The Capability Frontier: Benchmarks Miss 82% of Model Performance

Anonymous Authors[1]

Abstract

Existing benchmarks typically report accuracy for a single model on a single run. This systematically understates real-world LLM capabilities: (i) different models get different questions correct, allowing for ensembling gains, and (ii) given a budget, some models can be run multiple times to improve results. We introduce the concept of a *Capability Frontier*: a Pareto frontier for a set of models, characterizing the best achievable performance at each cost level. Our construction of the Capability Frontier corrects for two biases: underestimation from evaluating a single model on a single run, and overestimation from taking the maximum over several noisy models or runs. To understand the impact of these corrections, we study 21 LLMs across 16 widely used benchmarks (coding, reasoning, medicine, factuality, instruction following, and agentic tasks) and compare the performance of the Capability Frontier at matched cost to each benchmark's top-performing model. Correcting for single-model evaluation yields a 54% average accuracy improvement (reduction in error rate); additionally correcting for single runs yields an 82% improvement. Moreover, SOTA accuracy can be matched at 85% cost reduction on the Capability Frontier. These findings suggest that collective LLM capabilities are substantially underestimated, with immediate implications for both evaluation and deployment.

1. Introduction

Standard LLM benchmarks report accuracy for a single model on a single run. This methodology systematically understates achievable capabilities for two reasons: (i) different models excel on different prompts, so combining models can outperform any individual one, and (ii) given a fixed budget, generating multiple outputs and selecting among them can improve results. The gap between reported benchmark performance and what is actually achievable represents a substantial blind spot in how we measure progress.

To quantify this gap, we introduce the *Capability Frontier*: a Pareto frontier characterizing the best achievable performance at each cost level when optimally selecting among models and generations. However, naively estimating this frontier from finite samples introduces positive bias—taking the maximum over noisy estimates preferentially captures lucky outcomes. We develop debiasing methods to accurately measure the true frontier, revealing both the magnitude of benchmark underestimation and providing rigorous upper bounds for routing systems.

Foundational work on LLM routing by Shnitzer et al. (2023) demonstrated that an oracle router can achieve approximately 20% performance gains by switching models per-prompt. RouterBench (Hu et al., 2024) quantified model complementarity, finding that secondary models often provide unique correct answers on 10–30% of prompts. RouteLLM (Ong et al., 2025) achieved 2× cost reduction by identifying prompts where cheaper models suffice. Yet all prior oracle analyses suffer from the positive bias we identify and correct.

Our empirical analysis spans 21 LLMs across 16 benchmarks covering coding, reasoning, medicine, factuality, instruction following, and agentic tasks. The results reveal that standard single-model evaluation substantially understates achievable performance: at matched cost, the Capability Frontier achieves 54% average error reduction compared to each benchmark's top model. When additionally accounting for multi-run selection (posthoc routing), error reduction reaches 82%. Conversely, SOTA accuracy can be matched at 85% lower cost on the frontier. These gaps are not merely theoretical—they represent performance that is achievable today with existing models and straightforward inference-time strategies.

Accurately measuring these gains requires care. The standard oracle computation selects the model with the highest sample mean for each prompt, then reports this mean as achievable performance. Because this takes a maximum over noisy estimates, it preferentially captures positive outliers. With limited generations per prompt ($G \leq 10$ due to cost), this bias is substantial: we find it inflates accuracy estimates by up to 8.7% and cost estimates by up to 88% (Sec. 6.3). Our debiasing methods—extrapolation-based correction and probabilistic graphical modeling—enable accurate frontier estimation.

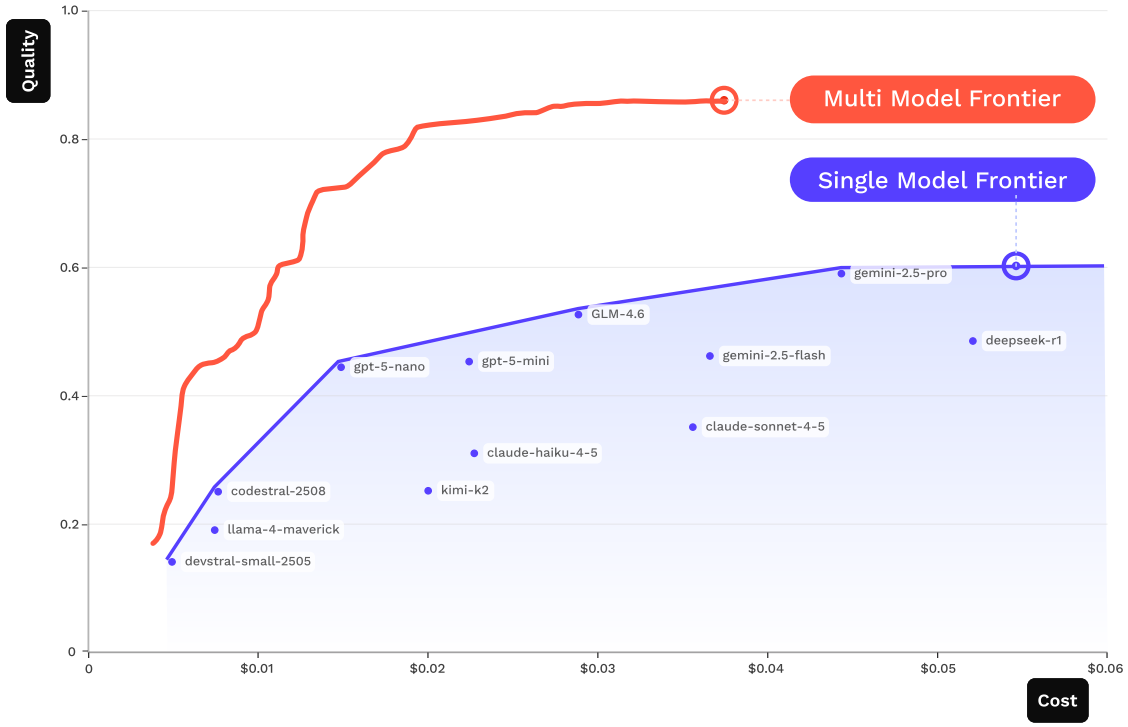


Figure 1. **The Capability Frontier:** Dynamic per-prompt LLM selection substantially outperforms any fixed LLM across our 16 benchmarks. Sample datapoints from App. B are shown. For any given cost budget, substantial quality improvements can be realized relative to a single LLM. Conversely, for a fixed quality threshold substantial cost savings can be realized through dynamic LLM selection.

Contributions. We make the following contributions:

1. **Capability Frontier:** We introduce a framework quantifying the gap between single-model benchmark evaluation and achievable performance, revealing that standard evaluations understate capabilities by 54–82% in error reduction terms (Sec. 4.3, 6).
2. **Debiasing methods:** We show that naive oracle estimators are positively biased and propose two correction methods—extrapolation and probabilistic modeling—with explicit assumptions and validation (Sec. 4.1, 4.2).
3. **Bias characterization:** We formally analyze oracle bias, showing it reduces as $O(G^{-\lambda})$ where G is generations per prompt and $\lambda \in [0.25, 0.75]$ (Sec. 4.1).
4. **Empirical evaluation:** Across 16 benchmarks with verifiable answers and 21 LLMs, we quantify both the underestimation in current benchmarks and the bias in naive oracle estimates (Sec. 6).

2. Related Work

The rapid proliferation of LLMs has increased research interest in LLM routing, the dynamic selection of models to balance quality, cost, and latency. Shnitzer et al. (2023) first formalized this problem using benchmark datasets, introducing the oracle router as a theoretical upper bound for performance gains. While they identified significant headroom beyond the "best-on-average" model, their oracle relied on biased sample means, a limitation our work addresses. Subsequent frameworks like RouterBench (Hu et al., 2024) have standardized evaluation across routing methods, though they similarly utilize these biased estimates.

Universal and Zero-Shot Routing Recent methods seek to solve the "model lock-in" problem, where routers must be retrained whenever the model pool changes. UniRoute (Jitkrittum et al., 2025) addresses this by representing LLMs as feature vectors based on anchor prompts, allowing for generalization to unseen models. Similarly, ZeroRouter (Yan et al., 2026) utilizes a universal latent space to decouple query difficulty from specific model profiles, enabling zero-shot selection across evolving model ecosystems.

Training-Free and Online Methods To support high-volume serving without extensive labeled data, several training-free approaches have emerged. Wu & Silwal (2025) proposed an online routing mechanism using approximate nearest neighbor search to estimate query features with the-

oretical performance guarantees. Building on this, CSCR (Shirkavand et al., 2025) employs cost-aware contrastive learning to map prompts and models into a shared embedding space, facilitating low-latency routing sensitive to both cost and quality. For environments lacking ground-truth labels, Smoothie (Guha et al., 2024) provides a label-free framework that leverages weak supervision and model ensembles to estimate query-specific quality.

Cascades and Multi-Objective Optimization Routing is often implemented as a cascade, where simpler models are queried before deferring to more expensive ones. FrugalGPT (Chen et al., 2023) pioneered this via learned cascades to reduce spend without degrading quality. More recently, C3PO (Valkanas et al., 2025) achieved cost-controlled cascades using conformal prediction to provide provable coverage bounds, while Dekoninck et al. (2025) derived optimal stopping rules for sequential model invocation. Other multi-objective systems, such as xRouter (Qian et al., 2025), utilize reinforcement learning with explicit monetary rewards to navigate the quality-cost trade-off surface.

Agreement-Based and Preference Routing Alternative signals for routing include model agreement and human preferences. ABC (Kolawole et al., 2025) uses agreement between models to make deferral decisions, while BEST-Route (Ding et al., 2025) jointly optimizes model selection and best-of-n sampling. In the absence of objective correctness, RouteLLM (Ong et al., 2025) trains routers on "Chatbot Arena" style preference data to maintain quality at significantly reduced costs. To improve transparency, LLM-Rank (Agrawal & Gupta, 2025) analyzes specific reasoning patterns to provide a granular understanding of model utility beyond aggregate scores.

Theoretical Foundations While the industry moves toward expert orchestration for safer and more capable systems (Quirke et al., 2025), a gap remains between implementable routers and theoretical optimality. Our work builds upon the foundations of oracle routing (Shnitzer et al., 2023; Hu et al., 2024), but departs from them by correcting for the "optimizer's curse"—a statistical bias well-documented in economics (Andrews et al., 2024; Capen et al., 1971) and decision analysis (Smith & Winkler, 2006). By introducing debiased oracles, we provide a more rigorous framework for quantifying the true headroom available in the Capability Frontier.

3. Problem Setting

Let $n \in [N]$ index dataset prompts, $l \in [L]$ index LLMs, and $g \in [G]$ index independent stochastic generations. For each prompt-model pair, we observe G generations and evaluate each using metric $\phi_{nlg} \in \mathbb{R}$ (e.g., correctness, cost, latency).

The standard formulation for routing is a two-dimensional objective that maximizes quality whilst minimizing cost:

$$\phi_{nlg} = \{(\mathbf{Q}, -\mathbf{C})\}_{nlg} \tag{1}$$

$\mathbf{Q}$, $\mathbf{C}$, and $\mathbf{T}^{95}$ are all tensors with identical dimensionality that represent Quality, Cost, and P95 latency.

The routing problem. A router $\pi : \mathcal{X} \to [L]$ maps each prompt to a model. The goal is to find π maximizing expected performance:

$$\max_{\pi} \frac{1}{N} \sum_{n} \mathbb{E}[\phi_{n,\pi(x_n),g}] \tag{2}$$

The oracle router. An oracle router has access to true expected performance $\mu_{nl} = \mathbb{E}[\phi_{nlg}]$ and selects optimally:

$$l^*(n) = \arg\max_{l} \mu_{nl} \tag{3}$$

The **true oracle value** is:

$$\mathcal{O}^{true} = \frac{1}{N} \sum_{n} \max_{l} \mu_{nl} \tag{4}$$

This is the fundamental upper bound for routing: the best achievable performance given perfect knowledge of each model's expected performance on each prompt.

The estimation problem. We cannot observe μ_{nl} directly, only noisy realizations ϕ_{nlg}. The standard approach estimates μ_{nl} with the sample mean $\bar{\phi}_{nl} = \frac{1}{G}\sum_g \phi_{nlg}$ and computes:

$$\mathcal{O}^{biased} = \frac{1}{N} \sum_{n} \max_{l} \bar{\phi}_{nl} \tag{5}$$

We show below this estimator is positively biased: $\mathbb{E}[\mathcal{O}^{biased}] > \mathcal{O}^{true}$.

4. Methodology

4.1. Characterizing Oracle Bias

Why the biased oracle is biased. The bias arises because taking the maximum over sample means preferentially selects models whose samples exceeded their true means. The bias crops in many fields from Economics Andrews et al. (2024), to Management Smith & Winkler (2006); however, it was first spotted in Auctions by Capen et al. (1971). This paper formalizes the bias in relation to LLM Routing presenting new methods to remove this bias. We formalize this under two distributional assumptions.

4.1.1. GAUSSIAN CASE

Assume $\phi_{nlg} \sim \mathcal{N}(\mu_{nl}, \sigma_{nl}^2)$ independently. The sample mean satisfies $\bar{\phi}_{nl} \sim \mathcal{N}(\mu_{nl}, \sigma_{nl}^2/G)$.

To derive the bias in closed form, we make a simplifying assumption:

$$\mu_{nl} = \mu_n, \quad \sigma_{nl}^2 = \sigma_n^2 \quad \forall l \tag{6}$$

Remark. Assumption (6) is used *only* to derive the functional form of bias decay, not to claim that $\mathcal{O}^{true} = \bar{\mu}$. Under heterogeneous means, the true oracle remains $\frac{1}{N}\sum_n \max_l \mu_{nl}$, which our debiasing methods estimate without requiring equal means.

Under (6), the expected maximum of L i.i.d. Gaussians with variance σ_n^2/G is approximately:

$$\mathbb{E}[\max_l \bar{\phi}_{nl}] \approx \mu_n + \sigma_n \sqrt{\frac{2\log L}{G}} \tag{7}$$

Averaging over prompts:

$$\mathcal{O}^{biased} \approx \underbrace{\bar{\mu}}_{\text{True Oracle}} + \underbrace{\bar{\sigma}\sqrt{\frac{2\log L}{G}}}_{\text{Bias}} \tag{8}$$

where $\bar{\mu} = \frac{1}{N}\sum_n \mu_n$ and $\bar{\sigma} = \frac{1}{N}\sum_n \sigma_n$.

Key insight: The bias reduces as $O(G^{-0.5})$ and increases with L (more models) and $\bar{\sigma}$ (higher variance). For $G = 10$ and $L = 21$, this bias is non-negligible.

4.1.2. BERNOULLI CASE

For binary metrics (correct/incorrect), let $\phi_{nlg} \sim \text{Bernoulli}(p_{nl})$. Under the simplifying assumption $p_{nl} = p_n$:

$$Y_{nl} = \sum_g \phi_{nlg} \sim \text{Binomial}(G, p_n) \tag{9}$$

$$\mathbb{E}[\max_l Y_{nl}] = \frac{1}{NG}\sum_{n,g}\left[1 - F(g; p_n)^L\right] \tag{10}$$

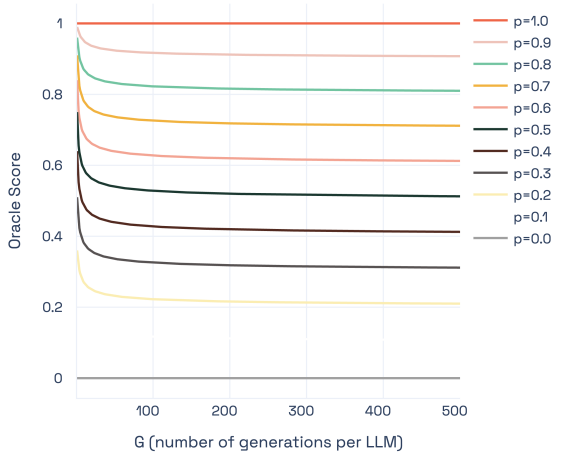
where $F(g; p_n)$ is the Binomial CDF.

There is no clean separation of true oracle and bias term however, through empirical study we can determine the characteristics of the bias decay.

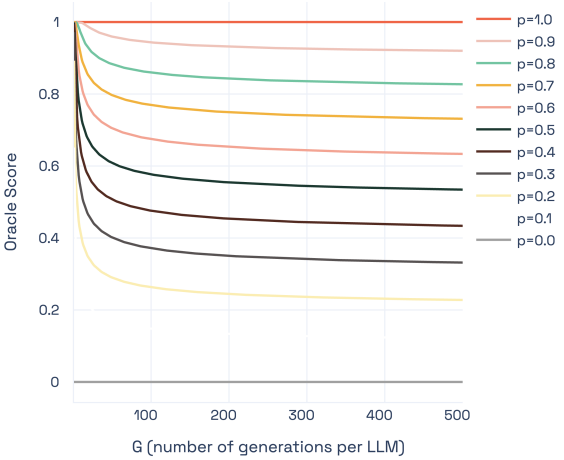
We know that for large G, the Oracle should tend towards p_n. Fig.2 shows how the bias decays in different scenarios. When $p = 0$ or $p = 1$, there is no variance in LLM performance per data point and so, the bias is zero for all G.

Through a synthetic study (Appendix A), we found the bias decayed with $O(G^{-0.5})$ $\forall$ $L > 1$, $p \in (0, 1)$ in the limit of large G for heterogeneous (μ_{nl}, σ_{nl}) generations across models, consistent with the Gaussian analysis. For correlations generations between models, we found the exponent varied in the range $[0.25, 0.75]$ for sensible hyperparameters. Both heterogeneous and correlated scenarios required roughly $G = 50$ generations in order to fit Eqn.11 accurately.

Key insight: The bias reduces as $O(G^{-\lambda})$ where $\lambda \in [0.25, 0.75]$. At least $G >= 50$ generations are needed for $O(G^{-\lambda})$ to be the dominate term in the bias decay.



(a) $L = 2$ LLMs



(b) $L = 10$ LLMs

Figure 2. **Oracle bias reduces with generations.** Oracle bias is greatest when each LLM is prompted once $(G = 1)$, and tends towards zero as G grows. Oracle bias decays with $O(G^{-0.5})$ in the limit. Curves show LLM success rates, p. Always correct/incorrect LLMs $(p = 0, 1)$ have no bias and curves are horizontal.

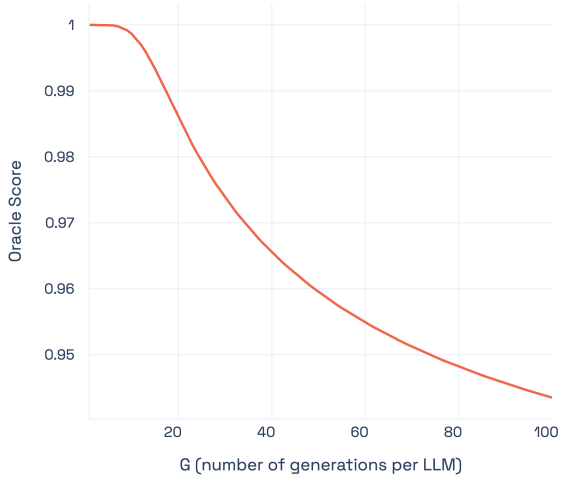


Figure 3. **Bias decay deviates from $O(G^{-0.5})$ for small G.** At $p = 0.9$, the curve only follows the asymptotic form for $G > 20$, motivating our smooth transition formulation. This curve is closeup of $p = 0.9$ from Fig. 2b

.

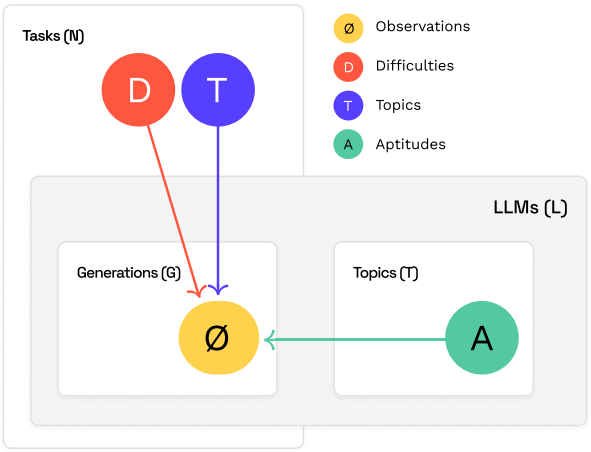


Figure 4. **Probabilistic graphical model (PGM).** We model an LLM's inherent accuracy, as indirectly observed over generations (G) for topics (T) as a function of the prompt difficulty (D) and the model aptitude (A). We depict this model here in Plate Notation, a standard way of writing the generative process for Bayesian models. D_n induces correlation across models for each prompt.

4.2. Debiasing Methods

4.2.1. Method 1: Extrapolation

Given that bias decays as $O(G^{-\lambda})$ where $\lambda \in [0.25, 0.75]$, we fit:

$$\mathcal{O}^{biased}(G) = \alpha + \beta G^{-\lambda} \tag{11}$$

and estimate $\mathcal{O}^{true} = \alpha$.

In practice, due to cost constraints we are not in the regime of $G \geq 50$ and Equation 11 does not hold as can be seen in Fig. 3. As such, a smooth transition formulation can be used to better approximate the bias decay:

$$\mathcal{O}^{biased}(G) = \alpha + \beta \left[1 + \left(\frac{G - \gamma}{\delta}\right)^2\right]^{-\lambda/2} \tag{12}$$

Limitations. With $G < 10$, extrapolation carries risk. We validate by: (1) testing on synthetic data with known ground truth, and (2) comparing to PGM estimates.

4.2.2. Method 2: Probabilistic Graphical Model

We introduce a generative model (Koller & Friedman, 2009) for observations ϕ_{nlg} (shown in Fig. 4) that allows direct estimation of true performance parameters.

The intuition behind the model is: (1) every prompt has a difficulty D, (2) every prompt belongs to a topic T, e.g. coding, math, or some weighted combination of them, (3) every LLM has some aptitude A on each topic. The observed performance of an LLM on a given prompt is a function of the prompt's difficulty, topic combination of the prompt, and the LLM's aptitude on those topics.

Latent variables:

- $D_n \in [0, 1]$: Task difficulty for prompt n
- $T_n \in \{1, \ldots, K\}$: Topic assignment for prompt n
- $A_{tl} \in [0, 1]$: Aptitude of model l on topic t

Generative process:

$$D_n \sim \text{Beta}(\alpha_D, \beta_D) \tag{13}$$
$$T_n \sim \text{Categorical}(\boldsymbol{\theta}) \quad \text{where } \boldsymbol{\theta} \sim \text{Dirichlet}(\boldsymbol{\alpha}) \tag{14}$$
$$A_{tl} \sim \text{Beta}(\alpha_{tl}, \beta_{tl}) \tag{15}$$
$$\phi_{nlg} \sim \text{Bernoulli}(\pi_{nl}) \tag{16}$$

Link function:

$$\pi_{nl} = f(D_n, A_{T_n,l}) \tag{17}$$

A simple multiplicative form of $(1 - D_n) \cdot A_{T_n,l}$ would capture the intuition that success requires both low difficulty and high model aptitude. However, we find the most accurate results were obtained using a feedforward neural network.

Limitations. The PGM has ad-hoc structural choices which can influence the results. Checking alignment against synthetic data with known ground truth across a variety of regimes de-risks these choices.

Inference. We use stochastic variational inference with factorized posterior $q(D_n)q(T_n)\prod_{t,l} q(A_{tl})$. We set uniform

priors ($\alpha_D = \beta_D = 1$, $\alpha_{tl} = \beta_{tl} = 1$, $\alpha_t = 1$) and run until convergence.

Computing the unbiased oracle:

$$\mathcal{O}^{true} = \frac{1}{N} \sum_n \max_l \hat{\pi}_{nl} \tag{18}$$

where $\hat{\pi}_{nl}$ are the inferred success probabilities.

Independence assumptions. Fig. 4 assumes conditional independence across generations given latent variables. This is reasonable when temperature-based sampling dominates, but may underestimate correlations when models share training data and architecture.

4.3. Capability Frontier for Multi-Objective Routing

Real routing decisions involve multiple objectives. We define the Capability Frontier as the Pareto-optimal surface achievable through routing.

For normalized quality Q^* and cost C^*:

$$\phi(\alpha) = \alpha Q^*_{nlg} + (1-\alpha)(-C^*_{nlg}) \tag{19}$$

$$Q^*_{nlg} = \frac{Q_{nlg} - \min \mathbf{Q}}{\max \mathbf{Q} - \min \mathbf{Q}} \tag{20}$$

$$C^*_{nlg} = \frac{C_{nlg} - \min \mathbf{C}}{\max \mathbf{C} - \min \mathbf{C}} \tag{21}$$

Sweeping $\alpha \in [0, 1]$ traces the Capability Frontier. For debiasing, we:

1. Use $\phi(\alpha)$ to determine routing decisions
2. Apply debiasing separately to quality and cost

For cost (positive real values), we replace the Bernoulli likelihood with LogNormal in our PGM.

4.4. Posthoc Oracle

When a verifier is available at inference time, we can select among multiple generations *after* observing outputs. With k generations per model and a perfect judge:

$$\mathcal{O}^{kshot}(k) = \frac{1}{N\binom{G}{k}} \sum_n \max_l \sum_{\substack{\mathcal{S}\subseteq[G] \\ |\mathcal{S}|=k}} \max_{j\in\mathcal{S}} \phi_{nlj} \tag{22}$$

Using the PGM:

$$\mathcal{O}^{kshot}(k) = \frac{1}{N} \sum_n \left[1 - \prod_l (1-\pi_{nl})^k\right] \tag{23}$$

Eqn. 22 & 23 formulation is the most naive form of a posthoc router, where all LLMs are queried for every prompt. A tighter bound can be achieved using more efficient posthoc techniques, such as sequential prompting LLMs with a return early rule. This paper does not discuss those approaches but we believe the gains can be attained at lower cost.

Critical caveats:

- Assumes a *perfect* judge (zero error)
- Assumes the judge is *free* (zero cost)

5. Experimental Setup

Benchmarks.

Benchmarks. We evaluate on 16 benchmarks with verifiable correct answers, spanning:

- **Coding:** LiveCodeBench (Jain et al., 2024), BigCodeBench (Zhuo et al., 2024), HumanEval-X-Python, HumanEval-X-CPP, HumanEval-X-Javascript, HumanEval-Java, HumanEval-X-Go (Zheng et al., 2023), MBPP (Austin et al., 2021), LeetCode Hard (LeetCode, 2026)
- **Reasoning:** LiveBench-Reasoning (White et al., 2024), GPQA Diamond (Rein et al., 2023)
- **Instruction-following:** LiveBench-IFEval (White et al., 2024)
- **Medical:** MedCalcBench (Khandekar et al., 2024)
- **Factuality:** TruthfulQA (Lin et al., 2022)
- **Agentic:** Terminal-Bench 2.0 (Institute & contributors, 2024), LiveCodeBench (Jain et al., 2024)

These benchmarks have binary correctness metrics (pass/fail for code, exact match for QA), enabling clean oracle analysis.

Models. We evaluate 21 LLMs spanning major providers:

- **OpenAI:** GPT-5-nano, GPT-5-mini, GPT-5.1 (OpenAI, 2025)
- **Anthropic:** Claude Haiku 4.5, Claude Sonnet 4.5 (Anthropic, 2025)
- **Google:** Gemini 2.5 Pro, Gemini 2.5 Flash, Gemini 2.5 Flash-Lite (Google Cloud, 2025)
- **Meta:** Llama 4 Scout, Llama 4 Maverick (Hugging Face, 2025)

Table 1. **Combining LLMs boosts Quality.** Benchmark level breakdown of quality comparing the SOTA LLM with $\mathcal{O}^{true}(\alpha = 1)$. On average the error rate is reduced by 53.7% compared to the SOTA LLM, at no additional cost.

Benchmark	% Quality		% Error Reduction ↑
	SOTA LLM	$\mathcal{O}^{\text{true}}(1)$	
LiveBench-Coding	82.2	86.8	26.2
BigCodeBench	35.8	49.1	20.6
LeetCode	79.1	87.7	41.3
HumanEval-X (Python)	97.4	99.5	79.6
HumanEval-X (CPP)	95.1	99.3	85.7
HumanEval-X (Javascript)	93.5	97.0	53.5
HumanEval-X (Java)	95.9	99.0	75.6
HumanEval-X (Go)	91.3	97.1	66.0
MBPP	86.2	92.3	44.3
MedCalcBench	70.5	86.4	53.9
TruthfulQA	98.8	99.5	57.1
LiveBench-IFEval	80.0	87.4	36.9
LiveBench-Reasoning	92.4	96.2	50.2
GPQA Diamond	94.0	99.0	82.7
Terminal-Bench 2.0 (agentic)	40.8	49.0	13.9
LiveBench-Coding (agentic)	85.5	96.0	72.1
Average	82.4	88.8	53.7

Table 2. **Combining LLMs reduces cost.** Benchmark level breakdown of cost comparing the SOTA LLM with $\mathcal{O}^{true}(\alpha = \alpha^*)$. Both SOTA LLM and $\mathcal{O}^{true}(\alpha = \alpha^*)$ have the same Quality by definition.

Benchmark	Cost (cents)		% Cost Saving ↑
	SOTA LLM	$\mathcal{O}^{\text{true}}(*)$	
LiveBench-Coding	1.06	0.26	75.6
BigCodeBench	0.43	0.07	84.6
LeetCode	1.10	0.33	70.0
HumanEval-X (Python)	0.32	0.02	94.8
HumanEval-X (CPP)	0.21	0.02	88.3
HumanEval-X (Javascript)	0.20	0.02	92.4
HumanEval-X (Java)	0.21	0.03	83.5
HumanEval-X (Go)	0.55	0.03	94.8
MBPP	0.14	0.01	96.3
MedCalcBench	0.23	0.04	83.3
TruthfulQA	0.38	0.00	99.5
LiveBench-IFEval	0.27	0.02	93.1
LiveBench-Reasoning	1.04	0.41	60.9
GPQA Diamond	0.54	0.02	96.3
Terminal-Bench 2.0 (agentic)	260.84	25.39	90.3
LiveBench-Coding (agentic)	3.34	1.37	58.9
Average	16.93	1.75	85.2

- **Mistral:** Codestral 2508, Devstral Medium 2505, Devstral Small 2505, Mistral Small Instruct (Mistral AI, 2025)
- **Qwen:** Qwen3 Coder Plus, Qwen3 Coder Flash, Qwen 2.5 Max, Qwen 2.5 72B Instruct (Qwen Team et al., 2025)
- **Moonshot:** Kimi K2 (Moonshot AI, 2025)
- **DeepSeek:** DeepSeek R1 (DeepSeek-AI et al., 2025)
- **Z.AI:** GLM-4.6 (Z.ai, 2025)

Generation parameters. For all models we use the provider's default hyper-parameters. Where benchmarks have a max tokens specified, we preserve that setting.

Metrics.

- **Quality:** Accuracy (fraction of prompts answered correctly). For coding benchmarks, we use execution based verification.
- **Cost:** Total API cost in USD (input + output tokens × provider pricing).

Generations. Each prompt-model pair evaluated with $G = 10$ independent generations. This yields $N \times L \times G$ total observations per benchmark.

Cost measurement. Costs computed using provider API pricing as of 01 Jan 2026.

Agentic Benchmarks. For agentic benchmarks, computing the true oracle is combinatorially hard since the optimal LLM may differ at each trajectory step. To simplify, we fix the LLM within each trajectory. This may *understate* routing benefits; true per-step routing could yield higher gains. We use the mini-SWE-agent (SWE-agent team, 2025) with default parameters.

6. Results

6.1. Finding #1: LLM Routing Gives Substantial Gains

Using debiased oracles, we quantify achievable routing gains (Table 1 & 2) through computation of the Capability Frontier as described in Sec. 4.3 using Eq. 12.

- SOTA LLM: The highest Quality LLM on average
- $\mathcal{O}^{true}(\alpha = 1)$: The most accurate router possible
- $\mathcal{O}^{true}(\alpha = \alpha^*)$: The true Oracle with alpha set to the value which achieves similar quality to the SOTA LLM, i.e., best SOTA cost saving.

Error rate reduction: Compared to SOTA LLM, $\mathcal{O}^{true}(\alpha = 1)$ achieves a 54% average error reduction.

Cost savings at SOTA quality: Compared to SOTA LLM, $\mathcal{O}^{true}(\alpha = \alpha^*)$ achieves 85% average cost savings.

6.2. Finding #2: Posthoc Routing Increases Gains

By leveraging a free and perfect judge at inference time as described in Eq.22, the error rate can be reduced further (Appendix. D Tab. 4 & 5). The results quantify not only that gain, but how quickly it changes as the number of attempts, k, increases from $1 \rightarrow 10$.

As described in Sec. 4.4, this paper uses the most naive form of a posthoc router. We believe these gains can be attained at significantly at lower cost.

k = 1: 66% error reduction vs SOTA LLM.

k=10: 82% error reduction vs SOTA LLM.

6.3. Finding #3: Naive Oracles Overestimate Gains

We compare $\mathcal{O}^{biased}$ to $\mathcal{O}^{true}$ across benchmarks (App. E Tab. 6, Fig. 11).

Quality bias: Average 1.2% overestimation. **Cost bias:** Average 37.5% overestimation.

The larger cost bias arises because cost distributions are more skewed, amplifying selection effects.

6.4. Finding #4: Model Reliability Varies Substantially

LLMs with default hyper parameters by design output different responses for the same input when prompted multiple times. An obvious question though, is how consistent are LLMs in solving the problem across these generations. Appendix. C Tab. 3 shows how different LLMs rank for reliability:

$$\text{reliability}(l) = 2 \times \frac{1}{N} \sum_{n} \left| \bar{\phi}_{nl} - 0.5 \right| \tag{24}$$

The most reliable LLM we tested was GPT-5-mini with a score of 90.2% and the least reliable was GLM-4.6 at 76.3%. There is no significant correlation between Reliability and either Quality or Cost.

7. Limitations

Limited generations. With $G = 10$, extrapolation carries uncertainty. We mitigate with (1) testing on synthetic data with known ground truth, and (2) comparing to PGM estimates. However larger G would improve estimates.

Agentic Benchmarks Gains May Be Underestimated. For agentic benchmarks, computing $\mathcal{O}^{true}$ is combinatorially hard since the optimal LLM may differ at each trajectory step. To simplify, we fix the LLM within each trajectory. This may understate routing benefits; true per-step routing could yield higher gains.

Perfect judge assumption. Posthoc oracles assume error-free, cost-free judges. Real verifiers introduce errors and costs that reduce achievable gains.

8. Conclusion

This work re-evaluates how the performance of large language models is measured. We show that standard benchmark evaluations, typically based on a single model and a single sampled output per prompt, do not capture the full range of performance that is already attainable with existing models and inference budgets. At the same time, we demonstrate that naive aggregation across models or runs can lead to overly optimistic estimates due to noise. To address both effects, we introduced the *Capability Frontier*, a quality-cost Pareto frontier that characterizes achievable performance while explicitly correcting for these opposing biases.

Empirically, across 21 LLMs and 16 benchmarks, the Capability Frontier substantially outperforms standard single-model evaluations. At matched cost, correcting for single-model evaluation reduces error by 54% on average, while additionally accounting for single-run variability yields an 82% reduction. Conversely, at matched accuracy, frontier points often achieve performance comparable to the SOTA LLM at a fraction of the cost. These results suggest that commonly reported benchmark scores can significantly understate achievable system-level performance.

Implications. Our findings have several implications for the evaluation and use of LLMs:

- **Evaluation methodology.** Single-model, single-run benchmarks provide a limited view of model capability. Capability Frontier based analysis offers a complementary perspective that accounts for model diversity and sampling effects, and can help contextualize results.
- **System design.** While the Capability Frontier itself is not a deployment strategy, it highlights regimes where simple routing or repeated sampling may be sufficient to achieve large gains, and where more sophisticated methods are necessary to approach the attainable limits.

Future work. Several extensions are clear. First, incorporating judge error and cost directly into posthoc frontier construction. Second, extending agentic evaluation beyond fixed trajectory routing. Third, developing and evaluating practical routing policies that can approach frontier performance under realistic deployment constraints. Finally, studying how system prompt selection and hyper-parameter sampling on LLMs can affect the frontier.

References

Agrawal, S. and Gupta, P. Llmrank: Understanding llm strengths for model routing. *arXiv preprint arXiv:2510.01234*, 2025. URL https://arxiv.org/abs/2510.01234.

Andrews, I., Kitagawa, T., and McCloskey, A. Inference on winners. *The Quarterly Journal of Economics*, 139(1): 305–358, 2024.

Anthropic. What's new in claude 4.5. https://platform.claude.com/docs/en/about-claude/models/whats-new-claude-4-5, 2025.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models. *arXiv*, 2021. arXiv:2108.07732.

Capen, E. C., Clapp, R. V., and Campbell, W. M. Competitive bidding in high-risk situations. *Journal of Petroleum Technology*, 1971.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023. URL https://arxiv.org/abs/2305.05176.

DeepSeek-AI et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv*, 2025. arXiv:2501.12948.

Dekoninck, J., Baader, M., and Vechev, M. A unified approach to routing and cascading for llms, 2025. URL https://arxiv.org/abs/2410.10347.

Ding, D., Mallick, A., Zhang, S., Wang, C., Madrigal, D., Garcia, M. D. C. H., Xia, M., Lakshmanan, L. V. S., Wu, Q., and Rühle, V. Best-route: Adaptive llm routing with test-time optimal compute, 2025. URL https://arxiv.org/abs/2506.22716.

Google Cloud. Gemini 2.5 updates: Flash/pro ga, sft, flash-lite on vertex ai. https://cloud.google.com/blog/products/ai-machine-learning/gemini-2-5-flash-lite-flash-pro-ga-vertex-ai, 2025.

Guha, N., Chen, M. F., Chow, T., Khare, I. S., and Ré, C. Smoothie: Label free language model routing. *arXiv preprint arXiv:2412.04692*, 2024. URL https://arxiv.org/abs/2412.04692.

Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., and Upadhyay, S. K. Routerbench: A benchmark for multi-llm routing system, 2024. URL https://arxiv.org/abs/2403.12031.

Hugging Face. Welcome llama 4 maverick & scout on hugging face. https://huggingface.co/blog/llama4-release, 2025.

Institute, L. and contributors. Terminal-bench. https://github.com/laude-institute/terminal-bench, 2024.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv*, 2024. arXiv:2403.07974.

Jitkrittum, W., Narasimhan, H., Rawat, A. S., Juneja, J., Wang, C., Wang, Z., Go, A., Lee, C.-Y., Shenoy, P., Panigrahy, R., Menon, A. K., and Kumar, S. Universal model routing for efficient llm inference. *arXiv arXiv:2502.08773*, 2025. URL https://arxiv.org/abs/2502.08773.

Khandekar, N., Jin, Q., Xiong, G., Dunn, S., Applebaum, S. S., Anwar, Z., Sarfo-Gyamfi, M., Safranek, C. W., Anwar, A. A., Zhang, A., Gilson, A., Singer, M. B., Dave, A., Taylor, A., Zhang, A., Chen, Q., and Lu, Z. Medcalc-bench: Evaluating large language models for medical calculations. *arXiv*, 2024. arXiv:2406.12036.

Kolawole, S., Dennis, D., Talwalkar, A., and Smith, V. Agreement-based cascading for efficient inference, 2025. URL https://arxiv.org/abs/2407.02348.

Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA, 2009.

LeetCode. Leetcode. https://leetcode.com/, 2026. Accessed: 2026-01-19.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of ACL 2022*, 2022.

Mistral AI. Announcing codestral 25.08 and the complete mistral coding stack for enterprises. https://mistral.ai/news/codestral-25-08, 2025.

Moonshot AI. Kimi k2: Open agentic intelligence. https://moonshotai.github.io/Kimi-K2/, 2025.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms with preference data, 2025. URL https://arxiv.org/abs/2406.18665.

OpenAI. Introducing gpt-5.1 for developers. https://openai.com/index/gpt-5-1-for-developers/, 2025.

Qian, C., Liu, Z., Kokane, S., Prabhakar, A., Qiu, J., Chen, H., Liu, Z., Ji, H., Yao, W., Heinecke, S., Savarese, S., Xiong, C., and Wang, H. xrouter: Training cost-aware llms orchestration system via reinforcement learning, 2025. URL https://arxiv.org/abs/2510.08439.

Quirke, P., Oozeer, N., Bandi, C., Abdullah, A., Hoelscher-Obermaier, J., Phillips, J. M., Greaves, J., Neo, C., Lan, M., Barez, F., and Upadhyay, S. Beyond monoliths: Expert orchestration for more capable, democratic, and safe language models, 2025. URL https://arxiv.org/abs/2506.00051.

Qwen Team et al. Qwen3 technical report. *arXiv*, 2025. arXiv:2505.09388.

Rein, D. et al. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv*, 2023. arXiv:2311.12022.

Shirkavand, R., Gao, S., Yu, P., and Huang, H. Cost-aware contrastive routing for llms. *arXiv preprint arXiv:2508.12491*, 2025. URL https://arxiv.org/abs/2508.12491.

Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., and Yurochkin, M. Large language model routing with benchmark datasets, 2023. URL https://arxiv.org/abs/2309.15789.

Smith, J. E. and Winkler, R. L. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

SWE-agent team. mini-swe-agent: The 100 line ai agent that solves github issues or helps you in your command line. https://github.com/SWE-agent/mini-swe-agent, 2025.

Valkanas, A., Pal, S., Rumiantsev, P., Zhang, Y., and Coates, M. C3po: Optimized large language model cascades with probabilistic cost constraints for reasoning, 2025. URL https://arxiv.org/abs/2511.07396.

White, C., Dooley, S., Roberts, M., Pal, A., Feuer, B., Jain, S., Shwartz-Ziv, R., Jain, N., Saifullah, K., Naidu, S., Hegde, C., LeCun, Y., Goldstein, T., Neiswanger, W., and Goldblum, M. Livebench: A challenging, contamination-free llm benchmark. *arXiv*, 2024. arXiv:2406.19314.

Wu, F. and Silwal, S. Efficient training-free online routing for high-volume multi-llm serving, 2025. URL https://arxiv.org/abs/2509.02718.

Yan, C., Zhang, W., Ning, Z., Xu, F., Tao, Z., Zhang, L., Yin, B., and Zhang, Y. Breaking model lock-in: Cost-efficient zero-shot llm routing via a universal latent space. *arXiv arXiv:2601.06220*, 2026. URL https://arxiv.org/abs/2601.06220.

Z.ai. Glm-4.6v: Open source multimodal models with native tool use. https://z.ai/blog/glm-4.6v, 2025.

Zheng, Q. et al. Humaneval-x: A new benchmark for multilingual program synthesis. CodeGeeX benchmark release, 2023. https://github.com/zai-org/CodeGeeX.

Zhuo, T. Y. et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv*, 2024. arXiv:2406.15877.

A. Synthetic Study of Bias Decay

We studied how Oracle bias decayed for varying numbers of LLMs, correlations between them, and LLM success probabilities. To do this, we leveraged the PGM defined in Sec. 4.2.2 to generate binary tensors of synthetic data.

For zero LLM correlation, Fig. 2 shows results of some of these experiments. As expected, we found that both increasing the number of LLMs, L, and LLM success probability, p, increased the number of generations, G, needed before the bias decayed with $O(G^{-0.5})$. We empirically proved with large enough G this is always the case.

For the regime $L <= 10; 0.3 <= p <= 0.7$, we found that at least $G = 50$ generations were needed before Eqn. 25 would fit with $c = 0.5 \pm 0.1$

$$y = ax^{-b} + c \tag{25}$$

When LLM correlation was increased, overall bias reduced along with the number of generations needed before the decay followed a predictable pattern. In the limit of large G, through empirical study we found the exponent varied, we found this ranged from 0.25 to 0.75 for sensible hyper-parameters. Similar to before, approximately $G = 50$ generations were needed for a consistent fit.

B. Benchmarks

Figs. 5-10 shows the Capability Frontier for six selected benchmarks.

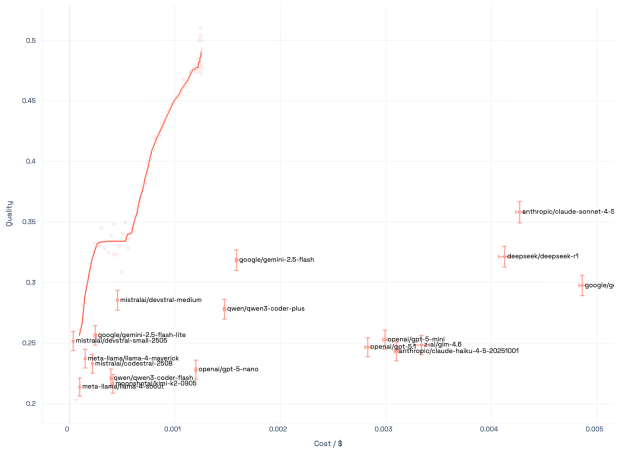


Figure 5. BigCodeBench

C. LLM Reliability Results

Tab. 3 summarizes the reliability of LLMs. The scores shown are averaged across all 16 benchmarks.

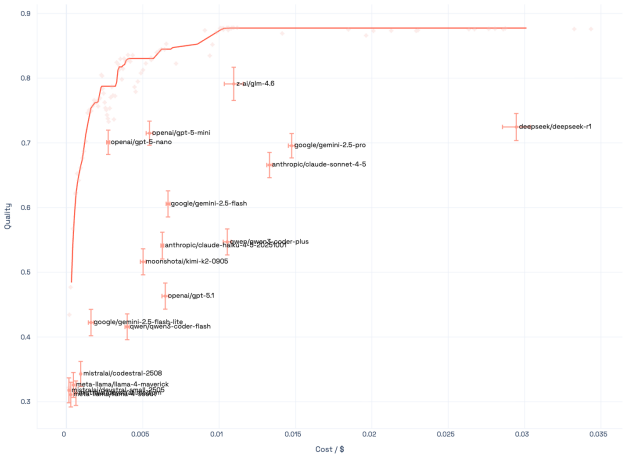


Figure 6. LeetCode

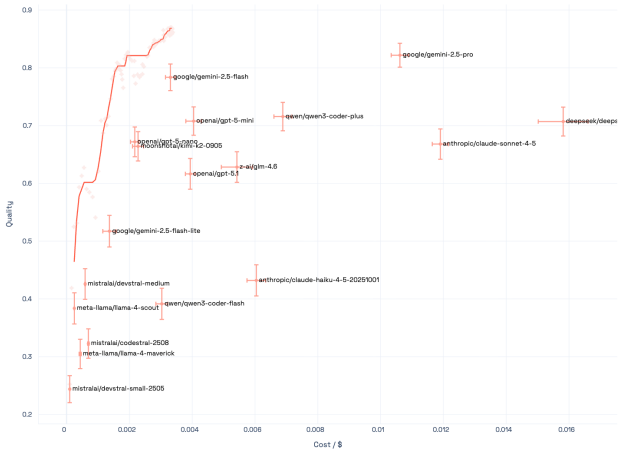


Figure 7. LiveBench-Coding

D. Posthoc Routing Gains

Tab.4 and Tab.5 shows the error rate reductions for $k = 1$ shot and $k = 10$ shot posthoc routing. The average error rate reduction is 66.1% and 82.4% respectively.

E. Quantifying the Bias

Fig.11 shows the bias quantification across 16 benchmarks.

Table 3. **LLMs have variable reliability** Comparison of LLMs for reliability in solving a problem. Scores averaged across all benchmarks.

Benchmark	Quality	Cost ($)	% Reliability
openai/gpt-5-mini	0.68	0.0040	90.2
anthropic/claude-sonnet-4-5	0.61	0.0615	87.8
google/gemini-2.5-pro	0.69	0.0174	87.6
mistralai/codestral-2508	0.44	0.1308	86.2
openai/gpt-5-nano	0.64	0.0014	85.8
mistralai/mistral-small-3.2-24b-instruct	0.50	0.0001	85.7
anthropic/claude-haiku-4-5-20251001	0.54	0.0256	85.7
mistralai/devstral-small	0.35	0.1158	84.8
mistralai/devstral-medium	0.46	0.0549	83.7
openai/gpt-5.1	0.58	0.0098	83.3
qwen/qwen-2.5-72b-instruct	0.48	0.0001	83.0
deepseek/deepseek-r1	0.52	0.0104	82.7
meta-llama/llama-4-scout	0.47	0.0005	82.7
qwen/qwen3-coder-flash	0.45	0.0039	82.4
google/gemini-2.5-flash-lite	0.52	0.0207	81.7
google/gemini-2.5-flash	0.65	0.0203	81.4
qwen/qwen3-coder-plus	0.53	0.0263	80.8
mistralai/devstral-small-2505	0.42	0.0051	80.1
qwen/qwen-max	0.53	0.0023	79.9
meta-llama/llama-4-maverick	0.51	0.0008	79.6
moonshotai/kimi-k2-0905	0.51	0.0043	77.1
z-ai/glm-4.6	0.45	0.0098	76.3
Average	0.52	0.0239	83.1%

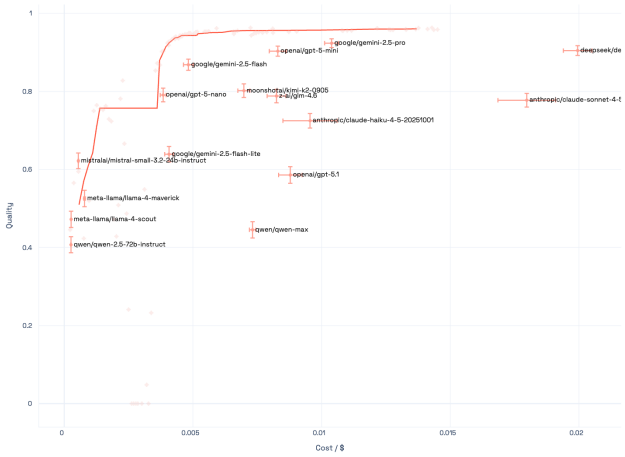


Figure 8. LiveBench-Reasoning

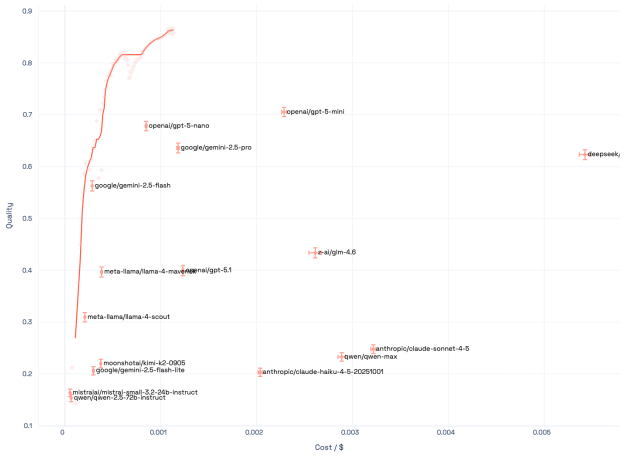


Figure 9. MedCalcBench

Table 4. Benchmark level breakdown comparing SOTA LLM with $\mathcal{O}^{kshot}(k=1)$.

Benchmark	% Quality		Cost ($)		% Error Rate Reduction ↑
	SOTA LLM	$\mathcal{O}^{\text{kshot}}(1)$	SOTA LLM	$\mathcal{O}^{\text{kshot}}(1)$	
LiveBench-Coding	82.2	89.8	0.011	0.079	43.0
BigCodeBench	35.8	66.7	0.004	0.032	48.1
LeetCode	79.1	89.3	0.011	0.129	48.8
HumanEval-X (Python)	97.4	99.6	0.003	0.029	83.7
HumanEval-X (CPP)	95.1	99.5	0.002	0.029	88.8
HumanEval-X (Javascript)	93.5	97.2	0.002	0.028	57.0
HumanEval-X (Java)	95.9	99.4	0.002	0.036	85.1
HumanEval-X (Go)	91.3	97.7	0.006	0.034	73.9
MBPP	86.2	93.4	0.001	0.029	51.8
MedCalcBench	70.5	90.8	0.002	0.023	68.6
TruthfulQA	98.8	99.6	0.004	0.009	66.3
LiveBench-IFEval	80.0	92.7	0.003	0.027	63.4
LiveBench-Reasoning	92.4	97.8	0.010	0.112	70.7
GPQA Diamond	94.0	100.0	0.005	0.036	100.0
Terminal-Bench 2.0 (agentic)	40.8	57.4	2.608	12.289	28.1
LiveBench-Coding (agentic)	85.5	97.1	0.033	2.017	79.7
Average	82.4	91.7	0.169	0.934	66.1

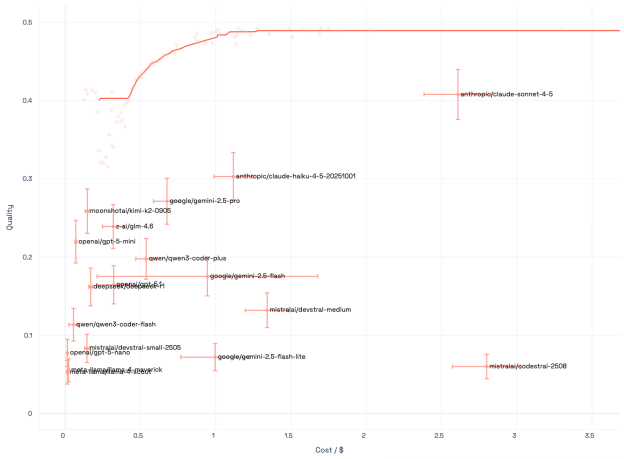


Figure 10. Terminal-Bench 2.0 (agentic)

Table 5. Benchmark level breakdown comparing SOTA LLM with $\mathcal{O}^{kshot}(k = 10)$.

Benchmark	% Quality		Cost ($)		% Error Rate Reduction ↑
	SOTA LLM	$\mathcal{O}^{\text{kshot}}(10)$	SOTA LLM	$\mathcal{O}^{\text{kshot}}(10)$	
LiveBench-Coding	82.2	93.8	0.011	0.789	64.9
BigCodeBench	35.8	77.5	0.004	0.318	65.0
LeetCode	79.1	93.9	0.011	1.292	70.6
HumanEval-X (Python)	97.4	100.0	0.003	0.290	100.0
HumanEval-X (CPP)	95.1	100.0	0.002	0.288	100.0
HumanEval-X (Javascript)	93.5	98.2	0.002	0.275	72.0
HumanEval-X (Java)	95.9	100.0	0.002	0.361	100.0
HumanEval-X (Go)	91.3	98.2	0.006	0.339	78.9
MBPP	86.2	95.5	0.001	0.286	67.3
MedCalcBench	70.5	96.6	0.002	0.234	88.5
TruthfulQA	98.8	99.9	0.004	0.088	89.1
LiveBench-IFEval	80.0	100.0	0.003	0.269	100.0
LiveBench-Reasoning	92.4	98.5	0.010	1.121	80.4
GPQA Diamond	94.0	100.0	0.005	0.358	100.0
Terminal-Bench 2.0 (agentic)	40.8	74.2	2.608	122.891	56.4
LiveBench-Coding (agentic)	85.5	97.8	0.033	20.166	85.1
Average	82.4	95.2	0.169	9.335	82.4

Table 6. **Naive oracles systematically overestimate.** Comparison of biased and debiased oracle estimates. Cost bias is substantially larger than quality bias.

Benchmark	Quality		Cost	
	$\mathcal{O}^{\text{biased}}$	$\mathcal{O}^{\text{true}}$	$\mathcal{O}^{\text{biased}}$	$\mathcal{O}^{\text{true}}$
LiveBench-Coding	87.3	86.8	0.31	0.33
BigCodeBench	53.6	49.1	0.12	0.12
LeetCode	87.9	87.7	0.74	1.05
HumanEval-X (Python)	99.5	99.5	0.02	0.02
HumanEval-X (CPP)	99.3	99.3	0.03	0.05
HumanEval-X (Javascript)	97.1	97.0	0.02	0.03
HumanEval-X (Java)	99.1	99.0	0.05	0.15
HumanEval-X (Go)	97.1	97.1	0.05	0.06
MBPP	92.5	92.3	0.03	0.06
MedCalcBench	87.3	86.5	0.10	0.11
TruthfulQA	99.5	99.5	0.01	0.01
LiveBench-IFEval	88.1	87.3	0.12	0.13
LiveBench-Reasoning	96.6	96.0	0.59	1.35
GPQA Diamond	99.0	99.0	0.02	0.02
Terminal-Bench 2.0 (agentic)	50.4	48.9	82.93	127.55
LiveBench-Coding (agentic)	96.0	96.0	1.55	1.76
Average	89.4	88.8	5.42	8.30

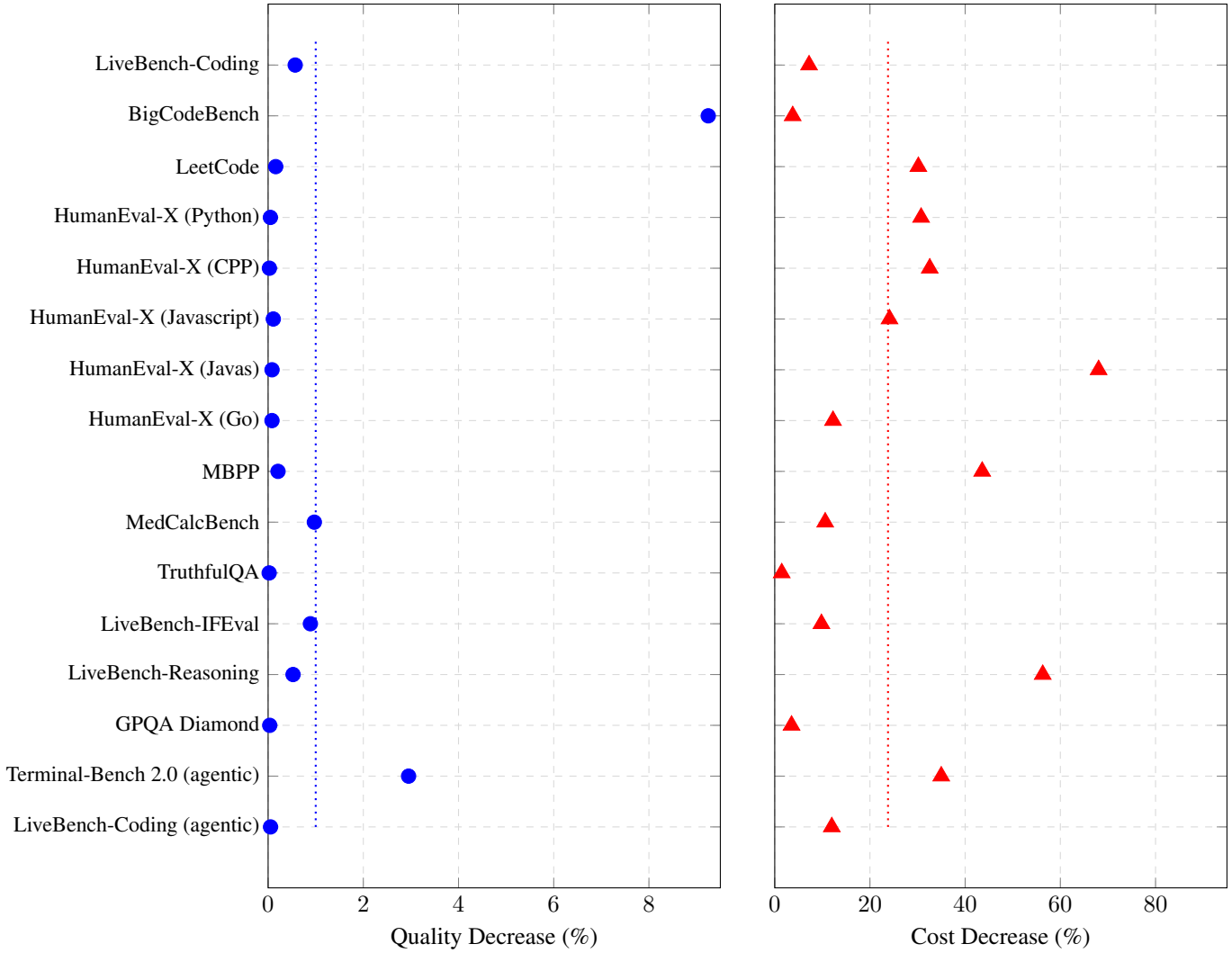


Figure 11. Bias quantification across benchmarks. The biased oracle $\mathcal{O}^{biased}$ overestimates both Quality (left, blue circles) and Cost (right, red triangles) relative to the true oracle $\mathcal{O}^{true}$. Quality bias is modest (under 9%, averaging $\approx 1.0\%$), while cost bias varies substantially (1.5–68%, averaging $\approx 23.8\%$). Dotted vertical lines indicate averages. See Tab.6 for detailed values.