# HM-ROUTER: JOINT MODEL AND HARNESS ROUTING FOR AGENTIC SYSTEMS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Agent performance depends on both the underlying model and the harness that manages its tool use and execution. Selecting a suitable pair requires accounting for their compatibility, yet training samples may cover only a subset of the growing combination space. We introduce HM-Router, a routing method that jointly selects a model and harness for each query. It learns separate model and harness representations shared across routes, with an interaction term inspired by canonical polyadic (CP) tensor decomposition to capture how their compatibility varies with the query. This sharing allows training samples from observed pairs to inform predictions for unobserved combinations. We curate HM-ROUTER-BENCH from 12 public agent benchmarks, covering 293 routes, 73 models, and 25 harnesses. HM-Router exceeds the strongest evaluated learned baseline by 7.3 percentage points in mean routing accuracy and leads at all seven evaluated cost budgets on the six-benchmark HM-ROUTE-COST subset. When 90% of routes have their training outcomes withheld, allowing unobserved combinations improves normalized accuracy by 15.8 points over restricting the same router to observed routes. HM-Router has also demonstrated training sample efficiency for new routes and components and generalization to unseen benchmarks.

## 1 INTRODUCTION

LLM-based agents are becoming increasingly capable and are being adopted across a broad range of complex tasks (Yao et al., 2022; Merrill et al., 2026; Xie et al., 2024; Sun et al., 2026). Deploying LLMs involves choosing among models with different capabilities and costs, motivating routing methods that select a suitable model for each query (Ong et al., 2025). In agentic settings, however, the model is only one component of the deployed system. Another design axis is the *harness*: the surrounding scaffold that governs how a model interacts with tools, observations, and the environment (Yang et al., 2024). Manually designed and automatically optimized harnesses create many ways to deploy the same model (Yang et al., 2024; Zhang et al., 2025a; Lee et al., 2026), expanding the routing decision to include both components. This motivates *model–harness co-routing*: jointly selecting a model and harness for each query based on compatibility, performance, and cost.

Model–harness co-routing introduces two challenges. **Challenge 1: Model–harness compatibility.** Existing routers primarily select models (Zhuang et al., 2025; Song et al., 2025; Feng et al., 2025), but a model's performance also depends on the harness with which it is paired. The highest-scoring model changes across harnesses (Figure 1a). Even when we select the best model for each query under one harness, those choices can become suboptimal under another, showing that model selection must account for the harness (Figure 1b). Joint routing must therefore capture model–harness compatibility to identify suitable pairs for each query. **Challenge 2: Generalization to unobserved combinations.** The number of candidate routes grows as the product of the model and harness counts, making execution coverage increasingly difficult as either set expands. Treating each pair as an independent routing option ties learning to that pair's own execution history, even when its components have been evaluated independently in other combinations. This creates a coverage bottleneck: potentially strong routes may be missing from direct evidence, and excluding them can sacrifice routing accuracy (Figure 1d). The challenge is to transfer evidence across shared components to assess and select unobserved combinations. Together, these challenges raise the question:

*How can we effectively route queries across the joint space of models and harnesses?*



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

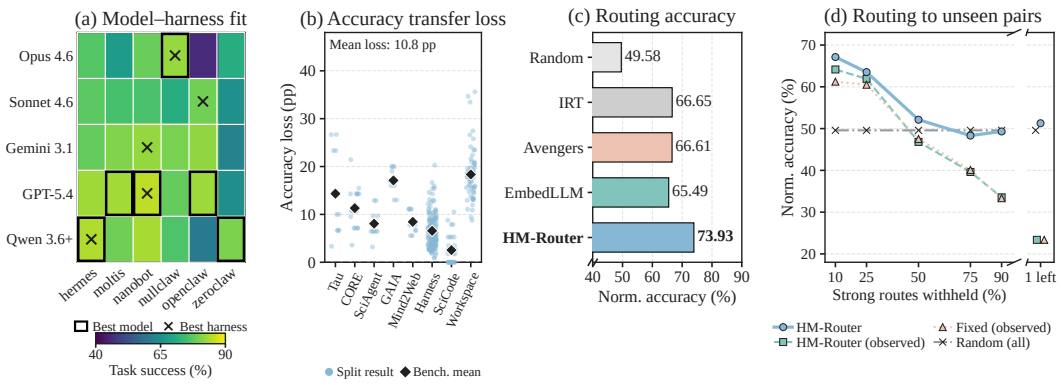



Figure 1: **Joint routing: motivation and benefits.** (a) HarnessBench accuracy: boxes mark the best model per harness, crosses the best harness per model. (b) Accuracy loss when transferring oracle model choices across harnesses; dots show split results, diamonds benchmark means. (c) Mean routing accuracy across 12 benchmarks (Table 1). (d) Routing with strong routes' training histories withheld (Table 2).

To study this question, we curate HM-ROUTER-BENCH from 12 existing public agent benchmarks, comprising 293 routes, 73 models, and 25 harnesses (Appendix B.1). Using this testbed, we introduce HM-Router, a routing method that learns separate representations for models and harnesses and composes them to score each candidate route. Shared component factors allow observations of one route to inform other routes containing the same model or harness. An explicit interaction term, inspired by canonical polyadic (CP) tensor decomposition (Kolda & Bader, 2009), allows harness preferences to depend on both the model and the query.

HM-Router exceeds the strongest evaluated learned router by 7.3 percentage points in mean accuracy and leads the cost comparison at every evaluated budget. Withholding route outcomes clarifies the practical value of shared representations: the scorer can retain unobserved combinations as candidates, increasingly preserving performance as strong observed routes are removed.

Our contributions are threefold:

1. We formulate query-level routing over the joint model–harness space and curate HM-ROUTER-BENCH, a controlled evaluation suite assembled from 12 existing public agent benchmarks, including a six-benchmark subset for cost analysis.
2. We introduce a routing method that learns separate model and harness representations to share information across routes, with an explicit interaction term capturing how their compatibility varies with the query.
3. We demonstrate that HM-Router supports both effective route selection and performance prediction, using shared component evidence to extend these capabilities to model–harness combinations with limited or no execution history.

## 2 Background and Related Work

**Agent harnesses and their optimization.** LLM agents interleave reasoning, tool use, and interaction with external environments (Yao et al., 2022). A *harness* orchestrates this process by managing prompts, observations, memory, and execution. Coding harnesses include Claude Code (Anthropic, 2025), Codex CLI (OpenAI, 2025), and Gemini CLI (Google, 2025). Research examples include SWE-agent's interface for navigating, editing, and testing code (Yang et al., 2024) and Voyager's curriculum, skill library, and feedback loop for refining executable programs (Wang et al., 2023). Recent methods automate harness evolution through approaches such as synthesizing code from environment feedback with AutoHarness (Lou et al., 2026), revising harness code from execution feedback with Meta-Harness (Lee et al., 2026), and guiding component edits with structured execution evidence in Agentic Harness Engineering (Lin et al., 2026). These methods expand the space of harnesses, motivating joint selection of a model and harness for each query.

**Model routing.** Model routing balances quality and cost through query-level model selection, as in RouteLLM (Ong et al., 2025), or cascades, as in FrugalGPT (Chen et al., 2023); RouterBench provides a common evaluation framework (Hu et al., 2024). Most closely related, EmbedLLM learns compact model representations for routing and performance prediction (Zhuang et al., 2025), while IRT-Router models latent model abilities and query difficulty (Song et al., 2025). GraphRouter captures relationships among tasks, queries, and models, and supports newly introduced models (Feng et al., 2025). These approaches learn variation across models without considering harnesses.

**Agentic routing.** Recent work incorporates execution evidence into model selection: Agent-as-a-Router accumulates verified execution experience for coding tasks (Zhou et al., 2026), SWE-Router uses partial trajectories to decide whether to escalate to a stronger model (Son et al., 2026), and OpenSquilla selects individual models or complementary ensembles conditioned on the current harness state (Liu et al., 2026a). Along the harness axis, Adaptive Auto-Harness evolves a tree of harnesses and selects a branch for each incoming task while keeping the solver model fixed (Liu et al., 2026b). These works select models or adapt harnesses; HM-Router jointly selects the model and harness for each query, learning their compatibility and reusing component evidence to score both observed and unobserved combinations.

## 3 Problem Formulation

Let $\mathcal{M}$ be a set of models and $\mathcal{H}$ a set of harnesses. A route $r = (m, h)$ pairs a model with the harness in which it runs. The candidate pool $\mathcal{R} \subseteq \mathcal{M} \times \mathcal{H}$ contains pairs that can be executed in the deployment setting. This pool need not include every combination: a harness may require interfaces or capabilities that some models do not support, or may integrate with only certain model providers. Feasibility is distinct from observation since a pair can be executable without having been evaluated.

The router learns from execution outcomes $y_{q,r}$ for training queries $q \in \mathcal{Q}_{\text{train}}$. Available observations are indexed by $\Omega \subseteq \mathcal{Q}_{\text{train}} \times \mathcal{R}$. Let $\mathcal{O} \subseteq \mathcal{R}$ contain routes with at least one observation. For a new query, the router selects

$$\hat{r}(q) = \arg\max_{r \in \mathcal{R}} S(q, r), \tag{1}$$

where $S$ estimates the suitability of a route for the query. Selection may include unobserved pairs in $\mathcal{R} \setminus \mathcal{O}$, so the candidate pool can extend beyond the available execution history.

This setting poses two challenges. **Model–harness compatibility:** a harness's suitability can vary across models, so it must be assessed together with the model it supports. **Incomplete observations:** the potential combination space grows as $|\mathcal{M}| \times |\mathcal{H}|$, while execution histories may cover only a subset. The router must therefore learn from observed pairs and use their shared components to predict unobserved combinations.

## 4 HM-Router: Compositional Model–Harness Routing

HM-Router first shares evidence across routes through separate model and harness representations. It then models their compatibility with an interaction term built from the same factors. Figure 2 summarizes the resulting routing pipeline.

### 4.1 Factorized route representations

The potential route space grows as $|\mathcal{M}| \times |\mathcal{H}|$, while execution histories may cover only a subset of combinations. We therefore learn separate model and harness representations, sharing each across all routes containing that component.

As illustrated in Figure 2a, each component is represented using its identity and behavioral evidence pooled across observed pairings; harnesses additionally use a textual description. Specifically,

$$\begin{aligned} \tilde{\mathbf{q}} &= W_q \mathbf{x}_q, \\ \mathbf{u}_m &= W_m[\mathbf{e}_m; P_m \mathbf{z}_m], \\ \mathbf{u}_h &= W_h[\mathbf{e}_h; P_h \mathbf{z}_h; T_h \mathbf{t}_h], \end{aligned} \tag{2}$$

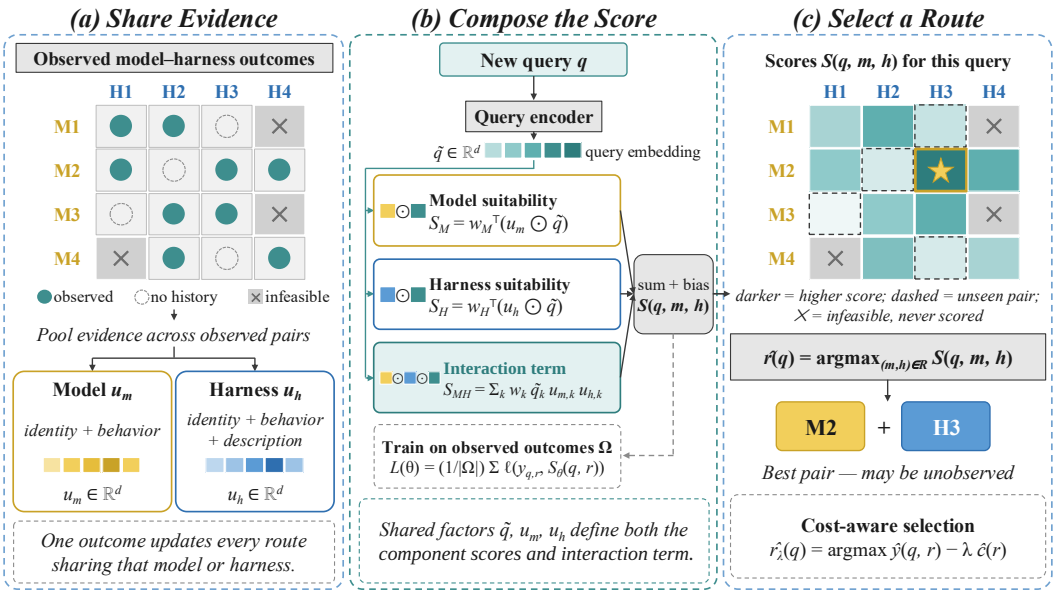



Figure 2: HM-Router overview. (a) Observed outcomes inform shared model and harness representations. (b) Component and interaction scores form a query-dependent route score. (c) The router selects the highest-scoring feasible pair, including unobserved combinations, optionally accounting for execution cost.

where $\mathbf{x}_q$ and $\mathbf{t}_h$ are frozen query and harness-description embeddings, $\mathbf{e}_m$ and $\mathbf{e}_h$ are learned identity embeddings, and $\mathbf{z}_m$ and $\mathbf{z}_h$ summarize each component's observed success vector over training queries. Harness descriptions are shared across benchmarks. The learned affine maps produce $\tilde{\mathbf{q}}, \mathbf{u}_m, \mathbf{u}_h \in \mathbb{R}^d$.

These representations define how well each component matches the query. Before feature normalization, the component scores are

$$S_M(q, m) = \mathbf{w}_M^\top(\mathbf{u}_m \odot \tilde{\mathbf{q}}), \qquad S_H(q, h) = \mathbf{w}_H^\top(\mathbf{u}_h \odot \tilde{\mathbf{q}}), \tag{3}$$

where $\odot$ denotes elementwise multiplication. Their sum gives

$$S_{\text{comp}}(q, m, h) = S_M(q, m) + S_H(q, h). \tag{4}$$

The encoders, identity embeddings, and scoring weights are learned jointly by predicting observed route outcomes (Section 4.4). Each observation thus informs representations shared across routes, allowing an unobserved pair to be scored from evidence about its components.

### 4.2 Model–Harness Compatibility

The additive component score cannot express a preference between harnesses that changes with the model for the same query. To capture this dependence, we extend Eq. (4) with an interaction term $S_{MH}$, omitting a shared bias for clarity:

$$S(q, m, h) = S_{\text{comp}}(q, m, h) + S_{MH}(q, m, h). \tag{5}$$

The interaction term captures how a particular model and harness work together beyond their individual suitability. Section 5.8 evaluates its contribution to routing accuracy on observed routes.

For pairs without training observations, $(m, h) \in \mathcal{R} \setminus \mathcal{O}$, we retain both the component and interaction terms. As described next, both use shared representations learned from the available observations, so scoring an unobserved combination requires no pair-specific parameters.

### 4.3 Shared Factors for the Interaction Term

We construct the interaction term $S_{MH}$ from the query, model, and harness representations learned for the component score, preserving the sharing introduced in Section 4.1. Inspired by canonical

polyadic (CP) tensor decomposition (Kolda & Bader, 2009), we extend their pairwise products to a three-way interaction term:

$$S_{MH}(q, m, h) = \sum_{k=1}^{d} w_k \tilde{q}_k u_{m,k} u_{h,k}. \tag{6}$$

Each coordinate contributes a model–harness compatibility pattern weighted by the query. Sharing factors lets the component scores and interaction term jointly shape the same representations. Before feature normalization, the resulting interaction score tensor has CP rank at most $d$; Appendix B.2 gives the implemented, normalized scorer.

### 4.4 Learning and Selection

We optimize all learned parameters $\theta$ through the full route score:

$$\mathcal{L}(\theta) = \frac{1}{|\Omega|} \sum_{(q,r)\in\Omega} \ell\big(y_{q,r}, S_\theta(q, r)\big), \tag{7}$$

where $\ell$ is the binary cross-entropy loss specified in Appendix B.2. For a new component, probe executions supply behavioral evidence for representing its candidate routes.

At inference, we select the highest-scoring feasible route using Eq. (1). When cost matters, we balance the predicted success probability $\hat{y}(q, r) = \sigma(S(q, r))$ against an estimated execution cost $\hat{c}(r)$:

$$\hat{r}_\lambda(q) = \arg\max_{r\in\mathcal{R}} \{\hat{y}(q, r) - \lambda \hat{c}(r)\}, \qquad \lambda \geq 0. \tag{8}$$

Here $\hat{c}(r)$ is the mean cost of the observed route based on the training dataset, and the same route score serves routing with and without a cost penalty.

## 5 Experimental Evaluation

We organize the main experiments around three research questions:

**RQ1.** Does HM-Router achieve higher routing accuracy than existing methods?

**RQ2.** Does HM-Router improve the cost–accuracy tradeoff in model–harness routing?

**RQ3.** Can our factored formulation generalize to unseen model–harness combinations?

### 5.1 Experimental Setup

**Benchmarks and protocol.** We evaluate on 12 benchmarks from HM-ROUTER-BENCH, covering 293 routes, 73 models, and 25 harnesses. All methods receive the same frozen Qwen3-Embedding-8B query features and observed training outcomes, using five fixed 70/30 query splits without per-benchmark tuning. We report test accuracy divided by the split's full-pool oracle accuracy, then average over splits and benchmarks; tables and figures express this ratio as a percentage. Appendix B.1 details benchmark coverage, route construction, and the exact model–harness pairs.

**Baselines.** We use Random, IrtNet (**IRT**) (Song et al., 2025), Avengers (Zhang et al., 2025c), and EmbedLLM with atomic route identities (Zhuang et al., 2025). Avengers-Pro (**Av-Pro**) (Zhang et al., 2025b) is used in the cost evaluation.

### 5.2 Routing Performance and Cost–Accuracy Tradeoffs

**Routing quality.** HM-Router improves routing accuracy over the strongest learned baseline by 7.3 percentage points and EmbedLLM by 8.4 points (Table 1). We compare full methods.

Table 1: Routing accuracy (%) over five splits. Values are oracle-normalized. **Bold** and underlined entries mark the best and second-best method in each row.

| **Benchmark** | Random | IRT | Avengers | EmbedLLM | HM-Router |
|---|---|---|---|---|---|
| TerminalBench 2.1 (Merrill et al., 2026) | 77.29 | 82.13 | 83.32 | 85.71 | **86.89** |
| TauBench Airline (Yao et al., 2024) | 35.66 | 42.74 | 51.52 | 52.59 | **57.52** |
| CORE-Bench Hard (Siegel et al., 2024) | 26.84 | 62.73 | 66.93 | 85.96 | **91.33** |
| ScienceAgentBench (Chen et al., 2025) | 30.28 | 35.60 | 41.56 | 36.60 | **56.47** |
| GAIA (Mialon et al., 2024) | 42.29 | 68.85 | 69.99 | 64.16 | **73.87** |
| Online Mind2Web (Deng et al., 2023) | 39.71 | 45.75 | 46.95 | 47.43 | **51.26** |
| SWE-bench Verified (Jimenez et al., 2024) | 61.65 | 86.27 | 86.12 | 88.06 | **88.22** |
| HarnessBench (Yao et al., 2026) | 79.71 | 87.83 | 82.95 | 86.44 | **89.33** |
| CORE-Bench 1.1 (Siegel et al., 2024) | 86.60 | **95.00** | 88.33 | 93.61 | 93.33 |
| SciCode (Tian et al., 2024) | 23.64 | **78.64** | 59.62 | 26.97 | 65.03 |
| SkillsBench (Li et al., 2026) | 40.10 | 52.29 | 56.00 | 54.12 | **66.23** |
| WorkspaceBench-Lite (Tang et al., 2026) | 51.24 | 61.94 | 66.01 | 64.21 | **67.69** |
| **Mean** | 49.58 | 66.65 | 66.61 | 65.49 | **73.93** |

**Cost–accuracy tradeoff.** To evaluate cost, we construct HM-ROUTE-COST, a subset of six benchmarks with execution-cost data. HM-Router maintains its accuracy advantage under spending limits, as it leads at all seven evaluated dollar budgets, exceeding Av-Pro by 8.5 percentage points at $2 per task (Figure 3).

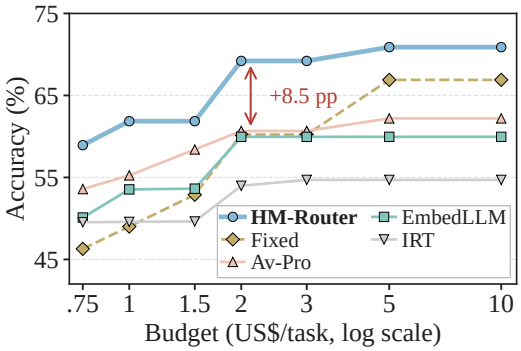



Figure 3: **Cost–accuracy tradeoff.** Mean normalized accuracy by cost budget on six benchmarks.

### 5.3 ROUTING WITH UNOBSERVED MODEL–HARNESS COMBINATIONS

**Withholding strong routes.** Execution histories may cover only some feasible model–harness combinations. We progressively remove train-best routes' outcomes, retaining the full candidate pool at inference. Component summaries and scorer training use only the retained outcomes. For components with no remaining observations, the cold-start policy sets their identity embeddings and behavioral summaries to zero while retaining available harness descriptions. This tests a progressively degraded observed portfolio.

**Access to hidden routes preserves routing options.** Allowing the same scorer to select hidden combinations improves over restricting it to observed routes: the gain is 1.6 points at 25% removal, 15.8 points at 90%, and 27.9 points with one observed route left (Table 2). At 90% removal, HM-Router also exceeds Oracle (observed). Component-based scoring can therefore recover useful options that no policy confined to the retained portfolio can select.

**Learned scoring improves selection among unobserved combinations.** HM-Router outperforms Comp. Avg. (all) at most removal levels, with gains of 3.3 percentage points at 25% removal and 11.9 points at 90%. These gains show the value of learned route scores beyond component success averages, particularly when many strong routes lack execution histories. Performance is comparable at 50% removal.

### 5.4 ROUTING WITH LIMITED SAMPLES

HM-Router achieves effective routing with small training datasets for new routes and components. To evaluate training sample efficiency, we remove all training outcomes for a route, model, or harness, then reveal its outcomes on sets of $k \in \{1, 5, 10, 15, 20\}$ training queries. Here, $k$ counts only

Table 2: **Routing with unobserved combinations.** Accuracy (%) at five target removal levels and with one surviving route. HM-Router (observed) selects only retained routes; Fixed (observed) selects their train-best route. Comp. Avg. (all) averages model and harness success rates over retained training outcomes, allowing hidden routes. Rankings exclude Oracle (observed).

| Removed | HM-Router | Comp. Avg. (all) | HM-Router (observed) | Fixed (observed) | Oracle (observed) |
|---|---|---|---|---|---|
| 10% | **67.13** | 63.52 | 64.15 | 61.24 | 98.30 |
| 25% | **63.52** | 60.22 | 61.92 | 60.50 | 94.22 |
| 50% | 52.12 | **52.89** | 46.82 | 47.53 | 80.29 |
| 75% | **48.32** | 40.30 | 39.62 | 40.10 | 63.48 |
| 90% | **49.31** | 37.43 | 33.54 | 33.33 | 42.22 |
| One route left | **51.28** | 47.54 | 23.37 | 23.37 | 23.37 |

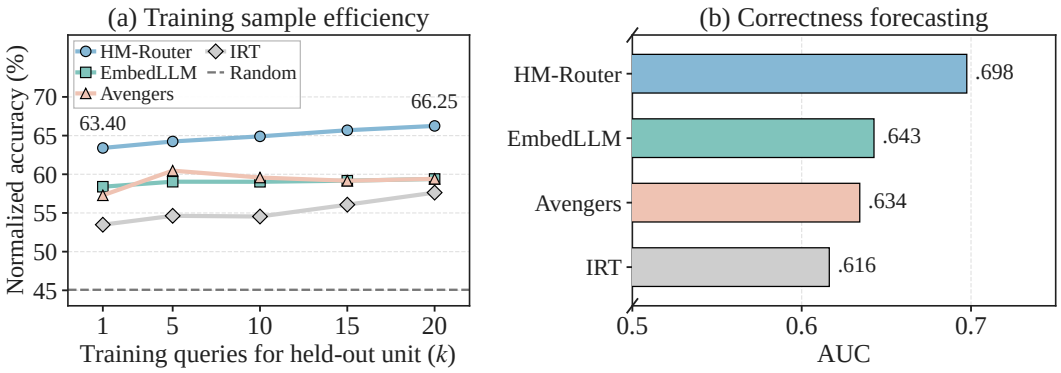



Figure 4: **Sample efficiency and correctness forecasting.** (a) Oracle-normalized routing accuracy versus the number of training queries $k$ observed for a held-out route, model, or harness. (b) AUC for forecasting whether each route execution succeeds.

queries observed for the held-out unit; training outcomes for other routes remain available. We aggregate results over held-out units whose removal reduces oracle accuracy by more than one percentage point. With a training dataset of just one query for the held-out unit, HM-Router exceeds EmbedLLM by 5.0 percentage points; at twenty queries, the gain increases to 6.9 points (Figure 4a). This sample efficiency may reflect shared model and harness representations across training samples: each outcome informs components reused across routes, allowing a small dataset for a new route or component to complement evidence from other pairings.

### 5.5 Routing Generalization to Held-Out Benchmarks

We evaluate routing on TauBench Airline, CORE-Bench Hard, and SciCode using only training samples from other benchmarks. Baselines learn a separate representation for each model–harness pair, whereas HM-Router learns model and harness representations separately and can combine them to score previously unseen pairs.

HM-Router achieves the highest accuracy on all three benchmarks (Figure 5), averaging 49.4% versus 32.9% for IRT, the strongest baseline overall.

### 5.6 Correctness Forecasting

HM-Router also forecasts route success more accurately, extending its usefulness beyond selecting a single route. It exceeds EmbedLLM by 0.055 AUC (Figure 4b), indicating better discrimination between successful and unsuccessful executions.

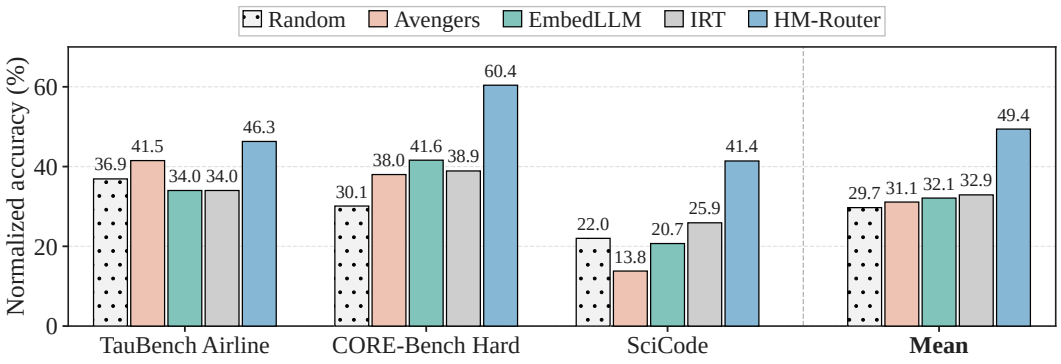



Figure 5: **Routing on held-out benchmarks.** Oracle-normalized accuracy (%) is reported. Random is the uniform routing expectation.

### 5.7 What do the learned representations capture?

We examine whether the learned representations capture similarities between harnesses and differences in how models and harnesses work together. The first analysis uses harness vectors $\mathbf{u}_h$ learned across benchmarks; the second examines the model–harness compatibility captured by the interaction term within each benchmark. Both use five query splits, with all evaluated routes represented in training, so the results concern generalization to new queries.

**Harness representations capture similarities in design and training data.** Harnesses with similar designs tend to have similar learned representations, even though their family labels are not used during training. Among the 18 harnesses from four families (terminal CLIs, personal assistants, SWE scaffolds, and web agents), 14 have their closest neighbor in the same family, compared with 4.3 expected when family labels are randomly shuffled (Figure 6a). This similarity may arise from both shared design and shared training samples. Among harness pairs that share no benchmark, average cosine similarity remains higher within families than across families (0.35 versus 0.22), suggesting that shared benchmark membership alone does not explain the pattern.

**The interaction term captures how models and harnesses work together.** The predicted deviations identify combinations that perform better or worse than expected from their components' individual suitability. Across 45 observed HarnessBench pairs, predicted and observed deviations from these separate contributions agree closely (Figure 6b). For example, OpenClaw (local) with Sonnet 4.6 performs 18.4 percentage points above this expectation, while its pairing with DeepSeek-V4 Flash performs 27.4 points below it; the predicted deviations capture both directions. We measure these deviations by subtracting the best-fitting sum of separate model and harness effects from predicted scores and test outcomes for each query, then averaging over queries and splits. This suggests that the interaction term captures compatibility beyond individual component suitability.

### 5.8 Analysis of the interaction term and routing overhead

**The interaction term improves route selection.** The interaction term improves routing accuracy by 5.4 percentage points over a version using only the component scores, with the other architecture details unchanged (Table 3). This suggests that the interaction term helps distinguish the best candidates for each query.

**The interaction term captures model-dependent harness preferences.** The interaction term helps predict how a harness's relative performance changes across models. Across 711 observed combinations of two models and two harnesses, HM-Router improves agreement with the observed direction of this change by 29 percentage points over the component-only version (Table 3). We measure the change as

$$I = [A(m_1, h_1) - A(m_1, h_2)] - [A(m_2, h_1) - A(m_2, h_2)], \tag{9}$$

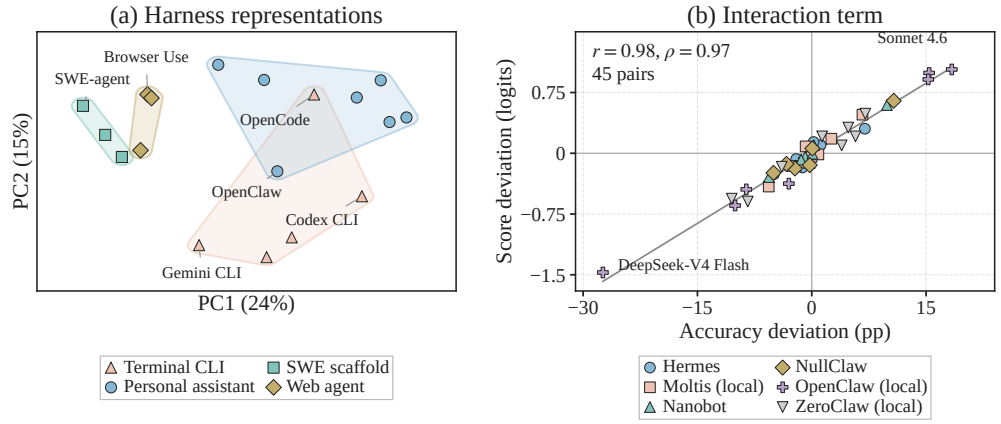



Figure 6: **Harness representations and the interaction term.** (a) PCA of 18 harness vectors from four design families, centered and unit-normalized per seed, then pooled through their mean cosine matrix. Colors, markers, and shading show annotated families, unused by the router; selected harnesses are labeled. (b) Predicted score versus observed accuracy deviations for all 45 observed HarnessBench pairs, removing separate model and harness effects and averaging over test queries and five splits. The line is a descriptive linear fit.

Table 3: **Interaction term ablation.** Sign agreement uses statistically resolved contrasts; Pearson measures contrast correlation.

| Metric | Component-only | HM-Router |
|---|---|---|
| Accuracy (%) | 68.51 | **73.93** |
| Sign agreement (%) | 55.7 | **84.8** |
| Pearson | 0.20 | **0.49** |

Table 4: **Routing overhead.** Routing time per query versus agent execution.

| Operation | Time |
|---|---|
| Query encoding | 25.4 ms/query |
| Candidate scoring | 0.2 ms/query |
| **Total routing** | **25.6 ms/query** |
| Agent execution | $>$ 100 s/task |

where $A$ is mean success on the same test queries or mean predicted success probability. Converting component-only scores to probabilities can also yield a nonzero $I$, but the higher agreement suggests that the interaction term better captures observed differences. This measures predictive agreement, not causal effects, without requiring a change in the preferred harness between models.

**Routing adds little execution time.** Routing introduces little latency relative to agent execution (Table 4). Scoring the candidate routes takes only 0.2 ms on one NVIDIA RTX PRO 6000, while training takes less than four seconds per benchmark. Nearly all routing time comes from encoding the query, consistent with the low cost of combining shared model and harness representations.

## 6 Conclusion

Our results support joint model and harness selection when training histories cover only part of their combination space. We introduced HM-Router, which shares model and harness representations across routes and captures their compatibility through an interaction term, and HM-ROUTER-BENCH to evaluate this setting. HM-Router improves mean routing accuracy across 12 benchmarks and leads at all evaluated budgets on the six-benchmark cost subset. It also generalizes to unobserved combinations, effective routing from small training datasets for new routes and components, transfer to held-out benchmarks, and correctness forecasting. The interaction term improves route selection, while shared representations allow training samples from one pairing to inform others. These representations could guide future evaluation of informative combinations and harness improvements for different models.

### AI USE STATEMENT

We used generative AI tools to assist with implementing parts of the evaluation scripts and interpreting aggregate results. We did not use these tools to generate synthetic datasets, formulate hypotheses or mathematical claims, or design the research methodology or experiments; other required-disclosure tasks are not applicable to this work. We also used generative AI to improve manuscript readability and revise figures. The authors reviewed all AI-assisted work, tested all AI-assisted code for correctness, and checked all AI-assisted text and figures against the underlying experimental data. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with AI assistance.

### ETHICS STATEMENT

Improved routing accuracy does not establish safe or fair agent behavior; deployment should retain safeguards for tool access and privacy and assess biases inherited from the underlying models and benchmarks.

### REPRODUCIBILITY STATEMENT

We detail the method in Section 4 and the evaluation protocol in Section 5.1. Appendix B provides exact model–harness pairs, scorer implementation, and hyperparameters.

## REFERENCES

Anthropic. Claude Code. Software repository, 2025. URL https://github.com/anthropics/claude-code. Accessed: September 11, 2026.

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

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, volume 2025, pp. 96934–96990, 2025.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=kiYqbO3wqw.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. In *International Conference on Learning Representations*, volume 2025, pp. 26186–26203, 2025.

Google. Gemini CLI. Software repository, 2025. URL https://github.com/google-gemini/gemini-cli. Accessed: September 11, 2026.

HKUDS. Nanobot. Software repository, 2026. URL https://github.com/HKUDS/nanobot. Accessed: September 18, 2026.

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

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, et al. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation. In *International Conference on Learning Representations*, volume 2026, pp. 98778–98849, 2026.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3): 455–500, September 2009. doi: 10.1137/07070111X.

LangChain. DeepAgents. Software repository, 2025. URL https://github.com/langchain-ai/deepagents. Accessed: September 18, 2026.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026.

Xinchen Liu, Hang Zhou, Yingjie Zong, Yuchuan Tian, Liuyang Song, Shuo Zhang, Yulong Li, Wei He, Mengyu Zheng, Runke Liu, et al. Agentic routing: The harness-native data flywheel. *arXiv preprint arXiv:2607.11399*, 2026a.

Zewen Liu, Zhan Shi, Yisi Sang, Bing He, Minhua Lin, Tianxin Wei, Dakuo Wang, Benoit Dumoulin, Wei Jin, and Hanqing Lu. Adaptive auto-harness: Sustained self-improvement for agentic system deployment on open-ended task streams. *arXiv preprint arXiv:2606.01770*, 2026b.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P Murphy. Autoharness: improving llm agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Moltis. Moltis: A secure persistent personal agent server. Software repository, 2026. URL https://github.com/moltis-org/moltis. Accessed: September 18, 2026.

Magnus Müller and Gregor Žunič. Browser use: Enable ai to control your browser, 2024. URL https://github.com/browser-use/browser-use.

Nous Research. Hermes Agent. Software repository, 2026. URL https://github.com/NousResearch/hermes-agent. Accessed: September 18, 2026.

NullClaw. NullClaw. Software repository, 2026. URL https://github.com/nullclaw/nullclaw. Accessed: September 18, 2026.

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, 2025. URL https://arxiv.org/abs/2406.18665.

OpenAI. Codex CLI. Software repository, 2025. URL https://github.com/openai/codex. Accessed: September 11, 2026.

OpenClaw. OpenClaw. Software repository, 2025. URL https://github.com/openclaw/openclaw. Accessed: September 18, 2026.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. Smolagents: a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

Erik Schluntz. Raising the bar on swe-bench verified with claude 3.5 sonnet. *Anthropic Engineering Blog*, 2025.

Zachary S Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. Core-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *arXiv preprint arXiv:2409.11363*, 2024.

Seongho Son, Sangwoong Yoon, Jiahua Tang, Shuhan Wang, Lorenz Wolf, and Ilija Bogunovic. Swe-router: Routing in multi-turn agentic software engineering tasks. *arXiv preprint arXiv:2607.00053*, 2026.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. Irt-router: Effective and interpretable multi-llm routing via item response theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15629–15644, 2025.

SST. OpenCode. Software repository, 2025. URL https://github.com/sst/opencode. Accessed: September 18, 2026.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, Weishu Zhang, Tyler Zeng, Ying Yan, Bo Liu, Hanson Wen, Mingyang Xu, Xiaoyuan Liu, Zimeng Chen, Weiyan Shi, Amanda Dsouza, Vincent Sunn Chen, Dawn Song, Patrick Bryant, Carl Boettiger, Yamini Rangan, Bradley Rothenberg, Kyle Steinfeld, Arvind Rao, Tapio Schneider, Georgios Yannakakis, Laure Zanna, Kaan Ozbay, Ida Sim, Tarek Zohdi, George Em Karniadakis, Jack Gallant, Teresa Head-gordon, et al. Agents' last exam. *arXiv preprint arXiv:2606.05405*, 2026.

Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Yukai Wu, Weizheng Wang, Hongzhang Huang, Wei Zhou, Jun Zhou, Jiachen Song, et al. Workspace-bench 1.0: Benchmarking ai agents on workspace tasks with large-scale file dependencies. *arXiv preprint arXiv:2605.03596*, 2026.

Minyang Tian, Luyu Gao, Shizhuo D Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*, 2025.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, et al. Harness-bench: Measuring harness effects across models in realistic agent workflows. *arXiv preprint arXiv:2605.27922*, 2026.

ZeroClaw Labs. ZeroClaw. Software repository, 2026. URL https://github.com/zeroclaw-labs/zeroclaw. Accessed: September 18, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025a.

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. In *Proceedings of the 2025 7th International Conference on Distributed Artificial Intelligence*, pp. 122–129, 2025b.

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

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=piecKJ2DlB.

Pengfei Zhou, Zhiwei Tang, Yixing Ma, Jiasheng Tang, Yizeng Han, Zhenglin Wan, Fanqing Meng, Wei Wang, Bohan Zhuang, Wangbo Zhao, et al. Agent-as-a-router: Agentic model routing for coding tasks. *arXiv preprint arXiv:2606.22902*, 2026.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models. In *International Conference on Learning Representations*, volume 2025, pp. 76913–76926, 2025.

# A Limitations

HM-ROUTER-BENCH covers a limited number of routes, and its model–harness grid is sparse. Larger route pools and denser grids would allow a more complete evaluation of compatibility effects and of routing to unobserved combinations. We will continue to add execution data to the benchmarks and to evaluate additional harnesses.

# B Additional Experimental Details

## B.1 Benchmark construction and coverage

Table 5 lists the exact pairs of models and harnesses across benchmarks. The cost subset comprises TerminalBench 2.1, TauBench Airline, CORE-Bench Hard, ScienceAgentBench, GAIA, and SciCode, whose route pools are re-derived by requiring a per-task cost on every query. The remaining six benchmarks either report no per-task cost or leave a degenerate cost pool.

Table 5: **Exact model–harness pairs in the retained pools.**

| Benchmark | Harness(es) | Model identifiers |
|---|---|---|
| TerminalBench 2.1 (Merrill et al., 2026) | Claude Code (Anthropic, 2025) | `claude-fable-5`; `claude-opus-4-7`; `claude-opus-4-8`; `claude-sonnet-5`; `glm-5-1` |
| TerminalBench 2.1 | Codex (OpenAI, 2025) | `gpt-5-5`; `gpt-5-6-luna`; `gpt-5-6-terra` |

*Continued on next page*

**Table 5: Exact model–harness pairs (continued)**

| Benchmark | Harness(es) | Model identifiers |
|---|---|---|
| TerminalBench 2.1 | Gemini CLI (Google, 2025) | `gemini-3-1-pro-preview`; `gemini-3-pro-preview` |
| TerminalBench 2.1 | Terminus 2 (Merrill et al., 2026) | `claude-fable-5`; `claude-opus-4-7`; `gemini-3-1-pro-preview`; `gemini-3-pro-preview`; `gpt-5-5` |
| TauBench Airline (Yao et al., 2024) | HAL Generalist (Kapoor et al., 2026) | `claude-opus-4`; `claude-opus-4-1`; `claude-sonnet-3-7`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gpt-4-1`; `gpt-5`; `o4-mini` |
| TauBench Airline | Taubench ToolCalling (Yao et al., 2024) | `claude-opus-4-1`; `claude-sonnet-3-5`; `claude-sonnet-3-7`; `gemini-2-0-flash`; `gpt-4-1`; `gpt-4o`; `gpt-5`; `o3`; `o4-mini` |
| CORE-Bench Hard (Siegel et al., 2024) | CORE-Agent (Siegel et al., 2024) | `claude-haiku-4-5`; `claude-opus-4-1`; `claude-opus-4-5`; `claude-sonnet-3-7`; `claude-sonnet-4`; `claude-sonnet-4-5`; `deepseek-chat-v3-1`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gemini-2-5-pro-preview-03-25`; `gemini-3-pro-preview`; `gpt-4-1`; `gpt-5`; `gpt-oss-120b`; `o3`; `o3-mini`; `o4-mini` |
| CORE-Bench Hard | Claude Code (Anthropic, 2025) | `claude-opus-4-1`; `claude-opus-4-5`; `claude-sonnet-4`; `claude-sonnet-4-5` |
| CORE-Bench Hard | HAL Generalist Agent (Kapoor et al., 2026) | `claude-3-7-sonnet:thinking`; `claude-opus-4-1`; `claude-opus-4-5`; `claude-sonnet-3-7`; `claude-sonnet-4-5`; `deepseek-chat-v3-0324`; `deepseek-r1`; `deepseek-r1-0528`; `deepseek-v3`; `gemini-2-0-flash`; `gemini-2-5-pro-preview-03-25`; `gemini-3-pro-preview`; `gpt-4-1`; `gpt-5`; `gpt-oss-120b`; `o3`; `o4-mini` |
| ScienceAgentBench (Chen et al., 2025) | HAL Generalist Agent (Kapoor et al., 2026) | `claude-sonnet-3-7`; `deepseek-v3`; `gpt-4-1`; `o3`; `o4-mini` |
| ScienceAgentBench | SAB Self-Debug (Chen et al., 2025) | `claude-haiku-4-5`; `claude-opus-4-1`; `claude-sonnet-3-7`; `claude-sonnet-4-5`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gemini-2-5-pro-preview-03-25`; `gpt-4-1`; `gpt-5`; `o3`; `o4-mini` |

*Continued on next page*

**Table 5: Exact model–harness pairs (continued)**

| Benchmark | Harness(es) | Model identifiers |
|---|---|---|
| GAIA (Mialon et al., 2024) | HAL Generalist Agent (Kapoor et al., 2026) | `claude-haiku-4-5`; `claude-opus-4`; `claude-opus-4-1`; `claude-sonnet-3-7`; `claude-sonnet-4-5`; `deepseek-chat-v3-0324`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gpt-4-1`; `gpt-5`; `o3`; `o3-mini`; `o4-mini` |
| GAIA | HF Open Deep Research (Roucher et al., 2025) | `claude-opus-4`; `claude-opus-4-1`; `claude-sonnet-3-7`; `claude-sonnet-4-5`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gpt-4-1`; `o3`; `o3-mini`; `o4-mini` |
| Online Mind2Web (Deng et al., 2023; Xue et al., 2025) | Browser-Use (Müller & Žunič, 2024) | `claude-sonnet-3-7`; `claude-sonnet-4`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gemini-2-5-pro-preview-03-25`; `gpt-4-1`; `gpt-5`; `o3`; `o4-mini` |
| Online Mind2Web | SeeAct (Zheng et al., 2024) | `claude-haiku-4-5`; `claude-sonnet-3-7`; `claude-sonnet-4`; `claude-sonnet-4-5`; `gemini-2-0-flash`; `gemini-2-5-pro-preview-03-25`; `gpt-4-1`; `gpt-5`; `o3`; `o4-mini` |
| SWE-bench Verified (Jimenez et al., 2024) | OpenHands (Wang et al., 2025) | `claude-opus-4-5`; `claude-sonnet-3-5`; `claude-sonnet-4`; `devstral-small-2505`; `glm-4-5`; `glm-4-6`; `gpt-5`; `kimi-k2-0711-preview`; `qwen-3-coder-30b-a3b-instruct`; `qwen-3-coder-480b-a35b-instruct`; `skywork-swe-32b` |
| SWE-bench Verified | SWE-agent (Yang et al., 2024) | `claude-opus-3`; `claude-sonnet-3-5`; `claude-sonnet-4`; `devstral-small-2507`; `gpt-4-1106-preview`; `gpt-4o`; `kimi-k2-instruct`; `swe-agent-lm-32b` |
| SWE-bench Verified | Tools (Schluntz, 2025) | `claude-haiku-3-5`; `claude-opus-4`; `claude-sonnet-3-5`; `claude-sonnet-3-7`; `claude-sonnet-4` |
| HarnessBench (Yao et al., 2026) | hermes (Nous Research, 2026); moltis-local (Moltis, 2026); nanobot (HKUDS, 2026); zeroclaw-local (ZeroClaw Labs, 2026) | `claude-opus-4-6`; `claude-sonnet-4-6`; `deepseek-v4-flash`; `gemini-3-1-pro`; `glm-5-1`; `gpt-5-4`; `kimi-k2-5`; `qwen-3-6-plus` |
| HarnessBench | nullclaw (NullClaw, 2026) | `claude-opus-4-6`; `claude-sonnet-4-6`; `gemini-3-1-pro`; `gpt-5-4`; `kimi-k2-5`; `qwen-3-6-plus` |

*Continued on next page*

**Table 5: Exact model–harness pairs (continued)**

| Benchmark | Harness(es) | Model identifiers |
|---|---|---|
| HarnessBench | openclaw-local (OpenClaw, 2025) | `claude-opus-4-6`; `claude-sonnet-4-6`; `deepseek-v4-flash`; `gemini-3-1-pro`; `glm-5-1`; `gpt-5-4`; `qwen-3-6-plus` |
| CORE-Bench 1.1 (Siegel et al., 2024) | CORE-Agent (Siegel et al., 2024); OpenCode (SST, 2025) | `claude-opus-4-5`; `claude-opus-4-6`; `gpt-5-4` |
| CORE-Bench 1.1 | Claude Code (Anthropic, 2025) | `claude-opus-4-5`; `claude-opus-4-6` |
| CORE-Bench 1.1 | Codex (OpenAI, 2025) | `gpt-5`; `gpt-5-1`; `gpt-5-2`; `gpt-5-3-codex`; `gpt-5-4` |
| SciCode (Tian et al., 2024) | HAL Generalist Agent (Kapoor et al., 2026); SciCode Zero Shot Agent (Tian et al., 2024) | `claude-sonnet-3-7`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gpt-4-1`; `o3`; `o4-mini` |
| SciCode | SciCode Tool Calling Agent (Tian et al., 2024) | `claude-haiku-4-5`; `claude-opus-4-1`; `claude-sonnet-3-7`; `claude-sonnet-4-5`; `deepseek-r1`; `deepseek-v3`; `gemini-2-0-flash`; `gpt-4-1`; `gpt-5`; `o3`; `o4-mini` |
| SkillsBench (Li et al., 2026) | Claude Code (Anthropic, 2025) | `claude-haiku-4-5`; `claude-opus-4-5`; `claude-opus-4-6`; `claude-opus-4-7`; `claude-sonnet-4-5` |
| SkillsBench | Codex (OpenAI, 2025) | `gpt-5.2-codex`; `gpt-5.5` |
| SkillsBench | Gemini CLI (Google, 2025) | `gemini-3-flash`; `gemini-3.1-pro` |
| SkillsBench | OpenHands (Wang et al., 2025) | `claude-opus-4-7`; `claude-opus-4-8`; `claude-sonnet-4-6`; `deepseek-v4-flash`; `deepseek-v4-pro`; `gemini-3.1-flash-lite`; `gemini-3.1-pro`; `gemini-3.5-flash`; `glm-5.1`; `gpt-5.4-mini`; `gpt-5.5`; `grok-4.3`; `hy3`; `kimi-k2.6`; `minimax-m2.7`; `minimax-m3` |
| WorkspaceBench-Lite (Tang et al., 2026) | Codex (OpenAI, 2025) | `gemini-3.1-pro`; `glm-5.1`; `gpt-5.4`; `grok-4.3`; `minimax-m2.7`; `qwen-3.6-plus` |
| WorkspaceBench-Lite | DeepAgent (LangChain, 2025); Hermes (Nous Research, 2026); OpenClaw (OpenClaw, 2025) | `gemini-3.1-pro`; `glm-5.1`; `gpt-5.4`; `grok-4.3`; `kimi-2.5`; `minimax-m2.7`; `qwen-3.6-plus` |

Table 6: **Accuracy attainable within a dollar budget** (%). Columns give absolute US dollars per task. Each entry is the best setting affordable by realized test spend on each of the six cost benchmarks, then averaged. Bold and underlining rank non-oracle methods; a dash means no setting fits on at least one benchmark.

| **Method** | $\leq$ \$0.75 | $\leq$ \$1 | $\leq$ \$1.50 | $\leq$ \$2 | $\leq$ \$3 | $\leq$ \$5 | $\leq$ \$10 |
|---|---|---|---|---|---|---|---|
| Random | — | — | — | — | 40.20 | 40.20 | 40.20 |
| Fixed | 46.29 | 49.01 | 52.89 | 60.23 | 60.23 | <u>66.90</u> | <u>66.90</u> |
| Av-Pro | <u>53.58</u> | <u>55.25</u> | <u>58.41</u> | <u>60.67</u> | <u>60.67</u> | 62.20 | 62.20 |
| IRT | 49.54 | 49.58 | 49.63 | 53.98 | 54.70 | 54.70 | 54.70 |
| EmbedLLM | 50.11 | 53.54 | 53.63 | 59.95 | 59.95 | 59.95 | 59.95 |
| HM-Router | **58.94** | **61.86** | **61.86** | **69.21** | **69.21** | **70.89** | **70.89** |
| Oracle | 99.95 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |

### B.2 Scorer implementation and optimization

Each component $j$ carries a behavioral summary $\mathbf{z}_j$ built from the training outcomes of the routes containing it. Write $s_{q,j}$ for the mean outcome over those routes on training query $q$, and $n_j$ for how many of them have any observation. Then $\mathbf{z}_j$ concatenates three parts:

- the sum of $(2s_{q,j} - 1)\mathbf{x}_q$ over queries with an observation, divided by the total number of training queries;
- the mean of $s_{q,j}$ over those same queries;
- $\log(1 + n_j)$.

A component with no observations gets a zero summary and a zero identity embedding. Summaries are recomputed whenever outcomes are withheld or added, and withheld outcomes enter neither the summaries nor the loss. In probe experiments the identity embeddings stay zero, the probe outcomes update the summaries, and the interaction term stays active.

The implemented scorer separately normalizes the three product features:

$$\begin{aligned} S(q, m, h) = b &+ \mathbf{w}_M^\top \mathrm{LN}_M(\mathbf{u}_m \odot \tilde{\mathbf{q}}) \\ &+ \mathbf{w}_H^\top \mathrm{LN}_H(\mathbf{u}_h \odot \tilde{\mathbf{q}}) \\ &+ \mathbf{w}_{MH}^\top \mathrm{LN}_{MH}(\mathbf{u}_m \odot \mathbf{u}_h \odot \tilde{\mathbf{q}}). \end{aligned} \tag{10}$$

Unlike the unnormalized interaction in Section 4.3, this feature-dependent normalization does not guarantee CP rank at most $d$.

We fit one scorer per benchmark and split on the binary cross-entropy of $\hat{y}(q, r) = \sigma(S(q, r))$, using $d = 32$, identity and side-feature width 16, and 250 Adam epochs with learning rate $10^{-3}$, weight decay $10^{-4}$, and Gaussian query noise $\sigma = 0.15$.