COST–ACCURACY AND BUDGET ADHERENCE FOR OUT-OF-DISTRIBUTION REASONING ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning routers decide when a language model should spend additional computation on a reasoning response. A useful router must both allocate reasoning to requests where it improves accuracy and continue meeting a compute budget when the request distribution changes. Cost–accuracy curves measure the first requirement, but not the second: deployment uses a threshold selected before the shift. We evaluate both requirements across two prediction models, diverse reasoning tasks, and unseen-dataset shifts by selecting thresholds on source validation data and applying them unchanged at test time. Scaling supervised routers from 0.8B to 27B yields only marginal in-distribution cost–accuracy gains, no consistent out-of-distribution gains, and greater variability in budget transfer. Moreover, routers with nearly identical cost–accuracy can differ substantially in realized cost and in how much their routing rates change, showing that ranking quality alone is insufficient. Existing supervised routers use only the question as input, whereas heuristic routers use signals from the direct prediction. We combine these approaches by also giving supervised routers the completed direct prediction and its confidence. This significantly improves in-distribution and out-of-distribution cost–accuracy at both 0.8B and 27B, and substantially reduces out-of-distribution budget-adherence error at 27B. Across our evaluated settings, the 0.8B router with these inputs remains the practical default. Reasoning routers should therefore be evaluated using cost–accuracy, cost adherence, and routing-rate adherence together under distribution shift.

1 INTRODUCTION

Reasoning can correct a language model's direct prediction, but it consumes additional computation and can replace a correct prediction with an incorrect one. A reasoning router must decide, for each request, whether the model should produce a separate reasoning prediction. Success in deployment has two parts: the router should spend computation where reasoning is most useful, and a threshold chosen to meet a compute budget before deployment should continue to meet that budget when the request distribution changes. We call these requirements *cost–accuracy* and *budget adherence*.

These requirements are related but not interchangeable. A cost–accuracy curve sweeps the routing threshold on a test population and measures how well router scores rank requests. Deployment instead fixes one numerical threshold in advance. After a distribution shift, that threshold may route a different fraction of requests, and the selected requests may require longer or shorter reasoning traces. A router can therefore preserve its cost–accuracy ranking while missing the budget it was selected to meet.

Most reasoning-router evaluations emphasize cost–accuracy curves or select and evaluate policies on the same benchmark (Lewis-Lim et al., 2025; Lu et al., 2025; Liu et al., 2025a; Li et al., 2025; Guo et al., 2026). We make budget transfer a co-primary evaluation target. Across two prediction models, 17 datasets, eight task families, and two complementary out-of-distribution (OOD) assignments, we select thresholds for six budgets on source validation data and apply them unchanged to in-distribution (ID) or OOD test data. We report cost mean absolute error (MAE), changes in the routing rate, and changes in the average reasoning cost of routed requests. Together, these measurements distinguish spending errors caused by routing more requests from errors caused by routing more expensive requests.

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.

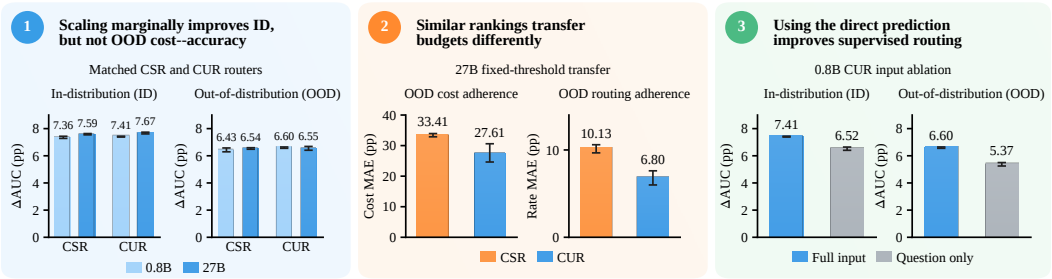

Figure 1: Overview of the main results. (1) Scaling Classifier Selective Reasoning (CSR) and Correctness Utility Routing (CUR) from 0.8B to 27B marginally improves ID cost–accuracy but yields no consistent OOD gain. (2) At 27B, their similar OOD rankings accompany different fixed-threshold cost and routing-rate adherence. (3) At 0.8B, adding the completed direct prediction and confidence improves CUR cost–accuracy ID and OOD. Tables 1–2 report every prediction-model and assignment setting. Error bars show one standard deviation across three matched router-training seeds.

Existing designs provide complementary information. Heuristic routers use signals from the direct prediction, whereas existing supervised routers use only the question. We combine these approaches by giving supervised routers the question, completed direct prediction, and its confidence. We primarily compare two label definitions grounded in prior work. Classifier Selective Reasoning (CSR) predicts whether reasoning helps; Correctness Utility Routing (CUR) predicts help, harm, or neutral outcomes and scores predicted help minus predicted harm (Li et al., 2025; Liang et al., 2025; Guo et al., 2026). As a supporting comparison, Direct Incorrectness Routing (DIR) predicts whether the direct prediction is wrong without observing the reasoning outcome.

Our results reveal three consistent lessons. First, when data, inputs, optimization, and supervision are held fixed, scaling CSR and CUR from 0.8B to 27B produces only marginal ID cost–accuracy gains and no consistent OOD gains; CUR also becomes more variable in budget transfer. Second, 27B CSR and CUR have nearly identical cost–accuracy but substantially different observed cost and routing-rate adherence, a separation reinforced by DIR. A scoring ablation suggests shared score drift as one possible cause, while also showing that routed-cost drift remains separate. Third, adding the direct prediction and its confidence significantly improves ID and OOD cost–accuracy at both 0.8B and 27B, and substantially improves 27B OOD budget adherence. These results make the 0.8B router with the additional inputs our practical recommendation, while the failure of 27B routing to deliver broader OOD gains remains a sobering scaling result.

Our main contributions are:

- We introduce an evaluation protocol for transferring source-selected budget thresholds under distribution shift and report cost, routing-rate, and routed-cost diagnostics alongside cost–accuracy.
- Holding all other conditions fixed, we show that scaling CSR and CUR from 0.8B to 27B yields little OOD benefit, and that similar cost–accuracy can conceal substantially different budget adherence. DIR provides supporting evidence for this separation at 27B.
- We combine supervised and heuristic routing signals by using the direct prediction and its confidence as additional inputs. This significantly improves ID and OOD cost–accuracy at 0.8B and 27B and improves 27B OOD budget adherence.

Together, these contributions support a concrete router design and a broader evaluation principle: cost–accuracy, fixed-threshold cost adherence, and routing-rate adherence should be reported together under distribution shift.

2 Related Work

Reasoning routing. Reasoning routers use confidence, consistency, or learned predictors to decide when the extra computation of a reasoning trace is worthwhile (Lewis-Lim et al., 2025; Lu et al.,

2025; Li et al., 2025; Liu et al., 2025a; Pan et al., 2024; Lou et al., 2025). Confidence-based methods use a completed prediction, whereas supervised routers learn semantic features from the request; our routers combine both. Existing supervised objectives predict improvement, mode preference, per-mode success, or a continuous compute-adjusted advantage (Li et al., 2025; Liu et al., 2025a; Zhang et al., 2025; Liang et al., 2025; Guo et al., 2026). CUR is a matched three-class parameterization of predicted advantage: it represents help, harm, or neutrality and subtracts predicted harm from predicted help.

Other adaptive inference. Reasoning can help on difficult predictions but can also waste computation or harm prediction accuracy (Wei et al., 2022; Chen et al., 2023; Sprague et al., 2025; Liu et al., 2025b). Related methods compress or skip reasoning, vary its depth, allocate more samples, or route among models (Cheng & Van Durme, 2024; Xu et al., 2025; Xia et al., 2025; Han et al., 2025; Pan et al., 2024; Lou et al., 2025; Chen et al., 2024; Yue et al., 2024; Ong et al., 2025; Hu et al., 2024). We focus on direct versus reasoning predictions from one model and the budget reliability of that choice under shift.

Budget control under distribution shift. Cost–accuracy curves and benchmark-specific thresholds do not establish that a source-selected threshold preserves its budget under shift. RACER instead trains a policy for each budget (Zhang et al., 2026); we study one reusable router with a fixed ID-selected threshold. Because predictive uncertainty can also deteriorate under shift (Ovadia et al., 2019), we distinguish probability fit and ranking quality from adherence to a transferred cost target.

3 Approach

3.1 Router inputs and routing policy

CUR and CSR use the question x, the complete direct prediction y_d, and its average token log probability $\bar{\ell}_d$:

$$u = (x, y_d, \bar{\ell}_d), \qquad \bar{\ell}_d = \frac{1}{T}\sum_{j=1}^{T} \log p(a_j \mid x, a_{<j}), \tag{1}$$

where $a_1, \ldots, a_T$ are the tokens in y_d. At threshold τ, the router returns a separate reasoning prediction y_r when $s(u) \geq \tau$ and otherwise returns y_d. Every request incurs direct-generation and router-scoring cost; only routed requests incur reasoning cost. The router sees neither y_r nor the reference target. We cache one direct/reasoning pair per example and reuse it for every method (Appendix A.1).

3.2 Router supervision labels

Let $c_d, c_r \in \{0, 1\}$ denote direct and reasoning correctness. CSR predicts whether reasoning helps, $(c_d, c_r) = (0, 1)$ (Li et al., 2025):

$$s_{\mathrm{CSR}} = \underbrace{p(c_d = 0, c_r = 1 \mid u)}_{p(\mathrm{help})}. \tag{2}$$

The expected accuracy change from choosing reasoning is

$$\mathbb{E}[c_r \mid u] - \mathbb{E}[c_d \mid u] = p(c_d = 0, c_r = 1 \mid u) - p(c_d = 1, c_r = 0 \mid u). \tag{3}$$

The both-correct and both-wrong outcomes cancel because either action has the same accuracy. Consequently, at a fixed routing rate, selecting the largest expected accuracy changes maximizes expected final accuracy; CSR omits the harm term from this decision. Thus reasoning helps for $(0, 1)$, harms for $(1, 0)$, and is neutral when $c_d = c_r$. CUR predicts these three outcomes and scores their expected accuracy change:

$$s_{\mathrm{CUR}} = \underbrace{p(c_d = 0, c_r = 1 \mid u)}_{p(\mathrm{help})} - \underbrace{p(c_d = 1, c_r = 0 \mid u)}_{p(\mathrm{harm})}. \tag{4}$$

All learned routers add a linear classification head and fine-tune rank-16 LoRA adapters for 2,500 steps with family-balanced sampling. They use the same inputs, backbone, examples, optimization budget, three seeds, and checkpoint criterion; the selected checkpoint maximizes family-macro validation $\Delta\mathrm{AUC}$. Appendix A.2 gives the full configuration.

4 Experimental Setup

4.1 Prediction models, workloads, and shifts

We evaluate paired generations from Qwen3.5-27B (Qwen Team, 2026) and Gemma-4-31B-IT (Gemma Team, 2026) on 17 datasets organized into eight task families. Mathematics contains GSM8K and MATH (Cobbe et al., 2021; Hendrycks et al., 2021b); logical and algorithmic tasks contain LogiQA and BBH (Liu et al., 2020; Suzgun et al., 2023); science contains ARC-Challenge, OpenBookQA, and SciQ (Clark et al., 2018; Mihaylov et al., 2018; Welbl et al., 2017); factual QA contains StrategyQA and TruthfulQA (Geva et al., 2021; Lin et al., 2022); commonsense reasoning contains CommonsenseQA and COPA (Talmor et al., 2019; Roemmele et al., 2011); reading comprehension contains BoolQ and MCTest (Clark et al., 2019; Richardson et al., 2013); language understanding contains WiC and RTE (Pilehvar & Camacho-Collados, 2019; Wang et al., 2018); and code generation contains LiveCodeBench and APPS (Jain et al., 2025; Hendrycks et al., 2021a). Each example has one direct and one separately generated reasoning prediction. Correctness uses task-normalized exact match or code tests, and every method uses the same prepared examples.

ID uses disjoint development and test splits of all datasets. Each of two OOD assignments removes selected datasets or BBH tasks from all router development and then tests on them. Training retains every task family, so this measures transfer to unseen datasets within represented families, not unseen families. Assignments A and B hold out different sources and cover both directions for two-dataset families; Appendix Table 6 lists them. The assignments concern router development; prediction-model pretraining is unknown.

All aggregates give the eight task families equal weight so that large datasets cannot dominate. Learned routers use seeds 0, 1, and 2; paired 95% Student-t intervals are computed from the three seed-level differences. They measure training variation conditional on fixed generations, datasets, and assignments, not uncertainty over new generations or future shifts. Direct and reasoning predictions are sampled once using their respective native inference configurations, so their paired outcomes compare complete modes rather than isolating the causal effect of adding a reasoning trace.

4.2 Methods and cost accounting

Our primary learned threshold routers use a Qwen3.5-0.8B backbone. We use a matched 27B configuration to study capacity. PPL routes high-perplexity direct predictions; first-token margin routes small gaps between the two most likely first tokens; and CAR fits correct/incorrect perplexity distributions to estimate direct error (Lewis-Lim et al., 2025; Lu et al., 2025).

Token Signature learns from a short probe generation's token-probability trend (Liu et al., 2025a). RouterBench-KNN embeds the question and averages neighbors' help-minus-harm outcomes (Hu et al., 2024). RACER trains a separate distributionally robust policy for each budget (Zhang et al., 2026); because it does not yield one reusable threshold curve, we report it in Appendix B.5. Appendix A.5 gives all implementations.

Weighted token cost is

$$C = N_{\mathrm{input}} + 5N_{\mathrm{output}}. \tag{5}$$

The fivefold weight reflects current short-context API price ratios (OpenAI, 2026a). Every policy pays for direct generation and its scoring computation; routed requests also pay for reasoning. PPL, margin, and CAR reuse direct-generation signals, whereas Token Signature, KNN, RACER, and the supervised routers incur their probe, embedding, or router-input tokens. This proxy consistently includes model inputs and generated outputs, but not parameter-dependent FLOPs, hardware utilization, latency, batching, or energy; it therefore does not price the 0.8B and 27B router passes differently. Appendix A.3 gives the rules, and Appendix B.3 confirms the conclusions with a sixfold output weight.

4.3 Evaluation objectives and metrics

Cost–accuracy. A cost–accuracy curve sweeps the routing threshold to evaluate how well the router allocates reasoning as the available budget varies. For each task family f, let $\bar{C}_{d,f}$ and $\bar{C}_{r,f}$ be the mean weighted-token costs of direct-only and reasoning-only generation, respectively. The family-averaged relative cost of direct inference is

$$b_d = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \frac{\bar{C}_{d,f}}{\bar{C}_{r,f}}, \tag{6}$$

where $\mathcal{F}$ contains the eight families. We summarize a method's average accuracy gain over direct inference as cost increases from b_d to the reasoning-only cost of one:

$$\Delta\mathrm{AUC}_m = \frac{1}{1 - b_d} \int_{b_d}^{1} \big[A_m(c) - A_d\big] \, \mathrm{d}c, \tag{7}$$

where $A_m(c)$ and A_d are routed and direct-only test accuracy. All costs and accuracies give each family equal weight. Because $\Delta\mathrm{AUC}$ uses test labels across all thresholds, it summarizes performance rather than selecting a deployment threshold.

Budget adherence. Budget adherence tests whether a numeric threshold selected on ID validation data meets its budget after shift. For each target $b \in \mathcal{B} = \{0.4, 0.5, \ldots, 0.9\}$, we choose the highest-routing validation threshold whose cost does not exceed b, freeze it, and apply it to test. This common interior grid leaves routing headroom above every method's direct-and-scoring cost and remains below the all-reasoning reference. Tied scores cross the threshold together, so discrete scores may undershoot a target. Let $\bar{C}_{T,f}(\tau_b)$ be test policy cost and $\bar{C}_{r,V,f}$ be validation all-reasoning cost in family f:

$$B_{T|V}(\tau_b) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \frac{\bar{C}_{T,f}(\tau_b)}{\bar{C}_{r,V,f}}, \qquad \mathrm{MAE} = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \big| B_{T|V}(\tau_b) - b \big| . \tag{8}$$

Families receive equal weight, policy cost includes scoring and routed reasoning, and reported MAE is multiplied by 100. Keeping the validation denominator reflects a budget committed before the future population is known; renormalizing by test cost would remove part of the deployment shift. We also report how well the routing rate itself transfers, which does not depend on this cost normalization. If cost is normalized using the new test population, it can only be interpreted after that population's costs are known. Fixing the fraction of requests sent to reasoning also misses changes in how expensive their reasoning traces are. Neither approach replaces tracking actual spending in deployment.

Using the same fixed thresholds, routing-rate MAE measures how much the fraction of requests sent to reasoning changes, on average, from validation to test. We also measure how the average reasoning cost of those routed requests changes. Reasoning spend per request depends on both how often reasoning is used and how expensive those requests are. Appendix B gives aggregation and interval details, and Appendix B.8 studies calibration with target costs.

5 Experiments and Results

5.1 Scaling marginally improves ID cost–accuracy but not OOD cost–accuracy

We begin with matched CSR and CUR routers at both capacities. Table 1 reports every setting and the aggregate results, alongside the baselines and supporting DIR comparison. DIR uses $s_{\mathrm{DIR}} = p(c_d = 0 \mid u)$, learning whether the direct prediction is wrong without observing the reasoning outcome. The individual settings remain visible because their $\Delta\mathrm{AUC}$ scales differ.

Increasing CUR from 0.8B to 27B raises aggregate ID $\Delta\mathrm{AUC}$ from 7.41 to 7.67 points, a paired gain of 0.26 points (95% CI $[0.15, 0.38]$). Aggregate OOD $\Delta\mathrm{AUC}$ changes from 6.60 to 6.55 points, a difference of -0.05 points ($[-0.45, 0.35]$). The per-setting OOD differences are -0.65 and $+0.11$ on Qwen A/B and $+0.32$ and $+0.02$ on Gemma A/B, so the 27B router has no consistent OOD

Table 1: Per-setting and aggregate ΔAUC (percentage points; higher is better). Learned values are mean (sample standard deviation) over three matched seeds. Aggregate ID averages prediction models; OOD also averages assignments. Bold marks the best value in each column.

	Qwen3.5-27B predictions			Gemma-4-31B-IT predictions			Aggregate	
Method	**ID**	**OOD A**	**OOD B**	**ID**	**OOD A**	**OOD B**	**ID**	**OOD**
CAR	9.39	8.91	10.54	2.46	2.44	1.32	5.93	5.80
PPL	9.47	8.91	10.69	2.22	2.44	1.19	5.85	5.81
Margin	7.74	6.34	10.27	4.18	**4.77**	1.86	5.96	5.81
Token sig.	5.66	5.09	6.52	2.23	2.55	0.30	3.95	3.61
RB-KNN	9.32	7.63	10.51	4.22	3.49	**2.02**	6.77	5.91
DIR 0.8B	9.94 (0.06)	9.36 (0.19)	10.69 (0.16)	4.20 (0.04)	3.80 (0.14)	1.51 (0.24)	7.07 (0.02)	6.34 (0.01)
DIR 27B	10.13 (0.19)	9.20 (0.16)	10.73 (0.22)	4.62 (0.16)	4.15 (0.08)	1.55 (0.09)	7.37 (0.05)	6.41 (0.09)
CSR 0.8B	10.29 (0.20)	8.95 (0.46)	11.09 (0.24)	4.42 (0.04)	4.08 (0.30)	1.62 (0.22)	7.36 (0.08)	6.43 (0.14)
CSR 27B	10.48 (0.12)	8.68 (0.05)	11.12 (0.16)	4.69 (0.07)	4.70 (0.05)	1.65 (0.09)	7.59 (0.05)	6.54 (0.06)
CUR 0.8B	10.39 (0.09)	**9.37 (0.30)**	11.19 (0.24)	4.42 (0.07)	4.15 (0.26)	1.68 (0.32)	7.41 (0.03)	**6.60 (0.05)**
CUR 27B	**10.62 (0.06)**	8.72 (0.26)	**11.30 (0.06)**	**4.72 (0.12)**	4.47 (0.17)	1.70 (0.20)	**7.67 (0.06)**	6.55 (0.14)

Table 2: Per-setting and aggregate fixed-threshold cost MAE (percentage points; lower is better). Learned values are mean (sample standard deviation) over three matched seeds. Aggregate ID averages prediction models; OOD also averages assignments. Bold marks the best value in each column.

	Qwen3.5-27B predictions			Gemma-4-31B-IT predictions			Aggregate	
Method	**ID**	**OOD A**	**OOD B**	**ID**	**OOD A**	**OOD B**	**ID**	**OOD**
CAR	3.93	21.19	23.81	1.35	30.09	14.95	2.64	22.51
PPL	3.39	21.19	20.74	2.21	30.09	17.90	2.80	22.48
Margin	2.41	22.78	28.01	2.22	25.79	25.56	2.31	25.53
Token sig.	3.60	**12.20**	**19.97**	**1.21**	17.20	**8.93**	2.41	**14.58**
RB-KNN	5.39	27.17	27.21	7.49	**14.56**	23.53	6.44	23.12
DIR 0.8B	2.96 (0.60)	25.98 (5.96)	25.15 (0.68)	1.44 (0.69)	41.20 (1.68)	18.08 (1.05)	2.20 (0.33)	27.60 (1.87)
DIR 27B	3.21 (0.46)	42.51 (0.78)	26.73 (0.79)	1.82 (0.20)	38.58 (3.48)	24.09 (0.25)	2.51 (0.31)	32.98 (0.58)
CSR 0.8B	2.30 (0.35)	27.26 (5.98)	22.18 (0.95)	1.81 (0.59)	40.71 (3.85)	22.39 (1.02)	2.05 (0.47)	28.13 (1.96)
CSR 27B	2.50 (0.59)	42.55 (0.58)	26.31 (0.80)	1.26 (0.37)	41.98 (2.49)	22.79 (1.41)	1.88 (0.38)	33.41 (0.53)
CUR 0.8B	2.25 (0.76)	29.71 (8.60)	23.58 (2.94)	1.32 (0.40)	39.23 (5.41)	19.59 (3.53)	**1.79 (0.52)**	28.03 (1.15)
CUR 27B	**1.83 (0.90)**	31.59 (6.31)	27.16 (0.85)	1.89 (1.15)	31.68 (7.33)	20.02 (2.39)	1.86 (0.70)	27.61 (2.99)

advantage. CSR shows the same broad pattern: its aggregate OOD $\Delta\mathrm{AUC}$ changes only from 6.43 to 6.54 points. The scale also differs sharply by prediction model, with Qwen results around 9–11 points and Gemma results around 2–5 points.

DIR follows this pattern: scaling raises ID $\Delta\mathrm{AUC}$ from 7.07 to 7.37 but changes OOD $\Delta\mathrm{AUC}$ only from 6.34 to 6.41. It remains supporting evidence, not a third primary label definition.

5.2 Similar cost–accuracy can transfer budgets differently

The more consequential difference between the 0.8B and 27B routers appears in budget transfer, where we select thresholds to meet budgets on validation data and then apply those same thresholds unchanged to test data. Table 2 reports fixed-threshold cost MAE for the baselines and the same six routers, including every setting and the aggregate ID and OOD results.

At 27B, CSR and CUR remain close in cost–accuracy: their aggregate ID $\Delta\mathrm{AUC}$ values are 7.59 and 7.67, and their OOD values are 6.54 and 6.55. The paired CUR-minus-CSR OOD difference is 0.01 points (95% CI $[-0.20, 0.22]$). Their OOD cost MAEs, however, are 33.41 and 27.61 points. The mean difference is substantial, although its three-seed CI is wide (-5.79 points, 95% CI $[-12.72, 1.14]$). At 0.8B, CSR and CUR have similar OOD cost MAE (28.13 and 28.03 points). Scaling therefore leaves CUR's mean OOD cost MAE nearly unchanged but raises CSR's by 5.28 points.

At 27B, CUR reduces cost MAE relative to CSR by 10.96 points on Qwen OOD A and 10.30 on Gemma OOD A, but is 0.85 points worse on Qwen OOD B. Transfer is therefore heterogeneous despite almost identical aggregate cost–accuracy.

DIR supports this result at 27B: its OOD $\Delta\mathrm{AUC}$ is 6.41, close to CSR's 6.54 and CUR's 6.55, but its OOD cost MAE is 32.98 points versus 27.61 for CUR. Thus the observed separation is not unique to CSR and CUR, although the three-seed uncertainty is substantial.

Capacity also changes stability across training runs. CUR's OOD cost-MAE standard deviation rises from 1.15 points at 0.8B to 2.99 points at 27B, even though its mean and cost–accuracy remain similar. The 27B router has thus failed to convert its much greater capacity into better OOD ranking or more reliable budget transfer. We recommend the 0.8B router as the practical default, but this is a sobering scaling result: the stronger backbone produces a small ID gain while leaving the deployment-facing OOD objectives unimproved.

Budget transfer has multiple views. Table 3 places cost MAE beside routing-rate MAE, the mean absolute change in the routed fraction at the same frozen thresholds.

Table 3: Aggregate OOD ranking and fixed-threshold adherence. Cost MAE is validation-referenced budget error; rate MAE is the mean absolute validation-to-test routing-rate change across the six targets. The fixed-fraction floor routes each request with its ID-selected probability and therefore has zero expected routing-rate error, while its cost still changes with population trace lengths. All values are percentage points; bold marks the best value in each column.

Method	**OOD ΔAUC $\uparrow$**	**Cost MAE $\downarrow$**	**Rate MAE $\downarrow$**
Fixed fraction	–	20.27	**0.00**
CAR	5.80	22.51	5.65
PPL	5.81	22.48	5.14
Margin	5.81	25.53	4.60
Token sig.	3.61	**14.58**	4.96
RouterBench-KNN	5.91	23.12	3.45
DIR 0.8B	6.34	27.60	9.99
DIR 27B	6.41	32.98	12.72
CSR 0.8B	6.43	28.13	9.36
CSR 27B	6.54	33.41	10.13
CUR 0.8B	**6.60**	28.03	8.89
CUR 27B	6.55	27.61	6.80

Routing-rate MAE looks only at changes in the fraction of requests sent to reasoning, so it does not depend on the validation cost used to normalize spending. It nevertheless disagrees with cost–accuracy ranking. At 27B, CSR and CUR have routing-rate MAEs of 10.13 and 6.80 points despite nearly identical OOD $\Delta\mathrm{AUC}$. DIR's similar OOD $\Delta\mathrm{AUC}$ but 12.72-point routing-rate MAE reinforces this comparison. CAR, PPL, and margin rank below the learned routers but generally transfer routing rates and costs better. The fixed-fraction policy anchors routing-rate MAE at zero while retaining 20.27 points of cost MAE, showing that stable routing frequency cannot prevent shifts in reasoning-trace cost. Cost MAE, routing-rate MAE, and routed-cost change therefore describe distinct aspects of budget transfer.

Post-hoc calibration does not remove this distinction. Appendix B.7 shows that source-only temperature, sigmoid, or isotonic calibration leaves adherence unchanged or improves some shifts while worsening others. Appendix B.8 shows that using target-population costs can improve target adherence, but degrades source adherence as the calibration mixture moves toward the target. Budget control is therefore population-specific rather than a consequence of probability calibration alone.

5.3 CUR HARM SUBTRACTION AS A CANDIDATE MECHANISM AT 27B

CUR-help is a mechanistic ablation that reuses the trained CUR router and scores only $p(\mathrm{help})$. At 27B, its OOD $\Delta\mathrm{AUC}$ is 6.57 and its error is 33.96, so subtracting harm changes $\Delta\mathrm{AUC}$ by -0.02

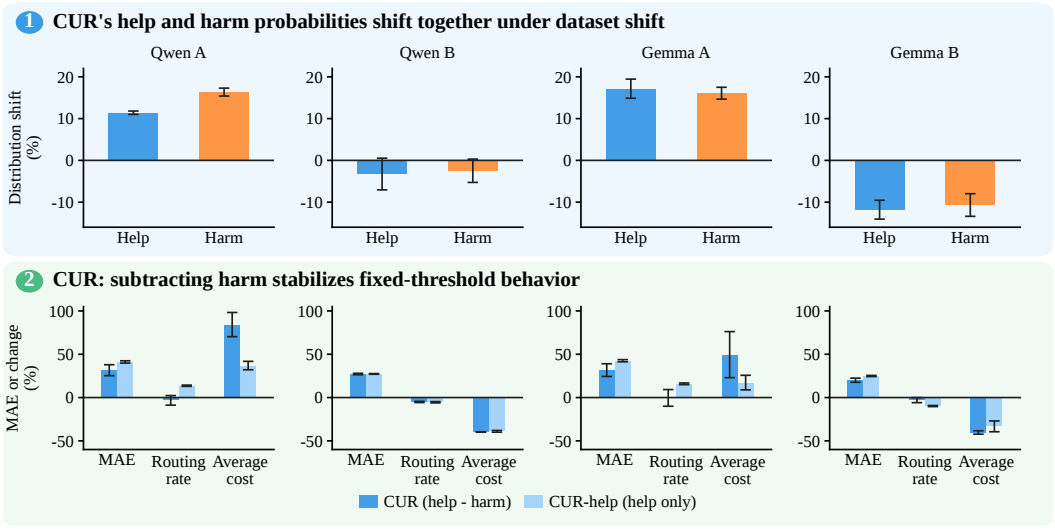

Figure 2: A CUR-specific candidate explanation for the 27B adherence observation. Both rows reuse the same trained CUR router. Top: CUR's predicted help and harm probabilities move together from validation to OOD test, so their difference cancels shared movement. Bottom: the deployed CUR help-minus-harm score (dark) has lower mean adherence MAE and smaller routing-rate changes than the CUR-help help-only ablation (light), although intervals are wide and routed reasoning cost can change more. Bars average six budget targets; error bars show one standard deviation across three router-training seeds.

points ($[-0.18, 0.14]$) while lowering error by 6.35 points ($[-13.98, 1.29]$). At 0.8B, CUR's error is instead 1.41 points higher than CUR-help's ($[-3.03, 5.84]$), so the observed direction does not generalize across sizes.

Figure 2 is consistent with a score-drift explanation for the 27B observation. Predicted help and harm move in the same direction from validation to OOD test in all four model–assignment settings. Their difference cancels part of this shared movement, keeping more scores on the same side of a frozen numerical threshold. A cost–accuracy curve, by contrast, sweeps thresholds and depends primarily on ranking, so it is largely insensitive to this score-level shift. The adherence CI includes zero, and the 0.8B reversal shows that this mechanism varies with router size. Moreover, help-minus-harm scoring controls neither which examples cross the threshold nor their trace lengths. In Qwen A it moves the routing-rate change from $+13.6$ to -3.3 points, yet mean routed reasoning-cost growth increases from 36.9% to 84.3%. Routed-cost drift is therefore a separate failure mode that none of the scoring rules studied here addresses, and another behavior a cost–accuracy curve cannot reveal.

5.4 Using the direct prediction as input improves routing and 27B budget adherence

Finally, we compare the recommended 0.8B routers with heuristics and existing routing methods, then isolate the effect of using the direct prediction as input. CAR, PPL, and first-token margin use inexpensive signals from the direct prediction; RouterBench-KNN retrieves training questions with similar embeddings. The learned routers jointly read the question, completed direct prediction, and its confidence. Table 1 reports their aggregate and per-setting cost–accuracy results together.

CUR achieves the strongest aggregate cost–accuracy among the evaluated methods: 7.41 points ID and 6.60 points OOD. Relative to RouterBench-KNN, the strongest aggregate existing-router baseline, its paired gains are 0.64 points ID (95% CI $[0.56, 0.72]$) and 0.69 points OOD ($[0.57, 0.81]$). It leads RouterBench-KNN in five of six settings; differences range from -0.34 points on Gemma OOD B to $+1.74$ on Qwen OOD A.

Table 4: CUR input ablation at 0.8B and 27B. Values are mean (sample standard deviation) percentage points over three seeds; bold marks the better input within each size.

		ΔAUC ↑		Adherence error ↓	
Router	**Input**	**ID**	**OOD**	**ID**	**OOD**
CUR 0.8B	Question only	6.52 (0.12)	5.37 (0.13)	2.40 (0.80)	28.50 (1.28)
	Question + prediction + confidence	**7.41 (0.03)**	**6.60 (0.05)**	**1.79 (0.52)**	**28.03 (1.15)**
CUR 27B	Question only	6.74 (0.03)	6.04 (0.21)	2.44 (0.36)	33.55 (1.08)
	Question + prediction + confidence	**7.67 (0.06)**	**6.55 (0.14)**	**1.86 (0.70)**	**27.61 (2.99)**

The matched input ablation explains this advantage. At 0.8B, adding the completed prediction and confidence improves CUR $\Delta\mathrm{AUC}$ by 0.89 points ID (95% CI $[0.56, 1.22]$) and 1.22 points OOD ($[0.97, 1.48]$); both gains are significant across the three matched seeds. Its ID and OOD cost-MAE changes, -0.61 and -0.48 points, are small relative to their uncertainty. At 27B, the corresponding $\Delta\mathrm{AUC}$ gains are 0.94 points ID (95% CI $[0.84, 1.04]$) and 0.50 points OOD ($[0.12, 0.89]$). More strikingly, OOD cost MAE falls from 33.55 to 27.61 points, a 5.94-point improvement ($[1.19, 10.68]$). Thus the direct prediction improves ranking at both capacities and, at 27B, also improves how well the source-selected budget transfers.

6 Limitations

A larger, randomly sampled collection of shifts would better establish the generality of the observed adherence patterns. Our two complementary assignments withhold data from all eight task families and cover both transfer directions for two-dataset families, but not unseen task families, transfer between prediction models, or arbitrary future shifts. Multiple independent direct–reasoning generations per example would reveal variation from generation sampling. We use one pair per example, so the three router-training seeds measure training variation conditional on fixed generations, not generation variation.

Multiple reasoning-effort levels would test whether the same supervision and budget-adherence conclusions extend from a binary decision to a richer compute allocation problem. We evaluate only one direct and one native reasoning configuration per prediction model. Intermediate sizes and other architectures are needed to establish a scaling trend and test whether the input and harm-subtraction results generalize. The 27B adherence improvement from harm subtraction is imprecise across three seeds and reverses slightly at 0.8B. CUR's higher 27B adherence variability is also based on three seeds and is not consistent across scoring rules. Automatically checkable tasks also do not test subjective human evaluation.

Our three-seed intervals cover router-training variation conditional on fixed predictions, not variation from resampling generations, datasets, or shifts. Wide adherence intervals therefore limit claims about population means.

Measured hardware cost would better describe deployment efficiency than our proxy, which prices input tokens equally across models and ignores router parameter count, utilization, batching, latency, and energy. Our 0.8B recommendation therefore reflects quality and footprint, not latency or price.

7 Conclusion

Scaling CSR and CUR from 0.8B to 27B yields marginal ID gains, no OOD gain, and greater CUR variation. Direct-prediction input improves cost–accuracy at both sizes and 27B OOD adherence. Even so, 0.8B remains our practical default.

At 27B, CSR and CUR have nearly identical cost–accuracy but different observed adherence, reinforced by DIR's similar ranking and worse adherence than CUR. Thus cost–accuracy does not determine budget transfer. Routing-rate and routed-cost drift are complementary; CUR-help suggests shared score drift but is imprecise, size-dependent, and does not control routed cost. Evaluation should therefore report all three: cost–accuracy, cost adherence, and routing-rate adherence.

AI USE STATEMENT

OpenAI Codex assisted with problem formulation, experimental design, analysis, implementation, related-work discovery, and manuscript editing. The authors reviewed all AI-assisted code, results, figures, tables, citations, claims, and prose against saved artifacts and primary sources, and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

The anonymized code records model and dataset revisions, split and generation seeds, prompts, decoding settings, router configurations, and analysis commands. Appendices A and B document implementation, evaluation, and threshold selection. Machine-readable metadata record split checksums and prediction provenance; repository scripts reproduce the figures and tables.

REFERENCES

Jiuhai Chen, Lichang Chen, Heng Huang, and Tianyi Zhou. When do you need chain-of-thought prompting for chatgpt? *arXiv preprint arXiv:2304.03262*, 2023.

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*, 2024. URL https://openreview.net/forum?id=cSimKw5p6R.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Google. Gemini developer API pricing, 2026. URL https://ai.google.dev/gemini-api/docs/pricing. Accessed September 18, 2026.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Huizhong Guo, Tianjun Wei, Dongxia Wang, Yingpeng Du, Ziyan Wang, Jie Zhang, and Zhu Sun. Think when needed: Model-aware reasoning routing for LLM-based ranking. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 471–481, 2026. doi: 10.1145/3805712.3809709. URL https://arxiv.org/abs/2601.18146.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021a. URL https://openreview.net/forum?id=sD93GOzH3i5.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021b.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Samuel Lewis-Lim, Xingwei Tan, Zhixue Zhao, and Nikolaos Aletras. Can confidence estimates decide when chain-of-thought is necessary for LLMs? *arXiv preprint arXiv:2510.21007*, 2025.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. When thinking fails: The pitfalls of reasoning for instruction-following in LLMs. *arXiv preprint arXiv:2505.11423*, 2025.

Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. ThinkSwitcher: When to think hard, when to think fast. *arXiv preprint arXiv:2505.14183*, 2025. URL https://arxiv.org/abs/2505.14183.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 3214–3252, 2022.

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

Peijie Liu, Fengli Xu, and Yong Li. Token signature: Predicting chain-of-thought gains with token decoding feature in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 39947–39967. PMLR, 2025a. URL https://proceedings.mlr.press/v267/liu25ci.html.

Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 38489–38517. PMLR, 2025b. URL https://proceedings.mlr.press/v267/liu25t.html.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. AdaCoT: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*, 2025.

Jinghui Lu, Haiyang Yu, Siliang Xu, Shiwei Ran, Guozhi Tang, Siqi Wang, Bin Shan, Teng Fu, Hao Feng, Jingqun Tang, Han Wang, and Can Huang. Prolonged reasoning is not all you need: Certainty-based adaptive routing for efficient LLM/MLLM reasoning. *arXiv preprint arXiv:2505.15154*, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 625–632, 2005. doi: 10.1145/1102351.1102430.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

OpenAI. API pricing, 2026a. URL https://developers.openai.com/api/docs/pricing. Accessed September 23, 2026.

OpenAI. GPT-5.4 model, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.4. Accessed September 18, 2026.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/8558cb408c1d76621371888657d2eb1d-Abstract.html.

Jiabao Pan, Yan Zhang, Chen Zhang, Zuozhu Liu, Hongwei Wang, and Haizhou Li. DynaThink: Fast or slow? a dynamic decision-making framework for large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 14686–14695. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.814. URL https://aclanthology.org/2024.emnlp-main.814/.

Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: The word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 1267–1273, 2019.

Qwen Team. Qwen3.5-27B model card. Hugging Face, 2026. URL https://huggingface.co/Qwen/Qwen3.5-27B.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 193–203, 2013.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 2011.

Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To CoT or not to CoT? chain-of-thought helps mainly on math and symbolic reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=w6nlcS8Kkn.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4149–4158, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, pp. 353–355, 2018.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.165. URL https://aclanthology.org/2025.emnlp-main.165/.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations*, 2024.

Wenbo Zhang, Lijinghua Zhang, Liner Xiang, and Hengrui Cai. Reasoning is not free: Robust adaptive cost-efficient routing for LLM-as-a-judge. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.10805.

Wencheng Zhang, Shiqin Qiao, Lingjie Luo, Yinfeng Li, Chuanyang Zheng, Qian Xu, Meng Li, Yong Gui, Yijun He, Jianing Qiu, Jindong Hong, and Jiankai Sun. SynapseRoute: An auto-route switching framework on dual-state large language model. *arXiv preprint arXiv:2507.02822*, 2025. URL https://arxiv.org/abs/2507.02822.

A Implementation Details

A.1 Prediction generation

We generate one direct and one reasoning completion for every prepared example and cache the pair before training or evaluating any router. Generation uses vLLM in bfloat16 with a maximum model context of 65,536 tokens and random seed 0. Qwen3.5-27B uses tensor-parallel size one, while Gemma-4-31B-IT uses size two. The exact Hugging Face model revisions are fixed in the repository and copied into the metadata file for each saved set of predictions. Table 5 gives the decoding parameters. Both modes use the model's chat template and the same task-specific output instruction. Direct generation disables thinking and begins from a response prefill; reasoning generation enables the model's native thinking mode. Non-code predictions must place the final prediction in `\boxed{}`, while code tasks require one fenced Python program. We request the top two log probabilities for every direct token, retain the sampled-token log probabilities, and use these values for the router confidence feature and confidence baselines. Each prompt is sampled once; there is no majority vote or best-of selection. Across all datasets, the prepared splits contain 75,604 training, 9,102 validation, and 17,055 test examples, for 101,761 examples per prediction model. Producing both modes therefore requires 203,522 generations per prediction model and 407,044 generations across the two prediction models. We treat direct and reasoning as complete inference configurations. Because they use mode-specific settings, paired outcomes should not be interpreted as the causal effect of adding a reasoning trace while holding every other generation parameter fixed.

Table 5: Prediction-generation settings. Values separated by a slash denote direct/reasoning modes. Both models use temperature 1.0, minimum probability 0, repetition penalty 1.0, and maximum completion lengths of 16,384/61,440 tokens.

Setting	Qwen3.5-27B	Gemma-4-31B-IT
Top-p	1.00 / 0.95	0.95 / 0.95
Top-k	40 / 20	64 / 64
Presence penalty	2.0 / 1.5	0 / 0
Tensor-parallel size	1	2
Direct batch size	256	8
Reasoning batch size	64	4

A.2 Router architecture and optimization

Our primary learned routers fine-tune Qwen3.5-0.8B; matched capacity and scoring ablations fine-tune Qwen3.5-27B. Both use a linear classification head on the final non-padding token. The input contains labeled fields for the question, the completed direct prediction, and the mean selected-token log probability of that prediction. Inputs are right padded and truncated from the left to 2,048 tokens, preserving the end of the direct prediction and confidence field. The head is unbiased and has two outputs for DIR and CSR, and three for CUR. The scoring ablation reuses the same trained three-class router and differs only in how its class probabilities form the route score.

We train rank-16 LoRA adapters with scaling parameter 16, zero dropout, and no adapter bias. Adapters are applied to the query, key, value, output, gate, up, and down projection modules; the classification head is trained jointly. Model weights are loaded in bfloat16 with FlashAttention-2. The 0.8B runs use one GPU and eight gradient-accumulation steps; the 27B runs use full-shard FSDP across two GPUs with activation checkpointing and four accumulation steps. Both use a per-device batch size of 4 and effective batch size of 32. A family-balanced weighted sampler draws examples with replacement, but the cross-entropy loss itself is unweighted and uses no label smoothing.

Optimization uses fused AdamW with learning rate 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, zero weight decay, and gradient-norm clipping at 1.0. The learning rate decays linearly to zero without warmup. Each run lasts 2,500 optimizer steps. We evaluate and save every 500 steps and export the saved checkpoint with the largest in-distribution (ID) validation $\Delta\mathrm{AUC}$ after giving each task family equal weight. Every label definition uses the same configuration and router seeds 0, 1, and 2;

we do not tune hyperparameters separately by label, prediction model, or out-of-distribution (OOD) assignment.

A.3 Cost accounting details

The output-token weight in Equation 5 follows current OpenAI short-context API pricing. GPT-6 Astra, Sol, and Luna each list output-token prices five times their input-token prices across standard and fast processing rates (OpenAI, 2026a). The ratio provides a common token-based proxy and does not estimate the monetary cost of running our open-weight models.

Every request pays for its direct generation and any method-specific computation needed to produce a routing score. Each learned router makes a separate forward pass with its experiment's backbone over the question, completed direct prediction, and mean direct-prediction log probability. We count every token in that router input, even when the request is not sent to reasoning; the classification head produces no generated tokens. PPL, margin, and CAR reuse values already produced by the direct call and therefore add no scoring tokens. Token Signature includes its separate question-only probe generation on every request. RouterBench-KNN and RACER likewise include the input tokens processed by their embedding models. If a request is selected, it additionally pays for an independent reasoning generation.

We assign the same input-token price to the 0.8B and 27B routers and both prediction models. This provides one consistent token accounting rule across methods; the models can differ in measured runtime. We normalize all policy costs by the cost of reasoning on every request and average the eight task families equally. The proxy accounts for prediction- and router-model input and output tokens, but it does not separately model router forward-pass FLOPs, hardware utilization, latency, or batching.

A.4 Data splits and OOD assignments

Dataset preparation uses split seed 0. We preserve official train, validation, and test partitions when all three contain public labels. When a dataset has an official train and test split but no validation split, 10% of training examples form validation. For a single public labeled pool, deterministic partitioning assigns 10% to validation and 10% to test. Stratification is applied by MATH subject, BBH task, LiveCodeBench difficulty, and APPS source split where applicable. The resulting prepared examples and split fingerprints are shared by both prediction models and recorded in the metadata files. Table 6 lists every dataset and BBH task in the family taxonomy and the examples excluded in each OOD assignment. All entries not named in an assignment column remain available for router training and validation.

Table 6: Complete grouping of datasets and BBH tasks into the eight task families, together with the test datasets or tasks for each OOD assignment. The corresponding training and validation examples are excluded from router training, baseline fitting, checkpoint selection, and threshold calibration. Entries prefixed by "BBH:" are individual tasks within the BBH dataset. Assignment B uses all 15 logical/algorithmic BBH tasks listed in that row as its OOD test set.

Task family	All datasets and BBH tasks	OOD A test	OOD B test
Mathematical problem solving	GSM8K; MATH; BBH: multistep arithmetic two, object counting	MATH	GSM8K
Logical and algorithmic tasks	LogiQA; BBH: boolean expressions, date understanding, Dyck languages, formal fallacies, geometric shapes, logical deduction (three, five, and seven objects), navigate, temporal sequences, tracking shuffled objects (three, five, and seven objects), web of lies, word sorting	LogiQA	All listed BBH tasks
Science question answering	ARC-Challenge; OpenBookQA; SciQ	SciQ	ARC-Challenge
Factual question answering	StrategyQA; TruthfulQA MC1; BBH: movie recommendation, sports understanding	StrategyQA	TruthfulQA MC1
Commonsense reasoning	CommonsenseQA; COPA; BBH: causal judgement	CommonsenseQA	COPA
Reading comprehension	BoolQ; MCTest; BBH: penguins in a table, reasoning about colored objects	BoolQ	MCTest
Language understanding	WiC; RTE; BBH: disambiguation QA, hyperbaton, ruin names, salient translation error detection, snarks	WiC	RTE
Code generation	LiveCodeBench v6; APPS competition	APPS competition	LiveCodeBench v6

A.5 Baseline implementation and evaluation

PPL is the exponential of the negative mean sampled-token log probability, and the first-token margin is the difference between the two largest first-token probabilities. Both scores rank requests directly. CAR fits family-balanced class priors and one-dimensional Gaussian likelihoods for the PPL of correct and incorrect direct predictions on the permitted training sources, then uses the posterior probability of direct error as its routing score. Token Signature generates at most 50 tokens greedily, computes the Spearman correlation between token position and sampled-token probability, and fits a scikit-learn logistic regression to distinguish help from harm. Its fitting set contains at most 50 training examples per dataset on which direct and reasoning predictions differ in correctness, selected with seed 42, and its extra generation cost is included for every evaluated request. Appendix A.6 details how this separate generation matches the procedure of Liu et al. (2025a) and differs from our direct and reasoning generations. PPL, margin, and CAR have zero added token cost.

For every learned router and baseline, tied scores enter a cost–accuracy curve together. Thresholds use the inclusive rule $s \geq \tau$. Accuracy, cost, and sampling weights are first computed within the eight task families and then averaged with equal weight for each family, so a large dataset cannot dominate a reported aggregate. The mathematically computed Random reference independently routes each example with a common probability and therefore has no fitted score threshold or scoring cost.

A.6 Token Signature reproduction

Token Signature requires a third generation that is distinct from both prediction modes evaluated elsewhere in this work. Liu et al. (2025a) obtain a "standard" response from a question-only prompt using greedy decoding, record the probabilities of the first 50 generated tokens, and define the instance-level feature as

$$\rho_i = \mathrm{Spearman}((1, \ldots, L_i), (p_{i,1}, \ldots, p_{i,L_i})), \qquad L_i \leq 50, \tag{9}$$

where $p_{i,t}$ is the probability assigned to the token selected at position t. Their Dynamic CoT classifier fits a one-dimensional logistic regression to this feature using 50 labeled examples per benchmark. The labels retain only examples on which direct and reasoning predictions differ in correctness: reasoning correct and direct incorrect is class one, while direct correct and reasoning incorrect is class zero.

We reproduce the signal-producing generation separately for each prediction model. The user message contains only the dataset question and is rendered with the model's native chat template with thinking disabled. It contains no task-specific output-format instruction, no direct-prediction prefill, and no CoT trigger. This matches Token Signature's standard question-only prompt and released model-specific chat formatting. In contrast, our ordinary direct and reasoning generations share a task-specific output instruction; direct mode disables thinking and begins from a response prefill, while reasoning mode uses the model's native thinking mode.

The Token Signature response is decoded greedily with temperature zero, top-p one, top-k disabled, minimum probability zero, repetition penalty one, presence penalty zero, and generation seed zero. We request the log probability of the selected token at every step and stop after at most 50 tokens or a model stop token. The released implementation permits a longer standard response and then retains its first 50 probabilities. Stopping at 50 produces the same autoregressive prefix while avoiding generation that cannot affect the feature. If a response terminates earlier, we compute the correlation from its available non-stop tokens, as in the released implementation; for the degenerate case of fewer than two such tokens, we set $\rho_i = 0$ and record this replacement in the saved results. These settings differ deliberately from the temperature-one, model-specific sampling parameters and much larger completion limits used for our direct and reasoning predictions.

We adapt the reported fitting protocol to our multi-dataset transfer setting. With selection seed 42, we sample up to 50 training examples from each permitted dataset on which direct and reasoning predictions differ in correctness, and pool them into one scikit-learn logistic regression for that ID or OOD assignment. OOD datasets and BBH tasks remain excluded from this fitting set. This preserves the authors' feature, binary target, per-dataset sample allowance, and logistic model while ensuring that the OOD baseline receives no examples from the evaluated datasets. We use the predicted probability of class one as a continuous routing score. The authors deploy Dynamic CoT

at probability 0.5; we instead sweep or calibrate this score under the same threshold protocol as every other method so that cost–accuracy and budget-adherence comparisons use the same requested budgets.

Finally, we charge the question-only probe on every request: its prompt tokens and up to 50 generated tokens enter the policy cost before any selected reasoning generation. The resulting curves therefore include Token Signature's nonzero scoring overhead. The released Dynamic CoT calculation sums the selected direct or CoT completion; our accounting additionally charges the generation used to obtain the routing feature.

B Additional Evaluation Details

B.1 Evaluation protocol

All language-model comparisons use the same prepared examples. Router training uses training rows, while checkpoint and ID-validation threshold selection use validation rows; reported accuracies use test rows. For OOD runs, the excluded sources are removed before router and baseline development. We evaluate only examples that have both direct and reasoning results, a nonempty normalized direct prediction, and the required confidence values. Non-code correctness uses normalized exact match. Multiple-choice predictions map to canonical letters, boolean predictions to their allowed labels, and MATH predictions through a deterministic boxed-expression normalizer. Code must pass every provided test.

For checkpoint selection, we compute $\Delta\mathrm{AUC}$ within each family and then give each family equal weight. The reported test curves instead vary one global threshold, so differences in score scales across families can affect the result. We construct each test curve by varying the routing-score threshold and adding all examples with a tied score together. Any scoring overhead appears as an initial horizontal segment from the direct-only cost. We linearly interpolate the curve, clip it at relative cost one, and use the same integration interval for all methods within a prediction-model and evaluation-setting comparison.

Budget adherence error includes both validation undershoot from discrete scores and changes in cost between validation and test data. Test policy cost is always divided by the corresponding validation family's mean all-reasoning cost; no test-derived denominator enters the metric. We compute the error within each assignment and router seed before averaging. Headline ID values average the two prediction models; OOD values average the four combinations of two prediction models and two assignments. For paired comparisons, we first average the fixed evaluation settings within each seed and then apply a two-sided Student-t interval to the three seed-level differences (two degrees of freedom). These intervals measure training variation conditional on the fixed generations, models, datasets, and assignments. Approximate normality cannot be checked with three seeds, so the intervals are necessarily imprecise. When one method is deterministic, its value is held fixed across seeds.

For the CUR diagnostic in Figure 2, the top-row shift is the centered probability-of-superiority statistic $100[\Pr(s_{\mathrm{OOD}} > s_{\mathrm{val}}) + 0.5\Pr(s_{\mathrm{OOD}} = s_{\mathrm{val}}) - 0.5]$ for independently sampled scores. At each threshold, reasoning spend is the routing rate times mean reasoning cost among routed examples. Direct and scoring costs remain included in adherence MAE but not in this decomposition.

B.2 Per-setting cost–accuracy curves

Figure 3 expands the aggregate cost–accuracy comparison into all six prediction-model and evaluation settings. The plotted cost is the realized family-macro test cost relative to always reasoning, and accuracy is averaged equally across task families. CUR and CSR remain close throughout most of the evaluated range. They generally trace a stronger frontier than RouterBench-KNN; the exception is Gemma OOD B, where KNN is slightly more accurate across much of the range. The per-setting panels also expose variation that is hidden when ID and OOD results are averaged across prediction models and assignments.

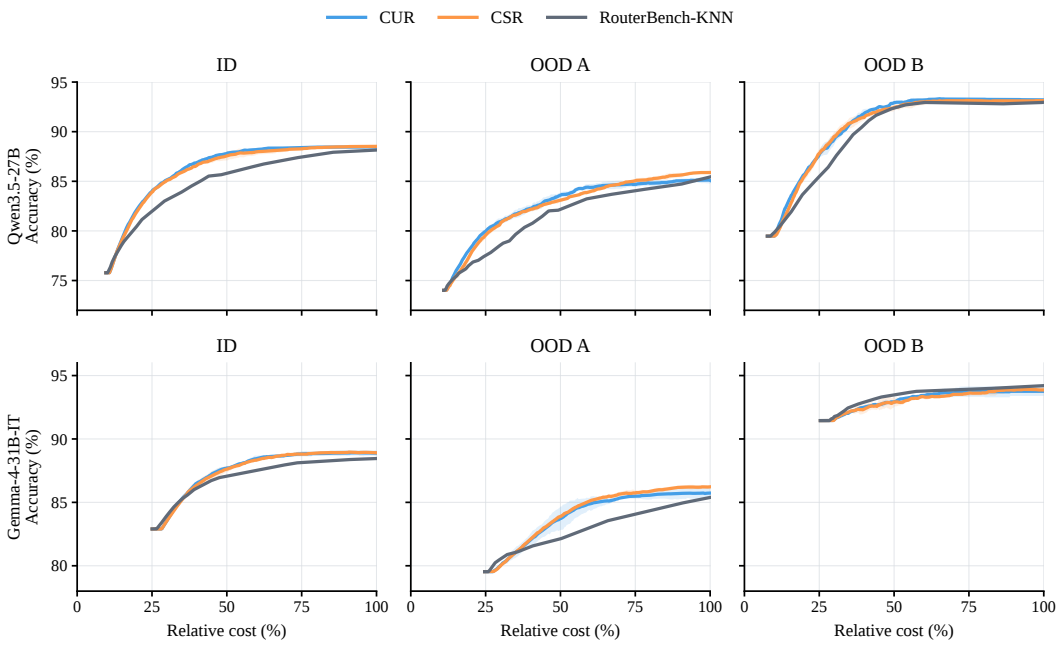

Figure 3: Cost–accuracy curves for CUR, CSR, and RouterBench-KNN in every prediction-model and evaluation setting. Rows correspond to the Qwen3.5-27B and Gemma-4-31B-IT prediction models; columns show ID, OOD Assignment A, and OOD Assignment B. Cost is relative to always generating the reasoning prediction and includes direct-generation and router-scoring overhead. Accuracy and cost give each task family equal weight. CUR and CSR curves are pointwise means over three router-training seeds after interpolation on realized test cost; shaded bands show one sample standard deviation. RouterBench-KNN is deterministic. Output tokens are weighted five times input tokens.

B.3 Output-token cost sensitivity

For the 27B router, our primary cost proxy weights an output token five times as much as an input token. As a nearby alternative, we repeat the main evaluation with a weight of six. This remains representative of recent pricing: GPT-5.4 lists \$2.50 per million input tokens and \$15 per million output tokens, while Gemini 3.1 Pro lists \$2 and \$12, respectively, for prompts no longer than 200K tokens (OpenAI, 2026b; Google, 2026). We change only the cost multiplier: predictions, correctness labels, routing scores, and router checkpoints remain fixed. We recompute direct, reasoning, and scoring costs from their token counts, reselect every validation threshold, and then recompute test ΔAUC and budget adherence error.

Under the sixfold weight, the supervised routers remain stronger than the best existing-router baseline in cost–accuracy: CUR exceeds RouterBench-KNN by 0.92 ID ΔAUC points (95% CI $[0.76, 1.08]$) and 0.65 OOD points ($[0.30, 1.00]$). CUR and CSR remain nearly identical in cost–accuracy, with CUR-minus-CSR differences of 0.09 points ID ($[0.03, 0.14]$) and 0.01 points OOD ($[-0.21, 0.22]$), while CUR has lower OOD budget adherence error: 28.00 versus 33.98 points, a difference of -5.97 points ($[-12.66, 0.72]$). Thus, the cost–accuracy and budget-adherence findings persist under the sixfold output-token weight.

B.4 RouterBench-KNN adaptation

RouterBench-KNN makes predictions without training a classifier: it embeds a new prompt, retrieves its nearest training prompts, and estimates each candidate model's performance by the unweighted mean of its observed scores on those neighbors (Hu et al., 2024). The released method then combines predicted performance with a model-level cost through a willingness-to-pay parameter. For two actions with a fixed cost difference, varying willingness to pay is equivalent to thresholding the predicted performance difference. We therefore use the neighbor-average direct-versus-

Table 7: Sensitivity results with output tokens weighted six times input tokens. $\Delta\mathrm{AUC}$ and budget adherence error are percentage points; adherence uses validation all-reasoning costs as fixed references. Higher $\Delta\mathrm{AUC}$ and lower error are better. ID averages two prediction models, and OOD averages both models and both assignments. Learned methods average three router seeds. Bold marks the best value in each column.

	$\Delta\mathrm{AUC}$		Adherence error	
Method	ID	OOD	ID	OOD
CAR	5.94	5.81	2.66	22.75
PPL	5.86	5.82	2.86	22.73
Margin	5.97	5.82	2.30	25.85
Token sig.	3.98	3.65	2.41	**14.98**
RouterBench-KNN	6.79	5.94	6.74	24.47
CSR 27B	7.63	6.58	1.92	33.98
CUR 27B	**7.71**	**6.59**	**1.87**	28.00

reasoning difference

$$s_{\mathrm{KNN}}(x) = \frac{1}{k} \sum_{j \in \mathcal{N}_k(x)} (c_{r,j} - c_{d,j}) \tag{10}$$

as a reusable scalar score and subject it to the same procedures for measuring the curve across all thresholds and for reusing an ID-selected threshold on OOD data. Actual direct, scoring, and reasoning costs for each request enter curve construction and threshold calibration rather than the KNN score.

We preserve the released RouterBench representation and neighbor rule. The input is the question only, embedded by `sentence-transformers/all-MiniLM-L12-v2`; it does not include the completed direct prediction or its log probability. We truncate to the model's 128-token limit, mean-pool non-padding token states, normalize the resulting 384-dimensional vector, and retrieve exact cosine neighbors. Neighbor labels receive uniform weight. For each prediction model and ID/OOD assignment, the index contains every training example permitted by that assignment that passed preprocessing. We select $k \in \{5, 10, 40\}$ by source-validation $\Delta\mathrm{AUC}$ after giving each task family equal weight, breaking ties toward smaller k; $k = 40$ is selected in all six evaluated settings. No test example or held-out dataset is used to select k.

We charge the MiniLM input-token count as scoring cost on every request under Equation 5; the lookup produces no output tokens. This uses the paper's common weighted-token proxy even though it does not reflect MiniLM's much smaller parameter count or hardware cost. The method is deterministic, so its table entries have no router-seed variation.

KNN's aggregate OOD budget adherence error is lower than CUR's, whereas its aggregate OOD $\Delta\mathrm{AUC}$ is lower. Its score can take only discrete multiples of $1/k$, which also produces substantial validation undershoot and a 6.44-point aggregate ID budget adherence error. The baseline therefore shows that subtracting harm can work outside a trained classifier, but does not dominate the trained routers.

B.5 Comparison with budget-conditioned robust routing

RACER is a complementary approach that trains a distinct probabilistic routing policy for each requested budget using distributionally robust reward and cost objectives (Zhang et al., 2026). We adapt the authors' released implementation to our setting, in which the router first observes a direct prediction, and evaluate it on exactly the same paired predictions, OOD assignments, and task-family populations as CUR 0.8B. Each request is represented by a fixed BGE-M3 embedding of the question, direct prediction, and direct-prediction mean log probability. A four-layer MLP with widths 256, 128, and 64 outputs the probability of requesting reasoning. As in the released implementation, training uses 60 epochs, batch size 64, AdamW learning rate 10^{-4}, dual learning rate 10^{-3}, entropy coefficient 0.005, reward-robustness temperature 1, and cost-robustness temperature 50. Because our normalized costs are approximately one order of magnitude smaller than those in

Table 8: OOD cost–accuracy and fixed-budget adherence for RACER and CUR 0.8B on the same six absolute relative-cost targets used in the main evaluation. Each g/c entry is the accuracy gain over direct inference (g) and realized OOD cost for the cost–accuracy comparison (c), in percentage points. Cost MAE is the mean absolute difference between validation-referenced OOD cost and the requested target. Rate MAE is the mean absolute validation-to-OOD change in routing probability. Values give each task family equal weight and average two prediction models, two OOD assignments, and three seeds. RACER trains a separate policy for every target; CUR reuses one router with six source-selected thresholds. Bold marks the higher gain in each target column and the lower value in each adherence column.

	Requested relative-cost target: OOD gain/cost						Adherence	
Method	40%	50%	60%	70%	80%	90%	Cost MAE	Rate MAE
CUR 0.8B	**5.98/38.77**	**7.18/48.91**	**7.82/60.45**	**8.28/74.34**	**8.52/88.44**	**8.51/98.16**	28.03	8.89
RACER	4.59/41.06	5.88/52.60	6.72/63.76	7.47/76.35	7.86/86.30	8.13/94.41	**25.28**	**4.68**

the released implementation, we multiply both action costs and the constraint target by 10 during optimization. This leaves feasibility and all reported costs unchanged while preserving the intended numerical scale of the cost-robustness temperature. We give each task family equal weight during sampling so the training population matches an evaluation that gives each family equal weight.

The action costs require one substantive adaptation. RACER originally chooses between mutually exclusive non-reasoning and reasoning calls. Our router acts after observing a direct prediction, so both actions incur the direct and router-scoring costs, and the reasoning action additionally incurs the reasoning-generation cost. We include the BGE-M3 input tokens in the scoring cost. We use the same six absolute relative-cost targets as the main budget evaluation, 0.4 through 0.9 in increments of 0.1. For each target, we train a separate RACER policy and select the source-validation checkpoint with the highest accuracy among those satisfying the target. If none is feasible, the declared rule selects the checkpoint closest to it. We then apply that policy unchanged to the OOD population. Reported accuracy, cost, and routing rate are expectations under its probability of requesting reasoning.

Table 8 makes the comparison with CUR 0.8B on the paper's current evaluation. RACER does not receive a $\Delta\mathrm{AUC}$ value because its six points come from six separately trained policies; a CUR cost–accuracy curve instead comes from sweeping the threshold of one reusable router. The g/c cells therefore show each policy's OOD accuracy gain and realized OOD cost directly. As elsewhere in the paper, cost–accuracy uses the OOD population's all-reasoning cost as its denominator. Budget adherence instead keeps the source-validation denominator fixed and measures absolute error from the budget chosen before the shift. Thus, the two reported costs answer different questions by design. Aggregate g/c cells can also hide positive and negative errors across settings, whereas cost MAE takes the absolute error within each setting before averaging.

The cost–accuracy result is straightforward: RACER has a smaller accuracy gain than CUR 0.8B at every target shown in the table. Because their realized costs are not identical, we additionally interpolate the CUR OOD curve at each RACER cost. At matched cost, RACER is 1.05 accuracy points below CUR 0.8B (95% CI $[0.84, 1.25]$ for the CUR advantage). This comparison uses 68 of the 72 matched operating points. Four RACER policies at the 0.9 target exceed the cost–accuracy domain's upper endpoint of one on OOD data, so no CUR interpolation exists for those points; all 72 points remain in the adherence calculation.

Budget adherence favors RACER. Its validation-referenced OOD cost MAE is 25.28 points, compared with 28.03 for CUR 0.8B, a paired RACER-minus-CUR difference of -2.75 points (95% CI $[-5.44, -0.06]$). Its routing-rate MAE is also lower: 4.68 versus 8.89 points, a difference of -4.21 points (95% CI $[-6.37, -2.05]$). Both methods overspend on average by a similar amount—6.53 points for RACER and 6.44 for CUR—so RACER's lower MAE should not be read as eliminating overspend. In plain English, RACER keeps how often it invokes reasoning more stable after the distribution shift and consequently misses the requested budgets by less, but CUR 0.8B chooses more useful requests and obtains higher accuracy for the same realized cost. RACER also requires a separate trained policy for every budget, whereas CUR 0.8B uses one router and changes only its threshold.

Table 9: Fixed-threshold budget-adherence diagnostics. Routing-rate change is test minus validation in percentage points. Mean routed reasoning-cost change is the percentage change from validation to test in the family-balanced mean reasoning cost among routed examples. Positive values denote more routing or more expensive routed reasoning on test. Each value averages the six budget targets; learned methods also average three router seeds. Assignments remain separate because their signed shifts can cancel. Bold marks the value closest to zero in each column and panel.

	ID		OOD			
Method	**Qwen**	**Gemma**	**Qwen A**	**Qwen B**	**Gemma A**	**Gemma B**
Routing-rate change (percentage points)						
Fixed fraction	**+0.0**	**+0.0**	**+0.0**	**+0.0**	**+0.0**	**+0.0**
CAR	-1.5	+0.7	-1.3	-7.5	+7.8	-4.7
PPL	-0.5	+0.2	-1.3	-0.2	+7.8	-6.1
Margin	-0.6	-1.7	+0.2	-6.2	-5.1	-4.7
Token sig.	-2.2	-0.6	-7.2	+0.2	-6.4	-0.9
RouterBench-KNN	-0.3	+0.5	+1.9	+5.0	-0.7	-0.2
CSR 0.8B	+0.1	+2.4	+4.8	+3.0	+18.3	-7.1
CSR 27B	-0.9	+0.2	+14.3	-4.3	+14.7	-7.2
CUR 0.8B	+0.0	+1.3	+5.3	+0.1	+16.8	-4.4
CUR 27B	-0.5	+0.8	-3.3	-5.1	-0.4	-3.0
Mean routed reasoning-cost change (percent)						
Fixed fraction	-3.0	-3.1	+44.8	-39.4	+42.3	-37.2
CAR	-1.7	-5.6	+80.8	-38.1	+29.5	-36.5
PPL	-2.6	-6.3	+80.8	-45.3	+29.5	-30.8
Margin	+2.3	+3.4	**-2.0**	**-6.6**	+35.6	**-14.1**
Token sig.	-4.6	**+0.1**	+41.2	-39.0	+58.1	-30.8
RouterBench-KNN	+0.8	-6.3	+40.9	-41.3	+26.7	-40.0
CSR 0.8B	-3.9	-9.3	+52.2	-45.5	+13.0	-38.3
CSR 27B	-1.8	-6.4	+37.9	-39.1	+23.6	-34.6
CUR 0.8B	-3.2	-4.7	+55.6	-43.4	**+4.9**	-30.0
CUR 27B	**-0.6**	-1.7	+84.3	-39.8	+49.6	-40.3

B.6 Additional score-transfer diagnostics

Table 9 reports the two signed validation-to-test diagnostics for the same methods as Table 3. Assignment A generally makes routed reasoning more expensive, whereas Assignment B makes it less expensive; an aggregate signed OOD mean would therefore hide the shift. Changes in routing rate also vary by method. In particular, CUR 27B keeps its routing rate near or below its validation level in Assignment A, even though routed reasoning becomes markedly more expensive. The validation-referenced adherence error retains both effects in the realized policy cost.

This score comparison holds the evaluated examples constant. For each OOD test source, we compare the primary OOD router, trained without that source, with a reference router trained with all sources. The reference router is not a deployable OOD policy because its training and validation data include the evaluated source. On identical test examples, we first compute correlations separately for each prediction model, assignment, and seed, then average these values. Pooled Spearman correlations range from 0.77 to 0.92 across methods. For the within-source statistic, we average source-specific correlations equally before the same outer aggregation. DIR and CSR retain within-source correlations of 0.76–0.82, whereas CUR has correlations of about 0.54. Thus the ordering of entire sources is more stable for labels based on changes in correctness than their ordering of examples inside a source. Correlation measures score agreement, not utility; positive OOD $\Delta\mathrm{AUC}$ supplies the complementary evidence that the orderings are useful rather than merely similar.

Table 10: Source-only probability calibration. Each cell reports test cost–accuracy $\Delta\mathrm{AUC}$ / validation-referenced cost MAE / routing-rate MAE in percentage points, averaged over three training seeds. ID calibrators and thresholds are fit on ID validation and transferred unchanged to ID test; A and B use retained-source validation and held-out-target test. Cost MAE uses the six absolute relative-cost targets from 0.4 through 0.9 and retains the validation reasoning-cost reference on test. Higher $\Delta\mathrm{AUC}$ and lower MAE are better. Within each router and prediction model, bold marks the best value for each metric and column.

Router	Calibration	Predictions	ID	OOD A	OOD B
CSR 27B	Raw	Qwen	**10.48 / 2.50 / 0.92**	**8.68** / 42.55 / 14.26	**11.12 / 26.31** / 4.36
		Gemma	**4.69 / 1.26** / 0.47	**4.70** / 41.98 / 14.73	**1.65 / 22.79** / 7.18
	Temperature	Qwen	**10.48 / 2.50 / 0.92**	**8.68** / 42.55 / 14.26	**11.12 / 26.31** / 4.36
		Gemma	**4.69 / 1.26** / 0.47	**4.70** / 41.98 / 14.73	**1.65 / 22.79** / 7.18
	Sigmoid	Qwen	**10.48 / 2.50 / 0.92**	**8.68** / 42.55 / 14.26	**11.12 / 26.31** / 4.36
		Gemma	**4.69 / 1.26** / 0.47	**4.70** / 41.98 / 14.73	**1.65 / 22.79** / 7.18
	Isotonic	Qwen	10.47 / 6.06 / 0.99	8.65 / **33.50 / 11.82**	**11.12** / 30.09 / **3.52**
		Gemma	4.68 / 4.84 / **0.43**	4.66 / **32.39 / 12.07**	**1.65** / 24.41 / **7.01**
CUR 27B	Raw	Qwen	10.62 / 1.83 / 1.11	8.72 / **31.59** / 7.07	11.30 / 27.16 / **5.09**
		Gemma	4.72 / 1.89 / 0.95	4.47 / **31.68** / 9.03	1.70 / **20.02** / 6.01
	Temperature	Qwen	**10.67 / 1.76** / 1.00	8.80 / 32.98 / **5.69**	**11.33 / 26.98** / 5.12
		Gemma	**4.74** / 1.88 / 0.98	4.58 / 33.55 / **7.53**	1.71 / 20.19 / **5.68**
	Sigmoid	Qwen	10.64 / 2.22 / 0.95	8.93 / 42.82 / 14.67	**11.33** / 27.79 / 6.02
		Gemma	**4.74 / 1.59** / 0.76	4.58 / 43.62 / 14.42	**1.78** / 24.18 / 8.66
	Isotonic	Qwen	10.63 / 4.16 / **0.92**	**8.97** / 39.95 / 12.88	11.32 / 29.02 / 5.42
		Gemma	**4.74** / 1.68 / **0.57**	**4.61** / 38.56 / 13.53	1.74 / 23.91 / 7.74

B.7 Source-only probability calibration does not reliably improve budget adherence

We test whether standard post-hoc calibration of the router's class probabilities can reduce the fixed-threshold adherence error. For every router, prediction model, assignment, and seed, we fit temperature scaling, one-vs-rest sigmoid calibration, and one-vs-rest isotonic calibration on the same source validation population used to select budget thresholds (Guo et al., 2017; Niculescu-Mizil & Caruana, 2005). Fits use the true router label and give each task family equal total weight. We renormalize the one-vs-rest outputs to sum to one, recover CSR's help probability or CUR's help-minus-harm score, select the six cost-only thresholds on the source validation population, and apply both calibrator and thresholds unchanged to test. Thus, no target or test label, score, or cost enters fitting or threshold selection. ID uses ID validation and test; OOD A and B use the retained-source validation and held-out-target test populations.

Table 10 reports cost–accuracy, cost MAE, and routing-rate MAE. Temperature and sigmoid mappings are strictly monotone for binary CSR, so validation threshold selection induces the same policies as the raw score and leaves all three metrics unchanged. Isotonic CSR changes policies through ties: it reduces Assignment A budget adherence error by about 9 points for both prediction models, but increases it in ID and Assignment B. Its cost–accuracy changes by at most 0.04 points.

For multiclass CUR, calibration can change the ordering because the deployed score is a difference of two calibrated probabilities. Temperature scaling changes $\Delta\mathrm{AUC}$ by only $+0.01$–$+0.11$ points across the six settings, but slightly increases aggregate OOD cost MAE from 27.61 to 28.43 points, while routing-rate MAE falls from 6.80 to 6.00 points. The more flexible sigmoid and isotonic mappings increase cost MAE further to 34.60 and 32.86 points, respectively, even while slightly improving OOD $\Delta\mathrm{AUC}$ in some settings; their routing-rate MAEs also rise to 10.94 and 9.89 points. Source-fitted probability calibration therefore does not generally ensure budget adherence under shift: probability fit, cost–accuracy ranking, cost adherence, and routing-rate adherence are distinct properties.

B.8 Target calibration trades target adherence against source adherence

We distinguish source and target validation and test populations in each assignment. Source populations contain datasets available during router development; target populations contain the excluded datasets. In the primary analysis, we fix the calibration sample at 512 examples and vary the target

count over $\{0, 64, 128, 256, 384, 448, 512\}$, filling the remainder with source validation examples. Sampling cycles across the eight task families without replacement, and we repeat each mixture 100 times. The sample's observed scores and direct, scoring, and reasoning costs select the threshold. As in the main evaluation, this validation sample also defines the family-specific all-reasoning cost references. We freeze both thresholds and references before evaluating the complete source and target test populations. Thus, each mixture uses the same validation-referenced adherence definition as the main analysis, but represents a new predeployment calibration population rather than retaining the original source-only budget reference. No test example selects a threshold, and no correctness label is used for calibration. When the estimated cost of producing the direct prediction and scoring it already exceeds a requested budget, the policy routes no examples to reasoning and its resulting budget deviation remains in the mean. This analysis estimates how much new-population data would be needed after deployment; it does not guarantee a future budget.

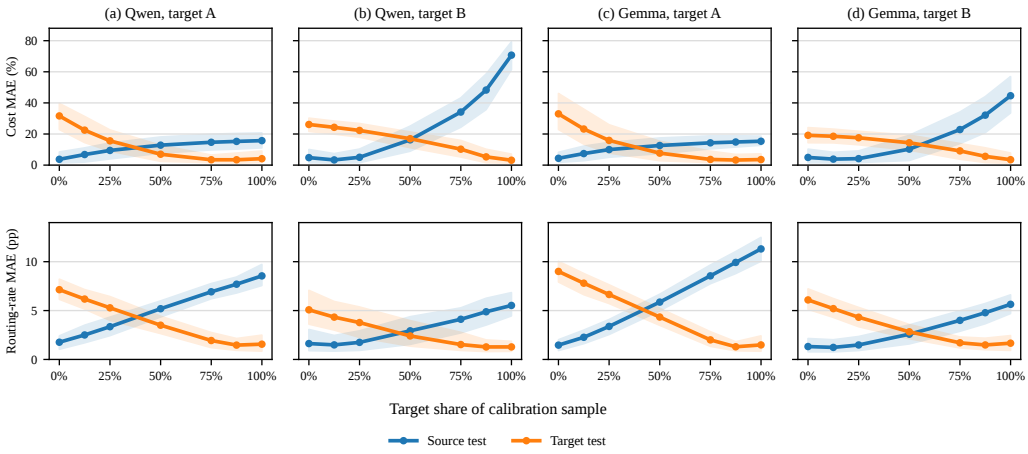

Figure 4: Threshold calibration trades CUR 27B adherence between source and target distributions. Top: validation-referenced cost MAE. Bottom: routing-rate MAE. As target validation occupies more of the fixed calibration sample, both forms of target test error generally fall while source test error rises. Thus, adding target calibration data shifts budget control toward that distribution instead of improving it uniformly. Curves average across three router seeds; bands span the 5th–95th percentiles over 100 validation draws that cycle across task families.

Figure 4 evaluates the calibration question on CUR 27B, the learned 27B router with the lowest aggregate OOD cost MAE among the main supervised methods in Table 2. Increasing the target share moves the selected threshold toward the relationship between scores and costs on the target data. Target test budget adherence error can therefore fall while source test budget adherence error rises. The reverse happens as source validation receives more weight. The pattern and its magnitude differ between Assignments A and B, so a single pooled calibration curve would hide relevant directionality. For Qwen, moving from a source-only to a target-only calibration sample reduces target test adherence error from 31.63 to 4.13 points in Assignment A and from 26.13 to 3.15 in Assignment B; source test adherence error rises from 3.73 to 15.74 and from 4.82 to 70.71, respectively. For Gemma, target test adherence error falls from 32.97 to 3.57 and from 19.17 to 3.48, while source test adherence error rises from 4.38 to 15.35 and from 4.94 to 44.61.

Routing-rate adherence shows the same population trade-off. Across the four model–assignment settings, moving from source-only to target-only calibration reduces target routing-rate MAE from 5.08–9.01 points to 1.28–1.66 points, while source routing-rate MAE rises from 1.32–1.76 points to 5.53–11.31 points. The cost and routing-rate curves need not move by the same amount because trace-cost drift affects only cost MAE.

Holding the total calibration count fixed isolates population mixture from the effect of simply supplying more calibration data. The appropriate mixture therefore depends on which future populations the policy must serve. Every target-validation example additionally requires its observed reasoning cost, including a reasoning call that deployment might otherwise skip. Target examples

therefore shift budget control toward their distribution rather than providing a uniform calibration improvement.

Repeating the analysis with 256 and 1,024 calibration examples preserves the same source–target trade-off; sample size changes its uncertainty but does not remove it.