

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

William Lugoloobi^{1,*}, Thomas Foster^{2,†}, William Bankes^{3,†}, Chris Russell¹

¹ Oxford Internet Institute, University of Oxford

² FLAIR, University of Oxford

³ Department of Computer Science, University College London

 [Code](#)  [Probes](#)

Abstract

Running LLMs with extended reasoning on every problem is expensive, but determining which inputs actually require additional compute remains challenging. We investigate whether a model’s likelihood of success is recoverable from its 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 match 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.

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). 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 invites the question: Can we predict whether a model will succeed *before* it generates any output?

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

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

[†]equal contribution

match high-compute accuracy at 70% cost reduction, while identifying critical failure modes where probe reliability becomes the bottleneck.

Main Contributions.

- **Human and model difficulty are encoded differently in LLMs.** Using the AMC subset of Easy2Hard-Bench (E2H-AMC), which provides both human difficulty labels (estimated via Item Response Theory, IRT) and model accuracy on identical questions, we show that linear probes can extract both signals from pre-generation activations (Spearman $\rho = 0.83\text{--}0.87$ for human difficulty, $0.40\text{--}0.64$ 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, though this is partially recoverable with non-linear MLP probes (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 5).

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 $\approx 0.6\text{--}0.7$). 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 IRT (Embretson, 2000; 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 the AMC subset of Easy2Hard-Bench (Ding et al., 2024), where human IRT scores and model performance are available, 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 such as input length, perplexity, or heuristic confidence measures (Chen et al., 2024; Ding et al., 2023). Chen et al. (2024) use multiple API calls to estimate confidence, while Ding et al. (2023) route based on input complexity heuristics. Song et al. (2025) propose an IRT-based approach that models question difficulty and model ability via learned latent traits from embeddings. In contrast, our probe-based method directly predicts model-specific success from pre-generation representations, requiring no additional generation or separate embedding models at routing time. This yields 17–70% cost savings while matching high-capability model performance. Critically, we find that routing effectiveness is limited by the reliability of the underlying success estimates, not the routing policy itself.

3 Predicting Difficulty

Difficulty is subjective: it depends on the actor facing the task. Just as students in a class find different questions hard, models have their own strengths and weaknesses. Difficulty also depends on what we count as success; a task that is hard on a single attempt may be much easier when several are allowed. Given this subjectivity, we consider a range of human- and model-derived difficulty definitions, and introduce activation probes as a tool for examining how a model perceives each of them.

3.1 Different Notions of Difficulty

Human difficulty. IRT models question difficulty in a similar manner to Elo in games such as chess. The IRT difficulty score $b(q) \in [0, 1]$ ranks questions from easiest (0) to hardest (1).

Model difficulty: Expected success rate. Another notion of difficulty is that of the expected success rate, which captures the probability of a stochastic agent succeeding at a task or challenge. For a model with a stochastic decoding policy π and question q with a ground-truth answer y^* , we define the expected success rate as

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

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

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

with $a_k \sim \pi(\cdot | q)$ i.i.d. 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. Difficulty can also be thought of from a binary perspective; thus we also consider binary success under different deterministic aggregation rules commonly used by LLMs. Specifically, we evaluate:

- **Greedy:** The model either succeeds or fails on a single deterministic generation.

- **Maj@K** (Majority voting): Select the most frequent parsed answer from K samples and verify correctness.
- **Pass@K**: The question is considered answered if at least one of the K sampled answers is correct (Chen et al., 2021).

Whilst both the binary and expected success metrics capture the probability of a model being successful, they are different transformations of the full answer distribution and thus capture different information about model capacity, serving as distinct prediction targets.

3.2 Activation Probes

Example suffixes:

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

Figure 1: Example chat-template suffixes for several model families.

Let $h_i^\ell \in \mathbb{R}^D$ be the residual-stream activation, where D is the size of the hidden dimension, ℓ the layer index of the model, and $i \in [S]$ the sequence position of a prompt with length S . Depending on the notion of difficulty we are probing we use different linear probe setups. For expected success rate and human IRT difficulty we use the MSE loss and for binary success under different decoding strategies we use a logistic regression probe with a cross-entropy loss. Across all objectives we include an L2-regularization term in the loss.

During training we perform 5-fold cross validation on the probe, and sweep three hyperparameters: the layer index ℓ , the L2 regularisation weight α , and the token position i , selected from the final P non-padding tokens of the chat template (Arditi et al., 2024); see Figure 1.

4 Experiments

4.1 Experimental Setup

Models. We evaluate on a heterogeneous pool of models spanning sizes and reasoning capabilities: GPT-OSS-20B (low/medium/high reasoning) (OpenAI, 2025), DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025), Qwen2.5-Math (1.5B, 7B) (Yang et al., 2024), Qwen2.5-Coder (3B, 7B) (Hui et al., 2024), and Qwen2.5-1.5B (Team, 2024). Full generation configurations are provided in Appendix E.

Datasets. We construct a range of datasets to train model-difficulty probes. For E2H-AMC, GSM8K, MATH, AIME (1983-2024), and LiveCodeBench (Ding et al., 2024; Cobbe et al., 2021; Hendrycks et al., 2021; Veeraboina, 2023; Balunović et al., 2025; Jain et al., 2024); we generate multiple answers for each of the models above. Using these answers we construct a label for each prompt, as defined in Section 3.1. Given the quantity of data and expense of generating multiple rollouts for the larger GPT-OSS model, we set $K=5$ (the number of Monte Carlo rollouts per prompt) for this model, and $K=50$ for the smaller Qwen based models. For LiveCodeBench we use temporal splits based on each model’s release date to control for data contamination in our analysis. Where test and validation splits are available we use those provided, otherwise we split the data into test splits using an 80/20 ratio. AIME-2025 (Balunović et al., 2025) is used only as a test set. Full dataset details can be found in Appendix B.

A particularly important dataset in our analysis is the *E2H-AMC dataset* Ding et al. (2024), which contains 4k mathematics problems annotated with an IRT difficulty score $b(q)$, calibrated from large-scale student performance data from secondary school students who sat the AMC. This provides a model-agnostic measure of human difficulty. This dataset is central to our experiments as it uniquely allows us to compare the representations of human and model perceived difficulty on the same set of questions.

Metrics. To evaluate the performance of our probes, we report the Spearman rank correlation for the linear regression probe used on the expected success rate, $\hat{s}_{MC}$. The rank ordering of questions by difficulty is often a key requirement for downstream tasks such as difficulty-routing, see Section 5. For binary success predictions, we report the AUROC, which measures discrimination quality irrespective of cutoff.

Baselines. Alongside the linear probes described in Section 3.2 we also evaluate a naive question-length baseline, a TF-IDF linear model, and a more complex MLP probe. We provide full details for all of these models in Appendix A.

4.2 Results

Table 1: **Human and Model notions of difficulty are both linearly represented within the model activations.** We report the Spearman correlation coefficient for probes trained on the E2H-AMC Dataset with both human IRT and expected success rate labels. We note that: i) For both notions of difficulty the linear probe outperforms the TF-IDF and length baselines; ii) The Human IRT difficulty is more linearly accessible within the model than expected success rate difficulty; iii) the expected success rate probe performance monotonically decreases as the reasoning level of GPT-OSS increases.

Method	Qwen2.5-Math		GPT-OSS-20B		
	1.5B	7B	Low	Med	High
<i>Human IRT difficulty $b(q)$</i>					
Linear Probe	0.85	0.87	0.84	0.83	0.83
MLP Probe	0.83	0.87	0.77	0.79	0.80
TF-IDF	0.72	0.72	0.74	0.74	0.74
Length	0.15	0.15	0.15	0.15	0.15
<i>Model difficulty $\hat{s}_{MC}$</i>					
Linear Probe	0.64	0.64	0.58	0.50	0.40
MLP Probe	0.65	0.61	0.44	0.44	0.11
TF-IDF	0.47	0.47	0.42	0.31	0.25
Length	0.27	0.27	0.30	0.24	0.19

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, suggesting that models robustly encode what humans find difficult, even when that differs from their own performance characteristics.

Critically, the model success rate becomes *less recoverable* as reasoning capability increases. For GPT-OSS-20B, probe performance drops from $\rho = 0.58$ (low reasoning) to $\rho = 0.40$ (high reasoning), despite higher task success rate. Non-linear MLP probes do not recover this loss, and in fact degrade more rapidly than linear probes under increased reasoning. Together, these results suggest that while human-aligned difficulty is stably encoded, model-specific success signals are more fragile and sensitive to inference-time computation.

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:

Table 2: **We report the AUROC for probes and baselines trained on the binary notion of difficulty under different decoding procedures.** The results are grouped by task: *Math* or *Code*, and decoding regime: *Maj@5*, *Greedy*, and *Pass@5*. *Math* results show the AUROC averaged over the MATH, GSM8K, and AIME-2025 datasets, *Code* reports the AUROC on LiveCodeBench with correctness defined w.r.t Pass@5 success. We also report the task success rate (Task Succ.) for each model to provide context on the number of positive and negative samples in training each probe.

Model	Inference Regime	Task Succ. $\uparrow$	Linear $\uparrow$	MLP $\uparrow$	TF-IDF $\uparrow$	Length $\uparrow$
<i>Math: Greedy vs. Maj@5</i>						
Qwen2.5-Math-1.5B	Greedy	0.724	0.84	0.83	0.64	0.61
	Maj@5	0.763	0.76	0.75	0.63	0.66
Qwen2.5-Math-7B	Greedy	0.809	0.79	0.79	0.68	0.67
	Maj@5	0.827	0.80	0.77	0.72	0.66
Qwen2.5-1.5B	Greedy	0.525	0.68	0.75	0.63	0.73
	Maj@5	0.583	0.85	0.77	0.65	0.69
<i>Math: Maj@5 with Variable Reasoning Budget</i>						
GPT-OSS-20B	Reasoning: Low	0.866	0.78	0.79	0.68	0.62
	Reasoning: Medium	0.914	0.70	0.67	0.63	0.55
	Reasoning: High	0.920	0.64	0.76	0.58	0.46
<i>Code: LiveCodeBench (target = Pass@5)</i>						
Qwen2.5-Coder-3B	Pass@5	0.14	0.91	0.56	0.86	0.64
Qwen2.5-Coder-7B	Pass@5	0.15	0.90	0.89	0.84	0.61
DeepSeek-R1-Distill-Qwen-7B	Pass@5	0.30	0.81	0.80	0.83	0.59
GPT-OSS-20B (low)	Pass@5	0.77	0.71	0.65	0.66	0.64
GPT-OSS-20B (medium)	Pass@5	0.77	0.67	0.59	0.64	0.60
GPT-OSS-20B (high)	Pass@5	0.79	0.69	0.64	0.64	0.62

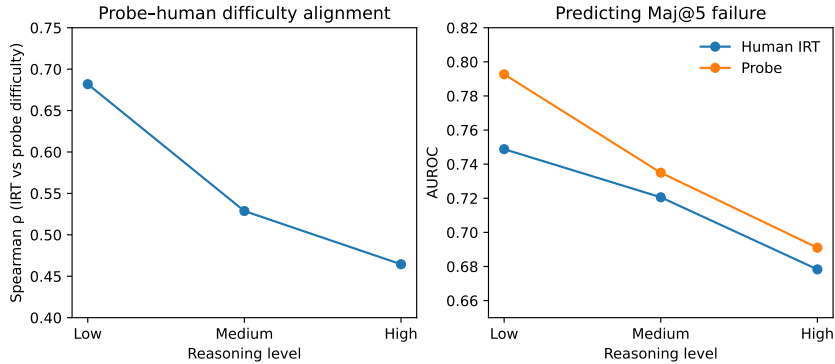


Figure 2: **Human and model difficulty diverge with increased reasoning.** We analyse how reasoning efforts affect the probe-able notions of human and binary difficulty on the E2H-AMC dataset using the GPT-OSS model. **Left:** We plot the Spearman correlation between IRT difficulty scores and the probe predicted difficulty across reasoning levels. As the reasoning level increases the correlation between probe-predicted model difficulty and human IRT difficulty labels decreases reflecting the diverging capabilities of the student examinees and model capabilities at higher reasoning levels. **Right:** Here we plot the AUROC of the probe and human IRT ground truth labels, which reflect probability of success on a question, against the model success at Maj@5, across reasoning levels. The difference between the human and probe scores indicates the probe is likely using an internal representation of model difficulty different from the human scores.

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. However, unlike in the success-rate setting, this degradation is partially recoverable with non-linear probes: MLPs match or exceed linear probe performance at higher reasoning levels (e.g., 0.76 vs 0.64 at high reasoning). This suggests that while extended reasoning introduces representation drift, the underlying success signal remains present but becomes less linearly separable.

Human and model difficulty diverge with increased reasoning. Figure 2 illustrates how the relationship between human and model difficulty changes as GPT-OSS-20B’s reasoning budget increases. Figure 2 [Left] shows that alignment between probe-predicted model difficulty and human IRT difficulty decreases monotonically with reasoning level (Spearman ρ drops from ~ 0.65 to ~ 0.45). This indicates that as models become better at solving human-hard problems through extended reasoning, the notions of difficulty diverge.

Figure 2 [Right] demonstrates that despite this divergence, probe-based predictions of model difficulty consistently outperform human difficulty for predicting 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.

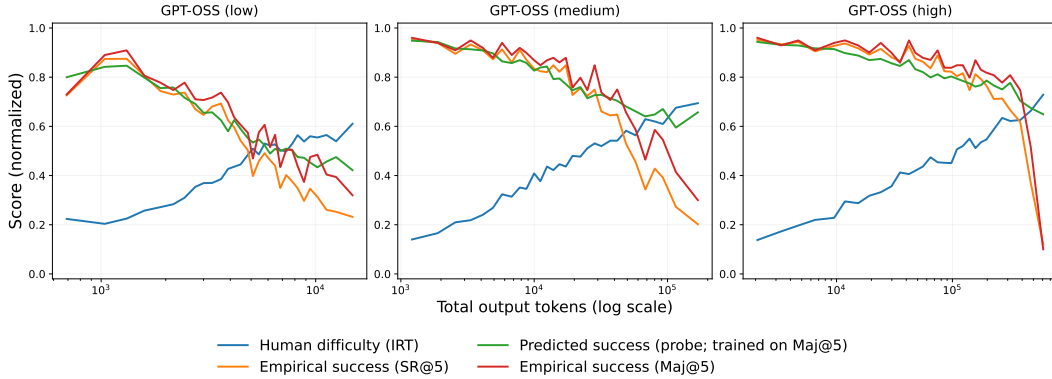


Figure 3: Chain-of-thought length tracks human difficulty but diverges from model success. On E2H-AMC, we plot binned chain-of-thought length (total output tokens, log-scale) against mean normalized human IRT difficulty, empirical success, and probe-predicted success (SR@5 and Maj@5) for GPT-OSS-20B across low, medium, and high reasoning modes. Across all settings, output length increases with human difficulty but decreases with both empirical and predicted success. This effect strengthens with higher reasoning budgets, indicating that generation length increasingly reflects human-aligned difficulty rather than model-relative likelihood of failure.

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 3 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. This observation aligns with concurrent work by [Chen et al. \(2026\)](#), which finds that longer chain-of-thought traces are not a reliable indicator of correctness. Our results provide a complementary perspective: we show that this disconnect arises because reasoning length

tracks human-aligned difficulty, while model-relative success remains predictable from pre-generation representations.

As such, extended reasoning amplifies a human-aligned difficulty signal 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.

5 Practical Applications: 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.

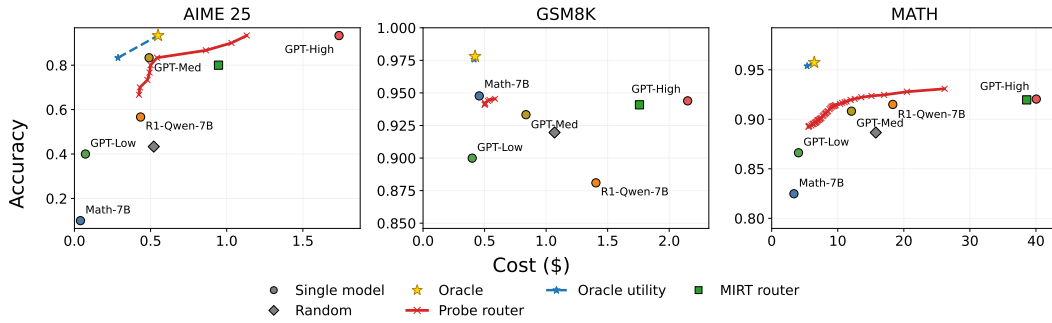


Figure 4: **Probe-based routing generalizes across diverse reasoning benchmarks.** On the hardest benchmark (AIME 25), the router matches GPT-OSS-20B-high’s 93.3% accuracy at 37% lower cost, tracing a Pareto frontier that dominates all single-model and IRT Router baselines. On the saturated benchmark (GSM8K), it pivots to cost minimization, identifying Math-7B as the cost-optimal choice and avoiding expensive models that offer no accuracy gain. On the intermediate benchmark (MATH), it achieves a Pareto improvement over all baselines, matching GPT-OSS-20B-high accuracy while reducing cost by approximately 70%. Together, these results show that a single probe-based routing mechanism adapts to difficulty distributions across benchmarks without retraining. All results use maj@5 with $K=5$ generations. For full results, see Appendix F.

5.1 Routing Strategies

We evaluate a probe-based utility routing strategy that uses predicted success probabilities $\hat{p}_M(x)$ to allocate queries across a set of models M , with different capabilities and costs. A cascading strategy is explored within the Appendix F.2.

Utility-Based Probe Routing For routing among a heterogeneous pool of models, we use a simple utility-based rule. Let $\{M_1, \dots, M_K\}$ denote available models with expected costs $\{\hat{c}_1, \dots, \hat{c}_K\}$ based on the average output cost from the training set. We normalize the expected cost $\hat{c}_i$ to be between [0-1] such that for a prompt x , we select:

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

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

5.2 Routing Experiment Setup

We evaluate using a pool of five models: Qwen2.5-Math-7B-Instruct ([Yang et al., 2024](#)), Deepseek-R1-Qwen-7B ([Guo et al., 2025](#)), and GPT-OSS-20B ([OpenAI, 2025](#)) with

low/medium/high reasoning budgets. We vary λ to trace the performance-cost frontier. Following prior routing work that uses API pricing to estimate deployment costs (Chen et al., 2024; Ding et al., 2023), we use Fireworks AI’s inference pricing to emulate realistic conditions, see Appendix F. All experiments use maj@5 accuracy with K=5 generations on the MATH Hendrycks et al. (2021) dataset, and probes trained to predict maj@5 success as described in Section 3.

Baselines We compare against three baselines: an IRT router baseline, 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. The IRT router follows prior work on modeling question difficulty and model ability via latent trait estimation using learned embeddings (Song et al., 2025). Unlike our approach, which directly extracts model-specific success signals from internal representations, IRT-based routers rely on learned item and model embeddings and require additional training to estimate these latent variables.

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.

We report additional routing configurations, including the Self-Consistency(SC)-Entropy baseline and routing considering input-prompt costs, in Appendix F.1.

5.3 Results

Utility routing achieves strong cost-accuracy tradeoffs and adapts to benchmark difficulty. Figure 4 shows that utility routing yields strong gains across benchmarks, achieving substantial cost reductions while maintaining accuracy. On MATH, the router matches GPT-OSS-20B-high’s 92% accuracy at a 70% cost reduction. This behavior generalizes across benchmarks with different difficulty distributions. On AIME 2025, where model performance varies widely (40%–93%), the router matches the strongest model’s performance at a 37% cost reduction (\$1.15 vs \$1.75). In contrast, on GSM8K, where performance saturates across models (85%–95%), the router identifies the cost-optimal model, selecting Math-7B (94.5% at \$0.34) over significantly more expensive high-reasoning models (GPT-OSS-20B-high: 94.4% at \$2.4). These results demonstrate performance-aware allocation: routing preferentially uses stronger models when task difficulty varies, and shifts toward efficient models when accuracy plateaus. See Appendix F for full results.

5.4 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 success rate, it consistently reduces the linear accessibility of pre-generation success signals. Across reasoning modes in GPT-OSS-20B, the linear 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. Probe-guided routing approaches oracle-utility performance when probes achieve high discrimination (AUROC), but exhibits substantial gaps when probe quality degrades. This suggests routing effectiveness is constrained by the reliability of success estimates rather than model capability alone. Even in lower-quality regimes, the probe consistently identifies cost-effective models, selecting cheaper models on saturated benchmarks (GSM8K) and higher-capability models on harder tasks (AIME). Unlike embedding-based approaches such as IRT router (Song et al., 2025), which depend on learned latent representations from another model, probe-based routing from the very model whose performance it is predicting.

6 Conclusion and Limitations

We show that a model’s likelihood of success is encoded before generation begins and can be extracted with simple linear probes. These signals reflect a notion of difficulty that differs from human judgments but more reliably predicts model performance. While they generalize across decoding strategies, they become harder to access as test-time compute increases. When accessible, they enable routing strategies that approach oracle performance.

Limitations. We focus on linear probes at a single post-instruction position. While effective for base and lightly instruction-tuned models, probe performance degrades under extended reasoning, and we do not probe during generation or do cross-domain transfer (e.g., math to code). Our probes are sensitive to token position, and our routing policies use fixed- k majority voting rather than learned or adaptive selection of k .

7 Acknowledgements

WL is supported by the Rhodes Scholarship. WB was supported by the Engineering and Physical Sciences Research Council EP/S021566/1. Lastly, we thank the reviewers for their helpful feedback.

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>.

- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Wei-Lin Chen, Liqian Peng, Tian Tan, Chao Zhao, Blake JianHang Chen, Ziqian Lin, Alec Go, and Yu Meng. Think Deep, Not Just Long: Measuring LLM Reasoning Effort via Deep-Thinking Tokens, February 2026. URL <http://arxiv.org/abs/2602.13517>. arXiv:2602.13517 [cs].
- 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].
- SE Embretson. Item response theory for psychologists. *Lawrence E*, 2000.
- Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koepl, 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, Shitong 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.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shangkuan Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder Technical Report, November 2024. URL <http://arxiv.org/abs/2409.12186>. arXiv:2409.12186 [cs.CL].
- 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, Stanislaw 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].

- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.
- 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 Lugolooobi 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/>.
- OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL <https://arxiv.org/abs/2508.10925>.
- Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. IRT-Router: Effective and Interpretable Multi-LLM Routing via Item Response Theory. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15629–15644, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.761. URL <https://aclanthology.org/2025.acl-long.761/>.
- Qwen Team. Qwen2.5: A Party of Foundation Models, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.
- 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.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. URL <https://arxiv.org/abs/2409.12122>.

A Probe and Baseline 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: $\{-P, \dots, -1\}$. These are our candidate positions to probe, see Figure 1.

Training linear probes. For each candidate pair $(\ell, p) \in \mathcal{L} \times \mathcal{P}$, where $\mathcal{L}$ spans all transformer layers and $\mathcal{P}$ the post-instruction suffix positions, we train a single linear probe on the activation vector h_p^ℓ . 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. Test evaluation happens exactly once with the selected probe.

MLP probes. While our main results use linear probes for efficiency and interpretability, we also implement MLP-based probes to capture non-linear relationships in internal representations.

For each candidate pair $(\ell, p) \in \mathcal{L} \times \mathcal{P}$, we train a small 2-layer MLP on the activation vector $h_p^\ell \in \mathbb{R}^D$. The architecture consists of:

- **Input layer:** D dimensions (matching model hidden dimension)
- **Hidden layer:** 256 dimensions with ReLU activation
- **Output layer:** 1 dimension (for regression or binary classification)

For regression tasks (success-rate prediction), we use MSE loss with output activation identity. For binary classification (majority vote correctness), we use binary cross-entropy loss with sigmoid output activation.

MLP training details. We apply ℓ_2 regularization with strength α tuned via grid search: $\alpha \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Optimisation is performed with Adam (learning rate 10^{-3}) for up to 100 epochs with early stopping on validation loss (patience=10). Data is split 80/20 for train-validation. For classification, we apply Platt scaling on validation data to calibrate probabilities. Best configuration (ℓ^*, p^*, α^*) is selected by validation performance and evaluated exactly once on a held-out test set.

TF-IDF probe. Using spaCy, we preprocess each prompt with lemmatisation and stop-word removal, then represent it using TF-IDF features. We train Ridge regression for continuous targets and logistic regression for binary targets. We select the regularisation strength on the validation set using the same grid and metrics as the linear probes, then evaluate once on the test set.

Length probe. We represent each prompt using a single feature: its whitespace-separated word count. We train Ridge regression for continuous targets and logistic regression for binary targets. Regularisation is selected on the validation set using the same procedure as the TF-IDF probe, followed by evaluation on a held-out test set.

Comparison. Comparing linear and MLP probes, three patterns emerge: (1) for human IRT difficulty prediction, the two probe types perform comparably, suggesting this signal is robustly linear; (2) for binary success prediction under standard decoding, MLP probes offer marginal or no improvement; (3) under variable reasoning with GPT-OSS, MLP probes partially recover discrimination lost by linear probes (e.g., AUROC 0.76 vs 0.64 for GPT-OSS-20B-high on Maj@5), suggesting that reasoning transforms representations in ways that break linear separability while preserving the signal in a nonlinear subspace. However, this recovery is task-dependent: for continuous success-rate regression, MLP probes degrade more severely than linear probes at high reasoning (Table 1), indicating that the underlying signal may become genuinely less accessible rather than merely nonlinear.

B Dataset Statistics

Table 3: Train, validation, and test split sizes for each dataset.

Dataset	Train	Val	Test
MATH	6,000	1,500	5,000
E2H-AMC	640	160	2,975
AIME-1983–2024	598	149	30
AIME-2025	–	–	30
GSM8K	4,784	1,195	1,319
LiveCodeBench	458	114	341

C Probe Layer Sweep

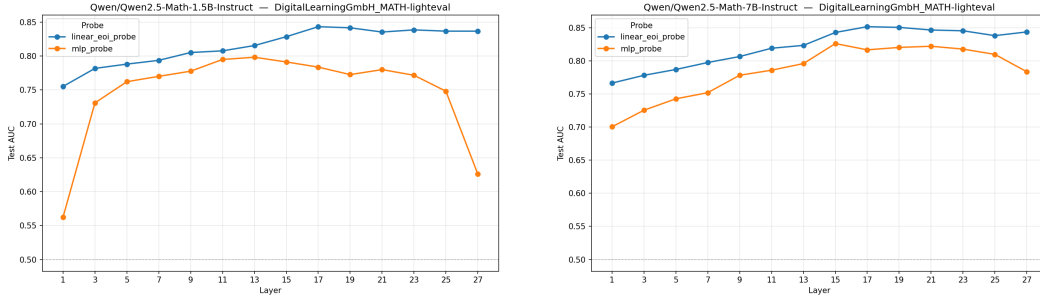


Figure 5: The validation performance of the linear and MLP probes at different layers for the QWEN2.5-MATH-1.5B-INSTRUCT and QWEN2.5-MATH-7B-INSTRUCT models on the MATH dataset. In both models the linear probe outperforms the MLP probe. The linear probe performance tends to increase as the layer index increases.

Table 4: Best probe layer and token position selected on validation, with the corresponding validation and test scores. Note the best token position is indexed backward from the final token.

Model	Dataset	Layer	Pos.	Val score	Test score
Qwen2.5-Math-7B-Instruct	MATH	21	4	0.891	0.846
	AMC	22	4	0.881	0.848
	AIME	28	1	0.867	0.679
	GSM8K	20	4	0.754	0.767
DeepSeek-R1-Distill-Qwen-7B	MATH	27	2	0.793	0.739
	AIME	13	1	0.753	0.701
	GSM8K	23	1	0.647	0.640
gpt-oss-20b (high)	MATH	15	0	0.850	0.860
	AMC	16	2	0.757	0.691
	AIME	10	2	0.761	0.286
	GSM8K	17	0	0.701	0.686
gpt-oss-20b (low)	MATH	17	0	0.845	0.845
	AMC	12	0	0.810	0.793
	AIME	24	0	0.825	0.778
	GSM8K	17	0	0.790	0.761
gpt-oss-20b (medium)	MATH	15	0	0.840	0.850
	AMC	13	1	0.784	0.735
	AIME	19	2	0.778	0.568
	GSM8K	16	1	0.921	0.629

D Additional Probe Results

Table 5: AUROC for Predicting Accuracy under Different Decoding Strategies

Linear Probe			
Decoding	Dataset	Qwen2.5-Math-1.5B	Qwen2.5-Math-7B
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
MLP Probe			
Decoding	Dataset	Qwen2.5-Math-1.5B	Qwen2.5-Math-7B
Maj@5	MATH	0.834	0.846
	AIME	0.712	0.703
	GSM8K	0.692	0.77
Greedy	MATH	0.825	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	0.70	0.70
	AIME	0.83	0.66
	GSM8K	0.66	0.62

Table 6: Maj@5 Probe Performance Comparison across GPT-OSS-20B thinking modes

Linear Probe				
Difficulty	Dataset	Low	Medium	High
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
MLP Probe				
Difficulty	Dataset	Low	Medium	High
gpt-oss-20b	MATH-lighteval	0.831	0.25	0.823
	AMC	0.793	0.735	0.691
	AIME	0.769	0.504	0.858
	GSM8K	0.763	0.691	0.610
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 7: 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

Table 8: Cross-dataset transfer. Each probe is trained on one dataset and evaluated on all three. Bold on the diagonal marks in-distribution evaluation.

Model	Trained on	MATH	GSM8K	AIME	Avg. $\pm$ SD
Qwen2.5-1.5B-Instruct	MATH	0.844	0.727	0.931	0.834 ± 0.084
	GSM8K	0.796	0.762	1.000	0.853 ± 0.105
	AIME	0.783	0.709	0.931	0.808 ± 0.092
Qwen2.5-Math-1.5B-Instruct	MATH	0.841	0.730	0.654	0.742 ± 0.077
	GSM8K	0.760	0.758	0.538	0.685 ± 0.104
	AIME	0.740	0.605	0.712	0.686 ± 0.058
Qwen2.5-Math-7B-Instruct	MATH	0.846	0.720	0.901	0.822 ± 0.076
	GSM8K	0.794	0.767	0.877	0.813 ± 0.047
	AIME	0.745	0.640	0.679	0.688 ± 0.043
DeepSeek-R1-Distill-Qwen-7B	MATH	0.739	0.515	0.643	0.632 ± 0.092
	GSM8K	0.589	0.640	0.502	0.577 ± 0.057
	AIME	0.634	0.511	0.701	0.615 ± 0.079
gpt-oss-20b (low)	MATH	0.845	0.710	0.611	0.722 ± 0.096
	GSM8K	0.662	0.761	0.361	0.595 ± 0.170
	AIME	0.650	0.614	0.778	0.681 ± 0.070
gpt-oss-20b (medium)	MATH	0.850	0.725	0.504	0.693 ± 0.143
	GSM8K	0.540	0.629	0.504	0.558 ± 0.053
	AIME	0.558	0.577	0.568	0.568 ± 0.008
gpt-oss-20b (high)	MATH	0.860	0.692	0.679	0.744 ± 0.082
	GSM8K	0.656	0.686	0.214	0.519 ± 0.216
	AIME	0.594	0.522	0.286	0.467 ± 0.132

Table 9: Probe performance across Qwen2.5-7B variants with different degrees of post-training.

Qwen2.5-7B	MATH	AIME	GSM8K
Base	0.7694	–	0.6661
Math	0.8463	0.6790	0.7665
Instruct	0.8375	0.7284	0.7766
Reasoning (R1-Distill)	0.7391	0.7014	0.6401

E Optimal Model Settings

All rollouts were performed using VLLM [Kwon et al. \(2023\)](#) 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 10: 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. For expected success rate math datasets the Qwen series models use K=50.

F Additional Routing Results and Data

Table 11: Fireworks AI Pricing for routed model costs. Note GPT-OSS-20B has more than 16B params but uses its own pricing structure.

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)

Table 12: Routing strategies comparison on MATH.

Strategy	Model	Accuracy	Cost
<i>Single-model baselines</i>			
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
<i>Routing strategies</i>			
Random Routing	ensemble	0.889	15.43
Oracle (Perfect Knowledge)	ensemble	0.957	6.37
<i>Probe router</i>			
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 13: Routing strategies comparison on AIME 25.

Strategy	Model	Accuracy	Cost
<i>Single-model baselines</i>			
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
<i>Routing strategies</i>			
Random Routing	ensemble	0.433	0.52
Oracle (Perfect Knowledge)	ensemble	0.933	0.55
<i>Probe router</i>			
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 14: Routing strategies comparison on GSM8K.

Strategy	Model	Accuracy	Cost
<i>Single-model baselines</i>			
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
<i>Routing strategies</i>			
Random Routing	ensemble	0.918	0.99
Oracle (Perfect Knowledge)	ensemble	0.978	0.40
<i>Probe router</i>			
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

F.1 Full-Cost Routing with an SC-Entropy Baseline

We revisit the routing results under more complete cost accounting. Under this formulation for probe-based routing, we include the input-token cost of obtaining activations from each candidate model, in addition to the selected model’s generation cost. This captures the cost of computing the routing signal itself. Based on our observations, this cost is minimal, as we perform only one forward pass per model per question.

SC-Entropy Routing. This baseline replaces the probe-estimated success probability $\hat{p}_i(x)$ with a confidence score derived from self-consistency entropy (SC-Entropy), while keeping the utility-based routing formulation of Section 5.1 unchanged. For model M_i , we compute the Shannon entropy $H_i(x)$ over the answers extracted from its K sampled outputs and define

$$c_{\text{SC},i}(x) = 1 - \frac{H_i(x)}{\log K}.$$

The selected model is therefore

$$\hat{M}_{\text{SC}}(x) = \arg \max_i (c_{\text{SC},i}(x) - \lambda \hat{c}_i).$$

Unlike the probe score, this confidence signal is available only after all K outputs have been generated for every candidate model. We therefore charge the SC-entropy baseline for the full generation cost across the model pool. It serves as an expensive, output-based reference rather than a practical cost-saving router.

F.2 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. We evaluate using Qwen2.5-Math-1.5B as M_s and Qwen2.5-Math-7B as M_l .

Table 15: GSM8K routing results. Model columns give the fraction of queries routed to each model.

Strategy	Cost (\$)	Acc.	Routing distribution				
			Math-7B	R1-Qwen-7B	oss-low	oss-med	oss-high
Oracle	0.4223	97.8%	6%	1%	92%	1%	0%
Probe ($\lambda = 1.00$)	0.4318	94.1%	39%	0%	60%	1%	0%
Probe ($\lambda = 0.59$)	0.4588	94.4%	47%	0%	51%	2%	0%
Probe ($\lambda = 0.53$)	0.4713	94.5%	47%	0%	49%	3%	0%
Probe ($\lambda = 0.37$)	0.5170	94.5%	49%	0%	43%	8%	0%
Random	0.9986	92.0%	20%	21%	19%	21%	19%
SC-Entropy	5.2471	95.8%	94%	3%	1%	2%	1%
MIRT Router	2.0130	94.1%	14.4%	1.5%	0.3%	6%	77.8%

Table 16: AIME routing results. Model columns give the fraction of queries routed to each model.

Strategy	Cost (\$)	Acc.	Routing distribution				
			Math-7B	R1-Qwen-7B	oss-low	oss-med	oss-high
Oracle	0.5490	93.3%	17%	17%	30%	27%	10%
Random	0.5187	43.3%	30%	30%	17%	10%	13%
SC-Entropy (best)	2.7740	93.3%	0%	30%	3%	33%	33%
Probe ($\lambda = 0.98$)	0.4216	66.7%	3%	10%	30%	57%	0%
Probe ($\lambda = 0.27$)	0.5401	83.3%	0%	13%	3%	77%	7%
Probe ($\lambda = 0.08$)	0.8607	86.7%	0%	7%	0%	57%	37%
Probe ($\lambda = 0.04$)	1.1288	93.3%	0%	3%	0%	43%	53%
MIRT Router	0.9460	80.0%	0%	0%	3%	70%	27%

Our oracle in this setting 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.

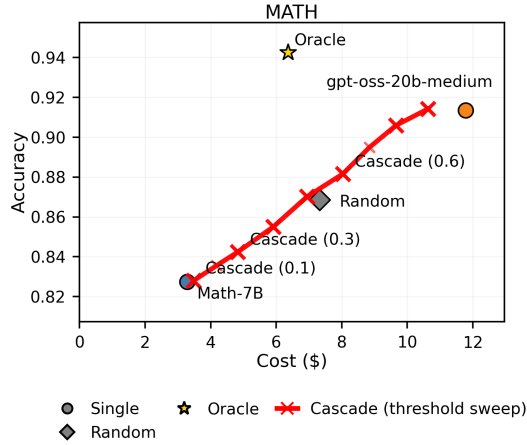


Figure 6: The cascade strategy (left) routes between Qwen2.5-Math-7B and GPT-OSS-20B-medium, substantially outperforming random allocation across the Pareto frontier. At $\tau=0.6$, the cascade matches GPT-OSS-20B-medium’s 91.2% accuracy while reducing cost by 17%.

Table 17: MATH routing results. Model columns give the fraction of queries routed to each model.

Strategy	Cost (\$)	Acc.	Routing distribution				
			Math-7B	R1-Qwen-7B	oss-low	oss-med	oss-high
Oracle	6.4429	95.7%	87%	1%	9%	3%	0%
Random	15.7569	88.7%	20%	20%	20%	21%	20%
SC-Entropy (best)	77.8804	92.4%	74%	20%	2%	2%	2%
Probe ($\lambda = 1.00$)	5.5612	89.2%	61%	1%	35%	3%	0%
Probe ($\lambda = 0.10$)	12.5633	92.1%	40%	16%	23%	21%	1%
Probe ($\lambda = 0.02$)	20.4636	92.8%	28%	25%	12%	27%	9%
Probe ($\lambda = 0.00$)	26.1820	93.1%	19%	26%	5%	16%	33%
MIRT Router	38.5990	92.0%	2%	4%	0%	0%	94%