LLMs Encode Their Failures: Predicting Success from Pre-Generation Activations

William Lugoloobi[1,*], **Thomas Foster**[2], **William Bankes**[3], **Chris Russell**[1]

[1] Oxford Internet Institute, University of Oxford
[2] FLAIR, University of Oxford
[3] Department of Computer Science, University College London

Abstract

Running LLMs with extended reasoning on every problem is expensive, but determining which inputs actually require additional compute remains challenging. We investigate whether their likelihood of success is recoverable from their internal representations prior to generation, and whether this signal can guide more efficient inference. We train linear probes on pre-generation activations to predict policy-specific success on math and coding tasks, substantially outperforming surface features such as question length and TF-IDF. Using E2H-AMC, which provides both human and model performance on identical problems, we show that models encode a model-specific notion of difficulty that is distinct from human difficulty, and that this distinction increases with extended reasoning. Leveraging these probes, we demonstrate that routing queries across a pool of models can exceed the performance of the best-performing model whilst reducing inference cost by up to 70% on MATH, showing that internal representations enable practical efficiency gains even when they diverge from human intuitions about difficulty. Our code is available at: `https://github.com/KabakaWilliam/llms_know_difficulty`

1 Introduction

Large Language Models (LLMs) have achieved remarkable performance on mathematics and programming tasks (Hendrycks et al., 2021; Cobbe et al., 2021; Jain et al., 2024), where correctness can be objectively verified. Because model outputs are typically generated via stochastic decoding, performance is naturally characterized by a *success rate*—the probability that a model will correctly answer a given query. Accurately estimating success rates is critical for model routing systems that direct queries to the model most likely to succeed (Chen et al., 2024; Ding et al., 2023), but obtaining low-variance estimates requires multiple costly rollouts per input. This begs the question; Can we predict whether a model will succeed *before* it generates any output?

In this work, we show that LLMs internally encode estimates of their own success in pre-generation activations, and that these estimates can be efficiently extracted using linear probes. Prior work has shown that models contain correctness-related signals (Kadavath et al., 2022; Azaria & Mitchell, 2023; Burns et al., 2024), but it remains unclear what notion of difficulty these signals represent and whether they are reliable enough for practical decision-making.

We conduct an empirical study across mathematics (MATH, GSM8K, AIME, E2H-AMC) and coding (LiveCodeBench) domains, training linear probes on pre-generation activations to predict success under various decoding policies. Our investigation reveals that LLMs encode a *model-specific* notion of difficulty that differs systematically from human judgments and varies with the inference-time policy. We demonstrate that probe-guided routing can match high-compute accuracy at 40% cost reduction, while identifying critical failure modes where probe reliability becomes the bottleneck.

Main Contributions.

[*]Corresponding author: `william.lugoloobi@oii.ox.ac.uk`

- **Human and model difficulty are encoded differently in LLMs.** Using E2H-AMC, where human IRT difficulty labels and model performance are available on identical questions, we show that linear probes can extract both signals from pre-generation activations (Spearman $\rho = 0.83$–0.87 for human difficulty, 0.40–0.64 for model difficulty). Crucially, these represent *distinct information*: model-derived difficulty proves more predictive of actual performance, and as models solve harder problems through extended reasoning, their internal representations increasingly diverge from human difficulty judgments (Section 3).
- **Probes reliably predict model performance across decoding settings and reasoning modes.** Binary classification of success under fixed decoding policies (greedy, Maj@K) achieves strong discrimination (AUROC > 0.7 for several models) and remains stable across sampling temperatures and majority voting thresholds. However, probe reliability degrades with extended test-time compute, suggesting that difficulty information becomes less linearly accessible as reasoning chains lengthen (Section 3).
- **Probe-guided routing achieves substantial cost savings with minimal accuracy loss.** Simple threshold-based and utility-maximizing routing policies match the highest-capability single-model performance at 70% lower inference cost on MATH, with similar gains on AIME and GSM8K. In some configurations, our router exceeds the best baseline while approaching oracle-level accuracy, demonstrating that reliable difficulty estimates—not routing sophistication—are the key to effective model allocation (Section 4).

2 Related Work

Predicting model correctness. For routing, abstention, and compute allocation decisions, we need estimates of whether a model will answer correctly. Prior work shows that LLMs contain correctness-related signals. Kadavath et al. (2022) demonstrate that models can predict their own correctness when explicitly prompted for P(True)-style self-assessments, but this requires generation overhead unsuitable for routing. A complementary line identifies "truthfulness" or "correctness directions" in internal activations (Azaria & Mitchell, 2023; Burns et al., 2024; Li et al., 2023), which implicitly estimate confidence by predicting correct responses (Geng et al., 2024). Most directly related, Cencerrado et al. (2025) extract correctness directions via difference-of-means between pass and fail activation centroids, then test whether these directions transfer to predict success on new questions. They find strong performance in factual settings but substantially weaker results on mathematical reasoning (GSM8K AUROC $\approx$ 0.6–0.7). We take a different approach: rather than extracting unsupervised directions, we train supervised linear classifiers directly on labeled pass/fail examples to predict binary success. This supervised formulation achieves stronger discrimination on reasoning tasks (AUROC > 0.7 for several models) and enables us to systematically investigate: (1) what notion of difficulty these probes encode, and (2) how probe reliability varies with extended reasoning.

Difficulty estimation. Recent work shows that pre-generation activations contain linearly decodable difficulty signals (Lugoloobi & Russell, 2025; Lee et al., 2025), but it remains unclear whether these represent human difficulty, model-specific difficulty, or both. Lugoloobi & Russell (2025) demonstrate that models encode problem difficulty but focus primarily on correlation with Item Response Theory (IRT; Woodruff & Hanson, 1996) scores—psychometric measures calibrated from large-scale human performance data.Lee et al. (2025) probe difficulty perception mechanisms without systematically comparing human versus model difficulty or evaluating routing applications. We provide the first direct comparison using E2H-AMC, where human Information Response Theory (IRT) scores and model performance are available on identical questions, establishing that these are *distinct* signals. Critically, we show this divergence intensifies with extended reasoning—models allocate computation according to human difficulty even when reliably solving those problems.

Test-time compute scaling. Sampling-based methods like self-consistency (Wang et al., 2022) aggregate k reasoning paths through majority voting (maj@k) to improve accuracy on complex reasoning tasks. Cobbe et al. (2021) train verifiers to re-rank generated solutions, substantially improving performance on math tasks. Recent models with extended reasoning capabilities (e.g., DeepSeek-R1, o1-series) scale test-time compute by generating longer chain-of-thought responses Guo et al. (2025); OpenAI (2024). While prior work focuses on the accuracy-compute tradeoff, we provide the first systematic investigation of how test-time scaling—both through majority voting

and extended reasoning—affects the linear accessibility of difficulty information in pre-generation activations. Our finding that probe quality degrades with increased reasoning budget (AUROC: $0.78 \rightarrow 0.64$) despite improved accuracy ($86.6\% \rightarrow 92.0\%$) has implications for adaptive inference systems that rely on difficulty estimates extracted before generation.

Model routing. Prior routing work relies on indirect proxies like input length, perplexity, or heuristic confidence measures (Chen et al., 2024; Ding et al., 2023). Chen et al. (2024) demonstrate cost reductions using ensemble-based routing with multiple API calls for confidence estimation, while Ding et al. (2023) propose hybrid routing based on input complexity heuristics. Our probe-based approach requires no additional generation at routing time and achieves 17–70% cost savings while matching high-capability model performance. Critically, we identify probe reliability—not routing sophistication—as the primary bottleneck: even oracle routing policies cannot overcome unreliable difficulty estimates from extended reasoning models.

3 Predicting Difficulty

Adaptive systems for model routing and training-data selection depend on accurate difficulty prediction. Prior work shows that pre-generation activations contain linearly decodable signals that anticipate downstream performance and correlate with perceived difficulty Lugoloobi & Russell (2025); Cencerrado et al. (2025); Lee et al. (2025). However, it remains unclear what this signal represents: human difficulty, model-specific difficulty under a particular decoding policy, or a conflation of both.

In this section we disentangle these notions. Using E2H-AMC from the Easy2HardBench dataset (Ding et al., 2024), where we have human IRT difficulty labels and can also estimate model success on the *same* questions via rollouts, we train linear probes for each target from identical pre-generation activations.

We find that both human IRT difficulty and model difficulty are linearly predictable, but they are not the same signal. Furthermore, we distinguish between two related but distinct prediction targets: (i) the expected success rate across multiple stochastic rollouts, and (ii) binary success under a fixed decoding policy (e.g., Maj@K). The former captures model-specific difficulty as a continuous ranking, while the latter enables direct decision-making for routing applications. Finally, we show that both formulations generalize to success prediction on GSM8K, MATH, AIME, and LiveCodeBench, motivating the routing applications in later sections.

3.1 Two Notions of Difficulty

Human difficulty (IRT). On E2H-AMC, each question q is annotated with a human IRT difficulty $b(q)$, where larger values indicate questions that are harder for humans.

Model difficulty: Expected success rate. For a model with stochastic decoding policy π and question q with ground-truth answer y^*, we define the expected success rate as

$$s(\pi, q) \;=\; \mathbb{E}_{a\sim\pi(\cdot|q)}\big[\mathbb{I}(\mathrm{parser}(a) = y^*)\big], \tag{1}$$

where $\mathrm{parser}(\cdot)$ extracts a final answer from response a. We estimate $s(\pi, q)$ with K Monte Carlo rollouts:

$$\hat{s}_{\mathrm{MC}}(\pi, q) \;=\; \frac{1}{K}\sum_{k=1}^{K}\mathbb{I}\big(\mathrm{parser}(a_k) = y^*\big), \tag{2}$$

with $a_k \sim \pi(\cdot \mid q)$ i.i.d. Unless otherwise stated, we use temperature $T = 1$ and $K = 50$. This formulation provides a continuous measure of model-specific difficulty that ranks questions by expected performance under stochastic decoding.

Model success: Binary outcome under specified decoding. For routing and other decision-making applications, we also consider binary success under deterministic aggregation rules. Specifically, we evaluate:

- **Greedy decoding** ($T = 0$): The model either succeeds or fails on a single deterministic generation.

- **Majority voting** (Maj@K): Select the most frequent parsed answer from K samples and verify correctness.

Unlike $\hat{s}_{\mathrm{MC}}$, which estimates a probability, these targets predict whether a specific inference procedure will succeed. Because Maj@K depends on the full answer distribution rather than individual samples, it captures different information about model capability and serves as a distinct prediction target.

3.2 Experimental Setup

E2H-AMC: Controlled human–model difficulty comparison. The AMC subset of Easy2Hard Bench (Ding et al., 2024) contains 4k mathematics problems from the American Mathematics Competitions, spanning algebra, geometry, number theory, and combinatorics. Each question is annotated with a psychometric IRT difficulty score $b(q)$ calibrated from large-scale student performance data, providing a model-agnostic measure of human difficulty.

This dataset is central to our comparison because it uniquely provides both (i) human difficulty labels and (ii) the ability to estimate model-specific difficulty via rollouts on identical questions. We train three types of probes on the same activation features:

- A **human-difficulty probe** predicting $b(q)$ (regression, MSE loss)
- A **success-rate probe** predicting $\hat{s}_{\mathrm{MC}}(\pi, q)$ (regression, MSE loss)
- **Binary success probes** predicting Maj@K or greedy success (classification, BCE loss)

This design enables direct comparison of human difficulty, expected success rate, and policy-specific success within a single model.

Additional benchmarks for model difficulty. To test whether model-difficulty probes generalize beyond E2H-AMC, we construct success-rate datasets using $K = 50$ rollouts per question on: GSM8K, MATH, AIME (1983-2024), and LiveCodeBench (Cobbe et al., 2021; Hendrycks et al., 2021; Veeraboina, 2023; Balunović et al., 2025; Jain et al., 2024). For LiveCodeBench we use contamination-aware temporal splits based on each model's release date.

Linear probe. Let $A \in \mathbb{R}^{S \times D}$ denote residual stream activations (pre-layer norm) from a fixed layer. Following Arditi et al. (2024), we extract activations at post-instruction template positions (the final tokens before generation begins but after the user input).

We train simple linear probes from each layer and position using an 80/20 train–validation split for hyperparameter selection, and report our best probe results on a held-out test set. For success-rate prediction, we use MSE loss; for binary success (Maj@K, greedy) we use binary cross-entropy. We apply Platt scaling on a validation set to calibrate probabilities from our best-trained classification probes.

Additional training details are in Appendix 5.1

Baselines and metrics. We compare against text-only baselines: TF-IDF features fed into a linear model and question length. For success-rate prediction ($\hat{s}_{\mathrm{MC}}$), we report Spearman rank correlation, since ordering is the key requirement for curriculum learning and prioritisationå. For binary success prediction (Maj@K, greedy) we report AUROC, as this measures discrimination quality for routing decisions.

3.3 Results

Human and model difficulty are both linearly decodable but encode different information. Table 1 shows that linear probes can predict both human IRT difficulty and model success rates from identical pre-generation activations, but with different levels of accessibility. Human difficulty is consistently more linearly decodable (Spearman $\rho = 0.83$–0.87) than model success rate ($\rho = 0.40$–0.64) across all models. This suggests that models robustly encode what humans find difficult, even when that differs from their own performance characteristics.

Table 1: **Human and model difficulty are both linearly decodable but represent distinct signals.** Difficulty prediction performance (Spearman ρ) on E2H-AMC comparing human IRT difficulty $b(q)$ versus model-specific difficulty $\hat{s}_{\mathrm{MC}}$ from identical pre-generation activations. Linear probes substantially outperform text-based baselines (TF-IDF, question length) for both targets. Human IRT difficulty is consistently more linearly accessible ($\rho = 0.83$–0.87) than model difficulty ($\rho = 0.40$–0.64) across all models. Critically, model difficulty becomes *less* linearly accessible as reasoning capability increases: for GPT-OSS-20B, ρ drops from 0.58 (low) to 0.40 (high) despite improved accuracy, suggesting that extended chain-of-thought obscures pre-generation difficulty signals. Model difficulty estimated from $K = 50$ rollouts for Qwen models; $K = 5$ for GPT-OSS-20B due to computational cost.

	Qwen2.5-Math		**GPT-OSS-20B**		
Method	**1.5B**	**7B**	**Low**	**Med**	**High**
Human IRT difficulty $b(q)$					
Linear Probe	0.85	0.87	0.84	0.83	0.83
TF–IDF	0.72	0.72	0.74	0.74	0.74
Length	0.15	0.15	0.15	0.15	0.15
Model difficulty $\hat{s}_{\mathrm{MC}}$					
Linear Probe	0.64	0.64	0.58	0.50	0.40
TF–IDF	0.47	0.47	0.42	0.31	0.25
Length	0.27	0.27	0.30	0.24	0.19

Critically, we observe that model success rate becomes *less* linearly accessible as reasoning capability increases. For GPT-OSS-20B, probe performance drops from $\rho = 0.58$ (low reasoning) to $\rho = 0.40$ (high reasoning), despite the model achieving higher accuracy. This suggests that extended chain-of-thought may encode difficulty information in ways that are not linearly separable at the pre-generation stage, foreshadowing the routing challenges we address in Section 4.

Table 2: Probe performance (AUROC) and task accuracy across models and inference regimes in Math and Coding domains. **Task Acc.** shows the model's average benchmark performance. **Math**: AUROC averaged over MATH, GSM8K, and AIME-2025, comparing Greedy vs. Maj@5. For GPT-OSS-20B we fix Maj@5 and vary the internal reasoning levels. **Code**: LiveCodeBench with target Pass@5 (a problem is correct if any of 5 sampled generations passes all test cases). Per-dataset results are reported in Appendix 5.

Model	Inference Regime	Task Acc. ↑	Linear ↑	TF-IDF ↑	Length ↑
Math: Greedy vs. Maj@5					
Qwen2.5-Math-1.5B	Greedy	0.724	0.84	0.64	0.61
	Maj@5	0.763	0.76	0.63	0.66
Qwen2.5-Math-7B	Greedy	0.809	0.79	0.68	0.67
	Maj@5	0.827	0.80	0.72	0.66
Qwen2.5-1.5B	Greedy	0.525	0.68	0.63	0.73
	Maj@5	0.583	0.85	0.65	0.69
Math: Maj@5 with Variable Reasoning Budget					
GPT-OSS-20B	Reasoning: Low	0.866	0.78	0.68	0.62
	Reasoning: Medium	0.914	0.70	0.63	0.55
	Reasoning: High	0.920	0.64	0.58	0.46
Code: LiveCodeBench (target = Pass@5)					
Qwen2.5-Coder-3B	Pass@5	0.14	0.91	0.86	0.64
Qwen2.5-Coder-7B	Pass@5	0.15	0.90	0.84	0.61
DeepSeek-R1-Distill-Qwen-7B	Pass@5	0.30	0.81	0.83	0.59
GPT-OSS-20B (low)	Pass@5	0.77	0.71	0.66	0.64
GPT-OSS-20B (medium)	Pass@5	0.77	0.67	0.64	0.60
GPT-OSS-20B (high)	Pass@5	0.79	0.69	0.64	0.62

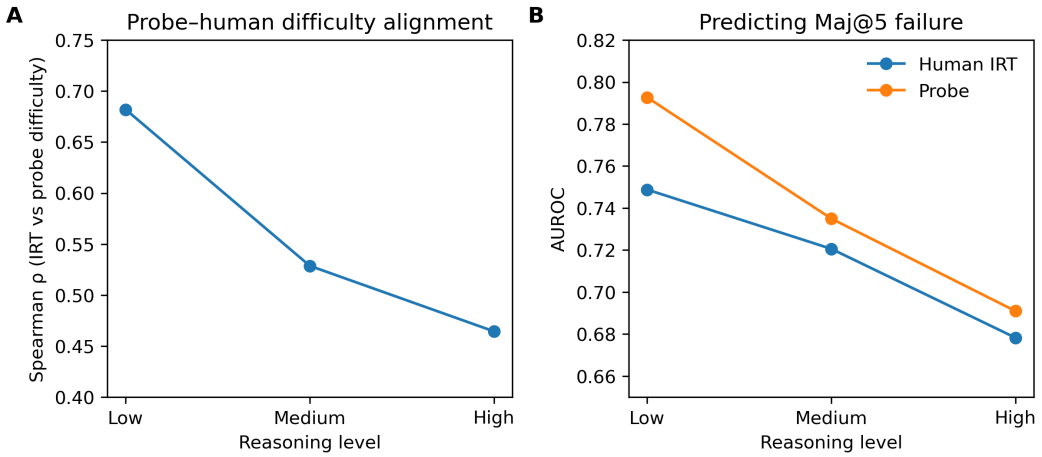

Figure 1: **Human and model difficulty diverge with increased reasoning.** On E2H-AMC, as the reasoning level in GPT-OSS-20B is increased, difficulty becomes less human-aligned and more model-specific. **Left: (A)** Alignment between probe-predicted model difficulty and human IRT difficulty decreases with higher reasoning, indicating that correctness-related signals become less linearly accessible as models solve questions that are typically difficult for humans. **Right: (B)** Despite this divergence, probe-based predictions consistently outperform human difficulty for predicting Maj@5 failure across reasoning modes, demonstrating that internal activations encode a model-relative notion of difficulty that is distinct from human difficulty.

Binary success under specified decoding policies is more predictable than success rate. While success-rate prediction shows moderate correlation (Table 1), binary classification of success under fixed decoding policies achieves substantially stronger discrimination. Table 2 shows that probes predicting Maj@5 or greedy success achieve AUROC > 0.7 across most settings, with several exceeding 0.8.

We observe three key patterns:

1. **Greedy vs. sampling**: Greedy decoding generally yields higher probe AUROC than Maj@5 for the same model (e.g., Qwen2.5-Math-1.5B: 0.84 vs 0.76), likely because deterministic generation reduces noise in the prediction target.
2. **Model capability matters**: Smaller or less capable models (e.g., Qwen2.5-1.5B, base variant) show stronger probe performance for Maj@5 than greedy, suggesting that sampling-based aggregation helps models solve problems they find marginally difficult, and this regime is easier to predict.
3. **Reasoning budget degrades probe quality**: For GPT-OSS-20B, increasing the reasoning level from low to high decreases AUROC from 0.78 to 0.64 even under fixed Maj@5 decoding. This pattern, which we also saw for success-rate prediction, indicates that extensive chain-of-thought reasoning obscures pre-generation difficulty signals.

Code domain shows high probe quality. On LiveCodeBench with Pass@5 as the target, we observe strong probe performance (AUROC $= 0.81$–0.91) for Qwen2.5-Coder and DeepSeek-R1 models. However, GPT-OSS-20B again shows weaker probe quality (AUROC ≈ 0.67), consistent with the pattern in math domains. This suggests that probe accessibility is a model-family property that generalizes across domains rather than a task-specific phenomenon.

Human and model difficulty diverge with increased reasoning. Figure 1 illustrates how the relationship between human and model difficulty changes as GPT-OSS-20B's reasoning budget increases. Panel (A) shows that alignment between probe-predicted model difficulty and human IRT difficulty decreases monotonically with reasoning level (Spearman ρ drops from $\sim$0.65 to $\sim$0.45). This indicates that as models become better at solving human-hard problems through extended reasoning, their internal notion of difficulty diverges from human judgments.

Panel (B) demonstrates that despite this divergence, probe-based predictions of model difficulty consistently outperform human difficulty for predicting actual Maj@5 failures across all reasoning modes. This confirms that models encode a model-relative notion of difficulty that is distinct from, and more predictive of their own performance than, human difficulty.

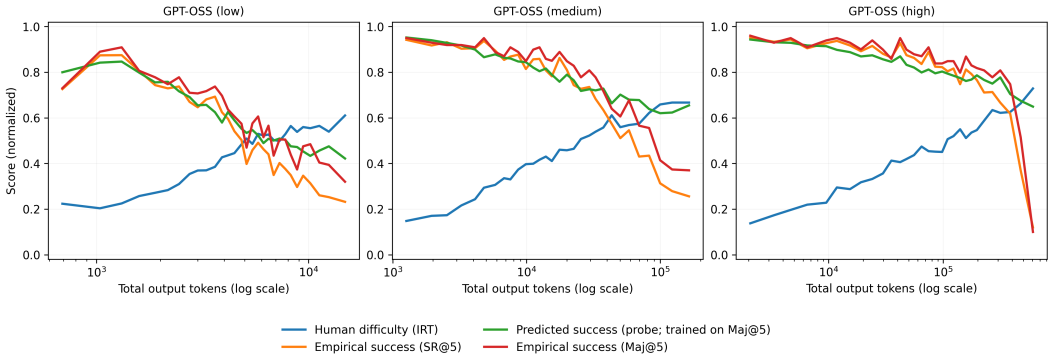

Figure 2: **Chain-of-thought length tracks human difficulty but diverges from model success.** We plot binned chain-of-thought length (total output tokens, log-scale) against expected values (means) of normalized human IRT difficulty, empirical correctness, empirical success rates, and probe-predicted success (SR@5 and Maj@5) for GPT-OSS-20B at low, medium, and high reasoning modes. Across all settings, output length is positively correlated with human difficulty and negatively correlated with both empirical and predicted success. This effect strengthens with increased reasoning mode, indicating that extended reasoning causes models to allocate more computation to problems humans find difficult, even when those problems are unlikely to be failed. thereby decoupling generation length from model-relative uncertainty.

Reasoning length reflects human difficulty rather than model uncertainty. To understand why human and model difficulty decouple under extended reasoning, we examine how chain-of-thought length (total output tokens) relates to human difficulty, empirical success, and probe-predicted model success across reasoning budgets.Figure 2 shows that as reasoning depth increases, output length becomes increasingly correlated with human IRT difficulty, while simultaneously becoming negatively-correlated with both empirical success and probe-predicted success. For GPT-OSS, this pattern is consistent across reasoning modes and strengthens at higher budgets: the model spends more tokens on problems humans find difficult, even when those problems are well within the model's competence.

As a result, extended reasoning amplifies a human-aligned difficulty signal in generation dynamics that is distinct from the model's own likelihood of success, helping explain why probe-predicted model difficulty remains useful even as alignment with human difficulty deteriorates.

4 Probe-Guided Routing

Prior work on routing between models with different capabilities and inference costs typically relies on indirect proxies for difficulty, such as input length, perplexity, or heuristic confidence measures Chen et al. (2024); Ding et al. (2023). We demonstrate that probe-derived success estimates enable effective routing decisions, yielding meaningful performance-cost tradeoffs in both cascade and utility-based settings.

4.1 Routing Strategies

We evaluate two probe-based routing strategies that use predicted success probabilities $\hat{p}_M(x)$ to allocate queries across models with different capabilities and costs. All experiments use maj@5 accuracy with K=5 generations on MATH Hendrycks et al. (2021), and probes trained to predict maj@5 success as described in Section 3.

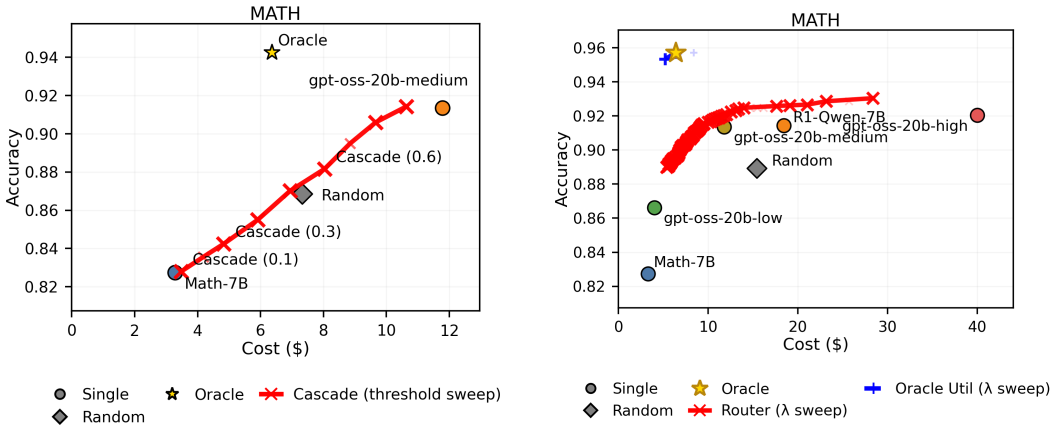

Figure 3: **Probe-based routing achieves strong performance-cost tradeoffs on MATH. Left (Cascade):** Binary routing between Qwen2.5-Math-7B-Instruct and GPT-OSS-20B-medium. The cascade strategy (red curve) substantially outperforms random routing (gray diamond) across the Pareto frontier, matching GPT-OSS-20B-medium accuracy (orange circle) at 17% lower cost. **Right (Utility):** Model selection from a pool of five models with varying capabilities and costs. The utility router (red curve) achieves a Pareto improvement over all single-model baselines, exceeding GPT-OSS-20B-high accuracy (red circle) while reducing cost by approximately 70%. Both strategies route difficult queries (low $\hat{p}$) to more capable models. Oracle performance (gold star) represents an upper bound with perfect difficulty prediction. All results use maj@5 with $K = 5$ generations.

Cascade Routing Let M_s denote a base model and M_l a stronger model with higher inference cost. For each input x, we use a threshold-based rule:

$$M(x) = \begin{cases} M_l & \text{if } \hat{p}_s(x) < \tau \\ M_s & \text{otherwise} \end{cases}$$

where $\hat{p}_s(x)$ is the probe's estimated probability that M_s will answer correctly, and $\tau \in [0, 1]$ controls the tradeoff between performance and cost. Higher τ escalates more queries to M_l, increasing both accuracy and cost.

We evaluate cascade routing with Qwen2.5-Math-1.5B as M_s and Qwen2.5-Math-7B as M_l. This formulation naturally extends to multi-stage cascades with ordered model sets $\{M_1, \ldots, M_K\}$, where queries escalate sequentially until a model's estimated success probability exceeds its threshold.

Utility-Based Routing For routing among a heterogeneous pool of models, we use a simple utility-based rule. Let $\{M_1, \ldots, M_K\}$ denote available models with expected costs $\{\hat{c_1}, \ldots, \hat{c_K}\}$ based on average output cost from the train set. We normalise the expected cost $\hat{c}_i$ to be between [0-1] such that for a prompt x, we select:

$$\hat{M}(x) = \arg\max_i \left(\hat{p}_i(x) - \lambda \hat{c}_i\right)$$

where $\hat{p}_i(x)$ is the probe-estimated success probability for model M_i, and λ trades off success probability against cost. This rule requires training separate probes for each model in the pool.

We evaluate on a pool of five models: Qwen2.5-Math-7B-Instruct, Deepseek-R1-Qwen-7B, and GPT-OSS-20B with low/medium/high reasoning budgets. We vary λ to trace the performance-cost frontier, without tuning for specific operating points. Following prior routing work that uses public API pricing to estimate deployment costs (Chen et al., 2024; Ding et al., 2023), we compute costs from total output tokens using Fireworks AI's pricing—a platform that provides realistic cost estimates for deploying open-source models at scale (see Appendix 5.4).

Baselines We compare against two baselines: random routing and an oracle with perfect knowledge of model success. Random routing assigns each problem uniformly at random to one of the available models, independent of difficulty or cost. For utility-based routing, the oracle replaces probe predictions with ground-truth correctness labels ($p(x) = [\text{correct}_i(x)]$) and selects

$\hat{M}(x) = \arg\max_i(\mathbb{I}[\text{correct}_i(x)] - \lambda\hat{c}_i)$ sweeping λ to trace the theoretical best-case Pareto frontier. For cascade routing, the oracle iterates through models from cheapest to most expensive and routes to the cheapest model that solves the problem correctly, escalating only on actual failures. If no model succeeds, it defaults to the cheapest.

4.2 RESULTS

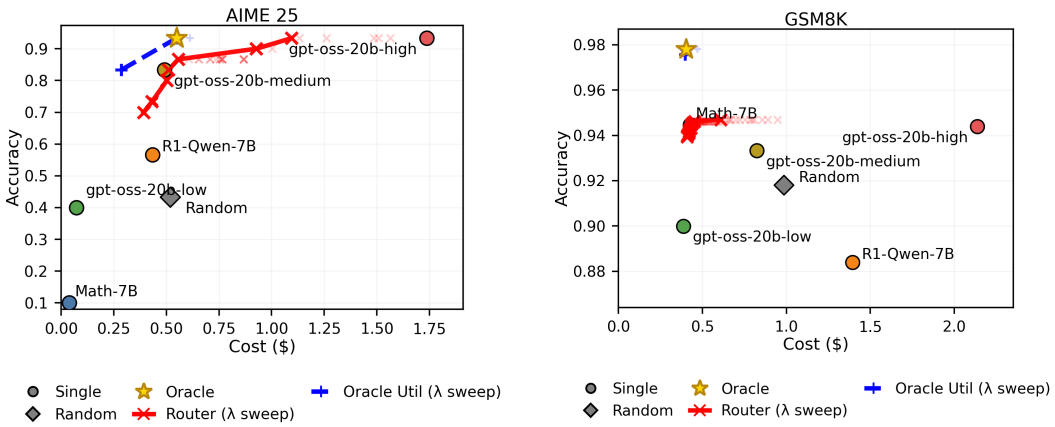

Figure 4: **Probe-based routing generalizes across diverse reasoning benchmarks. Left (AIME 2025):** Utility-based routing on a challenging competition mathematics benchmark. The router (red curve) traces a Pareto frontier that dominates all individual models, matching GPT-OSS-20B-high's 93.3% accuracy (red circle) at approximately 37% lower cost ($1.15 vs $1.75). The oracle (gold star, 95.6%) represents the theoretical upper bound with perfect prediction, while oracle utility (blue dashed line) shows the cost-optimal oracle policy. Our router matches oracle accuracy but at a higher cost. **Right (GSM8K):** Utility-based routing on a saturated benchmark with models achieving high accuracies. The router (red curve) substantially outperforms random routing (gray diamond) and efficiently identifies the cost-optimal operating point near Math-7B (red crosses, 94.5%). In this saturated regime, the router correctly avoids expensive high-reasoning models (GPT-OSS-20B-high: 94.4% at $2.4) in favor of the cheaper Math-7B with comparable accuracy—demonstrating cost-aware selection rather than simple accuracy maximization. Both benchmarks show that probe-guided routing adapts to difficulty distributions: preferring stronger models when accuracy varies widely (AIME), and selecting efficient models when performance plateaus (GSM8K). All results use maj@5 with K=5 generations. The full table of results is in the Appendix Section 5.5.

Cascade and utility routing achieve strong cost-accuracy tradeoffs. Figure 3 demonstrates probe-based routing on MATH. The cascade strategy (left) routes between Qwen2.5-Math-7B and GPT-OSS-20B-medium, substantially outperforming random allocation across the Pareto frontier. At threshold τ=0.6, the cascade matches GPT-OSS-20B-medium's 91.2% accuracy while reducing cost by 17%. Utility-based routing (right) over five heterogeneous models achieves even stronger gains: matching GPT-OSS-20B-high's 92% accuracy at $15; a 70% reduction from the $28 cost of using GPT-OSS-20B-high exclusively. The full set of results is in the Appendix Section 5.5.

Routing adapts to benchmark difficulty characteristics. Figure 4 shows generalization across benchmarks with different difficulty distributions. On AIME 2025 (left), where model accuracies range from 40% to 93%, utility routing matches the strongest model's performance at 37% cost reduction ($1.15 vs $1.75). On GSM8K (right), where models achieve high accuracies (85–95%), the router identifies the cost-optimal point: Math-7B's 94.5% at $0.34 makes expensive high-reasoning models unnecessary (GPT-OSS-20B-high: 94.4% at $2.4). This demonstrates cost-aware optimization—the router preferentially allocates to stronger models when accuracy varies widely, but selects efficient models when performance plateaus.

4.3 DISCUSSION

Reasoning changes what difficulty means, not just performance Our results show that increased test-time reasoning fundamentally changes how difficulty is represented in LLMs. While extended reasoning improves task accuracy, it consistently reduces the linear accessibility of pre-generation success signals. Across reasoning modes in GPT-OSS-20B, probe AUROC drops monotonically as reasoning budgets increase, even as accuracy improves. Analysis of chain-of-thought length reveals a key mechanism: with deeper reasoning, generation length becomes increasingly correlated with human difficulty rather than the model's own likelihood of failure. As a result, reasoning traces amplify human-aligned difficulty signals that decouple from model-relative uncertainty, explaining why probes degrade precisely when reasoning is most effective.

Human difficulty and model difficulty are distinct—and diverge with capability The divergence between human and model difficulty has broader implications beyond routing. Using E2H-AMC, we show that LLMs robustly encode human psychometric difficulty even when that signal no longer predicts model failure. As reasoning capability increases, models increasingly solve problems that humans find difficult, yet their internal representations continue to track human-aligned difficulty through longer reasoning traces. This creates a growing mismatch: human difficulty remains linearly accessible, while model-relative difficulty becomes harder to extract during extended reasoning. For applications such as curriculum learning, data selection, or evaluation, this suggests that human difficulty labels may increasingly mischaracterise what models actually find challenging.

Routing effectiveness is mediated by probe reliability. Across datasets, probe-guided routing approaches oracle-utility performance when probes achieve high discrimination (AUROC), but exhibits substantial gaps to oracle performance when probe quality degrades. This pattern suggests that the effectiveness of simple routing policies is constrained by the reliability of the underlying success estimates, rather than by model capability alone. Notably, even in these regimes, the probe consistently identifies which models are cost-effective for a given dataset, selecting cheaper models on saturated benchmarks (GSM8K) and higher-capability models on more challenging tasks (AIME). Together, these results indicate that probe-based success estimates are informative for routing, while improved probe reliability and calibration are likely required to close the remaining oracle gap.

4.4 CONCLUSION, LIMITATIONS AND FUTURE WORK

We show that simple linear probes trained on model activations can estimate model-relative difficulty and support effective routing decisions. Across tasks, we find that human and model notions of difficulty diverge, with model-derived difficulty providing a more reliable predictor of performance. Probes generalize across decoding strategies and reasoning modes, but their reliability degrades as test-time compute increases. When probe estimates are reliable, probe-guided routing achieves substantial cost savings, matching the strongest single-model baselines at 17–70% lower cost on MATH, and approaching oracle-level performance in several settings.

Limitations. We focus on linear probes applied at a single post-instruction position, following prior work. While effective for base and lightly instruction-tuned models, this design has clear limitations. Probe performance degrades under extended reasoning, and we do not explore alternative probing positions or non-linear probes. We also do not study cross-domain or cross-dataset probe transfer (e.g., MATH to GSM8K or math to code), leaving open questions about generalization. Additionally, our probes are sensitive to the token position we probe from, which constrains their usability. Finally, our routing policies are intentionally simple (fixed-k majority voting with threshold-based and utility-based rules) rather than learned or adaptive routing strategies.

Future work. Several directions follow naturally. First, exploring non-linear probes may reveal whether difficulty information is lost under extended reasoning or merely becomes less linearly accessible. Second, probing at multiple or intermediate generation positions could recover signals that are absent pre-generation. Third, understanding cross-domain and cross-dataset probe transfer would enable more practical, reusable probes. Finally, adaptive routing strategies that dynamically adjust k based on estimated difficulty or combine probe signals with input features may further narrow the gap to oracle performance. That said, our results suggest a clear bottleneck: improvements in routing will ultimately depend on improving the reliability of the underlying difficulty estimates.

REFERENCES

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in Language Models Is Mediated by a Single Direction, October 2024. URL http://arxiv.org/abs/2406.11717. arXiv:2406.11717 [cs].

Amos Azaria and Tom Mitchell. The Internal State of an LLM Knows When It's Lying, October 2023. URL http://arxiv.org/abs/2304.13734. arXiv:2304.13734 [cs].

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on Uncontaminated Math Competitions, February 2025. URL https://matharena.ai/.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering Latent Knowledge in Language Models Without Supervision, March 2024. URL http://arxiv.org/abs/2212.03827. arXiv:2212.03827 [cs].

Iván Vicente Moreno Cencerrado, Arnau Padrés Masdemont, Anton Gonzalvez Hawthorne, David Demitri Africa, and Lorenzo Pacchiardi. No Answer Needed: Predicting LLM Answer Accuracy from Question-Only Linear Probes, September 2025. URL http://arxiv.org/abs/2509.10625. arXiv:2509.10625 [cs].

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*, July 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=cSimKw5p6R.

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.

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. October 2023. URL https://openreview.net/forum?id=02f3mUtqnM.

Mucong Ding, Chenghao Deng, Jocelyn Choo, Zichu Wu, Aakriti Agrawal, Avi Schwarzschild, Tianyi Zhou, Tom Goldstein, John Langford, Anima Anandkumar, and Furong Huang. Easy2Hard-Bench: Standardized Difficulty Labels for Profiling LLM Performance and Generalization, September 2024. URL http://arxiv.org/abs/2409.18433. arXiv:2409.18433 [cs].

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A Survey of Confidence Estimation and Calibration in Large Language Models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6577–6595, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.366. URL https://aclanthology.org/2024.naacl-long.366/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

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. *arXiv preprint*, 2024.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language Models (Mostly) Know What They Know, November 2022. URL http://arxiv.org/abs/2207.05221. arXiv:2207.05221 [cs].

Sunbowen Lee, Qingyu Yin, Chak Tou Leong, Jialiang Zhang, Yicheng Gong, Shiwen Ni, Min Yang, and Xiaoyu Shen. Probing the Difficulty Perception Mechanism of Large Language Models, October 2025. URL http://arxiv.org/abs/2510.05969. arXiv:2510.05969 [cs].

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-Time Intervention: Eliciting Truthful Answers from a Language Model. November 2023. URL https://openreview.net/forum?id=aLLuYpn83y.

William Lugoloobi and Chris Russell. LLMs Encode How Difficult Problems Are, October 2025. URL http://arxiv.org/abs/2510.18147. arXiv:2510.18147 [cs].

OpenAI. Learning to reason with LLMs, September 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

Hemish Veeraboina. gneubig/aime-1983-2024 · Datasets at Hugging Face, 2023. URL https://huggingface.co/datasets/gneubig/aime-1983-2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. September 2022. URL https://openreview.net/forum?id=1PL1NIMMrw.

David J. Woodruff and Bradley A. Hanson. Estimation of Item Response Models Using the EM Algorithm for Finite Mixtures. Technical report, ACT Research Report Series, P, September 1996. URL https://eric.ed.gov/?id=ED405356. ERIC Number: ED405356.

5 APPENDIX

5.1 PROBING FORMULATION

Linear probe. Let $h_i^{(\ell)} \in \mathbb{R}^D$ denote the residual-stream hidden state at layer ℓ and token position i from the frozen language model. These are the internal representations we probe.

Finding end-of-instruction positions. We identify where instructions end by applying the model's chat template to a placeholder input, then tokenizing the post-instruction suffix. This gives us P token positions relative to the last non-padding token: $\{-P, \ldots, -1\}$. These are our candidate positions to probe.

Example suffixes:

Qwen2.5: `<|im_end|>\n<|im_start|>assistant\n`
DeepSeek-R1: `<|Assistant|><think|>\n`
GPT-OSS: `<|end|><|start|>assistant`

Training probes. For each candidate pair $(\ell, p) \in \mathcal{L} \times \mathcal{P}$, where $\mathcal{L}$ spans all transformer layers and $\mathcal{P}$ the EOI positions, we train a single linear probe on the activation vector $h_p^{(\ell)}$. The probe has no bias term, just a linear map from the D-dimensional activation to predictions.

The task type is determined automatically: for continuous success-rate targets we use Ridge regression (evaluated with Spearman's ρ); for binary correctness labels we use ℓ_2-regularized logistic regression (evaluated with ROC-AUC). The regularisation strength α is tuned via grid search on validation data over the range $\alpha \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$.

Data and evaluation. We hold out 20% of the original training set as validation. The best configuration (ℓ^*, p^*, α^*) is selected by validation performance. For classification, we apply Platt scaling on validation data to calibrate probabilities and report expected calibration error before and after. Test evaluation happens exactly once with the selected probe.

5.2 PROBE PERFORMANCE

Table 3: AUROC for Predicting Accuracy under Different Decoding Strategies

Decoding	Dataset	Qwen2.5-Math-1.5B	Qwen2.5-Math-7B
Linear Probe			
Maj@5	MATH	0.838	0.845
	AIME	0.712	0.765
	GSM8K	0.717	0.784
Greedy	MATH	0.837	0.827
	AIME	0.911	0.753
	GSM8K	0.762	0.778
Tfidf Probe			
Maj@5	MATH	0.771	0.761
	AIME	0.529	0.840
	GSM8K	0.582	0.568
Greedy	MATH	0.762	0.760
	AIME	0.589	0.741
	GSM8K	0.581	0.537
Length Probe			
Maj@5	MATH	0.70	0.68
	AIME	0.66	0.72
	GSM8K	0.62	0.59
Greedy	MATH	XX	XX
	AIME	XX	XX
	GSM8K	XX	XX

Table 4: Maj@5 Probe Performance Comparison across GPT-OSS-20B thinking modes

Difficulty	Dataset	Low	Medium	High
Linear Probe				
gpt-oss-20b	MATH-lighteval	0.848	0.842	0.855
	AMC	0.793	0.735	0.691
	AIME	0.731	0.600	0.375
	GSM8K	0.767	0.660	0.680
Tfidf Probe				
gpt-oss-20b	MATH-lighteval	0.771	0.781	0.783
	AMC	0.692	0.649	0.625
	AIME	0.602	0.528	0.339
	GSM8K	0.670	0.580	0.627

Table 5: Benchmark Performance across Different Models

GPT-OSS-20B Performance				
Difficulty	Dataset	Low	Medium	High
gpt-oss-20b	MATH-lighteval	0.866	0.914	0.920
	AMC	0.603	0.778	0.837
	AIME	0.620	0.913	0.963
	GSM8K	0.900	0.933	0.944
	AIME2025	0.400	0.833	0.933
Other Models Performance				
Decoding	Model	Dataset	Maj@5	Greedy
Comparison	Qwen2.5-1.5B -Instruct	MATH-lighteval	0.583	0.525
		AIME	0.072	0.059
		GSM8K	0.758	0.687
		AIME2025	0.033	0.000
	Qwen2.5-Math -1.5B-Instruct	MATH-lighteval	0.763	0.724
		AIME	0.320	0.278
		GSM8K	0.855	0.835
		AIME2025	0.167	0.067
	Qwen2.5-Math -7B-Instruct	MATH-lighteval	0.827	0.809
		AIME	0.340	0.281
		GSM8K	0.945	0.937
		AIME2025	0.100	0.100

5.3 Optimal Model Settings

All rollouts were performed using VLLM with the following configurations:

Model	**Max Length**	**Temperature**	**k**
GPT-OSS-20B	131,072	1.0	5
DeepSeek-R1-Distill-Qwen-7B	32,768	0.6	5
Qwen2.5-Math-1.5B-Instruct	3,000	0.7	5
Qwen2.5-Math-7B-Instruct	3,000	0.7	5
Qwen2.5-Coder-XB-Instruct	4,096	0.2	5

Table 6: Hyperparameters used for model rollouts. All models were evaluated using VLLM with maj@k sampling, where k denotes the number of samples generated per problem.

5.4 Routing Setup

Table 7: Fireworks AI Pricing

Model Size / Type	**Price (USD per million tokens)**
Less than 4B parameters	$0.10
4B – 16B parameters	$0.20
More than 16B parameters	$0.90
OpenAI gpt-oss-20b	$0.07 (input), $0.30 (output)

5.5 Routing Tables

Table 8: Routing strategies comparison on MATH.

Strategy	Model	Accuracy	Cost
Single-model baselines			
Math-7B	Qwen2.5-Math-7B-Instruct	0.827	3.28
gpt-oss-20b-low	gpt-oss-20b (low)	0.866	4.00
gpt-oss-20b-medium	gpt-oss-20b (medium)	0.914	11.78
R1-Qwen-7B	DeepSeek-R1-Distill-Qwen-7B	0.914	18.43
gpt-oss-20b-high	gpt-oss-20b (high)	0.920	40.00
Routing strategies			
Random Routing	ensemble	0.889	15.43
Oracle (Perfect Knowledge)	ensemble	0.957	6.37
Probe router			
Probe Router ($\lambda = 0.00$)	ensemble	0.930	28.37
Probe Router ($\lambda = 0.20$)	ensemble	0.917	10.35
Probe Router ($\lambda = 0.40$)	ensemble	0.909	8.29
Probe Router ($\lambda = 0.60$)	ensemble	0.902	7.02
Probe Router ($\lambda = 0.80$)	ensemble	0.894	5.98
Probe Router ($\lambda = 1.00$)	ensemble	0.890	5.34

Table 9: Routing strategies comparison on AIME 25.

Strategy	Model	Accuracy	Cost
Single-model baselines			
Math-7B	Qwen2.5-Math-7B-Instruct	0.100	0.04
gpt-oss-20b-low	gpt-oss-20b (low)	0.400	0.07
R1-Qwen-7B	DeepSeek-R1-Distill-Qwen-7B	0.567	0.43
gpt-oss-20b-medium	gpt-oss-20b (medium)	0.833	0.49
gpt-oss-20b-high	gpt-oss-20b (high)	0.933	1.74
Routing strategies			
Random Routing	ensemble	0.433	0.52
Oracle (Perfect Knowledge)	ensemble	0.933	0.55
Probe router			
Probe Router ($\lambda = 0.00$)	ensemble	0.933	1.57
Probe Router ($\lambda = 0.20$)	ensemble	0.867	0.74
Probe Router ($\lambda = 0.40$)	ensemble	0.800	0.50
Probe Router ($\lambda = 0.60$)	ensemble	0.733	0.44
Probe Router ($\lambda = 0.80$)	ensemble	0.733	0.43
Probe Router ($\lambda = 1.00$)	ensemble	0.700	0.39

Table 10: Routing strategies comparison on GSM8K.

Strategy	Model	Accuracy	Cost
Single-model baselines			
gpt-oss-20b-low	gpt-oss-20b (low)	0.900	0.39
Math-7B	Qwen2.5-Math-7B-Instruct	0.945	0.43
gpt-oss-20b-medium	gpt-oss-20b (medium)	0.933	0.82
R1-Qwen-7B	DeepSeek-R1-Distill-Qwen-7B	0.884	1.39
gpt-oss-20b-high	gpt-oss-20b (high)	0.944	2.14
Routing strategies			
Random Routing	ensemble	0.918	0.99
Oracle (Perfect Knowledge)	ensemble	0.978	0.40
Probe router			
Probe Router ($\lambda = 0.00$)	ensemble	0.940	0.95
Probe Router ($\lambda = 0.20$)	ensemble	0.946	0.55
Probe Router ($\lambda = 0.40$)	ensemble	0.945	0.47
Probe Router ($\lambda = 0.60$)	ensemble	0.942	0.43
Probe Router ($\lambda = 0.80$)	ensemble	0.941	0.42
Probe Router ($\lambda = 1.00$)	ensemble	0.939	0.41