RESPONSES REVEAL WHAT OUTCOMES HIDE: RESPONSE-INFORMED VALUE LEARNING FOR MULTI-TURN LLM ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-turn LLM routing selects a model at every turn before its response is available, yet task success is revealed only when the interaction ends, leaving each intermediate choice without a label of its own. Logged trajectories, however, retain the response each call produced, which exposes what each turn required and how well the chosen model served it, evidence that outcomes alone hide. We introduce RIVET (Response-Informed Value Estimation for Turn-Level Routing), which exploits this post-call evidence during training while keeping routing strictly pre-call at deployment. RIVET jointly trains two value estimators over a shared state–model representation: a deployable Routing Value Estimator that scores candidate models from pre-call information, and a training-only Response-Informed Value Estimator that additionally observes the realized response. Conditioning on the response turns response-induced return variation from residual noise into an observed covariate, and we show that this reduces the leading asymptotic estimation error of the shared representation. On the MINT benchmark, RIVET solves more held-out tasks than the strongest multi-turn baseline while using 74.7% less total call cost.

1 INTRODUCTION

Routing among language models of different capability and price is a standard way to trade accuracy for cost: a learned router predicts which model is worth calling for a query (Ong et al., 2025; Ding et al., 2024; Chen et al., 2024b; Zhuang et al., 2025; Feng et al., 2024; Zhang et al., 2025b), cascades escalate only when a cheaper answer looks insufficient (Chen et al., 2024a; Aggarwal et al., 2024), and benchmarks score the resulting accuracy–cost frontier (Hu et al., 2024). Yet the workloads where routing matters most are increasingly multi-turn: agents reason, invoke tools, observe feedback, and decide how to proceed (Wang et al., 2024a), so every turn is a fresh routing decision, and what each turn demands of a model is not something a static strong/weak ranking can tell (Figure 1).

Multi-turn routing is harder along three axes. **Credit assignment is delayed**: only the final outcome is graded, so intermediate decisions receive no label of their own. **Counterfactual evaluation is trajectory-dependent**: changing the model at one turn changes every later state, so alternative routes cannot be scored against a shared context. **Trajectory collection is expensive**: each trajectory is a sequence of paid model calls, so on-policy learning buys every sample at multi-turn prices. A multi-turn router must therefore propagate delayed consequences while extracting as much as possible from the experience already logged.

Logged experience, however, contains more than outcomes. The response recorded at each call shows what that turn actually required and whether the chosen model delivered it. Figure 1 illustrates this on a four-turn recommendation task. **A response exposes decision quality, not just spending**: whether it honors the user's hard constraint, picks up the preference just stated, keeps what earlier turns already settled, and answers at the length the turn deserves. None of this is visible in the final outcome, and all of it matters for which model to call next. **No single model wins every turn**: the small model succeeds where it keeps the constraint, the strongest model fails where it drifts from the conversation, and the mid-sized model over-deliberates or over-explains. These failure modes

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.

differ from turn to turn, so no fixed strong/weak ranking captures them. Responses therefore carry turn-level value beyond cost: they show what the turn asked for and whether the model answered it. A pre-call router must anticipate these effects from state–model information alone. This motivates using the realized response as privileged auxiliary information during training, while keeping routing strictly pre-call at deployment.

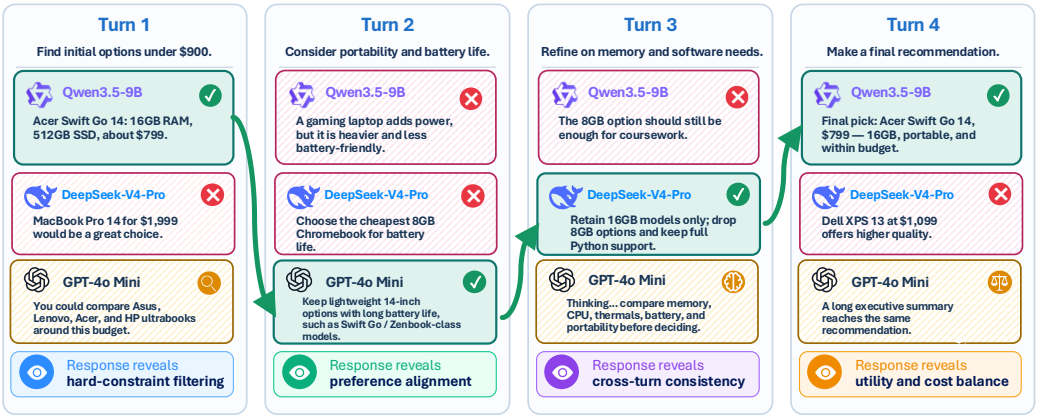


Figure 1: Intermediate responses reveal what outcomes hide. Three candidate models on a four-turn recommendation task. Each turn demands a different ability (bottom labels), and the response exposes whether the model met it: check and cross mark acceptable and unacceptable responses, and the remaining icons mark responses that are vague, over-deliberated, or needlessly verbose. The green path is the cost-efficient route and switches models across turns. Illustrative; each logged turn observes only the selected model's response.

To exploit both properties of logged experience, we introduce **RIVET** (Response-Informed Value Estimation for Turn-Level Routing), a sample-efficient off-policy value-learning framework for multi-turn LLM routing. RIVET formulates turn-level model selection as a cost-aware sequential decision problem with trajectory-level supervision, reuses logged transitions to propagate future routing consequences through Bellman updates, and requires only a small online adaptation budget instead of large-scale on-policy rollout collection.

RIVET uses the logged responses through two value estimators that share one trainable state–model representation. The deployed *Routing Value Estimator* (RVE) conditions only on the current state and candidate model; the training-only *Response-Informed Value Estimator* (RIVE) additionally observes the executed response. Both predict the same future return, but RIVE faces a cleaner problem: response-dependent variation that RVE must treat as noise is an observed input for RIVE. Because the representation is shared, what RIVE learns on this cleaner problem improves the representation RVE relies on; Section 4.3 shows that joint training suppresses the response-resolved component of return uncertainty in the estimation error of the shared representation. **At deployment RIVE is discarded**: routing remains strictly pre-call, with no additional model invocation or post-call stage.

On the MINT multi-turn benchmark, RIVET improves task accuracy over MTRouter while reducing model-call cost by 74.7%. Its cost advantage transfers across three out-of-distribution benchmarks while maintaining competitive task performance. Controlled experiments show that the gain depends on RIVE having access to the executed response (Section 5.4).

Our contributions are threefold:

① **Multi-turn routing as off-policy value learning.** We formulate turn-level model selection in agentic interaction as a cost-aware sequential value-learning problem with delayed trajectory-level supervision, and show that a small logged corpus plus limited online adaptation suffices for effective routing.

② **Response-informed auxiliary value learning.** A training-only estimator that additionally conditions on the executed response shares a state–model representation with the deployable

pre-call estimator, providing an auxiliary signal without changing the deployment routing path. We show this reduces the leading asymptotic estimation error of the shared representation.

③ **Sample-efficient multi-turn routing results.** From a small logged corpus and a limited online budget, RIVET reaches the high-accuracy end of the measured accuracy–cost frontier on MINT, retains strong cost efficiency under out-of-distribution evaluation, and maintains its accuracy–cost trade-off under candidate-model unavailability.

2 Related Work

LLM routing. Most routers choose a model for one query and stop. Pre-call predictors estimate query–model compatibility from preference data (Ong et al., 2025), predicted difficulty (Ding et al., 2024), contrastive query and model embeddings (Chen et al., 2024b; Zhuang et al., 2025), task–query–model graphs (Feng et al., 2024), or query clusters (Zhang et al., 2025b); cascades inspect a cheap answer before escalating (Chen et al., 2024a; Aggarwal et al., 2024); bandits adapt the choice online (Dai et al., 2024); and RouterBench standardizes accuracy–cost evaluation (Hu et al., 2024). All score candidates within a fixed query context. Agentic workloads make routing sequential, yet the multi-turn setting remains sparsely studied (Moslem & Kelleher, 2026). Router-R1, R2-Reasoner, and xRouter instantiate the router as an LLM that interleaves reasoning with model or tool calls and train it with on-policy policy gradients (Zhang et al., 2025a; Shao et al., 2025; Qian et al., 2025; Shao et al., 2024); GraphPlanner assembles model–role workflows from prior interactions (Feng et al., 2026); SWE-Router watches a cheap agent's partial trajectory before a one-shot escalation (Son et al., 2026); DialRouter distills MCTS rollouts into a dialogue router by behavior cloning (Zhang et al., 2026a); and MTRouter, which shares our decision unit, selects a model at every turn from a supervised outcome estimator over history–model embeddings (Zhang et al., 2026b). These methods buy each training sample with fresh rollouts, imitate search, or regress outcomes without propagating value across turns; **none learns turn-level action values off-policy from logged trajectories**.

Value-based RL and privileged supervision. Value-based methods are established for language decisions: ILQL learns token-level Q-functions offline (Snell et al., 2022), ArCHer shows that a turn-level TD critic is far more sample-efficient than on-policy training for multi-turn agents (Zhou et al., 2024), and conservative Q-learning makes bootstrapping from fixed logs safe (Kumar et al., 2020). RIVET brings this machinery to the discrete model-selection action space with Double DQN targets (van Hasselt et al., 2016) and CQL regularization. Its second ingredient draws on learning with privileged information (Vapnik & Vashist, 2009; Lopez-Paz et al., 2015) and asymmetric actor–critic methods, where a training-time critic observes what the deployed policy cannot (Pinto et al., 2017; Baisero & Amato, 2021), cast here as an auxiliary task (Jaderberg et al., 2016). Cascades and SWE-Router also exploit the response, but at deployment, paying for a call before deciding; **RIVET uses the response only during training**, so the deployed router stays pre-call, and we characterize when this auxiliary supervision improves the shared representation (Section 4.3).

3 Problem Formulation

We formulate multi-turn LLM routing as a cost-aware Markov decision process. For a task q and candidate model set $\mathcal{M}$, the state at turn t is $s_t = (q, \mathcal{H}_{<t})$, where $\mathcal{H}_{<t}$ is the interaction history before the current model call. The router selects a model $m_t \sim \pi(\cdot \mid s_t)$, which generates a response $o_t \sim p_{m_t}(\cdot \mid s_t)$; the environment then produces feedback f_t, and the interaction transitions to $s_{t+1} = T(s_t, o_t, f_t)$. Each call incurs a cost $c_t = c(s_t, m_t, o_t)$. Let $d_t \in \{0, 1\}$ indicate episode termination and $y \in \{0, 1\}$ the final task outcome. The per-turn reward is

$$r_t = -\lambda \frac{c_t}{c_{\text{ref}}} + d_t y, \tag{1}$$

where $c_{\text{ref}} > 0$ is a reference cost and $\lambda \geq 0$ controls the trade-off between task success and model-call cost. The objective is a routing policy that maximizes the expected discounted return

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right], \tag{2}$$

with discount $\gamma \in (0, 1]$ and p_π the trajectory distribution induced by π, the model responses, and the environment. Because each model choice changes the generated response and hence the states encountered at later turns, routing decisions are **sequentially coupled rather than independent turn-level predictions**.

4 METHOD: RIVET

RIVET jointly trains a deployable **Routing Value Estimator (RVE)** and a training-only **Response-Informed Value Estimator (RIVE)** over a shared state–model representation (Figure 2); the two differ only in the information available to their value heads.

4.1 DUAL VALUE ESTIMATORS WITH A SHARED STATE–MODEL REPRESENTATION

Shared state–model representation. RIVET represents each candidate action with a shared trainable state–model encoder,

$$z_\theta(s_t, m) = a_\theta(h_t,\ g_\theta(e_m, x_m))\,, \tag{3}$$

which jointly encodes the current interaction state and the candidate model. A frozen text encoder maps the serialized state s_t to h_t; the serialization omits earlier model identities. A trainable model-card projector g_θ maps each model's text embedding e_m and structured fields x_m to a model vector, and a trainable state adapter a_θ fuses it with h_t. The same representation feeds both value estimators, so training signals from pre-call and response-informed prediction both shape the representation used at deployment.

Routing Value Estimator (RVE). Given the shared state–model representation, RVE estimates the pre-call value of selecting each candidate model,

$$Q_{\text{route}}(s_t, m) = f_{\text{route}}(z_\theta(s_t, m))\,, \tag{4}$$

and directly determines the deployment-time routing decision,

$$m_t = \arg\max_{m \in \mathcal{M}} Q_{\text{route}}(s_t, m). \tag{5}$$

Response-Informed Value Estimator (RIVE). During training, RIVE augments the shared state–model representation of the executed model m_t with the realized response o_t. Let $e(o_t)$ denote the response representation, obtained by passing o_t through the frozen encoder and a trainable response projector. RIVE predicts the same return target,

$$Q_{\text{resp}}(s_t, m_t, o_t) = f_{\text{resp}}(z_\theta(s_t, m_t),\ e(o_t))\,. \tag{6}$$

RIVE is discarded at deployment and adds no model call or forward pass at inference time. Appendix B.3 specifies the encodings and network dimensions.

4.2 RESPONSE-INFORMED OFF-POLICY LEARNING

RIVET learns the shared state–model representation in three stages: it **first** grounds both value estimators with completed-trajectory returns, **then** performs off-policy Bellman refinement over logged replay, and **finally** adapts the routing policy with limited online interaction.

Response-informed joint learning. Both estimators regress to the same scalar return target. Let $q_{\text{route},t} = Q_{\text{route}}(s_t, m_t)$ and $q_{\text{resp},t} = Q_{\text{resp}}(s_t, m_t, o_t)$, let ℓ_δ denote the Huber loss, and let $\mathbb{E}_t$ average over the current replay distribution. For a scalar target Z_t, we optimize

$$\mathcal{L}_{\text{fit}}(Z) = \mathbb{E}_t[\ell_\delta(q_{\text{route},t}, Z_t) + \beta\, \ell_\delta(q_{\text{resp},t}, Z_t)]\,, \tag{7}$$

where $\beta \geq 0$ controls the strength of the response-informed auxiliary objective. For a fixed target, we refer to the first and second unweighted expected loss terms as $\mathcal{L}_{\text{route}}$ and $\mathcal{L}_{\text{resp}}$, respectively. Because both terms back-propagate through $z_\theta(s_t, m_t)$, the realized response can shape the state–model representation ultimately used by RVE. Section 4.3 analyzes this mechanism.

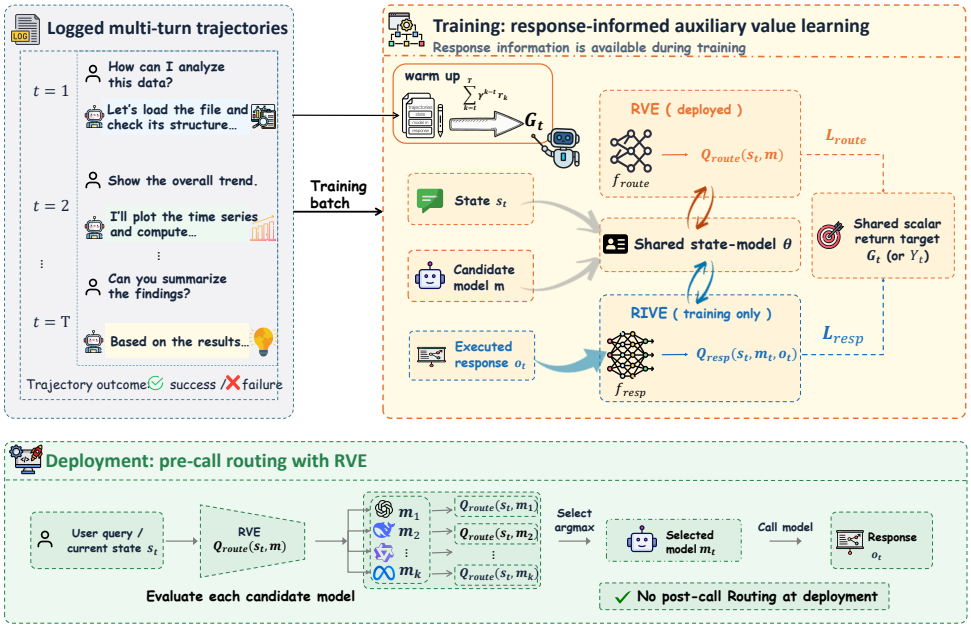


Figure 2: RIVET separates pre-call deployment from post-call training. Both estimators read the same trainable state–model representation $z_\theta(s_t, m)$. The Routing Value Estimator (RVE, deployed) scores every candidate model from pre-call information and makes the routing decision; the Response-Informed Value Estimator (RIVE, training only) additionally conditions on the executed response and shapes the shared representation through its own regression loss. Both estimators regress to the same scalar return target; deployment uses RVE alone.

Return-grounded Monte Carlo warm start. Let $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ denote the realized discounted return from turn t onward, with r_k as in Equation 1. We initialize both value estimators with $\mathcal{L}_{\text{MC}} = \mathcal{L}_{\text{fit}}(G)$. Completed trajectories thus provide direct return supervision before bootstrapping is introduced.

Off-policy Bellman refinement. After the Monte Carlo warm start, we perform temporal-difference learning with Double DQN (van Hasselt et al., 2016). Let ω collect the online RVE parameters, including θ, and let $\bar{\omega}$ denote their target-network copy. RVE selects the continuation action and the target network evaluates it:

$$m^\star_{t+1} = \arg\max_{m' \in \mathcal{M}} Q_{\text{route}}(s_{t+1}, m'; \omega), \tag{8}$$

$$Y_t = r_t + \gamma(1 - d_t) Q_{\text{route}}(s_{t+1}, m^\star_{t+1}; \bar{\omega}). \tag{9}$$

RVE defines the continuation because it is the estimator used for routing at deployment, while both RVE and RIVE regress to the same Bellman target through $\mathcal{L}_{\text{fit}}(Y)$.

Because each logged state contains only one executed action whereas RVE must rank the full candidate set at deployment, we additionally apply conservative Q-learning (CQL) (Kumar et al., 2020):

$$\mathcal{L}_{\text{CQL}} = \mathbb{E}_t \left[\log \sum_{m \in \mathcal{M}} \exp Q_{\text{route}}(s_t, m) - q_{\text{route},t} \right], \tag{10}$$

which suppresses optimistic values for weakly supported actions. The off-policy objective is therefore

$$\mathcal{L}_{\text{TD}} = \mathcal{L}_{\text{fit}}(Y) + \alpha \mathcal{L}_{\text{CQL}}, \tag{11}$$

where α controls the strength of conservative regularization.

Replay-anchored online adaptation. Offline training consists of the Monte Carlo warm start followed by off-policy Bellman refinement. We then collect a limited number of trajectories under the learned router and continue optimizing $\mathcal{L}_{\mathrm{TD}}$ on a balanced mixture of logged and newly collected transitions. Logged replay anchors learning, while fresh trajectories expose the router to states induced by its own decisions. Appendix B.2 provides the optimization and stopping details, including the default $\beta = 1$.

4.3 WHY RESPONSE-INFORMED AUXILIARY LEARNING HELPS

We show that the realized response makes RIVE a cleaner auxiliary value-prediction problem and that, through the shared representation $z_\theta(s, m)$, this cleaner signal improves the estimation of the deployed RVE. The analysis covers the joint regression component of the objective for a fixed target; CQL separately regularizes action ranking. Assumptions and complete proofs are in Appendix A.

Response-resolved uncertainty. Fix a turn and let $X = (S_t, M_t)$ denote the pre-call information, $O = O_t$ the realized response, and Y the scalar value target used in Section 4.2. Define

$$Q^\star_{\mathrm{route}}(X) = \mathbb{E}[Y \mid X], \qquad Q^\star_{\mathrm{resp}}(X, O) = \mathbb{E}[Y \mid X, O].$$

We decompose the return target as

$$Y = Q^\star_{\mathrm{route}}(X) + \Delta(X, O) + \epsilon, \tag{12}$$

where

$$\Delta(X, O) = Q^\star_{\mathrm{resp}}(X, O) - Q^\star_{\mathrm{route}}(X), \qquad \epsilon = Y - Q^\star_{\mathrm{resp}}(X, O).$$

Here Δ captures return variation that becomes predictable after observing the response, while ϵ denotes the remaining uncertainty.

Proposition 4.1 (Response-resolved uncertainty)**.** *For every pre-call input X,*

$$\mathrm{Var}(Y \mid X) = \mathrm{Var}(\Delta \mid X) + \mathbb{E}[\mathrm{Var}(Y \mid X, O) \mid X]. \tag{13}$$

Interpretation. RVE predicts the return from (s, m) alone and must treat both Δ and ϵ as residual variation; RIVE observes the response and explains Δ, leaving only ϵ. The response thus **converts a specific component of return variation from latent noise into an observed covariate**, making RIVE a cleaner auxiliary regression problem.

Benefit to the shared representation. We next ask whether this cleaner auxiliary problem improves the representation used by RVE. Around the population optimum, let $J(X)$ denote the sensitivity of the pre-call prediction to the shared representation parameters, and define

$$H = \mathbb{E}[J(X)J(X)^\top], \qquad \Sigma_\Delta = \mathbb{E}[J(X)J(X)^\top \Delta^2], \qquad \Sigma_\epsilon = \mathbb{E}[J(X)J(X)^\top \epsilon^2].$$

Under the local shared-signal and regularity conditions stated in Appendix A, we obtain the following result.

Theorem 4.2 (Response-informed auxiliary learning reduces representation estimation error)**.** *Let $\hat{\theta}_\beta$ denote the shared-representation parameters learned with $\mathcal{L}_{\mathrm{route}} + \beta\mathcal{L}_{\mathrm{resp}}$. Then*

$$\mathrm{Cov}(\hat{\theta}_\beta) = \frac{1}{n} H^{-1} \left(\Sigma_\epsilon + \frac{\Sigma_\Delta}{(1+\beta)^2} \right) H^{-1} + o(n^{-1}). \tag{14}$$

Relative to RVE-only training ($\beta = 0$), response-informed auxiliary learning therefore reduces the leading contribution of response-induced variation to the representation-estimation error by the factor $1/(1+\beta)^2$. Whenever $\Sigma_\Delta \neq 0$, the improvement is strict along directions containing response-resolved variation.

Interpretation. RIVE benefits the deployed router without changing its decision rule: it supplies an additional estimating signal for the same representation, free of Δ as residual noise, so joint learning **suppresses the effect of this variation on representation estimation by the factor** $1/(1+\beta)^2$, while the contribution of ϵ is unchanged.

5 EXPERIMENTS

Our experiments ask whether RIVET delivers a better accuracy–cost trade-off in multi-turn interaction and what produces it: in-distribution and out-of-distribution (OOD) performance (Section 5.2), the effect of the cost coefficient and online adaptation (Section 5.3), component ablations that isolate the response input (Section 5.4), and turn-level routing interventions (Section 5.5).

5.1 EXPERIMENTAL SETUP

Datasets. We train and evaluate on the MINT Reasoning subset (Wang et al., 2024a), which draws from GSM8K, HotpotQA, MATH, MMLU, and TheoremQA, using fixed, disjoint training, validation, and test manifests. For transfer beyond the training distribution we test on MuSiQue (Trivedi et al., 2022), DROP (Dua et al., 2019), and the mathematics, physics, and chemistry subsets of MMLU-Pro (Wang et al., 2024b). Appendix B.1 details domains, sampling, and trajectory collection.

Candidate models and baselines. The candidate pool contains DeepSeek V4-Pro, DeepSeek V4-Flash, Qwen3.7 Plus, Qwen3.5-9B, Meta Llama-3.3-70B-Instruct, and OpenAI GPT-4o-mini, spanning a range of capability and price. We compare against (i) the strongest fixed model, DeepSeek V4-Pro; (ii) the single-turn routers Weighted kNN (provided by ORBIT, Lai et al., 2026), RouterDC (Chen et al., 2024b), EmbedLLM (Zhuang et al., 2025), and Avengers-Pro (Zhang et al., 2025b), which score query–model compatibility from pre-call information without modeling long-horizon consequences; and (iii) the multi-turn routers Random Router and MTRouter (Zhang et al., 2026b), our closest baseline, which selects models per turn under a cost-aware objective.

Evaluation metrics and protocol. We report accuracy on MINT and MMLU-Pro and exact match on MuSiQue and DROP; model-call cost is measured over the full evaluation set relative to each table's reference policy. Hyperparameters and checkpoints are selected on the validation split, and deployment uses RVE alone. Appendix B records optimization and stopping settings.

5.2 ACCURACY–COST PERFORMANCE AND OOD GENERALIZATION

Table 1 reports the main comparison. On MINT, RIVET improves accuracy over MTRouter while reducing model-call cost by 74.7%, and over the strongest fixed model while reducing cost by 80.1%: stronger task performance does not require allocating expensive models uniformly across the trajectory. RIVET thus occupies a different accuracy–cost regime from both fixed-model and single-turn baselines: **it reaches the highest accuracy among routers at a cost comparable to the cheapest of them**, reflecting a better allocation of capability across interaction states rather than a globally cheaper model.

Table 1: In-distribution and OOD performance. MINT Test and MMLU-Pro report accuracy; MuSiQue and DROP report exact match. Cost is total evaluation API cost relative to DeepSeek V4-Pro (1.000) on the same dataset; annotations give score changes in points and relative cost changes from it. Bold marks the best router per column.

Type	Method	MINT Test		MuSiQue		DROP		MMLU-Pro	
		ACC ↑	Cost ↓	EM ↑	Cost ↓	EM ↑	Cost ↓	ACC ↑	Cost ↓
Model	DeepSeek V4-Pro	84.40	1.000	55.56	1.000	71.88	1.000	90.56	1.000
Single-turn router	RouterDC	79.43 $_{\downarrow 4.97}$	0.334 $_{\downarrow 66.6\%}$	50.56 $_{\downarrow 5.00}$	0.202 $_{\downarrow 79.8\%}$	70.00 $_{\downarrow 1.88}$	0.224 $_{\downarrow 77.6\%}$	87.22 $_{\downarrow 3.34}$	0.211 $_{\downarrow 78.9\%}$
	EmbedLLM	82.98 $_{\downarrow 1.42}$	0.220 $_{\downarrow 78.0\%}$	50.00 $_{\downarrow 5.56}$	0.588 $_{\downarrow 41.2\%}$	**71.25** $_{\downarrow 0.63}$	0.413 $_{\downarrow 58.7\%}$	89.94 $_{\downarrow 0.62}$	1.358 $_{\uparrow 35.8\%}$
	Avengers-Pro	85.82 $_{\uparrow 1.42}$	**0.133** $_{\downarrow 86.7\%}$	49.44 $_{\downarrow 6.12}$	0.217 $_{\downarrow 78.3\%}$	70.63 $_{\downarrow 1.25}$	0.188 $_{\downarrow 81.2\%}$	87.78 $_{\downarrow 2.78}$	0.226 $_{\downarrow 77.4\%}$
	Weighted kNN	85.82 $_{\uparrow 1.42}$	1.370 $_{\uparrow 37.0\%}$	52.20 $_{\downarrow 3.36}$	1.388 $_{\uparrow 38.8\%}$	**71.25** $_{\downarrow 0.63}$	0.997 $_{\downarrow 0.3\%}$	**90.00** $_{\downarrow 0.56}$	1.646 $_{\uparrow 64.6\%}$
Multi-turn router	Random Router	78.72 $_{\downarrow 5.68}$	0.404 $_{\downarrow 59.6\%}$	46.10 $_{\downarrow 9.46}$	0.272 $_{\downarrow 72.8\%}$	68.75 $_{\downarrow 3.13}$	0.226 $_{\downarrow 77.4\%}$	84.44 $_{\downarrow 6.12}$	0.276 $_{\downarrow 72.4\%}$
	MTRouter	85.11 $_{\uparrow 0.71}$	0.788 $_{\downarrow 21.2\%}$	48.33 $_{\downarrow 7.23}$	1.259 $_{\uparrow 25.9\%}$	70.63 $_{\downarrow 1.25}$	0.987 $_{\downarrow 1.3\%}$	**90.00** $_{\downarrow 0.56}$	1.204 $_{\uparrow 20.4\%}$
	☆ RIVET Offline	85.82 $_{\uparrow 1.42}$	0.281 $_{\downarrow 71.9\%}$	**55.00** $_{\downarrow 0.56}$	0.407 $_{\downarrow 59.3\%}$	70.63 $_{\downarrow 1.25}$	0.171 $_{\downarrow 82.9\%}$	87.22 $_{\downarrow 3.34}$	0.177 $_{\downarrow 82.3\%}$
	☆ RIVET Online	**87.94** $_{\uparrow 3.54}$	0.199 $_{\downarrow 80.1\%}$	51.11 $_{\downarrow 4.45}$	**0.190** $_{\downarrow 81.0\%}$	**71.25** $_{\downarrow 0.63}$	**0.107** $_{\downarrow 89.3\%}$	**90.00** $_{\downarrow 0.56}$	**0.092** $_{\downarrow 90.8\%}$

The cost advantage persists under distribution shift. Across the three OOD benchmarks, RIVET Online has the lowest cost among routers and matches or exceeds MTRouter's task scores; on MMLU-Pro it matches MTRouter's accuracy at 92.4% lower cost. **The learned cost-allocation behavior is thus not specific to the MINT training distribution.** The one exception is MuSiQue, where online adaptation settles on a cheaper policy and gives up some exact match; the offline router remains the most accurate router there. Appendix C.1 analyzes representative cases.

5.3 Cost control and online adaptation

Cost-sensitive routing through λ. Figure 3 sweeps the cost coefficient. Removing the penalty ($\lambda = 0$) raises model-call cost to 3.2× the default and also lowers accuracy, so **the cost term does more than enforce a budget**: without it the router overuses expensive models in states where additional capability contributes little. The default $\lambda = 0.05$ gives the best operating point; increasing λ shifts the policy toward cheaper models, and at the most cost-sensitive setting cost falls by 39.2% relative to the default while accuracy returns to MTRouter's level at a fraction of its cost.

Replay-anchored online adaptation. The offline router already exceeds MTRouter's MINT accuracy at lower cost (Table 1), and limited online interaction refines it: the full budget raises accuracy from 85.82% to 87.94% while reducing test-time cost by 29.2% relative to the offline router. **The improvement is thus not purchased through more expensive routing**; logged trajectories supply most of the routing structure, and fresh trajectories correct the mismatch between the offline behavior distribution and the learned policy. Figure 4 tracks one run as online trajectories are added in batches of 35: accuracy rises from 83.69% after 35 to 87.94% after 140, while final cost returns close to its value at the first checkpoint.

5.4 What drives the improvement?

Table 2 tests whether the gain comes from response-informed auxiliary learning by removing RIVE, removing only its response input, and removing CQL; all configurations deploy RVE alone.

Table 2: Component ablations on MINT Test-141. All configurations deploy RVE alone; cost is relative to Ours.

Configuration	RIVE	Response input	CQL	ACC ↑	Cost ↓
Ours	✓	✓	✓	**87.94%**	**1.000**
- w/o CQL	✓	✓	×	85.82% ↓2.12	2.163 ↑116.4%
- w/o RIVE	×	×	✓	80.14% ↓7.80	1.172 ↑17.2%
- w/o response	✓	×	✓	80.14% ↓7.80	2.454 ↑145.4%

Check: enabled component or access to the executed response; red cross: ablated component or zeroed response input. w/o response keeps RIVE but feeds a zero vector in place of the response. RIVE-enabled configurations use $\beta = 1$; removing RIVE sets $\beta = 0$; other settings are fixed. Deltas report ACC changes in points and cost changes from Ours (unrounded costs).

Role of the response-informed auxiliary estimator. Removing RIVE lowers accuracy by 7.80 points and raises cost, so its training contributes to the deployed router. Keeping RIVE but zeroing its response input produces exactly the same accuracy drop and a larger cost increase. **The improvement therefore does not come from adding a second value head**: RIVE is useful because it observes the realized response. This matches the zero-response case in Appendix A.7: without the response, RIVE cannot explain Δ, its residual equals RVE's, and the joint objective supplies no additional estimating signal. Appendix C.2 compares routing decisions of RIVET and this ablation on selected cases.

Role of conservative regularization. Removing CQL raises model-call cost by more than 100% and lowers accuracy by 2.12 points even with RIVE intact. The disproportionate cost increase indicates that **CQL mainly stabilizes how RVE ranks candidate actions weakly supported by logged replay**, preventing optimistic values for expensive models from dominating decisions.

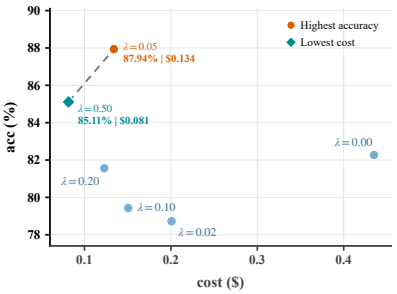


Figure 3: Cost-coefficient sensitivity on MINT Test-141. Highlighted settings achieve the highest accuracy ($\lambda = 0.05$) and lowest cost ($\lambda = 0.50$) among evaluated configurations. Cost denotes total test model-call cost in dollars.

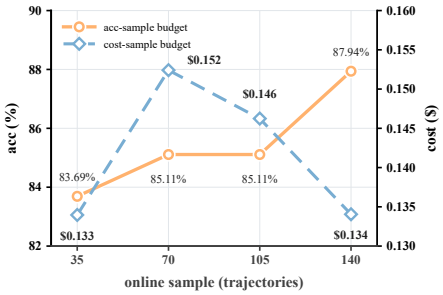


Figure 4: Online adaptation on MINT Test-141. Accuracy (left axis) and total test model-call cost in dollars (right axis) after cumulatively adding 35, 70, 105, and 140 online trajectories in batches within a single training run.

Table 3: Routing interventions on MINT evaluate retaining the first-turn model in episodes where RIVET originally switches and introducing a switch in episodes where it originally uses one model. Original: learned RIVET policy. Stay: retain the first-turn model throughout the episode. Switch: select the second-highest RVE-scoring model at turn two, then resume greedy routing. Cost is normalized to Original separately for each intervention.

Policy	ACC ↑	Cost ↓
Original	**84.75%**	1.000
Stay	72.88%	**0.825**
Original	82.61%	**1.000**
Switch	**84.78%**	1.429

5.5 Why turn-level adaptation matters

Table 3 examines both when to switch models and when to stay. On the 59 MINT episodes in which RIVET switches models, retaining the first-turn model throughout lowers cost by 17.5% but reduces successful episodes from 50 to 43; since both policies start from the same model, the difference isolates the later routing decisions. On 46 episodes in which RIVET keeps one model beyond the first turn, forcing the second-highest RVE-scoring model at turn two (greedy routing thereafter) produces 78 switches instead of zero but only one additional success, 38 to 39, at 42.9% higher cost. **Together, these results support switching models to improve task completion and retaining the current model when the tested alternative offers limited gains at higher cost.**

6 Conclusion

RIVET organizes multi-turn routing around when information becomes available: RVE makes every deployment decision from the pre-call state, while training-only RIVE conditions on the executed response and improves RVE only through their shared representation. Conditioning on the response removes response-induced return variation from the auxiliary regression problem, and this cleaner signal reduces the leading asymptotic estimation error of the deployed representation, using only responses and trajectories already recorded during interaction and no additional inference-time path. The current evaluation establishes a high-accuracy, lower-cost operating point and an end-to-end difference between the full and RVE-only configurations; cost-controlled response probes and matched transfer ablations remain necessary for a narrower causal attribution to response content.

Ethics Statement

This work studies routing among existing language models in multi-turn interaction, with the goal of reducing call cost while preserving task performance. Cost-effective routing lowers the compute and monetary cost of agentic systems and makes them accessible to a wider range of users. All experiments use publicly available benchmarks and models and involve no human subjects or personal data. Our method only selects which existing model to call and does not generate new content or alter model behavior, so it introduces no risks beyond those already present in the underlying models. We are not aware of any foreseeable negative societal impact of this work.

References

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically Mixing Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 131000–131034, 2024. doi: 10.52202/079017-4164.

Andrea Baisero and Christopher Amato. Unbiased asymmetric reinforcement learning under partial observability. arXiv preprint arXiv:2105.11674, 2021.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *Transactions on Machine Learning Research*, 2024a.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. RouterDC: Query-Based Router by Dual Contrastive Learning for Assembling Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 66305–66328, 2024b. doi: 10.52202/079017-2120.

Xiangxiang Dai, Jin Li, Xutong Liu, Anqi Yu, and John C. S. Lui. Cost-effective online multi-LLM selection with versatile reward models. arXiv preprint arXiv:2405.16587, 2024.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *The Twelfth International Conference on Learning Representations*, 2024.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1246.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A Graph-based Router for LLM Selections. arXiv preprint arXiv:2410.03834, 2024.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. GraphPlanner: Graph Memory-Augmented Agentic Routing for Multi-Agent LLMs. arXiv preprint arXiv:2604.23626, 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.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397, 2016.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020.

Guannan Lai, Haoran Hu, Hao-Xuan Ma, and Han-Jia Ye. ORBIT: An optimal routing and budgeted inference toolbox. *Frontiers of Computer Science*, 2026. doi: 10.1007/s11704-026-61310-5. Accepted manuscript.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. arXiv preprint arXiv:1511.03643, 2015.

Yasmin Moslem and John D. Kelleher. Dynamic model routing and cascading for efficient LLM inference: A survey. arXiv preprint arXiv:2603.04445, 2026. Accepted by Transactions on Machine Learning Research (2026); this entry identifies the arXiv version.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs from Preference Data. In *International Conference on Learning Representations*, 2025.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. arXiv preprint arXiv:1710.06542, 2017.

Cheng Qian, Zuxin Liu, Shirley Kokane, Akshara Prabhakar, Jielin Qiu, Haolin Chen, Zhiwei Liu, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. xRouter: Training cost-aware LLMs orchestration system via reinforcement learning. arXiv preprint arXiv:2510.08439, 2025.

Chenyang Shao, Xinyang Liu, Yutang Lin, Fengli Xu, and Yong Li. Route-and-reason: Scaling large language model reasoning with reinforced model router. arXiv preprint arXiv:2506.05901, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv preprint arXiv:2402.03300, 2024.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline RL for Natural Language Generation with Implicit Language Q Learning. arXiv preprint arXiv:2206.11871, 2022.

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.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-Hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094–2100, 2016. doi: 10.1609/aaai.v30i1.10295.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009. doi: 10.1016/j.neunet.2009.06.042.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in Multi-turn Interaction with Tools and Language Feedback. In *International Conference on Learning Representations*, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290, 2024b. doi: 10.52202/079017-3018.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-R1: Teaching LLMs Multi-Round Routing and Aggregation via Reinforcement Learning. arXiv preprint arXiv:2506.09033, 2025a.

Jiarui Zhang, Xiangyu Liu, Yong Hu, Chaoyue Niu, Hang Zeng, Shaojie Tang, Fan Wu, and Guihai Chen. From Myopic Selection to Long-Horizon Awareness: Sequential LLM Routing for Multi-Turn Dialogue. arXiv preprint arXiv:2604.12385, 2026a.

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, 2025b.

Yiqun Zhang, Hao Li, Zihan Wang, Shi Feng, Xiaocui Yang, Daling Wang, Bo Zhang, Lei Bai, and Shuyue Hu. MTRouter: Cost-Aware Multi-Turn LLM Routing with History–Model Joint Embeddings. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44206–44226. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.acl-long.2045.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 2024.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning Compact Representations of Large Language Models. In *International Conference on Learning Representations*, 2025.

A Theoretical Analysis: Proofs and Extensions

This appendix proves Proposition 4.1 and Theorem 4.2 and records five supporting facts that the main text invokes: the general form of Theorem 4.2 without the shared-signal and centering conditions (Appendix A.3), the structure of the response-resolved component Δ under an identity-blind environment (Appendix A.4), the behavior of the decomposition under bootstrapped targets (Appendix A.5), the quadratic regime of the Huber loss (Appendix A.6), and the effect of an imperfect response branch (Appendix A.7).

Setting. Fix a turn t and a continuation policy π that acts from turn $t+1$ onward. Write $\mathsf{R}_t = \sum_{j \geq t} \gamma^{j-t} r_j$ for the return from turn t, with $r_j = -\lambda c_j / c_{\text{ref}} + d_j y$. The regression target Y is G_t in the Monte Carlo stage ($\pi = \mu$) and the bootstrapped target Y_t of Equation 9 in the temporal-difference stage, where the local regression analysis holds the target network and selected continuation actions fixed. We write $X = (S_t, M_t)$, $O = O_t$, $Q^\star_{\text{route}}(X) = \mathbb{E}[Y \mid X]$, $Q^\star_{\text{resp}}(X, O) = \mathbb{E}[Y \mid X, O]$, related by the tower property

$$Q^\star_{\text{route}}(X) = \mathbb{E}_{O|X}\left[Q^\star_{\text{resp}}(X, O)\right], \tag{15}$$

and use Δ and ϵ as in Equation 12. Every statement below holds for any fixed π and any fixed target, so we omit the policy superscript. The superscript $\star$ on Q_{route}, Q_{resp}, and $U^\star$ marks a population conditional quantity; $m^\star$ and $V^\star$ denote a greedy action and its value. We use σ^2_{future} as shorthand for $\text{Var}(Y \mid X, O)$ when its arguments are clear. Bounded returns ensure that all conditional expectations and variances exist.

A.1 Proof of Proposition 4.1

Proof. By definition, $Y - Q^\star_{\text{route}}(X) = \Delta(X, O) + \epsilon$. Since Δ is measurable with respect to (X, O) and $\mathbb{E}[\epsilon \mid X, O] = 0$,

$$\mathbb{E}[\Delta\epsilon \mid X] = \mathbb{E}[\Delta\, \mathbb{E}[\epsilon \mid X, O] \mid X] = 0.$$

Expanding the square and using $\mathbb{E}[\Delta \mid X] = 0$,

$$\text{Var}(Y \mid X) = \mathbb{E}\left[(\Delta + \epsilon)^2 \mid X\right] = \text{Var}(\Delta \mid X) + \mathbb{E}[\epsilon^2 \mid X].$$

By the tower property, $\mathbb{E}[\epsilon^2 \mid X] = \mathbb{E}\left[\mathbb{E}[\epsilon^2 \mid X, O] \mid X\right] = \mathbb{E}[\text{Var}(Y \mid X, O) \mid X]$, which is Equation 13. The statements about the two estimators follow by reading the residual of RVE about $Q^\star_{\text{route}}$ as $\Delta + \epsilon$ and that of RIVE about $Q^\star_{\text{resp}}$ as ϵ. On a terminal turn, $Y = G_t = -\lambda c(s, m, o)/c_{\text{ref}} + y$; if the grader is a deterministic function of (s, o), then Y is constant given (X, O) and $\sigma^2_{\text{future}} = 0$. □

A.2 Proof of Theorem 4.2

The proof relies on the following local model of the shared representation.

Assumption A.1 (Local shared-signal model)**.** *(i)* Correct specification. $f_{\text{route}}(z_{\theta^\star}(X)) = Q^\star_{\text{route}}(X)$ *and* $f_{\text{resp}}(z_{\theta^\star}(X), e(O)) = Q^\star_{\text{resp}}(X, O)$ *at the population optimum of the heads. (ii)* Shared pre-call signal. *RIVE depends on* θ *only through the pre-call component:* $\nabla_\theta f_{\text{resp}}(z_{\theta^\star}(X), e(O)) = J(X)$. *(iii)* Centered response branch. *The head of RIVE is parameterized so that its response-dependent part is conditionally centered,* $\mathbb{E}[\nabla_{\psi_{\text{resp}}} f_{\text{resp}} \mid X] = 0$ *at the optimum. (iv)* $H \succ 0$*, and the standard regularity conditions for local asymptotic analysis of M-estimators hold.*

Condition (ii) says that $Q_{\text{resp}} = Q_{\text{route}} + h$ locally, where the response branch h absorbs Δ; condition (iii) is the corresponding normalization $\mathbb{E}[h \mid X] = 0$ and makes the shared and response-specific parameters asymptotically orthogonal. Appendix A.3 relaxes (ii) and (iii).

Proof. Write the per-sample squared losses as $\ell_{\text{route}}(\theta) = \frac{1}{2}\big(Y - f_{\text{route}}(z_\theta(X))\big)^2$ and $\ell_{\text{resp}}(\theta, \psi_{\text{resp}}) = \frac{1}{2}\big(Y - f_{\text{resp}}(z_\theta(X), e(O); \psi_{\text{resp}})\big)^2$, and let $\widehat{\mathcal{L}}_\beta$ be the empirical average of $\ell_{\text{route}} + \beta\ell_{\text{resp}}$ over n i.i.d. samples. Under Assumption A.1(i), the residuals at the population optimum are

$$Y - f_{\text{route}}(z_{\theta^\star}(X)) = \Delta + \epsilon, \qquad Y - f_{\text{resp}}(z_{\theta^\star}(X), e(O); \psi^\star_{\text{resp}}) = \epsilon.$$

Scores. Using Assumption A.1(ii), the per-sample gradients with respect to θ at the optimum are

$$g_{\text{route}} = -J(X)(\Delta+\epsilon), \qquad g_{\text{resp}} = -J(X)\,\epsilon, \qquad g_\beta = g_{\text{route}} + \beta g_{\text{resp}} = -J(X)\big[\Delta + (1+\beta)\epsilon\big].$$

Both scores have mean zero: $\mathbb{E}[J(X)\Delta] = \mathbb{E}[J(X)\mathbb{E}[\Delta \mid X]] = 0$ and $\mathbb{E}[J(X)\epsilon] = \mathbb{E}[J(X)\mathbb{E}[\epsilon \mid X, O]] = 0$. The same conditioning argument gives $\mathbb{E}[J(X)J(X)^\top\Delta\epsilon] = 0$, hence

$$\text{Var}(g_\beta) = \mathbb{E}\Big[JJ^\top\big(\Delta + (1+\beta)\epsilon\big)^2\Big] = \Sigma_\Delta + (1+\beta)^2\Sigma_\epsilon. \tag{16}$$

Hessian. The population Hessian of $\mathbb{E}[\ell_{\text{route}}]$ with respect to θ at $\theta^\star$ is $\mathbb{E}[JJ^\top] - \mathbb{E}[\nabla^2_\theta f_{\text{route}} \cdot (\Delta + \epsilon)] = H$, because $\mathbb{E}[\Delta + \epsilon \mid X] = 0$ removes the second term. The same argument with residual ϵ gives H for the θ-block of $\mathbb{E}[\ell_{\text{resp}}]$, so the θ-block of the joint Hessian is $H_\beta = (1+\beta)H$. The parameters ψ_{resp} of the response head are estimated jointly with θ, so the joint Hessian also contains the cross block $C = \mathbb{E}\big[J(X)\,\nabla_{\psi_{\text{resp}}} f_{\text{resp}}^\top\big]$. Under Assumption A.1(iii), $C = \mathbb{E}\big[J(X)\,\mathbb{E}[\nabla_{\psi_{\text{resp}}} f_{\text{resp}} \mid X]^\top\big] = 0$, so the joint Hessian is block diagonal in $(\theta, \psi_{\text{resp}})$ and the θ-block can be analyzed on its own.

Sandwich. Under Assumption A.1(iv), standard local asymptotics for M-estimators give

$$\text{Cov}(\hat\theta_\beta) = \frac{1}{n} H_\beta^{-1}\,\text{Var}(g_\beta) H_\beta^{-1} + o(n^{-1}).$$

Substituting $H_\beta = (1+\beta)H$ and Equation 16,

$$\begin{aligned}\text{Cov}(\hat\theta_\beta) &= \frac{1}{n}\,\frac{1}{(1+\beta)^2}\,H^{-1}\big[\Sigma_\Delta + (1+\beta)^2\Sigma_\epsilon\big]H^{-1} + o(n^{-1}) \\ &= \frac{1}{n}H^{-1}\left[\Sigma_\epsilon + \frac{1}{(1+\beta)^2}\Sigma_\Delta\right]H^{-1} + o(n^{-1}),\end{aligned}$$

which is Equation 14. Setting $\beta = 0$ gives the RVE-only covariance, and the difference is

$$\text{Cov}(\hat\theta_0) - \text{Cov}(\hat\theta_\beta) = \frac{1}{n}\left(1 - \frac{1}{(1+\beta)^2}\right)H^{-1}\Sigma_\Delta H^{-1} + o(n^{-1}) \;\succeq\; 0, \tag{17}$$

which is positive semidefinite because $\Sigma_\Delta \succeq 0$, and is positive definite on the range of Σ_Δ.

Excess risk. Since $\nabla_\theta \mathcal{L}_{\text{route}}(\theta^\star) = \mathbb{E}[g_{\text{route}}] = 0$ and $\nabla^2_\theta \mathcal{L}_{\text{route}}(\theta^\star) = H$, a second-order expansion gives

$$\mathcal{L}_{\text{route}}(\hat\theta_\beta) - \mathcal{L}_{\text{route}}(\theta^\star) = \tfrac{1}{2}(\hat\theta_\beta - \theta^\star)^\top H(\hat\theta_\beta - \theta^\star) + o_p(n^{-1}).$$

Taking expectations and using $\mathbb{E}[v^\top H v] = \mathrm{Tr}[H\,\mathrm{Cov}(v)]$ with Equation 14 yields

$$\mathbb{E}\Big[\mathcal{L}_{\mathrm{route}}(\hat{\theta}_\beta) - \mathcal{L}_{\mathrm{route}}(\theta^\star)\Big] = \frac{1}{2n}\,\mathrm{Tr}\left[H^{-1}\left(\Sigma_\epsilon + \frac{\Sigma_\Delta}{(1+\beta)^2}\right)\right] + o(n^{-1}), \tag{18}$$

which is nonincreasing in β because $H^{-1}\Sigma_\Delta$ has nonnegative trace. □

A.3 General form without the shared-signal and centering conditions

Assumption A.1(ii)–(iii) idealize the implemented architecture, in which RIVE has its own head and the two estimators share only the state–model encoder. Dropping both conditions changes the constants but not the direction of Theorem 4.2. Let $J_{\mathrm{route}}(X) = \nabla_\theta f_{\mathrm{route}}$ and $J_{\mathrm{resp}}(X,O) = \nabla_\theta f_{\mathrm{resp}}$ at the optimum, and define $H_{\mathrm{route}} = \mathbb{E}[J_{\mathrm{route}} J_{\mathrm{route}}^\top]$, $H_{\mathrm{resp}} = \mathbb{E}[J_{\mathrm{resp}} J_{\mathrm{resp}}^\top]$, $C = \mathbb{E}[J_{\mathrm{resp}} \nabla_{\psi_{\mathrm{resp}}} f_{\mathrm{resp}}^\top]$, $H_{\psi\psi} = \mathbb{E}[\nabla_{\psi_{\mathrm{resp}}} f_{\mathrm{resp}} \nabla_{\psi_{\mathrm{resp}}} f_{\mathrm{resp}}^\top]$, $\Sigma_\Delta^{\mathrm{route}} = \mathbb{E}[J_{\mathrm{route}} J_{\mathrm{route}}^\top \Delta^2]$, $\Sigma_\epsilon^{\mathrm{route}} = \mathbb{E}[J_{\mathrm{route}} J_{\mathrm{route}}^\top \epsilon^2]$, $\Sigma_\epsilon^{\mathrm{resp}} = \mathbb{E}[J_{\mathrm{resp}} J_{\mathrm{resp}}^\top \epsilon^2]$, and $\Sigma_\epsilon^{\mathrm{route,resp}} = \mathbb{E}[J_{\mathrm{route}} J_{\mathrm{resp}}^\top \epsilon^2]$. The joint score with respect to θ is $-J_{\mathrm{route}}(\Delta + \epsilon) - \beta J_{\mathrm{resp}}\epsilon$, and profiling out ψ_{resp} replaces the RIVE information by its Schur complement. The same sandwich argument then gives

$$\begin{aligned}\mathrm{Cov}(\hat{\theta}_\beta) &= \frac{1}{n}\,\tilde{H}_\beta^{-1}\big[\Sigma_\Delta^{\mathrm{route}} + \Sigma_\epsilon^{\mathrm{route}} + \beta\big(\Sigma_\epsilon^{\mathrm{route,resp}} + \Sigma_\epsilon^{\mathrm{route,resp}\,\top}\big) + \beta^2 \Sigma_\epsilon^{\mathrm{resp}}\big]\tilde{H}_\beta^{-1} + o(n^{-1}),\\ \tilde{H}_\beta &= H_{\mathrm{route}} + \beta\big(H_{\mathrm{resp}} - C H_{\psi\psi}^{-1} C^\top\big).\end{aligned} \tag{19}$$

The term $\Sigma_\Delta^{\mathrm{route}}$ enters without any β-dependent factor in the bracket and is divided by $\tilde{H}_\beta$ on both sides; since $\tilde{H}_\beta \succeq H_{\mathrm{route}}$ with equality only when RIVE carries no information about θ, the response-induced contribution is attenuated by a factor between $1/(1+\beta)^2$ and 1, with the clean factor recovered exactly when $J_{\mathrm{resp}} = J_{\mathrm{route}}$ and $C = 0$. RIVE therefore never increases the response-induced part of the estimation error, and the ϵ-part is reweighted rather than removed, matching the interpretation in the main text.

A.4 Structure of the response-resolved component

Theorem 4.2 makes the gain of RIVE proportional to Σ_Δ. The following separability result, which uses the specific structure of the routing environment, says what Δ is made of and hence when Σ_Δ is large. For this subsection, $Y = \mathsf{R}_t$ (and G_t is its realized trajectory sample); Appendix A.5 treats the bootstrapped target.

Assumption A.2 (Identity-blind environment)**.** *(i) Environment feedback, termination, and the task-success label depend on the executed response but not on the identity of the model that produced it: $P_{\mathrm{env}}(f_t \mid s_t, m_t, o_t) = P_{\mathrm{env}}(f_t \mid s_t, o_t)$ and $y = y(s_T, o_T)$. (ii) The call cost is a deterministic function of the call, $c(s,m,o) = p_m^{\mathrm{in}}|s| + p_m^{\mathrm{out}}|o|$, where p_m are the per-token prices of model m. (iii) The continuation policy reads only the next state s_{t+1}.*

All three conditions hold in our setting: MINT tools and graders consume response text only, API cost is price times token count, and the state serialization of Section 4.1 does not encode earlier model identities.

Proposition A.3 (Post-call separability)**.** *Under Assumption A.2, for every (s,m,o),*

$$Q^\star_{\mathrm{resp}}(s,m,o) = U^\star(s,o) - \lambda\,\frac{c(s,m,o)}{c_{\mathrm{ref}}}, \quad \textit{where} \quad U^\star(s,o) \triangleq \mathbb{E}[d_t y + \gamma(1-d_t)V(S_{t+1}) \mid s,o], \tag{20}$$

with $V = V^\pi$ the value of the continuation policy ($V^\star(s') = \max_{m'} Q^\star_{\mathrm{route}}(s',m')$ under the greedy continuation). Consequently,

$$\Delta(s,m,o) = \Big(U^\star(s,o) - \mathbb{E}_{o'\sim p_m(\cdot|s)}[U^\star(s,o')]\Big) - \lambda\,\frac{c(s,m,o) - \mathbb{E}_{o'\sim p_m(\cdot|s)}[c(s,m,o')]}{c_{\mathrm{ref}}}. \tag{21}$$

Proof. Write the return recursively as $\mathsf{R}_t = r_t + \gamma(1-d_t)\mathsf{R}_{t+1}$ and condition on $(S_t, M_t, O_t) = (s,m,o)$. By Assumption A.2(ii) the cost term $-\lambda c(s,m,o)/c_{\mathrm{ref}}$ is a constant given the conditioning event. By Assumption A.2(i), the joint conditional law of (d_t, y, f_t) given (s,m,o)

equals its law given (s, o). The next state is the deterministic function $S_{t+1} = (q, \mathcal{H}_{<t} \oplus (o, f_t))$ of (s, o, f_t), so its conditional law also depends on (s, o) only. By Assumption A.2(iii) the continuation policy reads S_{t+1} alone, hence $\mathbb{E}[\mathsf{R}_{t+1} \mid S_{t+1}, S_t, M_t, O_t] = \mathbb{E}[\mathsf{R}_{t+1} \mid S_{t+1}] = V(S_{t+1})$. Taking conditional expectations gives Equation 20. Equation 21 follows by subtracting the tower identity $Q^\star_{\text{route}}(s, m) = \mathbb{E}_{o' \sim p_m(\cdot|s)}[Q^\star_{\text{resp}}(s, m, o')]$ (Equation 15). □

Remark A.4 (Where RIVE helps). Equation 21 splits Δ into a progress term, the deviation of the response-conditioned progress $U^\star(s, o)$ from its mean over the model's response distribution, and a cost term, the deviation of the realized token cost from its mean. With $\lambda = 0.05$ and normalized costs of order one, the cost term is small, so Σ_Δ is dominated by states in which different responses of the same model lead to different downstream progress. These are precisely the situations that make single-turn outcome labels ambiguous: when a strong model fails on an unfavorable response realization, RVE alone attributes the failure to the model representation $z_\theta(s, m)$, whereas RIVE attributes it to the response content and leaves the shared representation to encode the model's typical behavior. When both a cheap and an expensive model succeed, Δ is nearly zero for both and the two estimators agree.

A.5 Bootstrapped targets

In the temporal-difference stage both estimators regress to $Y_t = r_t + \gamma(1 - d_t)Q_{\text{route}}(s_{t+1}, m^\star_{t+1}; \bar{\omega})$ with a fixed target network. Conditional on (s, m, o), the only remaining randomness in Y_t is the environment feedback f_t that determines s_{t+1}, so

$$\text{Var}(Y_t \mid s, m, o) = \gamma^2(1 - d_t)^2 \ \text{Var}_{f_t \sim P_{\text{env}}(\cdot|s,o)}\left[Q_{\text{route}}(s_{t+1}, m^\star_{t+1}; \bar{\omega})\right],$$

which vanishes on terminal turns and whenever the tool feedback is a deterministic function of (s, o). In this stage, therefore, σ^2_{future} is small or zero, and almost all of the target variance $\text{Var}(Y_t \mid s, m)$ is response-resolved: Δ contains the variation of the bootstrapped next-state value across the responses the model could have produced. The decomposition of Proposition 4.1 and the covariance reduction of Theorem 4.2 apply verbatim with $Y = Y_t$, and the bootstrapped stage is the regime in which the response-resolved share of the target variance, and hence the predicted benefit of RIVE, is largest. The reduction concerns the estimation of θ for a fixed target; it does not by itself bound the error introduced by bootstrapping from an imperfect target network, which is the usual caveat of fitted value iteration and is outside the scope of the local analysis.

A.6 Quadratic regime of the Huber loss

The implemented objective uses the Huber loss ℓ_δ with $\delta = 1$, whose minimizer coincides with the conditional mean only when residuals stay within $[-\delta, \delta]$, where $\ell_\delta(u) = u^2/2$. All regression targets in Section 4 are returns of the form $-\lambda \sum_j \gamma^{j-t} c_j / c_{\text{ref}} + \gamma^{T-t} y$ (Monte Carlo) or $r_t + \gamma(1 - d_t)Q_{\text{route}}(s_{t+1}, m^\star_{t+1}; \bar{\omega})$ (bootstrapped). With $\lambda = 0.05$, $y \in \{0, 1\}$, and at most ten turns whose normalized costs are of order one, the cost contribution is bounded by roughly 0.5 in magnitude, so every target lies in an interval of width below δ, and a predictor initialized inside that interval never leaves the quadratic regime. The squared-loss analysis of Theorem 4.2 therefore describes the implemented loss without modification; Huber is retained only as protection against occasional out-of-range predictions early in training.

A.7 Imperfect response modeling and the role of β

Assumption A.1(i) requires the response branch of RIVE to represent Δ exactly. If instead the best achievable response branch leaves an approximation residual $\rho(X, O) = \Delta(X, O) - h^\star(X, O)$ with $\mathbb{E}[\rho \mid X] = 0$, then the RIVE residual at the optimum is $\rho + \epsilon$ rather than ϵ, and the joint score becomes $-J\left[\Delta + \beta\rho + (1 + \beta)\epsilon\right]$. Repeating the proof of Theorem 4.2,

$$\text{Cov}(\hat{\theta}_\beta) = \frac{1}{n} H^{-1} \left[\Sigma_\epsilon + \frac{\mathbb{E}\left[JJ^\top(\Delta + \beta\rho)^2\right]}{(1 + \beta)^2}\right] H^{-1} + o(n^{-1}).$$

When $\rho = 0$ this is Equation 14. When $\rho \neq 0$ the response-induced term is no longer divided cleanly by $(1 + \beta)^2$: the part of Δ that RIVE explains, $h^\star = \Delta - \rho$, is attenuated, while the unexplained

part ρ is reintroduced with weight β. The gain therefore shrinks with the fraction of Σ_Δ that the response branch can represent, and for a sufficiently poor response branch a large β can hurt. This is why we treat β as a tuned hyperparameter rather than fixing it, and why the mechanism analysis in Section 5.4 includes a variant whose response input is zeroed: that variant has $h \equiv 0$, so $\rho = \Delta$ and Theorem 4.2 predicts no benefit over RVE-only training.

B Additional experimental details and results

B.1 Datasets, trajectory collection, and manifests

Table 4 reports the domain and sampled-to-source counts for each benchmark composition. We use the MINT Reasoning subset, MuSiQue-Ans dev, DROP validation, and MMLU-Pro test splits. The three sampled MMLU-Pro subjects contain 3,782 examples within the full 12,032-example test split.

Table 4: Dataset domains and sampling distributions. The final column gives the sampled count and source-pool size within each composition.

Dataset	Domain	Composition	Sample / Source size
MINT Test	Multi-task reasoning	GSM8K	21 / 48
		HotpotQA	19 / 43
		MATH	45 / 100
		MMLU	34 / 76
		TheoremQA	22 / 49
MuSiQue	Multi-hop QA	2-hop	60 / 1,252
		3-hop	60 / 760
		4-hop	60 / 405
DROP	Discrete text reasoning	Date	40 / 157
		Number	40 / 5,850
		Single-span	40 / 2,962
		Multi-span	40 / 567
MMLU-Pro	STEM reasoning	Mathematics	60 / 1,351
		Physics	60 / 1,299
		Chemistry	60 / 1,132

The MINT manifests contain 200 training, 35 validation, and 141 test tasks, with at most ten turns per episode. Six fixed-model policies, one per candidate model, and a Random Router each execute the 200 training tasks, yielding 1,400 offline trajectories and 7,492 transitions. Online adaptation uses the 140 reasoning tasks within the same training manifest and collects at most 140 additional trajectories. RIVET uses random seed 42, and the trace-level analyses in Section 5.5 and Appendix B.5 use the corresponding canonical run.

B.2 Optimization and stopping protocol

Except where noted, RIVET and its variants use $\gamma = 0.99$, $\lambda = 0.05$, $\beta = 1$, and $\alpha = 0.1$. Offline optimization runs for five Monte Carlo epochs followed by 20 temporal-difference epochs. During online adaptation, we sample offline and newly collected replay in equal proportions and anneal episode-indexed ϵ from 0.10 to 0.02. We validate every 35 trajectories. After 140 trajectories, early stopping uses patience two and caps adaptation at three passes through the task pool.

The Huber threshold is $\delta = 1$. AdamW uses batch size 64, weight decay 0.01, and gradient clipping at 5. Offline training uses learning rate 3×10^{-4}; online adaptation uses 3×10^{-5} and performs one gradient update per newly added transition. We synchronize the target network every 200 optimization steps.

For the online sample study, we add trajectories in batches of 35 within a single adaptation run. Figure 4 reports test accuracy and total test model-call cost at cumulative sample counts of 35, 70, 105, and 140 trajectories. Each point evaluates the corresponding router checkpoint on the same

MINT Test-141 set, tracing the progression of one training run as online data accumulate. In the cost-coefficient study, validation selects the $\lambda = 0$ checkpoint after 70 trajectories.

B.3 State, response, and model-card encodings

A frozen Qwen3-Embedding-0.6B encoder maps the serialized state s_t to $h_t \in \mathbb{R}^{1024}$. The serialization retains the task, the two latest complete turns, and earlier turns up to 16,384 tokens; it omits the identities of models used on earlier turns. Each candidate model has a frozen text embedding e_m, normalized structured fields x_m for context length, capabilities, and prices, and a missing-value mask. A trainable model-card projector produces

$$z_m = g_\theta(e_m, x_m) \in \mathbb{R}^{128}, \tag{22}$$

and a trainable state adapter fuses h_t with z_m into the shared state–model representation $z_\theta(s_t, m)$ of Equation 3. RVE computes $Q_{\text{route}}(s_t, m)$ from $z_\theta(s_t, m)$ for every candidate. RIVE passes the executed response o_t through the frozen encoder and a trainable response projector, then computes $Q_{\text{resp}}(s_t, m_t, o_t)$ from $z_\theta(s_t, m_t)$ and the response representation for the selected model. The two value heads have separate parameters; the adapter and the projector are shared.

B.4 Robustness to candidate-model unavailability

Table 5 tests routing stability under two changes to the candidate model pool. We remove DeepSeek V4-Pro to test the loss of a strong, high-cost option and Qwen3.5-9B to test the loss of a weaker, low-cost option. Without Qwen3.5-9B, RIVET matches MTRouter's 85.82% accuracy while reducing cost by 74.4%. Without DeepSeek V4-Pro, RIVET achieves 83.69% accuracy at 72.3% lower cost than MTRouter, with a 0.71-point accuracy gap. Across both conditions, RIVET preserves a favorable accuracy–cost operating point after the candidate pool loses either model.

Table 5: Routing stability under candidate-model unavailability on MINT Test-141. Each model pool removes the indicated model and uses RIVET Online as the reference. Colored annotations show accuracy changes from RIVET Online in percentage points and relative cost changes from RIVET Online, whose cost is set to 1.000. Bold values denote the best router result in each model pool.

Model pool	**Type**	**Method**	**MINT Test**	
			ACC ↑	**Cost ↓**
w/o Qwen3.5-9B	Single-turn router	RouterDC	80.85 ↓4.97	2.527 ↑152.7%
		EmbedLLM	79.43 ↓6.39	**0.820** ↓18.0%
		Avengers-Pro	84.40 ↓1.42	2.887 ↑188.7%
		Weighted kNN	84.40 ↓1.42	3.002 ↑200.2%
	Multi-turn router	MTRouter	**85.82** – 0%	3.906 ↑290.6%
		☆ RIVET Online	**85.82**	1.000
w/o DeepSeek V4-Pro	Single-turn router	RouterDC	80.14 ↓3.55	2.527 ↑152.7%
		EmbedLLM	**84.40** ↑0.71	1.029 ↑2.9%
		Avengers-Pro	83.69 – 0%	4.000 ↑300.0%
		Weighted kNN	82.97 ↓0.72	3.413 ↑241.3%
	Multi-turn router	MTRouter	**84.40** ↑0.71	3.610 ↑261.0%
		☆ RIVET Online	83.69	**1.000**

B.5 ROUTING AFTER EXECUTION ERRORS

Table 6 defines the error classes used in the post-error analysis. The classifier matches error names and response strings recorded in the interaction traces.

Table 6: Error-event categories used in the post-error routing analysis.

Category	Matched error signatures
Non-executable action	UnsafeCodeError; Invalid action; Illegal action; Unknown action; Action not allowed
Formatting error	SyntaxError; JSONDecodeError; Parse Error; FormatError; Malformed action
Non-network tool failure	TypeError; NameError; AttributeError; local-tool TimeoutError messages

The unit of analysis is an execution-error event at turn t. Error-event rate divides classified error events by all recorded turns, including execution, solution, and intermediate turns. Switch rate divides events for which the model at $t+1$ differs from that at t by error events with an observable next turn. We exclude final-turn errors from this denominator; none occur in these traces. Average switch cost is the mean cost of the changed-model call at $t+1$. Next-turn recovery requires that this call produces a valid execution or solution action without another classified error.

Across pooled canonical traces from MINT, MuSiQue, DROP, and MMLU-Pro, RIVET records an error-event rate of 3.96%, compared with 8.48% for MTRouter. Following an error event, it switches models in 26.3% of cases, compared with 4.2% for MTRouter, and spends 23.9% less on the subsequent switched call. The changed-model call recovers on the next turn in 93.3% and 100.0% of cases, respectively. These traces show that RIVET combines fewer recorded execution errors with more frequent model changes after an error. The freezing intervention in Section 5.5 separately tests the effect of preventing model changes.

Table 7: Post-error routing behavior pooled across MINT Test, MuSiQue, DROP, and MMLU-Pro canonical traces. The MTRouter switch cost is 1.000.

Method	Error-event rate	Switch rate	Switch cost ↓	Next-turn recovery
RIVET Online	**3.96%**	**26.3%**	**0.761**	93.3%
MTRouter	8.48%	4.2%	1.000	**100.0%**

B.6 STAGE-WISE MODEL-SELECTION PREFERENCES

We further examine whether each router changes its model-selection preference across the semantic progression of an interaction. Rather than dividing an episode by absolute turn index, we classify each recorded turn as *Analyze*, *Solve*, or *Submit* from its action and response content. Table 8 gives the classification rules. Explicit submission markers define Submit turns; among the remaining turns, tool use or substantive task-processing content defines Solve turns, while planning and pre-solution intermediate output define Analyze turns.

Table 8: Turn-stage definitions used in the stage-wise routing analysis.

Stage	**Classification rule**
Analyze (outer ring)	Planning; task interpretation; task decomposition; blank pre-solution output (no substantive computation or result).
Solve (middle ring)	Tool calls; code execution; tool feedback; calculations; evidence; intermediate conclusions; verification.
Submit (inner ring)	Solution action; submitted solution field; forced final answer; solution tag.

Figure 5 shows the resulting conditional model-selection distributions. The two RIVET variants select Qwen3.5-9B most often during Analyze and Solve turns, whereas DeepSeek V4-Flash becomes their most frequent choice during Submit turns. This shift indicates that the learned policies allocate models according to the semantic role of a turn, rather than following one fixed model mixture throughout an episode.

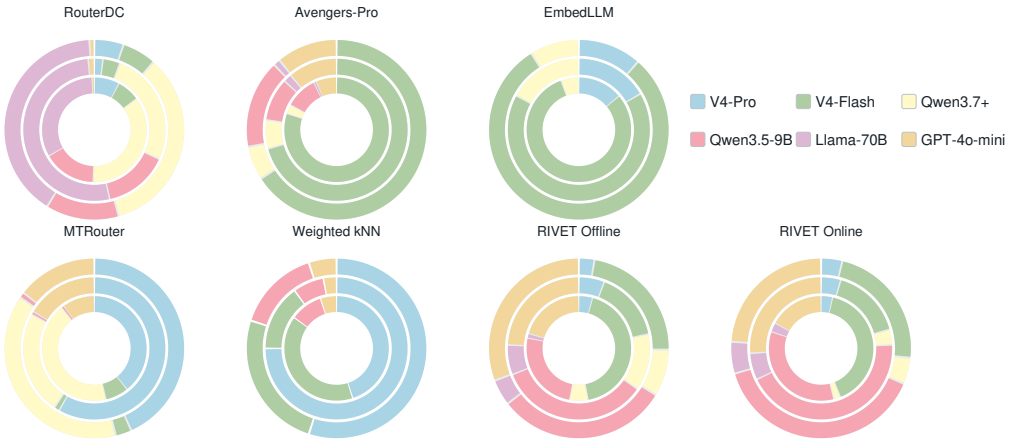


Figure 5: Stage-wise model-selection preferences for seven routers on MINT Test-141. Each panel shows the distribution of selected models within Analyze (outer ring), Solve (middle ring), and Submit (inner ring) turns; each ring is normalized independently to 100%. Segment colors denote candidate models.

C CASE STUDIES

Across all cases, we condense tasks and actions, quote recorded responses and tool outputs verbatim, and report total episode costs in USD to six decimal places.

C.1 COMPARISON WITH OTHER ROUTERS

The cases illustrate how RIVET combines correct task completion with lower episode cost. In Example 3, RIVET submits the correct answer after two turns, whereas RouterDC obtains the correct intermediate result but then introduces an unnecessary unit conversion and returns an incorrect answer. In Example 2, both RIVET and EmbedLLM answer correctly in two turns, but RIVET uses DeepSeek V4-Flash instead of DeepSeek V4-Pro, completing the task at lower cost.

The recovery trajectories also show how routers proceed after an execution error. Table 7 in Appendix B.5 reports post-error switch rates of 26.3% for RIVET and 4.2% for MTRouter. In Example 6, both routers initially select DeepSeek V4-Pro and encounter the same disallowed-import error. From identical histories, RIVET switches to Qwen3.5-9B and answers correctly from the supplied passages at turn two. MTRouter retains DeepSeek V4-Pro, corrects the tool call, and continues searching until it submits the same answer at turn eight. Both resolve the execution error on the next turn, but RIVET completes the task earlier at lower episode cost.

Example 1. Identifying the drummer of Maroon 5

Task. Who is the current drummer of the band who did the song "What Lovers Do"?

Turn-by-turn model choices and actions

Turn	RIVET	MTRouter
1	**OpenAI GPT-4o-mini** Plans to identify the band.	**OpenAI GPT-4o-mini** Plans to identify the band.
2	**OpenAI GPT-4o-mini** Searches for the song.	**OpenAI GPT-4o-mini** Searches for the song.
3	**OpenAI GPT-4o-mini** Identifies Maroon 5 and plans a drummer search.	**OpenAI GPT-4o-mini** Identifies Maroon 5 and plans a drummer search.
4	**OpenAI GPT-4o-mini** Searches for the drummer; the result names Matt Flynn.	**OpenAI GPT-4o-mini** Searches for the drummer; the result names Matt Flynn.
5	**OpenAI GPT-4o-mini** States Matt Flynn and prepares to submit.	**DeepSeek V4-Pro** Performs an additional verification search.
6	**OpenAI GPT-4o-mini** Submits Matt Flynn.	**OpenAI GPT-4o-mini** Submits Matt Flynn.

Selected responses and tool outputs

T4, RIVET / OpenAI GPT-4o-mini (tool output excerpt):

drummer Matt Flynn

T5, RIVET / OpenAI GPT-4o-mini (response excerpt):

The current drummer of Maroon 5 is Matt Flynn. I will now submit the final answer.

T5, MTRouter / DeepSeek V4-Pro (response excerpt):

The Wikipedia summary states that Maroon 5's drummer is Matt Flynn. I will verify this with a more specific search to ensure there hasn't been a recent change.

Final answers

RIVET: Matt Flynn. **Correct.** 6 turns; Cost: $0.000639.

MTRouter: Matt Flynn. **Correct.** 6 turns; Cost: $0.002832.

Task hotpotqa:33.

Example 2. The final three digits of a power

Task. What is the sum of the final three digits of the integer representation of 5^{100}?

Turn-by-turn model choices and actions

Turn	RIVET	EmbedLLM
1	**DeepSeek V4-Flash** Computes the last three digits (625) and their sum (13).	**DeepSeek V4-Pro** Computes the last three digits (625) and their sum (13).
2	**DeepSeek V4-Flash** Submits 13.	**DeepSeek V4-Pro** Submits 13.

Selected responses and tool outputs

T1, RIVET / DeepSeek V4-Flash (response excerpt):

print(sum(int(d) for d in str(pow(5, 100, 1000))))

T1, EmbedLLM / DeepSeek V4-Pro (response excerpt):

mod = pow(5, 100, 1000)

T2, RIVET / DeepSeek V4-Flash (response excerpt):

<solution>13</solution>

T2, EmbedLLM / DeepSeek V4-Pro (response excerpt):

<solution>13</solution>

Final answers

RIVET: 13. **Correct.** 2 turns; Cost: $0.000113.

EmbedLLM: 13. **Correct.** 2 turns; Cost: $0.002088.

Task math:73.

Example 3. Buying snowshoes for six sled dogs

Task. Mario has 6 sled dogs, each with four legs. A pair of snowshoes costs $12. How much does it cost to buy snowshoes for all his dogs? Return an integer.

Turn-by-turn model choices and actions

Turn	RIVET	RouterDC
1	**Qwen3.5-9B** Computes 12 pairs at 12 dollars per pair; tool output: 144.	**Meta Llama-3.3-70B-Instruct** Plans to calculate the required number of pairs.
2	**DeepSeek V4-Flash** Submits 144.	**Meta Llama-3.3-70B-Instruct** Computes the total; tool output: 144.0.
3	Episode ended	**Meta Llama-3.3-70B-Instruct** Converts dollars to cents; tool output: 14400.
4	Episode ended	**Meta Llama-3.3-70B-Instruct** Submits 14400.

Selected responses and tool outputs

T1, RIVET / Qwen3.5-9B (response excerpt):

total_cost = num_pairs * cost_per_pair

T3, RouterDC / Meta Llama-3.3-70B-Instruct (response excerpt):

the output should be in cents to avoid losing precision due to the decimal point.

T3, RouterDC / Meta Llama-3.3-70B-Instruct (response excerpt):

total_cost_cents = int(total_cost_dollars * 100)

T2, RIVET / DeepSeek V4-Flash (response excerpt):

<solution>144</solution>

Final answers

RIVET: 144. **Correct.** 2 turns; Cost: $0.000179.

RouterDC: 14400. **Incorrect.** 4 turns; Cost: $0.000403.

Task gsm8k:17.

Example 4. Area defined by floor functions

Task. What is the area of the region in the xy-plane satisfying $\lfloor x \rfloor \lfloor y \rfloor = 16$?

Turn-by-turn model choices and actions

Turn	RIVET	RouterDC
1	**DeepSeek V4-Flash** Submits 10.	**Meta Llama-3.3-70B-Instruct** Lists ten unit squares, then writes code that counts each positive ordered pair four times; makes no tool call.
2	Episode ended	**Meta Llama-3.3-70B-Instruct** Revises the code to count 8, 8, and 4 squares for the three positive unordered pairs; makes no tool call.
3	Episode ended	**Meta Llama-3.3-70B-Instruct** Repeats the code in a Markdown block; makes no tool call.
4	Episode ended	**Meta Llama-3.3-70B-Instruct** Claims a code output of 24 and submits 24; the log contains no tool execution.

Selected responses and tool outputs

T1, RIVET / DeepSeek V4-Flash (response excerpt):

<solution>10</solution>

T1, RouterDC / Meta Llama-3.3-70B-Instruct (response excerpt):

The area of each unit square is 1. Since we have 10 such pairs (considering both positive and negative factors), we might initially think to calculate the area based on these pairs.

T4, RouterDC / Meta Llama-3.3-70B-Instruct (response excerpt):

We've executed the code and now have the total area.

Final answers

RIVET: 10. **Correct.** 1 turn; Cost: $0.000126.

RouterDC: 24. **Incorrect.** 4 turns; Cost: $0.001318.

Task math:0.

Example 5. Expected number of die rolls

Task. Each morning Bob rolls a fair six-sided die, rerolling whenever he gets 1 and stopping on any other number. What is the expected total number of rolls in a non-leap year?

Turn-by-turn model choices and actions

Turn	RIVET	Avengers-Pro
1	**Qwen3.5-9B** Computes the expected annual total; tool output: 438.0.	**Qwen3.5-9B** Computes the expected annual total; tool output: 438.0.
2	**OpenAI GPT-4o-mini** Switches to OpenAI GPT-4o-mini and submits 438.	**Qwen3.5-9B** Keeps Qwen3.5-9B and submits 438 with a derivation.

Selected responses and tool outputs

T1, RIVET / Qwen3.5-9B (tool output excerpt):

Expected rolls per day: 1.2
Total expected rolls in 365 days: 438.0

T1, Avengers-Pro / Qwen3.5-9B (tool output excerpt):

Expected rolls per day: 1.2
Total expected rolls in 365 days: 438.0

T2, RIVET / OpenAI GPT-4o-mini (response excerpt):

<solution>438</solution>

T2, Avengers-Pro / Qwen3.5-9B (response excerpt):

Expected rolls per day = 1/p = 6/5 = 1.2

Final answers

RIVET: 438. **Correct.** 2 turns; Cost: $0.000283.

Avengers-Pro: D) 438. **Correct.** 2 turns; Cost: $0.000437.

Task mmlu:22.

Example 6. Answering after an invalid tool invocation

Task. What is the average yearly income for a person of the same nationality as the creator of MacGruder and Loud?

Turn-by-turn model choices and actions

Turn	RIVET	MTRouter
1	**DeepSeek V4-Pro** Imports wikipedia before calling the search function; the tool rejects the import.	**DeepSeek V4-Pro** Imports wikipedia before calling the search function; the tool rejects the import.
2	**Qwen3.5-9B** Switches to Qwen3.5-9B and submits the answer from the supplied paragraphs.	**DeepSeek V4-Pro** Removes the import and searches for MacGruder and Loud.
3	Episode ended	**DeepSeek V4-Pro** Searches for the creator of MacGruder and Loud.
4	Episode ended	**DeepSeek V4-Pro** Searches again for the television series creator.
5	Episode ended	**DeepSeek V4-Pro** Searches for the television series.
6	Episode ended	**DeepSeek V4-Pro** Searches for the cast.
7	Episode ended	**DeepSeek V4-Pro** Searches for MacGruder and Loud and Aaron Spelling.
8	Episode ended	**DeepSeek V4-Pro** Submits the answer with supporting paragraphs.

Selected responses and tool outputs

T1, RIVET / DeepSeek V4-Pro (response excerpt):

import wikipedia
print(wikipedia_search("MacGruder and Loud"))

T1, RIVET / DeepSeek V4-Pro (tool output excerpt):

UnsafeCodeError: Import is not allowed: wikipedia

T2, RIVET / Qwen3.5-9B (response excerpt):

Answer: $72,641
Support: 4, 8, 15

T2, MTRouter / DeepSeek V4-Pro (response excerpt):

print(wikipedia_search("MacGruder and Loud"))

T8, MTRouter / DeepSeek V4-Pro (response excerpt):

Answer: $72,641
Support: 15,8,4

Final answers

RIVET: $72,641. **Correct.** 2 turns; Cost: $0.003489.

MTRouter: $72,641. **Correct.** 8 turns; Cost: $0.015209.

Task musique_ans:3hop1__694534_160088_73916.

C.2 Comparison with response-ablated training

These cases illustrate how response-informed training affects accuracy and cost through the deployed routing policy. In Example 9, both variants answer correctly in one turn, but RIVET selects DeepSeek V4-Flash and RIVE w/o response selects DeepSeek V4-Pro. In this case, RIVET incurs 95.9% lower episode cost than RIVE w/o response.

Example 10 provides the clearest accuracy contrast. RIVET selects Qwen3.5-9B and returns the correct acceleration, -120, in one turn. RIVE w/o response selects OpenAI GPT-4o-mini, repeats its differentiation plan through turn nine, and submits -240 at turn ten. The ablated variant is incorrect and costs $15.6\times$ as much. Together, these cases illustrate how response-informed training guides model allocation across both accuracy and cost.

Example 7. Sum of proper divisors

Task. The sum of the proper divisors of 18 is 21. What is the sum of the proper divisors of 198?

Turn-by-turn model choices and actions

Turn	RIVET	RIVE w/o response
1	**DeepSeek V4-Flash** Runs a divisor-sum function; tool output: 270.	**DeepSeek V4-Flash** Runs the identical function; tool output: 270.
2	**Qwen3.5-9B** Reruns the function; tool output: 270.	**Qwen3.5-9B** Reruns the identical function; tool output: 270.
3	**DeepSeek V4-Flash** Switches to DeepSeek V4-Flash and submits 270.	**Qwen3.5-9B** Keeps Qwen3.5-9B and recomputes; output: 270.
4	Episode ended	**Qwen3.5-9B** Repeats the computation; output: 270.
5	Episode ended	**Qwen3.5-9B** Changes the function and subtracts 198; output: 72.
6	Episode ended	**Qwen3.5-9B** Restores the earlier function; output: 270.
7	Episode ended	**Qwen3.5-9B** Explicitly sums the proper divisors; output: 270.
8	Episode ended	**Qwen3.5-9B** Includes verification code and submits 270.

Selected responses and tool outputs

T1, RIVET / DeepSeek V4-Flash (response excerpt):

```
for i in range(1, n):
if n % i == 0:
total += i
```

T3, RIVET / DeepSeek V4-Flash (response excerpt):

<solution>270</solution>

T5, RIVE w/o response / Qwen3.5-9B (response excerpt):

```
return total - n # subtract n itself to get proper divisors only
```

T5, RIVE w/o response / Qwen3.5-9B (tool output excerpt):

```
72
None
```

Final answers

RIVET: 270. **Correct.** 3 turns; Cost: $0.000230.

RIVE w/o response: 270. **Correct.** 8 turns; Cost: $0.001073.

Task math:84.

Example 8. Evaluating a defined operation

Task. Let $a \star b = a^b - ab$. If $2 \star x = 22$, find x. Options: A: 5; B: 22; C: 11; D: 6.

Turn-by-turn model choices and actions

Turn	RIVET	RIVE w/o response
1	**OpenAI GPT-4o-mini** Rewrites the equation and plans to test the options.	**OpenAI GPT-4o-mini** Produces the identical equation and plan.
2	**Qwen3.5-9B** Switches to Qwen3.5-9B and evaluates each option.	**OpenAI GPT-4o-mini** Keeps OpenAI GPT-4o-mini and evaluates each option.
3	**DeepSeek V4-Flash** Selects DeepSeek V4-Flash and submits 5.	**OpenAI GPT-4o-mini** Keeps OpenAI GPT-4o-mini and submits 5.

Selected responses and tool outputs

T1, RIVET / OpenAI GPT-4o-mini (response excerpt):

Next, I will evaluate this equation for the given options: A) 5, B) 22, C) 11, D) 6.

T2, RIVET / Qwen3.5-9B (tool output excerpt):

x = 5: 2^x - 2x = 22

T2, RIVE w/o response / OpenAI GPT-4o-mini (tool output excerpt):

{5: 22, 22: 4194260, 11: 2026, 6: 52}

T3, RIVET / DeepSeek V4-Flash (response excerpt):

<solution>5</solution>

Final answers

RIVET: 5. **Correct.** 3 turns; Cost: $0.000343.

RIVE w/o response: 5. **Correct.** 3 turns; Cost: $0.000453.

Task mmlu:7.

Example 9. Minimizing a sum with fixed GCD and LCM

Task. Positive integers m, n have greatest common divisor 6 and least common multiple 126. What is the least possible value of $m + n$?

Turn-by-turn model choices and actions

Turn	RIVET	RIVE w/o response
1	**DeepSeek V4-Flash** Uses DeepSeek V4-Flash and submits 60.	**DeepSeek V4-Pro** Uses DeepSeek V4-Pro, derives the factor pairs, and submits 60.

Selected responses and tool outputs

T1, RIVET / DeepSeek V4-Flash (response excerpt):

<solution>60</solution>

T1, RIVE w/o response / DeepSeek V4-Pro (response excerpt):

Coprime factor pairs of 21: (1, 21) and (3, 7).

T1, RIVE w/o response / DeepSeek V4-Pro (response excerpt):

Corresponding (m, n): (6, 126) and (18, 42). Sums: 132 and 60.

Final answers

RIVET: 60. **Correct.** 1 turn; Cost: $0.000087.

RIVE w/o response: 60. **Correct.** 1 turn; Cost: $0.002121.

Task math:74.

Example 10. Helicopter acceleration

Task. A helicopter has height $s(t) = 300t - 4t^3$ metres, with t in minutes. Find its acceleration at $t = 5$, in $\mathrm{m/min^2}$. Return an integer.

Turn-by-turn model choices and actions

Turn	RIVET	RIVE w/o response
1	**Qwen3.5-9B** Uses Qwen3.5-9B and submits -120.	**OpenAI GPT-4o-mini** Plans to differentiate the height function.
2	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
3	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
4	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
5	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
6	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
7	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
8	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
9	Episode ended	**OpenAI GPT-4o-mini** Repeats the differentiation plan without a tool call.
10	Episode ended	**OpenAI GPT-4o-mini** Includes symbolic-differentiation code and submits -240.

Selected responses and tool outputs

T1, RIVET / Qwen3.5-9B (response excerpt):

<solution>-120</solution>

T9, RIVE w/o response / OpenAI GPT-4o-mini (response excerpt):

Let's execute the calculations.

T10, RIVE w/o response / OpenAI GPT-4o-mini (response excerpt):

a = sp.diff(v, t)

T10, RIVE w/o response / OpenAI GPT-4o-mini (response excerpt):

<solution>
-240
</solution>

Final answers

RIVET: -120. **Correct.** 1 turn; Cost: $0.000084.

RIVE w/o response: -240. **Incorrect.** 10 turns; Cost: $0.001307.

Task theoremqa:41.

D USE OF GENERATIVE AI

We used generative AI tools to polish the writing, explore related literature, assist with experiment implementation, and help check mathematical derivations. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the entire manuscript.