>[0.6732, 0.6824] | 0.8128<br>[0.8096, 0.8162] |
| Qwen L21 / Ridge               | 0.8014<br>[0.7975, 0.8055] | 0.7826<br>[0.7787, 0.7868] | 0.7502<br>[0.7462, 0.7546] | 0.6856<br>[0.6810, 0.6906] | 0.8209<br>[0.8177, 0.8244] |
| Qwen L21 / Family Adaptation   | 0.7927<br>[0.7885, 0.7969] | 0.7707<br>[0.7666, 0.7751] | 0.7313<br>[0.7270, 0.7358] | 0.6574<br>[0.6523, 0.6625] | 0.8096<br>[0.8060, 0.8131] |
| Qwen L21 / Validated kNN       | 0.8018<br>[0.7978, 0.8057] | 0.7836<br>[0.7795, 0.7876] | 0.7522<br>[0.7480, 0.7563] | 0.6886<br>[0.6840, 0.6932] | 0.8212<br>[0.8178, 0.8246] |
| Qwen L21 / kNN-100             | 0.8011<br>[0.7971, 0.8052] | 0.7834<br>[0.7794, 0.7874] | 0.7517<br>[0.7476, 0.7559] | 0.6890<br>[0.6844, 0.6935] | 0.8206<br>[0.8172, 0.8240] |

Table 28: SPROUT out-of-distribution utility with 95% intervals, part 1 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 0                          | 0.5                        | 1                          | 2                          | 5                          | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7891<br>[0.7857, 0.7924] | 0.7883<br>[0.7849, 0.7916] | 0.7875<br>[0.7841, 0.7908] | 0.7874<br>[0.7840, 0.7907] | 0.7830<br>[0.7796, 0.7864] | 0.7439<br>[0.7408, 0.7470] |
| GTE / Best single model        | 0.8445<br>[0.8414, 0.8475] | 0.8420<br>[0.8389, 0.8451] | 0.8040<br>[0.8008, 0.8070] | 0.8010<br>[0.7978, 0.8041] | 0.7669<br>[0.7634, 0.7704] | 0.7575<br>[0.7544, 0.7604] |
| GTE / KM-Prox                  | 0.7887<br>[0.7854, 0.7921] | 0.7873<br>[0.7840, 0.7906] | 0.7861<br>[0.7828, 0.7894] | 0.7837<br>[0.7803, 0.7869] | 0.7808<br>[0.7774, 0.7841] | 0.7460<br>[0.7429, 0.7491] |
| GTE / KM-Prox (validated)      | 0.7968<br>[0.7934, 0.8001] | 0.7948<br>[0.7914, 0.7981] | 0.7929<br>[0.7896, 0.7962] | 0.7889<br>[0.7855, 0.7921] | 0.7765<br>[0.7731, 0.7798] | 0.7483<br>[0.7453, 0.7513] |
| GTE / MLP                      | 0.8088<br>[0.8055, 0.8121] | 0.8056<br>[0.8022, 0.8089] | 0.8125<br>[0.8092, 0.8158] | 0.8071<br>[0.8037, 0.8104] | 0.7971<br>[0.7936, 0.8005] | 0.7564<br>[0.7536, 0.7592] |
| GTE / Realized oracle          | 0.9766<br>[0.9753, 0.9778] | 0.9764<br>[0.9751, 0.9776] | 0.9762<br>[0.9750, 0.9774] | 0.9758<br>[0.9746, 0.9770] | 0.9747<br>[0.9734, 0.9759] | 0.9601<br>[0.9588, 0.9613] |
| GTE / Ridge                    | 0.8328<br>[0.8296, 0.8359] | 0.8292<br>[0.8260, 0.8322] | 0.8256<br>[0.8225, 0.8288] | 0.8201<br>[0.8169, 0.8232] | 0.8097<br>[0.8065, 0.8129] | 0.7674<br>[0.7645, 0.7703] |
| GTE / Family Adaptation        | 0.8097<br>[0.8065, 0.8126] | 0.8061<br>[0.8028, 0.8091] | 0.8021<br>[0.7989, 0.8052] | 0.7918<br>[0.7883, 0.7949] | 0.7802<br>[0.7767, 0.7837] | 0.7603<br>[0.7573, 0.7632] |
| GTE / Validated kNN            | 0.8037<br>[0.8004, 0.8070] | 0.8020<br>[0.7988, 0.8053] | 0.8001<br>[0.7969, 0.8035] | 0.7961<br>[0.7928, 0.7995] | 0.7846<br>[0.7813, 0.7879] | 0.7534<br>[0.7504, 0.7564] |
| GTE / kNN-100                  | 0.8035<br>[0.8002, 0.8067] | 0.8013<br>[0.7980, 0.8046] | 0.7994<br>[0.7962, 0.8026] | 0.7955<br>[0.7922, 0.7988] | 0.7873<br>[0.7839, 0.7906] | 0.7529<br>[0.7498, 0.7559] |
| Qwen L21 / Avengers-Pro        | 0.8148<br>[0.8115, 0.8180] | 0.8143<br>[0.8110, 0.8175] | 0.8135<br>[0.8102, 0.8167] | 0.8113<br>[0.8080, 0.8145] | 0.8118<br>[0.8085, 0.8153] | 0.7714<br>[0.7683, 0.7745] |
| Qwen L21 / KM-Prox             | 0.8250<br>[0.8217, 0.8283] | 0.8221<br>[0.8188, 0.8253] | 0.8189<br>[0.8157, 0.8222] | 0.8118<br>[0.8085, 0.8151] | 0.7829<br>[0.7795, 0.7864] | 0.7709<br>[0.7679, 0.7739] |
| Qwen L21 / KM-Prox (validated) | 0.8195<br>[0.8163, 0.8229] | 0.8164<br>[0.8132, 0.8199] | 0.8134<br>[0.8102, 0.8169] | 0.8074<br>[0.8041, 0.8108] | 0.8055<br>[0.8022, 0.8088] | 0.7764<br>[0.7734, 0.7795] |
| Qwen L21 / MLP                 | 0.8139<br>[0.8107, 0.8173] | 0.8145<br>[0.8112, 0.8180] | 0.8080<br>[0.8046, 0.8114] | 0.8044<br>[0.8011, 0.8079] | 0.7997<br>[0.7965, 0.8030] | 0.7667<br>[0.7639, 0.7696] |
| Qwen L21 / Ridge               | 0.8103<br>[0.8070, 0.8135] | 0.8083<br>[0.8050, 0.8114] | 0.8064<br>[0.8032, 0.8095] | 0.8027<br>[0.7994, 0.8058] | 0.7806<br>[0.7772, 0.7838] | 0.7712<br>[0.7683, 0.7740] |
| Qwen L21 / Family Adaptation   | 0.8163<br>[0.8130, 0.8196] | 0.8129<br>[0.8095, 0.8161] | 0.8094<br>[0.8061, 0.8126] | 0.8025<br>[0.7992, 0.8058] | 0.8070<br>[0.8035, 0.8103] | 0.7767<br>[0.7737, 0.7797] |
| Qwen L21 / Validated kNN       | 0.8212<br>[0.8179, 0.8244] | 0.8177<br>[0.8143, 0.8208] | 0.8157<br>[0.8125, 0.8190] | 0.8104<br>[0.8070, 0.8137] | 0.8050<br>[0.8018, 0.8083] | 0.7739<br>[0.7709, 0.7769] |
| Qwen L21 / kNN-100             | 0.8243<br>[0.8212, 0.8275] | 0.8225<br>[0.8192, 0.8257] | 0.8200<br>[0.8169, 0.8232] | 0.8170<br>[0.8137, 0.8202] | 0.8073<br>[0.8040, 0.8106] | 0.7731<br>[0.7701, 0.7760] |

Table 29: SPROUT out-of-distribution utility with 95% intervals, part 2 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 10                         | 25                         | 50                         | 75                         | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7822<br>[0.7789, 0.7856] | 0.7799<br>[0.7765, 0.7832] | 0.7663<br>[0.7629, 0.7698] | 0.7548<br>[0.7514, 0.7582] | 0.7439<br>[0.7408, 0.7470] |
| GTE / Best single model        | 0.7576<br>[0.7542, 0.7611] | 0.7641<br>[0.7607, 0.7676] | 0.7546<br>[0.7511, 0.7581] | 0.7451<br>[0.7416, 0.7486] | 0.7575<br>[0.7544, 0.7604] |
| GTE / KM-Prox                  | 0.7796<br>[0.7762, 0.7829] | 0.7725<br>[0.7690, 0.7759] | 0.7579<br>[0.7543, 0.7614] | 0.7437<br>[0.7400, 0.7471] | 0.7460<br>[0.7429, 0.7491] |
| GTE / KM-Prox (validated)      | 0.7657<br>[0.7622, 0.7690] | 0.7699<br>[0.7665, 0.7732] | 0.7577<br>[0.7543, 0.7611] | 0.7443<br>[0.7409, 0.7478] | 0.7483<br>[0.7453, 0.7513] |
| GTE / MLP                      | 0.7908<br>[0.7873, 0.7940] | 0.7744<br>[0.7712, 0.7776] | 0.7452<br>[0.7416, 0.7487] | 0.7354<br>[0.7318, 0.7389] | 0.7564<br>[0.7536, 0.7592] |
| GTE / Realized oracle          | 0.9730<br>[0.9717, 0.9742] | 0.9692<br>[0.9680, 0.9705] | 0.9642<br>[0.9629, 0.9655] | 0.9595<br>[0.9582, 0.9608] | 0.9601<br>[0.9588, 0.9613] |
| GTE / Ridge                    | 0.7895<br>[0.7862, 0.7927] | 0.7667<br>[0.7633, 0.7701] | 0.7527<br>[0.7492, 0.7562] | 0.7430<br>[0.7395, 0.7466] | 0.7674<br>[0.7645, 0.7703] |
| GTE / Family Adaptation        | 0.7834<br>[0.7799, 0.7866] | 0.7756<br>[0.7721, 0.7789] | 0.7648<br>[0.7613, 0.7681] | 0.7579<br>[0.7543, 0.7614] | 0.7603<br>[0.7573, 0.7632] |
| GTE / Validated kNN            | 0.7816<br>[0.7782, 0.7851] | 0.7724<br>[0.7689, 0.7758] | 0.7596<br>[0.7562, 0.7630] | 0.7482<br>[0.7447, 0.7516] | 0.7534<br>[0.7504, 0.7564] |
| GTE / kNN-100                  | 0.7820<br>[0.7787, 0.7855] | 0.7734<br>[0.7698, 0.7767] | 0.7607<br>[0.7573, 0.7641] | 0.7496<br>[0.7462, 0.7530] | 0.7529<br>[0.7498, 0.7559] |
| Qwen L21 / Avengers-Pro        | 0.8020<br>[0.7985, 0.8053] | 0.7948<br>[0.7914, 0.7982] | 0.7866<br>[0.7832, 0.7899] | 0.7764<br>[0.7730, 0.7798] | 0.7714<br>[0.7683, 0.7745] |
| Qwen L21 / KM-Prox             | 0.7754<br>[0.7719, 0.7788] | 0.7750<br>[0.7715, 0.7783] | 0.7650<br>[0.7615, 0.7683] | 0.7697<br>[0.7661, 0.7729] | 0.7709<br>[0.7679, 0.7739] |
| Qwen L21 / KM-Prox (validated) | 0.8044<br>[0.8010, 0.8077] | 0.7747<br>[0.7712, 0.7779] | 0.7910<br>[0.7877, 0.7944] | 0.7829<br>[0.7795, 0.7863] | 0.7764<br>[0.7734, 0.7795] |
| Qwen L21 / MLP                 | 0.7869<br>[0.7836, 0.7901] | 0.7924<br>[0.7891, 0.7959] | 0.7522<br>[0.7486, 0.7556] | 0.7794<br>[0.7758, 0.7829] | 0.7667<br>[0.7639, 0.7696] |
| Qwen L21 / Ridge               | 0.7993<br>[0.7961, 0.8025] | 0.7897<br>[0.7864, 0.7928] | 0.7923<br>[0.7892, 0.7954] | 0.7624<br>[0.7591, 0.7655] | 0.7712<br>[0.7683, 0.7740] |
| Qwen L21 / Family Adaptation   | 0.8085<br>[0.8050, 0.8117] | 0.8042<br>[0.8008, 0.8075] | 0.7957<br>[0.7924, 0.7991] | 0.7873<br>[0.7839, 0.7906] | 0.7767<br>[0.7737, 0.7797] |
| Qwen L21 / Validated kNN       | 0.7992<br>[0.7958, 0.8025] | 0.7991<br>[0.7957, 0.8025] | 0.7884<br>[0.7850, 0.7918] | 0.7736<br>[0.7702, 0.7770] | 0.7739<br>[0.7709, 0.7769] |
| Qwen L21 / kNN-100             | 0.8009<br>[0.7974, 0.8042] | 0.7946<br>[0.7911, 0.7979] | 0.7830<br>[0.7796, 0.7864] | 0.7716<br>[0.7682, 0.7751] | 0.7731<br>[0.7701, 0.7760] |

Table 30: SPROUT out-of-distribution utility with 95% intervals, part 3 of 3. Mean averages all thirteen penalties.

| Method / $\lambda$             | 100                        | 150                        | 250                        | 500                        | Mean                       |
|--------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| GTE / Avengers-Pro             | 0.7423<br>[0.7390, 0.7458] | 0.7175<br>[0.7141, 0.7211] | 0.6413<br>[0.6375, 0.6449] | 0.5510<br>[0.5469, 0.5550] | 0.7439<br>[0.7408, 0.7470] |
| GTE / Best single model        | 0.7356<br>[0.7321, 0.7392] | 0.7166<br>[0.7131, 0.7203] | 0.7218<br>[0.7181, 0.7255] | 0.5930<br>[0.5890, 0.5971] | 0.7575<br>[0.7544, 0.7604] |
| GTE / KM-Prox                  | 0.7319<br>[0.7282, 0.7354] | 0.7123<br>[0.7086, 0.7159] | 0.6822<br>[0.6785, 0.6859] | 0.5908<br>[0.5866, 0.5951] | 0.7460<br>[0.7429, 0.7491] |
| GTE / KM-Prox (validated)      | 0.7330<br>[0.7295, 0.7364] | 0.7215<br>[0.7180, 0.7251] | 0.6907<br>[0.6870, 0.6944] | 0.5951<br>[0.5909, 0.5990] | 0.7483<br>[0.7453, 0.7513] |
| GTE / MLP                      | 0.7313<br>[0.7277, 0.7348] | 0.7164<br>[0.7129, 0.7201] | 0.7218<br>[0.7181, 0.7255] | 0.5863<br>[0.5821, 0.5905] | 0.7564<br>[0.7536, 0.7592] |
| GTE / Realized oracle          | 0.9549<br>[0.9535, 0.9562] | 0.9463<br>[0.9449, 0.9477] | 0.9316<br>[0.9300, 0.9330] | 0.9026<br>[0.9008, 0.9042] | 0.9601<br>[0.9588, 0.9613] |
| GTE / Ridge                    | 0.7339<br>[0.7304, 0.7375] | 0.7157<br>[0.7121, 0.7193] | 0.7210<br>[0.7172, 0.7246] | 0.6364<br>[0.6322, 0.6404] | 0.7674<br>[0.7645, 0.7703] |
| GTE / Family Adaptation        | 0.7514<br>[0.7478, 0.7549] | 0.7380<br>[0.7345, 0.7416] | 0.7127<br>[0.7090, 0.7164] | 0.6099<br>[0.6057, 0.6142] | 0.7603<br>[0.7573, 0.7632] |
| GTE / Validated kNN            | 0.7362<br>[0.7327, 0.7397] | 0.7143<br>[0.7109, 0.7177] | 0.6844<br>[0.6806, 0.6882] | 0.6114<br>[0.6072, 0.6157] | 0.7534<br>[0.7504, 0.7564] |
| GTE / kNN-100                  | 0.7379<br>[0.7345, 0.7413] | 0.7149<br>[0.7114, 0.7184] | 0.6786<br>[0.6748, 0.6824] | 0.6037<br>[0.5996, 0.6080] | 0.7529<br>[0.7498, 0.7559] |
| Qwen L21 / Avengers-Pro        | 0.7662<br>[0.7628, 0.7696] | 0.7458<br>[0.7424, 0.7493] | 0.7099<br>[0.7063, 0.7134] | 0.5803<br>[0.5760, 0.5844] | 0.7714<br>[0.7683, 0.7745] |
| Qwen L21 / KM-Prox             | 0.7668<br>[0.7633, 0.7702] | 0.7521<br>[0.7485, 0.7556] | 0.7216<br>[0.7180, 0.7253] | 0.6357<br>[0.6318, 0.6397] | 0.7709<br>[0.7679, 0.7739] |
| Qwen L21 / KM-Prox (validated) | 0.7662<br>[0.7628, 0.7697] | 0.7565<br>[0.7530, 0.7600] | 0.7216<br>[0.7180, 0.7253] | 0.6339<br>[0.6299, 0.6378] | 0.7764<br>[0.7734, 0.7795] |
| Qwen L21 / MLP                 | 0.7592<br>[0.7556, 0.7628] | 0.7440<br>[0.7404, 0.7475] | 0.7145<br>[0.7108, 0.7182] | 0.5977<br>[0.5937, 0.6018] | 0.7667<br>[0.7639, 0.7696] |
| Qwen L21 / Ridge               | 0.7701<br>[0.7667, 0.7733] | 0.7519<br>[0.7484, 0.7556] | 0.7217<br>[0.7180, 0.7255] | 0.6299<br>[0.6258, 0.6339] | 0.7712<br>[0.7683, 0.7740] |
| Qwen L21 / Family Adaptation   | 0.7772<br>[0.7739, 0.7806] | 0.7571<br>[0.7536, 0.7605] | 0.7210<br>[0.7173, 0.7247] | 0.5975<br>[0.5933, 0.6016] | 0.7767<br>[0.7737, 0.7797] |
| Qwen L21 / Validated kNN       | 0.7629<br>[0.7595, 0.7663] | 0.7416<br>[0.7382, 0.7451] | 0.7060<br>[0.7021, 0.7097] | 0.6200<br>[0.6160, 0.6241] | 0.7739<br>[0.7709, 0.7769] |
| Qwen L21 / kNN-100             | 0.7606<br>[0.7571, 0.7641] | 0.7384<br>[0.7348, 0.7419] | 0.6986<br>[0.6947, 0.7023] | 0.6119<br>[0.6079, 0.6160] | 0.7731<br>[0.7701, 0.7760] |

## D.2 SharedTrunkNet results and robustness

The implementation and selection procedures are specified in Appendix A.3. Table 31 also reports the fixed-L21 and common-layer controls. The per-model selection procedure used in the main comparison has the highest mean among these three SharedTrunkNet variants on both benchmarks.

Table 31: Out-of-distribution mean utility over thirteen penalties, with 95% intervals.

| Method                                    | RouterBench                | SPROUT                     |
|-------------------------------------------|----------------------------|----------------------------|
| SharedTrunkNet: fixed L21                 | 0.6841<br>[0.6809, 0.6875] | 0.7812<br>[0.7783, 0.7841] |
| SharedTrunkNet: common selected layer     | 0.6803<br>[0.6771, 0.6837] | 0.7806<br>[0.7777, 0.7835] |
| SharedTrunkNet: per-model selected layers | 0.6949<br>[0.6918, 0.6983] | 0.7829<br>[0.7800, 0.7857] |
| L21 kNN-100                               | 0.7007<br>[0.6972, 0.7043] | 0.7731<br>[0.7701, 0.7760] |
| L21 Family Adaptation                     | 0.7157<br>[0.7125, 0.7191] | 0.7767<br>[0.7737, 0.7797] |

**Paired utility differences.** Table 32 subtracts the main SharedTrunkNet variant from each L21 method, using the same queries and averaging all thirteen penalties. Intervals use the paired query bootstrap in Appendix A.1. Both L21 methods have higher mean utility on RouterBench and lower mean utility on SPROUT; none of these four difference intervals includes zero. These intervals condition on the fitted routers and evaluated families.

Table 32: Mean utility difference from SharedTrunkNet, with paired 95% intervals. Positive values favor the L21 method.

| Comparison                                 | RouterBench                   | SPROUT                        |
|--------------------------------------------|-------------------------------|-------------------------------|
| L21 kNN-100 minus SharedTrunkNet           | +0.0058<br>[+0.0043, +0.0072] | −0.0097<br>[−0.0112, −0.0083] |
| L21 Family Adaptation minus SharedTrunkNet | +0.0208<br>[+0.0195, +0.0221] | −0.0062<br>[−0.0073, −0.0050] |

**Results at every penalty.** Tables 33 and 34 report utility at all thirteen cost penalties. SharedTrunkNet denotes the per-model selection procedure used in the main text. On RouterBench, the Family Adaptation has higher utility at 13 of 13 penalties and kNN-100 at 7 of 13. On SPROUT, each has higher utility at 3 of 13 penalties. These counts describe the observed comparisons; penalties are not independent statistical replications.

Table 33: RouterBench: out-of-distribution utility at every penalty, with 95% intervals.

| $\lambda$ | SharedTrunkNet             | L21 kNN-100                | L21 Family Adaptation      |
|-----------|----------------------------|----------------------------|----------------------------|
| 0         | 0.7916<br>[0.7882, 0.7952] | 0.7854<br>[0.7818, 0.7891] | 0.7921<br>[0.7887, 0.7958] |
| 1         | 0.7887<br>[0.7853, 0.7925] | 0.7805<br>[0.7769, 0.7843] | 0.7889<br>[0.7854, 0.7926] |
| 2         | 0.7855<br>[0.7820, 0.7891] | 0.7771<br>[0.7734, 0.7809] | 0.7856<br>[0.7822, 0.7893] |
| 5         | 0.7754<br>[0.7719, 0.7790] | 0.7678<br>[0.7642, 0.7716] | 0.7759<br>[0.7724, 0.7796] |
| 10        | 0.7585<br>[0.7551, 0.7622] | 0.7587<br>[0.7549, 0.7626] | 0.7661<br>[0.7624, 0.7698] |
| 20        | 0.7207<br>[0.7171, 0.7246] | 0.7439<br>[0.7401, 0.7478] | 0.7523<br>[0.7486, 0.7560] |
| 30        | 0.6801<br>[0.6764, 0.6841] | 0.7279<br>[0.7240, 0.7319] | 0.7385<br>[0.7348, 0.7423] |
| 50        | 0.6786<br>[0.6746, 0.6826] | 0.6994<br>[0.6954, 0.7037] | 0.7126<br>[0.7090, 0.7164] |
| 75        | 0.6780<br>[0.6739, 0.6823] | 0.6691<br>[0.6648, 0.6735] | 0.6966<br>[0.6927, 0.7005] |
| 100       | 0.6323<br>[0.6279, 0.6366] | 0.6462<br>[0.6418, 0.6508] | 0.6689<br>[0.6647, 0.6733] |
| 150       | 0.6078<br>[0.6034, 0.6125] | 0.6189<br>[0.6143, 0.6235] | 0.6350<br>[0.6307, 0.6395] |
| 250       | 0.6015<br>[0.5972, 0.6060] | 0.5899<br>[0.5852, 0.5946] | 0.6191<br>[0.6147, 0.6235] |
| 500       | 0.5352<br>[0.5307, 0.5400] | 0.5441<br>[0.5394, 0.5487] | 0.5726<br>[0.5683, 0.5771] |
| Mean      | 0.6949<br>[0.6918, 0.6983] | 0.7007<br>[0.6972, 0.7043] | 0.7157<br>[0.7125, 0.7191] |

Table 34: SPROUT: out-of-distribution utility at every penalty, with 95% intervals.

| $\lambda$ | SharedTrunkNet             | L21 kNN-100                | L21 Family Adaptation      |
|-----------|----------------------------|----------------------------|----------------------------|
| 0         | 0.8212<br>[0.8179, 0.8244] | 0.8243<br>[0.8212, 0.8275] | 0.8163<br>[0.8130, 0.8196] |
| 0.5       | 0.8321<br>[0.8291, 0.8354] | 0.8225<br>[0.8192, 0.8257] | 0.8129<br>[0.8095, 0.8161] |
| 1         | 0.8145<br>[0.8113, 0.8178] | 0.8200<br>[0.8169, 0.8232] | 0.8094<br>[0.8061, 0.8126] |
| 2         | 0.8091<br>[0.8059, 0.8124] | 0.8170<br>[0.8137, 0.8202] | 0.8025<br>[0.7992, 0.8058] |
| 5         | 0.8118<br>[0.8085, 0.8150] | 0.8073<br>[0.8040, 0.8106] | 0.8070<br>[0.8035, 0.8103] |
| 10        | 0.8155<br>[0.8122, 0.8188] | 0.8009<br>[0.7974, 0.8042] | 0.8085<br>[0.8050, 0.8117] |
| 25        | 0.8074<br>[0.8042, 0.8106] | 0.7946<br>[0.7911, 0.7979] | 0.8042<br>[0.8008, 0.8075] |
| 50        | 0.7972<br>[0.7939, 0.8004] | 0.7830<br>[0.7796, 0.7864] | 0.7957<br>[0.7924, 0.7991] |
| 75        | 0.7746<br>[0.7712, 0.7779] | 0.7716<br>[0.7682, 0.7751] | 0.7873<br>[0.7839, 0.7906] |
| 100       | 0.7847<br>[0.7813, 0.7881] | 0.7606<br>[0.7571, 0.7641] | 0.7772<br>[0.7739, 0.7806] |
| 150       | 0.7566<br>[0.7531, 0.7601] | 0.7384<br>[0.7348, 0.7419] | 0.7571<br>[0.7536, 0.7605] |
| 250       | 0.7134<br>[0.7096, 0.7172] | 0.6986<br>[0.6947, 0.7023] | 0.7210<br>[0.7173, 0.7247] |
| 500       | 0.6392<br>[0.6351, 0.6431] | 0.6119<br>[0.6079, 0.6160] | 0.5975<br>[0.5933, 0.6016] |
| Mean      | 0.7829<br>[0.7800, 0.7857] | 0.7731<br>[0.7701, 0.7760] | 0.7767<br>[0.7737, 0.7797] |

### D.3 Breaking Down the Routing Gain by Family

We compare kNN-100 with GTE and Qwen L21 embeddings on each held-out family. Each query is routed using only training queries from the other families, with the same candidate models and cost penalties for both representations. We average realized utility across the thirteen penalties and across queries within each family. The gain  $\Delta$  is L21 minus GTE.

**Equal query and equal family weights.** Let  $\bar{u}_j$  be the mean utility within family  $j$ , containing  $n_j$  queries, and let  $N = \sum_{j=1}^J n_j$ . We report both

$$\bar{u}_{\text{query}} = \sum_{j=1}^J \frac{n_j}{N} \bar{u}_j, \quad \bar{u}_{\text{family}} = \frac{1}{J} \sum_{j=1}^J \bar{u}_j.$$

The equal-query average gives every test query the same weight, so larger families contribute more. It matches the aggregation used in the main text. The equal-family average first normalizes each family by its query count and then gives every family the same weight: 1/7 on RouterBench and 1/6 on SPROUT. Both summaries use the same saved routing decisions; the individual family rows are identical under either aggregation.

**Confidence intervals.** We use the same 2,000 paired bootstrap samples as Appendix A.1, resampling queries within each held-out family and retaining both representations and all penalties together. Each average and paired gain is recomputed in every sample. Brackets report the 2.5th and 97.5th percentiles. These pointwise intervals condition on the evaluated families, splits and fitted routers.

Table 35: RouterBench: kNN-100 utility by held-out family and under both averaging schemes, with 95% intervals.  $\Delta$  is L21 minus GTE.

| Family / aggregation | Queries | GTE                        | L21                        | $\Delta$                     |
|----------------------|---------|----------------------------|----------------------------|------------------------------|
| ARC-Challenge        | 1,470   | 0.9080<br>[0.8979, 0.9177] | 0.9126<br>[0.9015, 0.9232] | +0.0046<br>[-0.0021, 0.0107] |
| GSM8K                | 7,450   | 0.4125<br>[0.4088, 0.4162] | 0.5723<br>[0.5690, 0.5755] | +0.1598<br>[0.1567, 0.1628]  |
| HellaSwag            | 10,042  | 0.7220<br>[0.7155, 0.7288] | 0.7263<br>[0.7194, 0.7330] | +0.0043<br>[0.0006, 0.0081]  |
| MBPP                 | 425     | 0.5316<br>[0.4922, 0.5700] | 0.5730<br>[0.5323, 0.6112] | +0.0414<br>[0.0123, 0.0703]  |
| MMLU                 | 14,037  | 0.7253<br>[0.7192, 0.7316] | 0.7287<br>[0.7223, 0.7355] | +0.0034<br>[0.0002, 0.0066]  |
| MT-Bench             | 80      | 0.3995<br>[0.3170, 0.4805] | 0.5922<br>[0.5212, 0.6589] | +0.1927<br>[0.1155, 0.2836]  |
| WinoGrande           | 1,267   | 0.7151<br>[0.6963, 0.7343] | 0.7462<br>[0.7279, 0.7655] | +0.0311<br>[0.0193, 0.0444]  |
| Equal-query average  | 34,771  | 0.6615<br>[0.6583, 0.6651] | 0.7007<br>[0.6972, 0.7043] | +0.0391<br>[0.0371, 0.0410]  |
| Equal-family average | 34,771  | 0.6306<br>[0.6169, 0.6439] | 0.6930<br>[0.6810, 0.7046] | +0.0625<br>[0.0505, 0.0756]  |

Changing the representation increases mean utility in all seven RouterBench families, with paired 95% confidence intervals excluding zero in six. Most of the query-weighted gain comes from GSM8K, where utility increases from 0.4125 to 0.5723. This substantial improvement occurs in one of the larger families where GTE performs poorly, while gains also extend to the other families. MT-Bench, where GTE performs worst, shows the largest per-family improvement (+0.1927), although its contribution to the query-weighted average is small because it contains only 80 queries.

The particularly large gains on GSM8K and MT-Bench may not be coincidental. In this representation, GSM8K retrieves neighbors mainly from MBPP and MT-Bench instead of MMLU, while MT-Bench retrieves neighbors mainly from GSM8K and MBPP instead of MMLU and HellaSwag. For predicting relative model rewards, MT-Bench is the best-ranked training family for GSM8K, while GSM8K and MBPP rank first and third for MT-Bench, compared with sixth and fifth for MMLU and HellaSwag, respectively. These shifts provide a plausible explanation for the large gains.

Table 36: SPROUT: kNN-100 utility by held-out family and under both averaging schemes, with 95% intervals.  $\Delta$  is L21 minus GTE.

| Family / aggregation | Queries | GTE                        | L21                        | $\Delta$                      |
|----------------------|---------|----------------------------|----------------------------|-------------------------------|
| GPQA                 | 540     | 0.4037<br>[0.3692, 0.4384] | 0.4715<br>[0.4399, 0.5038] | +0.0678<br>[0.0375, 0.0989]   |
| MATH                 | 9,884   | 0.7601<br>[0.7526, 0.7675] | 0.7635<br>[0.7561, 0.7709] | +0.0034<br>[0.0003, 0.0067]   |
| MMLU-Pro             | 11,786  | 0.6698<br>[0.6628, 0.6772] | 0.6696<br>[0.6626, 0.6767] | -0.0002<br>[-0.0041, 0.0035]  |
| MuSR                 | 748     | 0.5500<br>[0.5216, 0.5785] | 0.5149<br>[0.4840, 0.5460] | -0.0351<br>[-0.0550, -0.0153] |
| OpenHermes           | 19,340  | 0.8092<br>[0.8060, 0.8123] | 0.8505<br>[0.8472, 0.8538] | +0.0414<br>[0.0391, 0.0437]   |
| RAGBench             | 1,589   | 0.8539<br>[0.8435, 0.8646] | 0.8831<br>[0.8732, 0.8926] | +0.0292<br>[0.0210, 0.0370]   |
| Equal-query average  | 43,887  | 0.7529<br>[0.7498, 0.7559] | 0.7731<br>[0.7701, 0.7760] | +0.0202<br>[0.0185, 0.0220]   |
| Equal-family average | 43,887  | 0.6744<br>[0.6664, 0.6823] | 0.6922<br>[0.6845, 0.7001] | +0.0177<br>[0.0113, 0.0238]   |

On SPROUT, significantly positive gains are spread across four families, while MMLU-Pro remains approximately unchanged. OpenHermes contributes most to the query-weighted improvement. MuSR is the only family with a significantly negative gain; the loss on this family could be further investigated.

In summary, mean utility increases in 11 of the 13 families. Paired 95% confidence intervals exclude zero for ten gains and one decrease, on MuSR; the differences on ARC-Challenge and MMLU-Pro remain compatible with zero.

## E Layer choice and robustness across depth

**Origin of the fixed layer.** L21 was fixed following exploratory ID difficulty prediction on an earlier RouterBench panel. A ridge probe compared layers  $\{0, 7, 14, 21, 28\}$  with last-token and mean pooling, predicting the number of models solving each query. L21 with last-token pooling gave the largest reduction in squared error beyond a family-mean predictor. This historical choice used RouterBench outcomes. We therefore test robustness across layers.

**Matched routing sweep.** We test whether the improvement depends on choosing L21 by repeating held-out-family routing with every upper-half layer,  $\ell = 15, \dots, 28$ , of Qwen2.5-7B-Instruct. Each representation is the unit-normalized state of the last input token, `hidden_states[ $\ell$ ]`, using the same raw prompts. Queries, candidate models, splits, cost penalties and routing rules are fixed. The L21 results are the same results reported in the main text.

Table 37: Held-out-family mean utility over all thirteen penalties.

| Layer | RouterBench |               |                   | SPROUT  |               |                   |
|-------|-------------|---------------|-------------------|---------|---------------|-------------------|
|       | kNN-100     | Validated kNN | Family Adaptation | kNN-100 | Validated kNN | Family Adaptation |
| GTE   | 0.6615      | 0.6566        | 0.7006            | 0.7529  | 0.7534        | 0.7603            |
| 15    | 0.6933      | 0.6895        | 0.7157            | 0.7671  | 0.7644        | 0.7592            |
| 16    | 0.6865      | 0.6878        | 0.7153            | 0.7667  | 0.7647        | 0.7592            |
| 17    | 0.6890      | 0.6897        | 0.7165            | 0.7665  | 0.7645        | 0.7593            |
| 18    | 0.7013      | 0.6977        | 0.7170            | 0.7673  | 0.7645        | 0.7610            |
| 19    | 0.7038      | 0.7061        | 0.7167            | 0.7641  | 0.7600        | 0.7612            |
| 20    | 0.7032      | 0.7051        | 0.7165            | 0.7719  | 0.7685        | 0.7751            |
| 21    | 0.7007      | 0.7034        | 0.7157            | 0.7731  | 0.7739        | 0.7767            |
| 22    | 0.7011      | 0.7034        | 0.7169            | 0.7754  | 0.7775        | 0.7767            |
| 23    | 0.7037      | 0.7047        | 0.7165            | 0.7758  | 0.7777        | 0.7751            |
| 24    | 0.7024      | 0.7034        | 0.7146            | 0.7762  | 0.7777        | 0.7758            |
| 25    | 0.6959      | 0.6969        | 0.7148            | 0.7773  | 0.7784        | 0.7769            |
| 26    | 0.6973      | 0.6962        | 0.7136            | 0.7765  | 0.7773        | 0.7769            |
| 27    | 0.6963      | 0.6954        | 0.7140            | 0.7745  | 0.7774        | 0.7798            |
| 28    | 0.6901      | 0.6782        | 0.7123            | 0.7636  | 0.7670        | 0.7726            |

Both kNN rules outperform their GTE counterparts at every tested layer on both benchmarks. The result is therefore not specific to L21. Layer choice still affects performance: SPROUT’s Family Adaptation falls slightly below GTE at layers 15–17 and exceeds it at layers 18–28.

As a small additional check, we repeat in-distribution routing at L23 and L25 (Table 38). These two layers were chosen arbitrarily from the tested range, without using performance results to guide the choice. Their mean utilities remain within 0.0006 of L21 for each router and benchmark.

Table 38: ID mean utility over five splits and thirteen penalties.

| Layer | RouterBench |               |                   | SPROUT  |               |                   |
|-------|-------------|---------------|-------------------|---------|---------------|-------------------|
|       | kNN-100     | Validated kNN | Family Adaptation | kNN-100 | Validated kNN | Family Adaptation |
| 21    | 0.7352      | 0.7360        | 0.7217            | 0.8206  | 0.8212        | 0.8096            |
| 23    | 0.7356      | 0.7366        | 0.7218            | 0.8202  | 0.8209        | 0.8097            |
| 25    | 0.7354      | 0.7355        | 0.7219            | 0.8203  | 0.8211        | 0.8096            |

## F Cost, Reward and Answer-Format Controls

**Predictions and evaluation.** We use the same out-of-distribution splits as the main comparison: each of the seven RouterBench and six SPROUT families is held out in turn. All 34,771 RouterBench and 43,887 SPROUT queries are evaluated once. For each test query  $x$  and candidate model  $m$ , kNN-100 predicts reward  $\hat{r}(x, m)$  and cost  $\hat{c}(x, m)$  by averaging that model’s recorded outcomes over the 100 nearest training queries. For each encoder, neighbors are retrieved by cosine similarity. Cost is the total dollar cost used in the routing objective, including both input and output costs where available; we do not measure answer length separately.

**Uncertainty.** All 95% intervals use 2,000 paired bootstrap samples of queries within each held-out family, keeping family sizes fixed and using the same sampled queries across methods and penalties (Appendix A.1). Each bootstrap mean is computed over the full penalty grid before taking interval endpoints. These intervals condition on the evaluated families, frozen training pools and predictors.

### F.1 Prediction errors

Mean squared error (MSE) and mean absolute error (MAE) average equally over queries and candidate models. To examine relative model preferences, we also center predictions and recorded outcomes separately across models within each query before computing the same errors. Centering removes a common increase or decrease shared by all models. The training-mean reference predicts each model’s average reward and cost over all available training queries. Table 39 reports the errors; Table 40 reports  $100(1 - E_{L21}/E_{GTE})$ , where  $E$  is the corresponding error. Positive percentages mean that L21 has lower error. Errors for reward and dollar cost have different units and should not be compared directly.

Table 39: Reward and cost prediction errors, with 95% intervals. Cost MSE and MAE are multiplied by  $10^6$  and  $10^3$ , respectively, for readability.

| Benchmark   | Predictor            | Reward MSE                 | Reward MAE                 | Cost MSE                   | Cost MAE                   |
|-------------|----------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| RouterBench | GTE                  | 0.2061<br>[0.2053, 0.2070] | 0.4077<br>[0.4068, 0.4086] | 2.5186<br>[2.4758, 2.5611] | 0.5757<br>[0.5725, 0.5788] |
| RouterBench | L21                  | 0.1780<br>[0.1770, 0.1791] | 0.3383<br>[0.3370, 0.3395] | 1.0323<br>[1.0077, 1.0580] | 0.3185<br>[0.3160, 0.3208] |
| RouterBench | GTE, matching format | 0.1902<br>[0.1894, 0.1909] | 0.3852<br>[0.3844, 0.3860] | 0.8648<br>[0.8402, 0.8895] | 0.3406<br>[0.3382, 0.3428] |
| RouterBench | Training-model means | 0.1899<br>[0.1893, 0.1905] | 0.3943<br>[0.3937, 0.3949] | 2.8637<br>[2.8255, 2.9016] | 0.7833<br>[0.7806, 0.7863] |
| SPROUT      | GTE                  | 0.2073<br>[0.2064, 0.2082] | 0.4031<br>[0.4020, 0.4042] | 4.8932<br>[4.5794, 5.2115] | 0.7142<br>[0.7086, 0.7199] |
| SPROUT      | L21                  | 0.1614<br>[0.1604, 0.1624] | 0.3221<br>[0.3209, 0.3234] | 5.0209<br>[4.7157, 5.3337] | 0.7436<br>[0.7377, 0.7493] |
| SPROUT      | GTE, matching format | 0.1773<br>[0.1763, 0.1783] | 0.3478<br>[0.3467, 0.3489] | 5.2243<br>[4.9110, 5.5502] | 0.7163<br>[0.7106, 0.7222] |
| SPROUT      | Training-model means | 0.1921<br>[0.1912, 0.1930] | 0.3787<br>[0.3778, 0.3798] | 5.6598<br>[5.3400, 5.9843] | 0.7527<br>[0.7467, 0.7586] |

Table 40: L21 error reduction relative to GTE (%), with paired 95% intervals. Negative values indicate higher error.

| Target | Error | Centering     | RouterBench                | SPROUT                     |
|--------|-------|---------------|----------------------------|----------------------------|
| Reward | MSE   | None          | +13.64<br>[+13.23, +14.05] | +22.15<br>[+21.67, +22.64] |
| Reward | MAE   | None          | +17.03<br>[+16.79, +17.27] | +20.07<br>[+19.74, +20.41] |
| Reward | MSE   | Across models | +5.84<br>[+5.58, +6.10]    | +2.43<br>[+2.26, +2.59]    |
| Reward | MAE   | Across models | +6.25<br>[+6.07, +6.44]    | +4.01<br>[+3.89, +4.14]    |
| Cost   | MSE   | None          | +59.01<br>[+58.11, +59.95] | -2.61<br>[-3.21, -2.03]    |
| Cost   | MAE   | None          | +44.67<br>[+44.19, +45.21] | -4.11<br>[-4.42, -3.78]    |
| Cost   | MSE   | Across models | +53.16<br>[+52.15, +54.21] | -3.21<br>[-3.83, -2.63]    |
| Cost   | MAE   | Across models | +41.35<br>[+40.80, +41.94] | -3.60<br>[-3.92, -3.28]    |

## F.2 Answer-format control

For each test query, we restrict the training pool to queries with the same coarse answer-format label, then retrieve its 100 nearest GTE neighbors. The two labels, multiple-choice and free-form, are defined in Appendix B.5. The query’s entire family remains excluded, and every restricted training pool contains at least 100 queries. We use the known task-format label, without generating or classifying any answers. This control changes both format agreement and the available source-family composition; it does not isolate answer format as the sole cause.

The answer-format restriction raises GTE utility to 0.6974 on RouterBench and 0.7584 on SPROUT, closing approximately 92% and 27% of the observed GTE–L21 gaps. Both restricted-GTE means remain below L21, with paired difference intervals excluding zero. These results support a substantial role for the format restriction on RouterBench and additional useful reward information in L21 on SPROUT.

Tables 41 and 42 report these means and paired differences; Tables 43 and 44 give the results at every penalty.

## F.3 Combining reward and cost estimates

We evaluate all four combinations of GTE and L21 reward and cost predictions. At each penalty  $\lambda$ , every combination selects  $\arg \max_m [\hat{r}(x, m) - \lambda \hat{c}(x, m)]$ . The GTE answer-format control uses both estimates from its restricted neighborhood. We average recorded routing utility equally over test queries and then equally over the thirteen penalties. Tables 41 and 42 give means and paired differences; Tables 43 and 44 retain all penalties.

Table 41: Mean out-of-distribution utility over thirteen penalties, with 95% intervals.

| Reward estimate                 | Cost estimate | RouterBench                | SPROUT                     |
|---------------------------------|---------------|----------------------------|----------------------------|
| GTE                             | GTE           | 0.6615<br>[0.6583, 0.6651] | 0.7529<br>[0.7498, 0.7559] |
| L21                             | GTE           | 0.6754<br>[0.6720, 0.6791] | 0.7709<br>[0.7679, 0.7739] |
| GTE                             | L21           | 0.6965<br>[0.6934, 0.7000] | 0.7560<br>[0.7529, 0.7590] |
| L21                             | L21           | 0.7007<br>[0.6972, 0.7043] | 0.7731<br>[0.7701, 0.7760] |
| GTE with matching answer format |               | 0.6974<br>[0.6942, 0.7010] | 0.7584<br>[0.7553, 0.7614] |
| Training-model means            |               | 0.6618<br>[0.6586, 0.6652] | 0.7575<br>[0.7544, 0.7604] |

Table 42: Mean utility differences, with paired 95% intervals. GTE and L21 alone denote ordinary kNN-100.

| Comparison                           | RouterBench                   | SPROUT                        |
|--------------------------------------|-------------------------------|-------------------------------|
| Replace GTE reward estimate with L21 | +0.0139<br>[+0.0120, +0.0156] | +0.0180<br>[+0.0163, +0.0198] |
| Replace GTE cost estimate with L21   | +0.0350<br>[+0.0342, +0.0357] | +0.0031<br>[+0.0027, +0.0036] |
| Replace both GTE estimates with L21  | +0.0391<br>[+0.0371, +0.0410] | +0.0202<br>[+0.0185, +0.0220] |
| GTE with matching format minus GTE   | +0.0359<br>[+0.0350, +0.0368] | +0.0055<br>[+0.0042, +0.0069] |
| GTE with matching format minus L21   | -0.0032<br>[-0.0050, -0.0015] | -0.0147<br>[-0.0163, -0.0131] |

**Results.** On RouterBench, replacing only GTE’s cost predictions raises mean utility from 0.6615 to 0.6965, whereas replacing only reward predictions gives 0.6754. On SPROUT, replacing reward predictions raises utility from 0.7529 to 0.7709, whereas replacing only cost predictions gives 0.7560. Thus, the contributions differ across benchmarks and are not additive. SPROUT’s L21 cost estimates have slightly higher MSE and MAE, yet still produce a small routing gain, illustrating that aggregate prediction error does not determine routing utility.

Table 43: RouterBench: utility at every penalty, with 95% intervals. The first four columns identify the reward/cost predictors; the last uses GTE neighbors matching the query answer format.

| $\lambda$ | GTE/GTE                    | L21/GTE                    | GTE/L21                    | L21/L21                    | GTE, matching format       |
|-----------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| 0         | 0.7900<br>[0.7866, 0.7937] | 0.7854<br>[0.7818, 0.7891] | 0.7900<br>[0.7866, 0.7937] | 0.7854<br>[0.7818, 0.7891] | 0.7902<br>[0.7868, 0.7939] |
| 1         | 0.7864<br>[0.7831, 0.7902] | 0.7804<br>[0.7768, 0.7842] | 0.7865<br>[0.7832, 0.7903] | 0.7805<br>[0.7769, 0.7843] | 0.7867<br>[0.7834, 0.7905] |
| 2         | 0.7831<br>[0.7798, 0.7869] | 0.7774<br>[0.7738, 0.7812] | 0.7832<br>[0.7799, 0.7870] | 0.7771<br>[0.7734, 0.7809] | 0.7832<br>[0.7799, 0.7870] |
| 5         | 0.7732<br>[0.7699, 0.7770] | 0.7676<br>[0.7640, 0.7713] | 0.7732<br>[0.7698, 0.7770] | 0.7678<br>[0.7642, 0.7716] | 0.7726<br>[0.7690, 0.7764] |
| 10        | 0.7563<br>[0.7529, 0.7601] | 0.7498<br>[0.7462, 0.7536] | 0.7557<br>[0.7524, 0.7595] | 0.7587<br>[0.7549, 0.7626] | 0.7638<br>[0.7601, 0.7676] |
| 20        | 0.7225<br>[0.7192, 0.7263] | 0.7180<br>[0.7144, 0.7219] | 0.7392<br>[0.7358, 0.7430] | 0.7439<br>[0.7401, 0.7478] | 0.7491<br>[0.7456, 0.7529] |
| 30        | 0.6881<br>[0.6846, 0.6919] | 0.6859<br>[0.6823, 0.6900] | 0.7247<br>[0.7213, 0.7286] | 0.7279<br>[0.7240, 0.7319] | 0.7341<br>[0.7306, 0.7380] |
| 50        | 0.6225<br>[0.6188, 0.6264] | 0.6226<br>[0.6187, 0.6269] | 0.6948<br>[0.6912, 0.6987] | 0.6994<br>[0.6954, 0.7037] | 0.7029<br>[0.6991, 0.7069] |
| 75        | 0.5491<br>[0.5453, 0.5535] | 0.5619<br>[0.5575, 0.5662] | 0.6574<br>[0.6537, 0.6616] | 0.6691<br>[0.6648, 0.6735] | 0.6638<br>[0.6600, 0.6680] |
| 100       | 0.5057<br>[0.5013, 0.5102] | 0.5945<br>[0.5900, 0.5994] | 0.6253<br>[0.6215, 0.6296] | 0.6462<br>[0.6418, 0.6508] | 0.6334<br>[0.6296, 0.6377] |
| 150       | 0.5320<br>[0.5270, 0.5370] | 0.6150<br>[0.6104, 0.6194] | 0.5967<br>[0.5922, 0.6013] | 0.6189<br>[0.6143, 0.6235] | 0.5937<br>[0.5897, 0.5981] |
| 250       | 0.5560<br>[0.5513, 0.5605] | 0.5859<br>[0.5813, 0.5905] | 0.5843<br>[0.5797, 0.5891] | 0.5899<br>[0.5852, 0.5946] | 0.5592<br>[0.5546, 0.5638] |
| 500       | 0.5353<br>[0.5307, 0.5398] | 0.5362<br>[0.5316, 0.5408] | 0.5434<br>[0.5388, 0.5481] | 0.5441<br>[0.5394, 0.5487] | 0.5341<br>[0.5296, 0.5386] |
| Mean      | 0.6615<br>[0.6583, 0.6651] | 0.6754<br>[0.6720, 0.6791] | 0.6965<br>[0.6934, 0.7000] | 0.7007<br>[0.6972, 0.7043] | 0.6974<br>[0.6942, 0.7010] |

Table 44: SPROUT: utility at every penalty, with 95% intervals. The first four columns identify the reward/cost predictors; the last uses GTE neighbors matching the query answer format.

| $\lambda$ | GTE/GTE                    | L21/GTE                    | GTE/L21                    | L21/L21                    | GTE, matching format       |
|-----------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| 0         | 0.8035<br>[0.8002, 0.8067] | 0.8243<br>[0.8212, 0.8275] | 0.8035<br>[0.8002, 0.8067] | 0.8243<br>[0.8212, 0.8275] | 0.7972<br>[0.7938, 0.8005] |
| 0.5       | 0.8013<br>[0.7980, 0.8046] | 0.8224<br>[0.8193, 0.8256] | 0.8018<br>[0.7986, 0.8051] | 0.8225<br>[0.8192, 0.8257] | 0.7956<br>[0.7923, 0.7990] |
| 1         | 0.7994<br>[0.7962, 0.8026] | 0.8204<br>[0.8173, 0.8235] | 0.7997<br>[0.7964, 0.8030] | 0.8200<br>[0.8169, 0.8232] | 0.7951<br>[0.7917, 0.7984] |
| 2         | 0.7955<br>[0.7922, 0.7988] | 0.8165<br>[0.8132, 0.8196] | 0.7962<br>[0.7929, 0.7995] | 0.8170<br>[0.8137, 0.8202] | 0.7940<br>[0.7906, 0.7973] |
| 5         | 0.7873<br>[0.7839, 0.7906] | 0.8075<br>[0.8041, 0.8108] | 0.7877<br>[0.7844, 0.7910] | 0.8073<br>[0.8040, 0.8106] | 0.7907<br>[0.7874, 0.7940] |
| 10        | 0.7820<br>[0.7787, 0.7855] | 0.8004<br>[0.7970, 0.8037] | 0.7820<br>[0.7786, 0.7853] | 0.8009<br>[0.7974, 0.8042] | 0.7878<br>[0.7844, 0.7911] |
| 25        | 0.7734<br>[0.7698, 0.7767] | 0.7926<br>[0.7892, 0.7960] | 0.7742<br>[0.7708, 0.7775] | 0.7946<br>[0.7911, 0.7979] | 0.7816<br>[0.7782, 0.7849] |
| 50        | 0.7607<br>[0.7573, 0.7641] | 0.7795<br>[0.7761, 0.7829] | 0.7631<br>[0.7597, 0.7665] | 0.7830<br>[0.7796, 0.7864] | 0.7714<br>[0.7679, 0.7747] |
| 75        | 0.7496<br>[0.7462, 0.7530] | 0.7671<br>[0.7637, 0.7706] | 0.7535<br>[0.7500, 0.7568] | 0.7716<br>[0.7682, 0.7751] | 0.7606<br>[0.7573, 0.7641] |
| 100       | 0.7379<br>[0.7345, 0.7413] | 0.7555<br>[0.7520, 0.7589] | 0.7431<br>[0.7396, 0.7465] | 0.7606<br>[0.7571, 0.7641] | 0.7497<br>[0.7464, 0.7531] |
| 150       | 0.7149<br>[0.7114, 0.7184] | 0.7318<br>[0.7282, 0.7353] | 0.7233<br>[0.7198, 0.7268] | 0.7384<br>[0.7348, 0.7419] | 0.7283<br>[0.7249, 0.7318] |
| 250       | 0.6786<br>[0.6748, 0.6824] | 0.6924<br>[0.6886, 0.6963] | 0.6929<br>[0.6891, 0.6966] | 0.6986<br>[0.6947, 0.7023] | 0.6904<br>[0.6865, 0.6942] |
| 500       | 0.6037<br>[0.5996, 0.6080] | 0.6115<br>[0.6071, 0.6157] | 0.6074<br>[0.6033, 0.6115] | 0.6119<br>[0.6079, 0.6160] | 0.6165<br>[0.6123, 0.6206] |
| Mean      | 0.7529<br>[0.7498, 0.7559] | 0.7709<br>[0.7679, 0.7739] | 0.7560<br>[0.7529, 0.7590] | 0.7731<br>[0.7701, 0.7760] | 0.7584<br>[0.7553, 0.7614] |