# Measuring Nines of Reliability: Sample-Efficient LLM Evaluation on Saturated GSM Benchmark

**Eungyeup Kim** [1] **Chenchen Gu** [*1] **Vashisth Tiwari** [*1] **J. Zico Kolter** [1]

## Abstract

While existing benchmarks demonstrate the near-perfect performance of large language models (LLMs) on various tasks, this apparent saturation often obscures the need for rigorous evaluation of their reliability. In real-world deployment, however, achieving high reliability, where each added 'nine' (e.g., "four-nines" ($99.99\%$) vs. "three-nines" ($99.9\%$)) is critical, as this gap yields an order-of-magnitude increase in failures, catastrophic in reliability-critical applications. Still, estimating such a rare failure probability with tight confidence bounds requires prohibitively large LLM inference sizes, making standard Monte Carlo evaluation infeasible under limited compute budgets. In this paper, we observe that, on parameterized GSM8K math-reasoning tasks, LLM failures exhibit strong systematic patterns: across these parameterized input spaces, a small subset of inputs disproportionately accounts for the majority of failures. Leveraging this observation, we propose to learn a sampling distribution concentrated on failure-prone inputs via the cross-entropy method (CEM). We evaluate our framework on three LLMs, Qwen2.5-Math-7B-Instruct, gpt-oss-20b-low, and Gemini 2.5 Flash Lite, across parameterized GSM8K templates and achieve up to $156.22\times$ reduction in required inferences compared to naive uniform sampling. Our estimates reveal that models with indistinguishable accuracy on standard benchmarks can differ substantially in estimated failure rates, underscoring that reliability is a distinct and measurable axis of model quality. Our simple framework takes a first step toward evaluating high-reliability regimes of LLMs on saturated GSM benchmarks, a distinct and underexplored dimension of evaluation beyond existing benchmarks.

[1]Carnegie Mellon University. Correspondence to: Eungyeup Kim <eungyeuk@cs.cmu.edu>.

*Proceedings of the $43^{rd}$ International Conference on Machine Learning*, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

## 1. Introduction

Large language models (LLMs) show rapid progress not just in completing tasks with shorter duration (METR, 2026), but in giving saturating performance in numerous benchmarks (Maslej et al., 2025). For example, in AIME 2025 (American Invitational Mathematics Examination), GPT-5.2 Thinking (OpenAI, 2025) and Gemini-3-Pro (Google DeepMind, 2025) achieve a perfect score, while Claude Opus 4.6 (Anthropic, 2026) reaches $99.8\%$. While these near-perfect performances give the impression that models seem reliable at first glance, they may obscure how genuinely reliable these models are in real-world deployments where reliability becomes extremely important. As an example, in safety-critical settings, e.g., refusing harmful requests (OpenAI, 2026; Anthropic, 2026), even a vanishingly small gap in accuracy, e.g., $99.99\%$ ("four-nines") vs. $99.9\%$ ("three-nines") over a million queries, translates into an order-of-magnitude difference in failures, e.g., 1000 vs. 100 failures. Thus, this dimension of reliability evaluation is increasingly important but remains under-explored in current evaluation pipelines of existing benchmarks (Vendrow et al., 2024).

One major practical bottleneck is, for such rare failures, achieving the tight confidence bounds requires a prohibitively large sample size via Monte Carlo sampling under a uniform distribution $P$. For instance, to estimate $p = 10^{-4}$ ($99.99\%$ accuracy) with $\varepsilon = 10^{-5}$, $n \approx p/\varepsilon^2 = 10^{-4}/(10^{-5})^2 = 10^6$. Combined with the expense of LLM inference (which grows with long reasoning traces in recent models), uniform sampling for evaluation becomes computationally infeasible.

In this paper, we observe that, *on parameterized GSM8K math-reasoning problems, LLM failures are not uniformly distributed but exhibit systematic patterns*. Specifically, in parameterized GSM8K (Mirzadeh et al., 2025), across a combinatorially large input space, only a small set of inputs consistently induces failures at disproportionately high rates. For example, gpt-oss-20b-low (Agarwal et al., 2025) shows that approximately $82\%$ of its failures strongly co-occur with the input prompts with a specific parameter value (e.g., `fraction of blue ball` $= 2/7$ for GSM template 7), while other values rarely cause failures (Figure 2 middle row, Figure 3 right panel). Such concen-

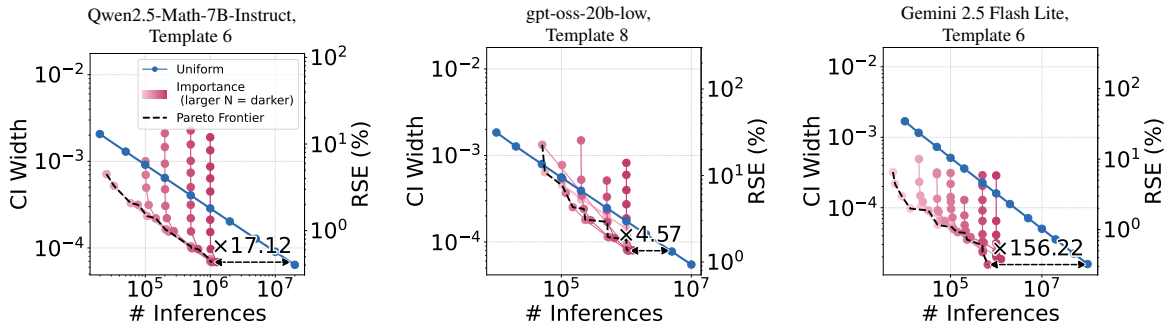


*Figure 1.* Each solid pink line connects estimations with varying number of inferences (pink dots) under importance sampling with same number of overhead inferences for CEM ($N$), and darker pink means larger $N$. Similarly, solid blue line for uniform sampling. Only estimates with coverage within $1\%$ tolerance of $99\%$ are shown. The Pareto-frontier (dotted line) is the lower-left envelope across all ($N$, # evaluation) combinations, which achieves a tight CI width, i.e., relative standard error (RSE) of around $1\%$, at a much smaller inference budget compared to uniform sampling (e.g., $156.22\times$ gain for Gemini 2.5 Flash Lite on Template 6).

tration appears across different models, such as Qwen2.5-Math-7B-Instruct (Yang et al., 2024) and Gemini 2.5 Flash Lite (Comanici et al., 2025), and other templates in GSM. This phenomenon sharpens under self-consistency decoding with majority voting (Wang et al., 2023), which filters out random errors and reveals inputs that consistently trigger failures across multiple generations.

This systematic failure structure motivates our approach: rather than sampling uniformly, we concentrate *sampling on failure-prone regions*. We cast this as a rare-event estimation problem and adopt the CEM (Rubinstein, 1999; De Boer et al., 2005) to learn a new sampling distribution $Q$ concentrated on failure-prone regions. Given just a few inferences, $Q$ iteratively converges to distribution that assigns higher probability to failure-prone input regions, which enables importance sampling for inference-efficient error estimation.

Our evaluation testbed includes multiple parameterized GSM8K templates (Cobbe et al., 2021; Mirzadeh et al., 2025), and three small but capable LLMs: Qwen2.5-Math-7B-Instruct, gpt-oss-20b-low, and Gemini 2.5 Flash Lite. Across models and templates, our method gains substantial inference efficiency, e.g., $156.22\times$ reduction in the required number of inferences for Gemini 2.5 Flash Lite on Template 6, for equivalent confidence bounds compared to uniform sampling (Figure 1 third plot). Notably, these gains grow as LLM failures become more concentrated, i.e., via increasing majority-vote ensemble size $K$. Specifically, efficiency improves from $3.68\times$ at $K = 4$ to $16.41\times$ at $K = 16$ for Qwen2.5-Math-7B-Instruct on Template 0 (Table 2). Our estimation results give $\sim 1\%$ relative standard errors for models with nearly $0.001$–$0.0001\%$ error rates, indicating that our method provides a practical evaluation pipeline for models' reliability on extremely saturated tasks. Applying this pipeline across models, we find that models all exceeding $99.9\%$ accuracy still differ by up to $2.4\times$ in estimated failure rates (e.g., gpt-oss-20b-low at $0.0254\%$ vs. Qwen2.5-Math-7B-Instruct at $0.0614\%$ on Template 0, $K{=}16$), confirming that saturated benchmarks retain discriminative power at the reliability level (Figure 5).

## 2. Related Work

**Evaluating reliability in saturated benchmarks.** As LLMs push the frontier, they are evaluated on new, challenging benchmarks (Jimenez et al., 2024; Merrill et al., 2026; Rein et al., 2024; Yue et al., 2024; 2025; Center for AI Safety et al., 2026; Foundation, 2026), leaving behind older, saturated ones (Cobbe et al., 2021; Zellers et al., 2019; Wang et al., 2018; Sakaguchi et al., 2019) on which frontier models are now considered to perform nearly perfectly. Yet near-perfect is not perfect: Vendrow et al. (2024) showed that frontier LLMs, such as o1-2024-12-17 (high) (OpenAI, 2024), retain rare failures, around $0.1\%$, on GSM8K (Cobbe et al., 2021) even after correcting noisy label errors. Relatedly, GSM-Symbolic (Mirzadeh et al., 2025) demonstrates that frontier models' near-perfect performance degrades on parameterized variants of the same problem templates. We focus on highly saturated GSM benchmarks, targeting rare-failure regimes up to roughly "four-nines" ($99.99\%$) accuracy, where the rarity of failures makes accurate estimation computationally costly compared to less saturated settings.

**Efficient LLM evaluation.** The autoregressive nature of LLMs makes inference expensive, and this cost grows as test-time compute scales (OpenAI, 2024; Guo et al., 2025). Several approaches reduce evaluation cost by selecting informative subsets of benchmark items (Maia Polo et al., 2024; Vivek et al., 2024; Perlitz et al., 2024; Hofmann et al., 2025), leveraging historical information (Wu et al., 2026), or using difficulty-adaptive testing (Truong et al., 2025). These studies target general capability evaluation on non-saturated benchmarks, whereas our work specifically addresses the

rare-failure regime of highly saturated benchmarks.

**Rare-event estimation with the CEM.** The CEM (Rubinstein, 1999; Rubinstein & Kroese, 2004; De Boer et al., 2005) is a method for estimating rare-event probabilities by iteratively learning a proposal distribution concentrated on the rare-event region, enabling importance sampling with substantially reduced variance. A few works have adopted the CEM for rare-event estimation, including estimating failure probabilities of black-box physical systems in safety-critical applications (Arief et al., 2021) and estimating rare predictions of linear predictors (Bai et al., 2022). Our work is the first to apply the CEM to LLM evaluation, where the rare event is a majority-vote failure, the parameter space is a high-dimensional categorical distribution over problem instances, and each LLM inference is expensive.

## 3. Problem and Experimental Setup

**Problem parameterization.** To evaluate a model over a broad problem distribution, we construct a parameterized representation of task instances (e.g., mathematical reasoning prompts) as a mapping from the parameters $Z$ to natural language prompts. Specifically, we define $Z = (Z_1, \ldots, Z_d)$ as a vector of $d$ independent categorical variables, where each $Z_i$ takes values in a finite domain $\mathcal{Z}_i$. Following Mirzadeh et al. (2025), we parameterize all types of variables — including nouns and numerical values. To illustrate, consider a template where specific nouns and values are treated as parameters:

> **Prompt (Template 0)**
> $\{Z_{\text{Name}}\}$ saw a $\{Z_{\text{x}}\}$-foot $\{Z_{\text{Fish}}\}$ with $\{Z_{\text{y}}\}$ remoras, each $\{Z_{\text{z}}\}$-inches long, attached to it. What percentage of the $\{Z_{\text{Fish}}\}$'s body length is the combined length of the remoras?

Each placeholder corresponds to a variable $Z_i$ sampled from its domain $\mathcal{Z}_i$ (e.g., $Z_{\text{Name}} \in \mathcal{Z}_{\text{Name}}$, $Z_x \in \mathcal{Z}_x$). This formulation provides a structured, factorized representation of the problem input space. Let $\mathcal{Z} = \prod_{i=1}^{d} \mathcal{Z}_i$ denote the full parameter space. Additional details are given in Appendix G.

**Sampling distribution and failure.** Given the parameter space $\mathcal{Z}$, we define a uniform sampling distribution $P(Z) = \frac{1}{|\mathcal{Z}|}$. Our goal is to estimate the failure probability under $P$,

$$\mu \coloneqq \mathbb{E}_{Z\sim P}\left[\mathbb{P}(f(Z) = 1)\right], \tag{1}$$

where for each input $Z$, $f(Z) \in \{0, 1\}$ is a Bernoulli random variable indicating whether majority voting over $K$ independent stochastic generations yields an incorrect prediction.

Modern LLMs often produce non-deterministic outputs (e.g., under non-zero temperature), especially for reasoning tasks. To obtain more consistent predictions, we adopt self-consistency decoding (Wang et al., 2023): for each input $Z$, we perform $K$ independent inference runs and aggregate the outputs via majority voting. This reduces output variance by aggregating over diverse reasoning paths, approximating the mode of the model's output distribution.

Formally, we define the failure indicator as

$$f(Z) \coloneqq \mathbb{1}\{\text{MajorityVote}\Big(\hat{y}_1(Z), \ldots, \hat{y}_K(Z)\Big) \neq y\}, \tag{2}$$

where $\hat{y}_k(Z)$ denotes the model output from the $k$-th independent inference on input $Z$, and $y$ is the ground-truth label. Ties are counted as failures.

**Datasets and models.** Building on GSM-Symbolic (Mirzadeh et al., 2025), we generate approximately 100K instances per template by varying parameter ranges, yielding nine templates in total. Details of all templates used throughout our paper can be found in Appendix G. We notice that for some template-model pairs, failures are extremely rare (i.e., the template is too easy for some model) which is potentially non-zero but undetectable within our feasible evaluation budget. Therefore, we only report results for pairs with a sufficient number of observed failures to support reliable estimation. Details of such excluded pairs are provided in Section C.

Due to compute budget limits, we focus on relatively efficient yet capable LLMs, including Qwen2.5-Math-7B-Instruct (Yang et al., 2024), gpt-oss-20b-low (Agarwal et al., 2025) with low reasoning efforts, and Gemini 2.5 Flash Lite (Comanici et al., 2025). All inferences follow the models' recommended decoding configurations (e.g., temperature, top-$p$, and max number of tokens). Further details are provided in Appendix A.

**Bootstrapping Inferences.** Ideally, evaluating an LLM requires fresh inferences for each generation, since LLM outputs are inherently stochastic (i.e., non-zero temperature), particularly during reasoning. However, this becomes computationally infeasible for evaluating with large sample sizes for rare failures, across multiple models, templates, and random seeds (for coverage). Therefore, we employ *bootstrap resampling* (Efron, 1979) to approximate the stochastic variation of LLM outputs without fresh inference for each input prompt. Specifically, we pre-compute and cache a pool of independent generations for each input, 16 generations for Qwen2.5-Math-7B-Instruct, 40 for gpt-oss-20b-low, and 32 for Gemini 2.5 Flash Lite, which is deemed large enough to approximate the true sampling. We randomly sample $K$ outputs with replacement from this cached pool to simulate independent generations for majority voting. This approximates the distribution of majority-vote outcomes given fresh inferences without additional inference costs.

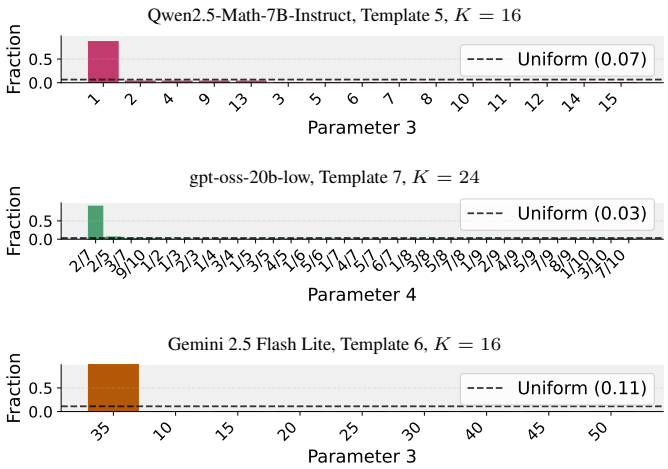


*Figure 2.* Distribution of failures across parameter values for model–template pairs. Each bar represents the proportion of failures for each parameter value, and the dotted line shows the expected uniform error rate.

| Model | ID | Distance | | |
|---|---|---|---|---|
| | | Top1 | Top2 | Top3 |
| Qwen2.5-Math -7B-Instruct ($K = 16$) | 0 | 0.900 | 0.750 | 0.509 |
| | 1 | 0.914 | 0.833 | 0.750 |
| | 2 | 0.538 | 0.249 | 0.019 |
| | 3 | 0.472 | 0.362 | 0.310 |
| | 4 | 0.339 | 0.264 | 0.185 |
| | 5 | 0.940 | 0.867 | 0.800 |
| | 6 | 0.879 | 0.752 | 0.247 |
| | 7 | 0.798 | 0.748 | 0.557 |
| gpt-oss -20b-low ($K = 24$) | 0 | 0.933 | 0.850 | 0.729 |
| | 2 | 0.699 | 0.541 | 0.052 |
| | 4 | 0.631 | 0.508 | 0.385 |
| | 7 | 0.895 | 0.633 | 0.537 |
| | 8 | 0.760 | 0.599 | 0.487 |
| Gemini 2.5 Flash Lite ($K = 16$) | 0 | 0.867 | 0.850 | 0.267 |
| | 2 | 0.907 | 0.750 | 0.080 |
| | 4 | 0.208 | 0.133 | 0.102 |
| | 6 | 0.983 | 0.889 | 0.189 |

*Table 1.* Total variation distance between failure distributions and uniform baseline. Parameters with the three largest distances are reported for each template.

**Prompt (Template 5)**
{Katy} makes {coffee} using teaspoons of sugar and cups of water in the ratio of {15}: {1}. If she used a total of {736} teaspoons of sugar and cups of water, calculate the number of teaspoonfuls of sugar she used.

---

**LLM Output (Qwen2.5-Math-7B-Instruct)**
To determine the number of teaspoonfuls of sugar Katy used, we start by
...
Since 1 cup of water is equivalent to 16 teaspoons (assuming standard US measurements
...
Therefore, the number of teaspoonfuls of sugar Katy used is: $\boxed{360}$
Answer: **690**

**Prompt (Template 7)**
{Jamie} is juggling at a {circus}. {Jamie} can juggle {168} balls. {1/2} of the balls are {golf} balls, and {2/7} of the {golf} balls are {blue}. How many {blue} {golf} balls are there?

---

**LLM Output (gpt-oss-20b-low)**
analysisCompute: total balls 168. Half are golf balls: 168/2=84 golf balls. Of those, 2/7 are blue: 84*(2/7)=12.assistantfinalJamie has
($\boxed{12}$) blue golf balls.
Answer: **24**

*Figure 3.* Example pairs of prompt-generations that illustrate repetitive failures by LLMs. Failure-inducing parameter values and output traces are highlighted in red.

## 4. LLM Failures Are Systematic, Not Random

**Failures concentrate on a small subset of parameters.** We observe that *on the parameterized GSM tasks we study, LLM failures are not random, but exhibit systematic structure*. Within the high-dimensional parameter space $\mathcal{Z}$, a small subset of parameter values disproportionately accounts for a large fraction of failures. Across models and templates, we observe striking concentration of failures on individual parameter values (Figure 2). We first iterate over all possible inputs for testing models, and aggregate all failures observed. We observe that a single parameter value accounts for the majority of errors: parameter `term of water cups` $= 1$ in Template 5 for Qwen2.5-Math-7B-Instruct (86% of failures), parameter `fraction of blue ball` $= 2/7$ in Template 7 for gpt-oss-20b-low (Agarwal et al., 2025) (82%), and parameter `area of fruits` $= 35$ in Template 6 for Gemini 2.5 Flash Lite (Comanici et al., 2025) (81%). Bar plots for all parameters are given in Appendix Figure 6.

To quantify this concentration, we compute the total variation (TV) distance between the normalized histogram of the failure counts over each parameter dimension ($Z_i$ such as $Z_{\text{Name}}$) and the uniform distribution (Table 1). The uniform distribution represents a baseline where the failures occur evenly across parameters. Thus, larger distances indicate more pronounced failure concentration, while a value of zero indicates that failures are spread evenly. For each setup, we report the top-3 parameter dimensions with the largest divergence. Across models and templates, we observe consistently large deviations from the uniform distribution, i.e., most Top-1 distances reach 0.9. Furthermore, as we increase $K$, we observe that the TV distance increases (Table 3), which implies that with larger $K$, the majority of failures becomes more pronounced, with fewer inconsistent failures. Specifically, larger $K$ filters out random errors—failures caused by incidental factors in individual

generations—while amplifying systematic failure patterns that persist across multiple samples. These results suggest that LLM failures are not spread randomly across the parameter space, but are instead driven by a small subset of failure-inducing inputs.

**Qualitative analysis of repetitive failures.** To understand how the failures concentrate in certain regions of the parameter space, we conduct a qualitative analysis of LLM-generated outputs that exhibit repetitive failures, as in Figure 3. We mark in red the input parameter values that frequently induce failures as well as the erroneous reasoning trace. For instance, Qwen2.5-Math-7B-Instruct consistently applies an unjustified unit conversion heuristic when the parameter value `term for cup of water` $= 1$ is given in Template 5. gpt-oss-20b-low fails at simple multiplication when $2/7$ is given as `fraction of blue ball` in Template 7. Similarly, Gemini 2.5 Flash Lite fails in $35 \times 43$ in Template 6 (Figure 27). These models exhibit such errors repeatedly across multiple generations, as illustrated in Figures 19,23, and 27. We provide additional qualitative analysis for other template-model pairs in Section I, which show various error types including wrong arithmetic operations, unnecessary approximation, problem misinterpretation, and others. These examples reveal that LLM errors in GSM problems stem from consistent, input-dependent reasoning behaviors, providing insight into the conditions under which models are likely to produce incorrect reasoning trajectories. Despite there being no a priori reason to expect systematic failure patterns to persist across template-model pairs, we find compelling empirical evidence that such patterns arise consistently across most pairs we examine.

## 5. Learning Failure-Prone Sampling Distributions

Our key finding of the concentration of failures in Section 4 naturally leads to a failure-prone sampling strategy—rather than drawing samples from the full parameter space under $P$, we concentrate sampling on failure-prone regions. To this end, we leverage the CEM to iteratively refine a proposal distribution $Q$ toward regions of the parameter space where failures are more likely, which we can use to estimate true error probability via importance sampling.

**Importance Sampling with $Q$.** To estimate $\mu$ defined in Eq. 1, a standard Monte Carlo estimator under $P$ is

$$\hat{\mu}_P = \frac{1}{N}\sum_{i=1}^{N} f(Z_i), \quad Z_i \sim P.$$

However, when failures are rare, the estimator suffers from high variance relative to $\mu$, requiring large numbers of samples to obtain tight confidence bounds. To improve efficiency, we introduce a proposal distribution $Q$ and estimate $\mu$ via importance sampling:

$$\hat{\mu}_Q = \frac{1}{N}\sum_{i=1}^{N} f(Z_i)\,\frac{P(Z_i)}{Q(Z_i)}, \quad Z_i \sim Q,$$

which yields an unbiased estimate under the support condition that $Q(Z) > 0$ whenever $\mathbb{P}(f(Z) = 1)\,P(Z) > 0$. To ensure that importance weights are bounded, we employ defensive sampling (Hesterberg, 1995), which mixes $Q$ with $P$ with mixing coefficient $\lambda$.

**Algorithm 1** Failure Probability Estimation via CEM

**Input:** Failure indicator $f$, original distribution $P$, sample size $N$, smoothing coefficient $\zeta$, iterations $T$, confidence level $\alpha$, defensive sampling coefficient $\lambda$
**Output:** Failure probability estimate $\hat{\mu}$, confidence interval $[\hat{\mu}_{\text{low}}, \hat{\mu}_{\text{high}}]$
**// CEM for updating $Q$**
Initialize $Q_0(Z_i)$ as uniform over $\mathcal{Z}_i$ for all $i = 1, \ldots, d$
**for** $t = 1$ to $T$ **do**
  **// Sampling**
  Sample $Z_1, \ldots, Z_N \sim Q_t(Z) = \prod_{i=1}^{d} Q_t(Z_i)$
  **// Elite set selection**
  Define elite set $\mathcal{E}_t \leftarrow \{n : f(Z_n) = 1\}$
  **if** $\mathcal{E}_t = \emptyset$ **then**
    Set $Q_{t+1} \leftarrow Q_t$ and **continue**
  **end if**
  **// Weighted update**
  **for** each dimension $i$ and value $j \in \mathcal{Z}_i$ **do**
    Compute importance weights $w(Z_n) \leftarrow P(Z_n)/Q_t(Z_n)$
    $Q'_{t+1}(Z_i)_j \leftarrow \frac{\sum_{n\in\mathcal{E}_t} w(Z_n)\mathbb{1}\{z_{n,i}=j\}}{\sum_{n\in\mathcal{E}_t} w(Z_n)}$
  **end for**
  **// Smoothing**
  $Q_{t+1} \leftarrow \zeta Q'_{t+1} + (1-\zeta)Q_t$
**end for**
Apply defensive sampling $\tilde{Q}_T(Z) \leftarrow (1-\lambda)Q_T(Z) + \lambda P(Z)$
**// Estimation with confidence bounds**
$\hat{\mu}, [\hat{\mu}_{\text{low}}, \hat{\mu}_{\text{high}}] \leftarrow$ COMPUTEESTIMATE$(f, P, \tilde{Q}_T, N, \alpha)$
**Return** $\hat{\mu}, [\hat{\mu}_{\text{low}}, \hat{\mu}_{\text{high}}]$

**CEM for learning $Q$.** The variance-minimizing proposal distribution is given by $Q^*(Z) = \frac{\mathbb{P}(f(Z)=1)P(Z)}{\mu}$, which concentrates probability mass on failure regions. The CEM seeks a proposal distribution $Q$ that approximates this optimal distribution by minimizing the Kullback-Leibler divergence, $\mathrm{KL}\,(Q^* \parallel Q)$. Since $Q^*$ depends on the unknown $\mu$, the CEM iteratively refines $Q$ by sampling from the current proposal and updating toward the empirical failure-weighted distribution proportional to $\mathbb{P}(f(Z) = 1)\,P(Z)$.

**Failure probability estimation via $Q$ learned from CEM.** To begin with, we model $Q$ as a fully factorized distribution $Q(Z) = \prod_{i=1}^{d} Q(Z_i)$, where each $Q(Z_i)$ is a categorical

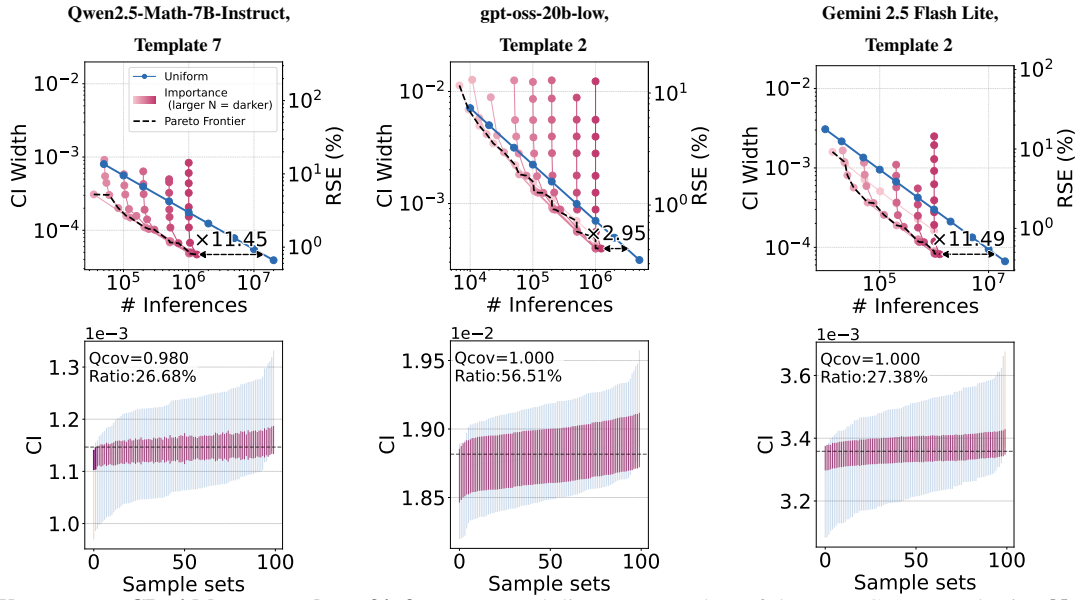


*Figure 4.* **Upper row: CI width vs. number of inferences.** Each line connects dots of the same CEM sample size $N$ (darker pink = larger $N$) at varying evaluation sizes; only estimates with coverage within 1% tolerance of 99% are shown. The Pareto-frontier (dotted line) is the lower-left envelope across all ($N$, evaluation size) combinations. Importance distribution $Q$ achieves RSE of 1% at a much smaller inference budget than uniform sampling $P$. **Lower row: sorted CIs over** 100 **sample sets.** At the evaluation size of the bottom-right Pareto-frontier dot, $Q$ (pink) yields much narrower confidence bounds than $P$ (blue) for the same sample size, while maintaining coverage close to 99%. We highlight the intervals that fail to cover the population mean in dark pink.

distribution over $\mathcal{Z}_i$. This independence assumption provides a tractable approximation to model the joint parameter space. We initialize $Q_0(Z)$ as a uniform distribution over each parameter dimension $Z_i \in \mathcal{Z}_i$. We then apply the CEM followed by importance sampling for estimating true failure probability with $Q$ (Algorithms 1 and 2).

## 6. Order-of-Magnitude Inference Efficiency Gains

**Setup for CEM** For CEM, we use fixed hyperparameters across all models and templates: $T$=10 iterations, smoothing coefficient $\zeta$=0.1, and defensive sampling coefficient $\lambda \in \{0.1, 0.3, 0.5\}$. The CEM sample size $N$, the number of data for updating $Q$ at each CEM iteration $t$, is set as a hyperparameter, ranging between 0.5K and 100K.

**Setup for estimation.** For the final estimation, we vary the evaluation sample size $M$ between 1K and 1M (million). We define the *number of inferences* as the number of samples from $P$ or $Q$ (one inference per $Z$, independent of $K$). Because the $T \times N$ samples drawn during CEM can be reused directly as importance sampling estimation samples, the total number of inferences under $Q$ is $\max(T \times N,\ M)$: if the CEM draws already reach or exceed $M$, no additional inferences are needed; otherwise, $M - T \times N$ samples are drawn from $\tilde{Q}_T$. For $P$, the cost is simply $M$ inferences with no CEM phase.

**Setup for evaluation.** For all evaluations, we construct 99% confidence intervals (significance level $\alpha = 0.01$) using the normal approximation applied to the importance-weighted estimator $\hat{\mu}_Q$. To verify the unbiasedness of our estimator, we draw 100 independent sample sets for each sample size and report the estimates whose coverage falls within $99 \pm 1\%$ (i.e., 1% tolerance of the nominal coverage level). For calculating coverage, we use $\hat{\mu}_P$ computed using 1B bootstrap samples from $P$, which converges to the unknown true mean.

We compare against uniform sampling from $P$ with confidence bounds computed via the Clopper-Pearson binomial exact method (Clopper & Pearson, 1934). To quantify the inference efficiency obtained by importance sampling, we present the *sampling efficiency gain*, calculated by the number of inferences under $P$ divided by that under $Q$ (i.e., $\max(T \times N, M)$) to obtain the same CI width. Across $N$ and evaluation sizes, we report the gains of the Pareto-frontier in the CI width vs. number of inferences trade-off (i.e., dotted line in Figures 1 and 4). We additionally report *RSE* defined as RSE $= \frac{\text{SE}}{\hat{\mu}}$ to evaluate the tightness of the confidence bounds.

**Failure-prone sampling saves an order of magnitude in inferences.** Our results demonstrate that the learned distribution $Q$, which concentrates on frequent failure patterns, yields substantially tighter confidence bounds compared to uniform sampling under $P$. In Figure 1 and upper row of

| | $\lambda = 0.1$ | | | | | | $\lambda = 0.3$ | | | | | | $\lambda = 0.5$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Gains** | | | **RSE** (%) | | | **Gains** | | | **RSE** (%) | | | **Gains** | | | **RSE** (%) | | |
| **Qwen2.5-Math-7B-Instruct** | | | | | | | | | | | | | | | | | | |
| ID | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 |
| 0 | 3.68 | 6.50 | 16.41 | 1.24 | 1.22 | 1.02 | 4.51 | 8.03 | 15.84 | 1.27 | 1.21 | 1.05 | 3.82 | 6.59 | 13.65 | 1.27 | 1.26 | 1.13 |
| 1 | 1.31 | 2.29 | 5.33 | 1.97 | 2.99 | 2.95 | 1.58 | 2.34 | 7.02 | 1.94 | 3.00 | 2.46 | 1.57 | 2.30 | 6.02 | 1.95 | 3.21 | 2.69 |
| 2 | 1.63 | 1.48 | 1.87 | 0.22 | 0.24 | 0.27 | 1.37 | 1.48 | 1.80 | 0.22 | 0.25 | 0.27 | 1.51 | 1.61 | 1.68 | 0.22 | 0.26 | 0.29 |
| 3 | 1.07 | 1.46 | 1.75 | 0.43 | 0.63 | 0.80 | 1.25 | 1.50 | 1.49 | 0.43 | 0.63 | 0.81 | 1.24 | 1.25 | 1.64 | 0.43 | 0.64 | 0.83 |
| 4 | 1.34 | 1.38 | 1.22 | 0.31 | 0.36 | 0.40 | 1.30 | 1.38 | 1.23 | 0.31 | 0.36 | 0.40 | 1.29 | 1.33 | 1.41 | 0.31 | 0.37 | 0.41 |
| 5 | 2.15 | 3.67 | 6.93 | 1.20 | 1.77 | 5.05 | 2.51 | 4.28 | 5.40 | 1.14 | 1.87 | 2.81 | 2.32 | 3.69 | 5.62 | 1.19 | 1.85 | 2.78 |
| 6 | 5.36 | 15.84 | 17.12 | 0.44 | 0.44 | 0.43 | 6.40 | 12.87 | 15.92 | 0.49 | 0.49 | 0.47 | 5.80 | 9.46 | 11.74 | 0.56 | 0.56 | 0.54 |
| 7 | 3.22 | 8.02 | 11.45 | 1.08 | 0.95 | 0.79 | 3.53 | 8.45 | 14.66 | 1.03 | 0.89 | 0.80 | 4.44 | 6.42 | 11.59 | 1.03 | 0.96 | 0.87 |
| 8 | 1.91 | — | — | 4.97 | — | — | 2.73 | — | — | 4.09 | — | — | 2.45 | — | — | 4.57 | — | — |
| **gpt-oss-20b-low** | | | | | | | | | | | | | | | | | | |
| ID | $K$=8 | $K$=16 | $K$=24 | $K$=8 | $K$=16 | $K$=24 | $K$=8 | $K$=16 | $K$=24 | $K$=8 | $K$=16 | $K$=24 | $K$=8 | $K$=16 | $K$=24 | $K$=8 | $K$=16 | $K$=24 |
| 0 | 3.34 | 3.81 | 3.63 | 2.56 | 2.95 | 3.92 | 2.95 | 3.79 | 6.77 | 2.52 | 3.10 | 2.85 | 2.71 | 4.45 | 3.54 | 2.63 | 3.11 | 4.15 |
| 2 | 2.27 | 2.70 | 2.95 | 0.28 | 0.37 | 0.41 | 2.28 | 2.86 | 3.28 | 0.30 | 0.39 | 0.43 | 1.93 | 2.35 | 2.81 | 0.32 | 0.42 | 0.46 |
| 4 | 1.30 | 1.93 | 2.65 | 0.79 | 1.19 | 1.36 | 1.55 | 1.69 | 2.30 | 0.79 | 1.16 | 1.34 | 1.53 | 1.91 | 2.09 | 0.80 | 1.19 | 1.40 |
| 7 | 2.89 | 2.80 | 4.45 | 1.61 | 2.76 | 2.29 | 3.07 | 3.50 | 5.22 | 1.62 | 2.14 | 2.18 | 2.39 | 4.29 | 5.39 | 1.65 | 2.11 | 2.31 |
| 8 | 4.36 | 4.41 | 4.57 | 0.84 | 1.18 | 1.36 | 4.02 | 4.42 | 4.65 | 0.90 | 1.21 | 1.42 | 2.99 | 3.43 | 3.70 | 0.97 | 1.32 | 1.49 |
| **Gemini 2.5 Flash Lite** | | | | | | | | | | | | | | | | | | |
| ID | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 | $K$=4 | $K$=8 | $K$=16 |
| 0 | 0.96 | 4.05 | 5.29 | 1.95 | 1.90 | 2.35 | 1.23 | 4.74 | 8.23 | 1.64 | 1.72 | 1.74 | 1.35 | 4.90 | 8.62 | 1.57 | 1.71 | 1.93 |
| 2 | 2.54 | 4.41 | 11.49 | 0.68 | 0.62 | 0.47 | 2.76 | 5.69 | 10.35 | 0.65 | 0.62 | 0.51 | 2.38 | 5.16 | 9.50 | 0.66 | 0.64 | 0.57 |
| 4 | 1.12 | 0.99 | 1.00 | 0.40 | 0.51 | 0.58 | 1.11 | 1.18 | 1.19 | 0.40 | 0.51 | 0.58 | 0.95 | 1.16 | 1.00 | 0.40 | 0.51 | 0.58 |
| 6 | 1.06 | 2.75 | 156.22 | 0.99 | 1.80 | 0.32 | 1.10 | 3.45 | 85.21 | 0.98 | 1.63 | 0.36 | 1.27 | 3.32 | 82.59 | 0.98 | 1.52 | 0.37 |

*Table 2.* Sampling efficiency gains and RSE (%) across templates, models, $K$, and $\lambda$ for majority voting. Cell shading for gains is proportional to the gain value, i.e., the higher the darker, capped at gain $\geq 10$. — denotes zero failures.

Figure 4, we present CI width under $Q$ (pink) for varying evaluation sizes (dots connected by lines) with different $N$, the number of samples for CEM (different shades). The Pareto-frontier (dotted line), which connects the minimum number of inferences required to achieve each CI width, decreases much more rapidly under $Q$ as the number of inferences increases than under $P$ (blue). This indicates that, for a tight bound, i.e., $1\%$ RSE, importance sampling under $Q$ requires significantly fewer inferences than uniform sampling from $P$. Specifically, such inference efficiency gains span orders of magnitude across models and templates, e.g., $17.12\times$ for Qwen2.5-Math-7B-Instruct on Template 6 ($K = 16$), $4.57\times$ for gpt-oss-20b-low on 8 ($K = 24$), and $156.22\times$ for Gemini 2.5 Flash Lite on 6 ($K = 16$).

These inference efficiency gains stem from the tight CIs obtained through importance sampling, as illustrated in Figure 4 bottom row. The plots compare the confidence bounds between $P$ and $Q$ across 100 independent sample sets (sorted by their center). Across evaluation sizes, we plot the bounds for the sample size corresponding to the bottom-right pink dot on the Pareto-frontier. Intervals under $Q$ (pink) are much narrower than those under $P$ (blue) for the same sample size. These 100 intervals faithfully bound the $\hat{\mu}_P$ (with $\pm 1$ tolerance), confirming that our method meets the nominal $99\%$ confidence level.

We report inference efficiency gains and their RSE across models and templates in Table 2. For each gain, we add darker shading for larger values for visibility. The results demonstrate that our method achieves consistent efficiency gains, where most exceed $2\times$. These estimations show substantially low RSE, where most are around just $1\%$, implying that we can estimate LLMs' true error probability with highly accurate estimation across templates and models.

We observe that inference efficiency gains improve as we use larger $K$ for self-consistency decoding (Wang et al., 2023), i.e., shading becomes darker as $K$ increases in gains. For instance, Qwen2.5-Math-7B-Instruct shows a gain of $3.68\times$ with $K = 4$ increasing to $16.41\times$ with $K = 16$ on Template 0, sharing the trends across setups. These results corroborate our empirical observations that systematic failures become more concentrated with higher $K$. In addition, overall gains and RSE remain consistent across different $\lambda$, which allows practitioners to use our method without careful hyperparameter tuning. Similarly, efficiency gains are strongly correlated with failure rarity: as models become more saturated on a task, failures concentrate on a smaller subset of inputs, making importance sampling more effective. As a concrete example, Gemini 2.5 Flash Lite shows higher error rate ($\hat{\mu}_Q$) on Template 4, i.e., $2.59\%$, than others ($0.335\%$-$0.0328\%$) with $K = 16$ correspondingly, those high error-rate templates also exhibit lower TV distance (Table 1) and smaller efficiency gains (Table 2),

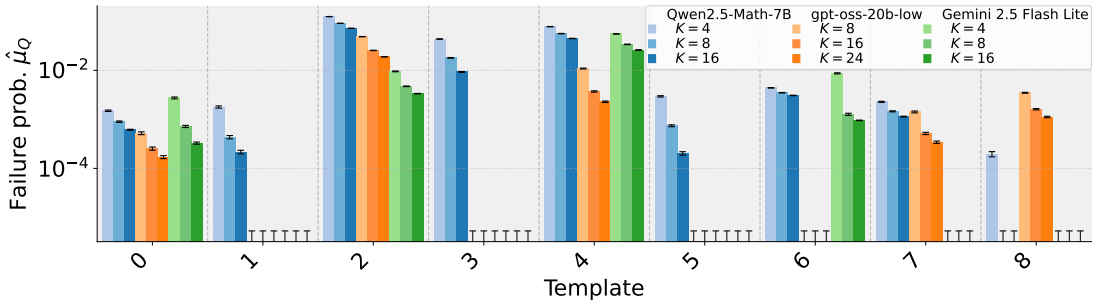


*Figure 5.* Estimated failure probability $\hat{\mu}_Q$ across templates and models at $\lambda = 0.1$ and evaluation size of $1M$, with error bars denoting the half-width of the confidence interval. For those with zero failures, we report the error bars of exact binomial bounds, i.e., (0, 5.30e-06).

which is consistent with our finding that stronger failure concentration drives higher importance sampling efficiency.

## 7. Reliability Evaluation of Models on GSM

Finally, Figure 5 reports each model's estimated rare failure probability $\hat{\mu}_Q$ across GSM templates, presented as a point estimate with confidence bounds. While all models achieve saturating performance with error rates well below $1\%$, our evaluation reveals meaningful differences in reliability at finer granularity. Specifically, gpt-oss-20b-low gives a failure rate of just $0.0254\%$ on Template 0 with $K = 16$, which outperforms Qwen2.5-Math-7B-Instruct ($0.0614\%$, $\sim 2.5\times$ higher) and Gemini 2.5 Flash Lite ($0.0328\%$, $\sim 1.3\times$ higher). The full table is provided in Table 4. These results highlight that such saturated benchmarks still carry discriminative signal for reliability.

## 8. Limitations, Future Work, and Conclusion

**Limitations.** Our method relies on the assumption that LLM errors concentrate on a small subset of the parameter space, which we validate empirically across the models and templates we study. When error patterns are diffuse, e.g., when the model is relatively weak for a task, the CEM may not find a substantially better proposal than uniform sampling. Our method may also be less effective when estimating failure rates without majority voting ($K$=1), where stochastic generation outputs produce inconsistent failure patterns that are harder to concentrate on. In addition, due to computational constraints, our experiments focus on relatively small LLMs. Evaluating reliability introduces significant overhead: the required number of inferences grows with the evaluation sample size, the majority-vote ensemble size $K$, and the number of independent trials for coverage verification, all of which increase as failures become rarer or as the desired confidence level tightens. Extending this framework to larger, more capable LLMs remains an important direction for future work.

**Lack of reliability benchmarks.** While nearly all existing benchmarks focus on evaluating how LLMs push the frontier in challenging tasks, to our knowledge, no benchmark specifically tests their reliability on already-saturated benchmarks. Rather than revisiting these saturated benchmarks, the community tends to curate novel, more challenging ones, leaving a gap in our understanding of model reliability. In this paper, we take a first step toward addressing this gap by introducing a prototypical framework for evaluating LLM reliability using parameterized GSM templates. However, reliability evaluation must be extended to broader domains, including safety guardrails, code generation for production systems, and medical or legal question answering, where even rare failures can have serious consequences. For instance, GPT-5.2-Thinking achieves a 99.9441% no-violation rate on internal safety benchmarks (OpenAI, 2026), yet the remaining $0.0559\%$ violation rate corresponds to over 500 failures per million queries, underscoring the importance of rigorous reliability measurement in near-perfect regimes.

**Surgical fixes for systematic failures.** Identifying model failures that are consistently triggered by certain input prompts could enable more targeted reliability improvements. Our method directly supports this: the learned proposal distribution $Q$ highlights the failure-inducing parameter values, which practitioners can use as targeted training data to surgically correct systematic failures via fine-tuning on problematic inputs or adding safeguards rather than retraining broadly. The ultimate goal is to leverage these systematic failures to push reliability further, adding nines toward the levels demanded by safe deployment in high-stakes tasks.

**Conclusion.** We present an inference-efficient framework for estimating the true error probability of highly accurate LLMs, demonstrated on parameterized GSM problems. LLM failures are strongly concentrated on a small subset of inputs, as evidenced by high TV distances from the uniform distribution across all models and templates we study. Leveraging this, we adopt the CEM to learn a failure-prone sampling distribution, achieving up to $156.22\times$ inference

savings over uniform sampling for equivalent confidence bounds. We call on the community to treat reliability as a first-class evaluation criterion alongside capability; our framework provides a practical foundation for rigorous reliability measurement in high-stakes LLM deployments.

## References


Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Anthropic. Claude opus 4.6 system card, 2026. URL https://anthropic.com/claude-opus-4-6-system-card.

Arief, M., Huang, Z., Koushik Senthil Kumar, G., Bai, Y., He, S., Ding, W., Lam, H., and Zhao, D. Deep probabilistic accelerated evaluation: A robust certifiable rare-event simulation methodology for black-box safety-critical systems. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 595–603. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/arief21a.html.

Bai, Y., Huang, Z., Lam, H., and Zhao, D. Rare-event simulation for neural network and random forest predictors. *ACM Trans. Model. Comput. Simul.*, 32(3), July 2022. ISSN 1049-3301. doi: 10.1145/3519385. URL https://doi.org/10.1145/3519385.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649:1139–1146, 2026. doi: 10.1038/s41586-025-09962-4. URL https://arxiv.org/abs/2501.14249.

Clopper, C. J. and Pearson, E. S. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934. ISSN 00063444, 14643510. URL http://www.jstor.org/stable/2331986.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Efron, B. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1 – 26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Foundation, A. P. Arc-agi-3: A new challenge for frontier agentic intelligence, 2026. URL https://arxiv.org/abs/2603.24621.

Google DeepMind. Gemini 3 pro model card, 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hesterberg, T. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2): 185–194, 1995.

Hofmann, V., Heineman, D., Magnusson, I., Lo, K., Dodge, J., Sap, M., Koh, P. W., Wang, C., Hajishirzi, H., and Smith, N. A. Fluid language model benchmarking. *arXiv preprint arXiv:2509.11106*, 2025.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. R. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Maia Polo, F., Weber, L., Choshen, L., Sun, Y., Xu, G., and Yurochkin, M. tinyBenchmarks: evaluating LLMs with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34303–34326. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/maia-polo24a.html.

Maslej, N., Fattorini, L., Perrault, R., Gil, Y., Parli, V., Kariuki, N., Capstick, E., Reuel, A., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Niebles, J. C., Shoham, Y., Wald, R., Walsh, T., Hamrah, A., Santarlasci, L., Lotufo, J. B., Rome, A., Shi, A., and Oak, S. Artificial intelligence index report 2025, 2025. URL https://arxiv.org/abs/2504.07139.

Merrill, M. A., Shaw, A. G., Carlini, N., Li, B., Raj, H., Bercovich, I., Shi, L., Shin, J. Y., et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

METR. Task-completion time horizons of frontier ai models. https://metr.org/time-horizons/, 03 2026.

Mirzadeh, S. I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., and Farajtabar, M. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

OpenAI. Openai o1 system card, 2024. URL https://openai.com/index/openai-o1-system-card/.

OpenAI. Introducing gpt-5.2, 2025. URL https://openai.com/index/introducing-gpt-5-2/.

OpenAI. Gpt-5.4 thinking system card. https://deploymentsafety.openai.com/gpt-5-4-thinking/gpt-5-4-thinking.pdf, 2026.

Perlitz, Y., Bandel, E., Gera, A., Arviv, O., Ein-Dor, L., Shnarch, E., Slonim, N., Shmueli-Scheuer, M., and Choshen, L. Efficient benchmarking (of language models). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2519–2536, 2024.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Rubinstein, R. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

Rubinstein, R. Y. and Kroese, D. P. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Truong, S. T., Tu, Y., Liang, P., Li, B., and Koyejo, S. Reliable and efficient amortized model-based evaluation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=HDbWrsgkB9.

Vendrow, J., Vendrow, E., Beery, S., and Madry, A. Large language model benchmarks do not test reliability. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL https://openreview.net/forum?id=XSeN6xZtZ9.

Vivek, R., Ethayarajh, K., Yang, D., and Kiela, D. Anchor points: Benchmarking models with much fewer examples. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1576–1601, 2024.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Linzen, T., Chrupała, G., and Alishahi, A. (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wu, S., Nair, Y., and Candès, E. J. Efficient evaluation of llm performance with statistical guarantees, 2026. URL https://arxiv.org/abs/2601.20251.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., Wei, C., Yu, B., Yuan, R., Sun, R., Yin, M., Zheng, B., Yang, Z., Liu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR*, 2024.

Yue, X., Zheng, T., Ni, Y., Wang, Y., Zhang, K., Tong, S., Sun, Y., Yu, B., Zhang, G., Sun, H., Su, Y., Chen, W., and Neubig, G. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of ACL*, 2025.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

# A. LLM Inference Setup

We describe the implementation details for LLM inference. We use the same system prompt for all models:

```
Please reason step by step, and put your
final answer within \\boxed{}.
```

and parse the final answer within the box. We follow the recommended decoding hyperparameters specified in each model card: temperature of 0.7 and top-p of 0.8 for Qwen2.5-Math-7B-Instruct, 1.0 and 1.0 for gpt-oss-20b-low, and 1.0 and 0.95 for Gemini 2.5 Flash Lite. For all models, we use a maximum of 32768 new tokens. If not specified, we use the default values.

# B. Algorithm for calculating confidence bounds

**Algorithm 2** COMPUTEESTIMATE: Importance Sampling with Confidence Bounds

**Input:** Failure indicator $f$, distributions $P$ and $Q$, sample size $N$, confidence level $\alpha$
**Output:** Estimate $\hat{\mu}$, confidence interval $[\hat{\mu}_{\text{low}}, \hat{\mu}_{\text{high}}]$
Sample $Z_1, \ldots, Z_N \sim Q$
Compute importance-weighted estimate: $\hat{\mu} \leftarrow \frac{1}{N}\sum_{i=1}^{N} f(Z_i)\frac{P(Z_i)}{Q(Z_i)}$
Compute sample variance: $\hat{\sigma}^2 \leftarrow \frac{1}{N-1}\sum_{i=1}^{N}\left(f(Z_i)\frac{P(Z_i)}{Q(Z_i)} - \hat{\mu}\right)^2$
Set $Z_{\alpha/2} \leftarrow$ critical value from standard normal for confidence level $\alpha$
Compute CI: $[\hat{\mu}_{\text{low}}, \hat{\mu}_{\text{high}}] \leftarrow \left[\hat{\mu} - Z_{\alpha/2}\frac{\hat{\sigma}}{\sqrt{N}}, \hat{\mu} + Z_{\alpha/2}\frac{\hat{\sigma}}{\sqrt{N}}\right]$
**Return** $\hat{\mu}$, $[\hat{\mu}_{\text{low}}, \hat{\mu}_{\text{high}}]$

Algorithm 2 details the COMPUTEESTIMATE subroutine called in Algorithm 1 (Section 5).

# C. Model-Template Pairs with Too Few Failures

The following model-template pairs have zero observed failures within our 100K evaluation budget to support reliable estimation (failure counts from the full 100K cache):

- gpt-oss-20b-low: Templates 1, 3, 5, 6 ($K = 8$)
- Gemini 2.5 Flash Lite: Templates 1, 3, 5, 7, 8 ($K = 8$)

We expect systematic failure patterns may emerge with a larger evaluation budget, and leave this for future work.

# D. Stronger Failure Concentration with Larger $K$

To supplement Table 1, we compare TV distance for each parameter dimension with increasing $K$. The distance is measured between the normalized failure distribution per parameter dimension and the uniform baseline; we report the three parameter dimensions with the largest distances. We observe that increasing $K$ leads to larger distances across models and templates. This implies that with larger $K$, the failure distribution has shorter tails (fewer inconsistent failures) and probability mass concentrates more on the dominant failure-inducing inputs. This empirically supports the efficiency gains that grow with $K$ in Table 2.

| **Qwen2.5-Math-7B-Instruct** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $K = 4$ | | | $K = 8$ | | | $K = 16$ | | |
| Template | Top1 | Top2 | Top3 | Top1 | Top2 | Top3 | Top1 | Top2 | Top3 |
| 0 | 0.822 | 0.705 | 0.400 | 0.884 | 0.749 | 0.433 | 0.899 | 0.749 | 0.509 |
| 1 | 0.629 | 0.485 | 0.477 | 0.857 | 0.713 | 0.710 | 0.914 | 0.833 | 0.750 |
| 2 | 0.424 | 0.213 | 0.069 | 0.495 | 0.234 | 0.022 | 0.538 | 0.249 | 0.019 |
| 3 | 0.230 | 0.218 | 0.184 | 0.364 | 0.296 | 0.219 | 0.472 | 0.362 | 0.310 |
| 4 | 0.324 | 0.173 | 0.126 | 0.347 | 0.229 | 0.154 | 0.339 | 0.264 | 0.185 |
| 5 | 0.752 | 0.446 | 0.266 | 0.799 | 0.599 | 0.559 | 0.939 | 0.866 | 0.799 |
| 6 | 0.835 | 0.623 | 0.333 | 0.862 | 0.708 | 0.269 | 0.878 | 0.752 | 0.246 |
| 7 | 0.682 | 0.588 | 0.380 | 0.773 | 0.696 | 0.528 | 0.797 | 0.748 | 0.556 |
| 8 | 0.850 | 0.750 | 0.524 | — | — | — | — | — | — |
| **gpt-oss-20b-low** | | | | | | | | | |
| | $K = 8$ | | | $K = 16$ | | | $K = 24$ | | |
| Template | Top1 | Top2 | Top3 | Top1 | Top2 | Top3 | Top1 | Top2 | Top3 |
| 0 | 0.708 | 0.621 | 0.394 | 0.900 | 0.800 | 0.695 | 0.933 | 0.850 | 0.729 |
| 2 | 0.550 | 0.390 | 0.041 | 0.648 | 0.459 | 0.053 | 0.699 | 0.541 | 0.052 |
| 4 | 0.419 | 0.292 | 0.223 | 0.552 | 0.396 | 0.335 | 0.631 | 0.508 | 0.385 |
| 7 | 0.694 | 0.412 | 0.390 | 0.836 | 0.555 | 0.468 | 0.895 | 0.633 | 0.537 |
| 8 | 0.680 | 0.541 | 0.300 | 0.732 | 0.590 | 0.413 | 0.760 | 0.599 | 0.487 |
| **Gemini 2.5 Flash Lite** | | | | | | | | | |
| | $K = 8$ | | | $K = 16$ | | | $K = 24$ | | |
| Template | Top1 | Top2 | Top3 | Top1 | Top2 | Top3 | Top1 | Top2 | Top3 |
| 0 | 0.739 | 0.664 | 0.118 | 0.829 | 0.792 | 0.237 | 0.867 | 0.850 | 0.267 |
| 2 | 0.696 | 0.691 | 0.032 | 0.848 | 0.750 | 0.062 | 0.907 | 0.750 | 0.080 |
| 4 | 0.201 | 0.093 | 0.086 | 0.224 | 0.110 | 0.107 | 0.208 | 0.133 | 0.102 |
| 6 | 0.747 | 0.701 | 0.118 | 0.969 | 0.875 | 0.143 | 0.983 | 0.889 | 0.189 |

*Table 3.* Top-3 largest TV distances between the failure distribution and the uniform baseline, reported across models, templates, and majority-vote ensemble sizes $K$.

## E. Failure Histograms for All Parameters

Figure 6 shows parameter-wise failure histograms for all parameters, supplementing Figure 2 in Section 4. Parameters such as `Name`, `Drinks`, and `Locations` are roughly uniformly distributed among failures. In contrast, parameters such as `n` (first column), `frac2` (second column), and `x` (third column) show strong concentration of failures at a few specific values.

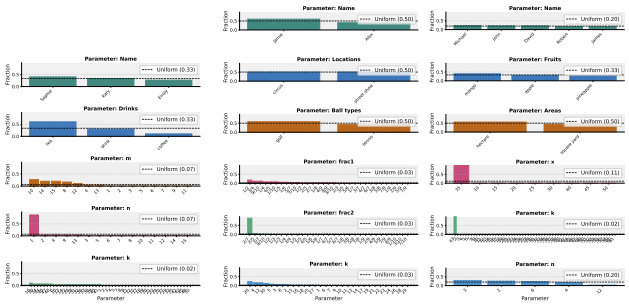

*Figure 6.* Parameter-wise failure histograms reveal strong concentration of errors.

## F. Table of Failure Probability Evaluation

This section provides the full table of estimated failure probabilities across models, templates, and $K$ ($\lambda = 0.1$), corresponding to Figure 5.

| ID | Estimate ($\hat{\mu}_Q$) ± Bounds | Estimate ($\hat{\mu}_Q$) ± Bounds | Estimate ($\hat{\mu}_Q$) ± Bounds |
|---|---|---|---|
| **Qwen2.5-Math-7B-Instruct** | | | |
| | $K$=4 | $K$=8 | $K$=16 |
| 0 | 1.50e-03 ± 4.90e-05 | 8.98e-04 ± 3.03e-05 | 6.14e-04 ± 1.50e-05 |
| 1 | 1.79e-03 ± 9.20e-05 | 4.34e-04 ± 3.36e-05 | 2.16e-04 ± 1.79e-05 |
| 2 | 1.25e-01 ± 6.97e-04 | 9.11e-02 ± 5.73e-04 | 7.21e-02 ± 4.98e-04 |
| 3 | 4.36e-02 ± 4.87e-04 | 1.80e-02 ± 2.95e-04 | 9.29e-03 ± 1.94e-04 |
| 4 | 7.79e-02 ± 6.22e-04 | 5.62e-02 ± 5.21e-04 | 4.47e-02 ± 4.61e-04 |
| 5 | 2.95e-03 ± 9.20e-05 | 7.41e-04 ± 3.29e-05 | 2.03e-04 ± 1.62e-05 |
| 6 | 4.40e-03 ± 5.02e-05 | 3.50e-03 ± 4.16e-05 | 3.08e-03 ± 3.48e-05 |
| 7 | 2.28e-03 ± 6.37e-05 | 1.46e-03 ± 3.50e-05 | 1.14e-03 ± 2.13e-05 |
| 8 | 1.96e-04 ± 2.36e-05 | 0.0, 5.30e-06 | 0.0, 5.30e-06 |
| **gpt-oss-20b-low** | | | |
| ID | $K$=8 | $K$=16 | $K$=24 |
| 0 | 5.22e-04 ± 3.40e-05 | 2.54e-04 ± 1.90e-05 | 1.70e-04 ± 1.19e-05 |
| 2 | 4.87e-02 ± 3.55e-04 | 2.55e-02 ± 2.42e-04 | 1.89e-02 ± 1.98e-04 |
| 4 | 1.09e-02 ± 2.24e-04 | 3.71e-03 ± 1.15e-04 | 2.29e-03 ± 7.96e-05 |
| 7 | 1.42e-03 ± 6.18e-05 | 5.17e-04 ± 2.70e-05 | 3.42e-04 ± 1.92e-05 |
| 8 | 3.49e-03 ± 7.65e-05 | 1.61e-03 ± 4.93e-05 | 1.12e-03 ± 3.86e-05 |
| **Gemini 2.5 Flash Lite** | | | |
| ID | $K$=4 | $K$=8 | $K$=16 |
| 0 | 2.74e-03 ± 1.21e-04 | 7.18e-04 ± 3.63e-05 | 3.28e-04 ± 1.65e-05 |
| 2 | 9.53e-03 ± 1.66e-04 | 4.73e-03 ± 7.29e-05 | 3.35e-03 ± 4.06e-05 |
| 4 | 5.54e-02 ± 5.69e-04 | 3.37e-02 ± 4.42e-04 | 2.59e-02 ± 3.87e-04 |
| 6 | 8.70e-03 ± 2.22e-04 | 1.27e-03 ± 6.26e-05 | 9.53e-04 ± 1.37e-05 |

*Table 4.* Estimated failure probability across templates, models, and $K$ at $\lambda = 0.1$. For all missing model-template pairs are those that have zero failures within our evaluation budgets, and they can be estimated via exact binomial bounds of (0, 5.30e-06).

## G. Parameterization of GSM8K Templates

Motivated by GSM-Symbolic (Mirzadeh et al., 2025), we construct parameterized GSM8K templates with extended variables and corrected grammatical flaws. Each template consists of a prompt with placeholders, a variable domain, and a formula determining the answer.

**Template 0**

- Prompt:
  $\{Z_{\text{Name}}\}$ saw a $\{Z_x\}$-foot $\{Z_{\text{Fish}}\}$ with $\{Z_y\}$ remoras, each $\{Z_z\}$-inches long, attached to it. What percentage of the $\{Z_{\text{Fish}}\}$'s body length is the combined length of the remoras?
- Variables:
  $Z_{\text{Name}} \in$ {Benny, Alex, Casey, Jordan, Taylor, Morgan, Riley, Sam, Jamie, Avery}
  $Z_{\text{Fish}} \in$ {dolphin, whale, shark}
  $Z_k \in \{2, \ldots, 21\}$
  $Z_y \in \{1, \ldots, 30\}$
  $Z_{\text{ratio}} \in \{4, 5, 10, 20, 25\}$
- Formula:
  $Z_y = Z_y \times 12$
  $Z_x = Z_{\text{ratio}} \times Z_k \times Z_y/12$
  answer $= 100/\text{ratio}$

---

**Template 1**

- Prompt:
  A $\{Z_{\text{weather}}\}$ rolls in from the ocean to cover $\{Z_{\text{city}}\}$. It takes $\{Z_t\}$ minutes to cover every $\{Z_d\}$ miles of the city. If the city is $\{Z_y\}$ miles across from the oceanfront to the opposite inland edge, how many minutes will it take for the $\{Z_{\text{weather}}\}$ to cover the whole city?
- Variables:
  $Z_{\text{city}} \in$ {Harborview, Lakewood, Riverton}
  $Z_{\text{weather}} \in$ {fog bank, marine layer, coastal mist}
  $Z_t \in \{2, \ldots, 71\}$
  $Z_d \in \{2, \ldots, 13\}$
  $Z_s \in \{1, \ldots, 12\}$
- Formula:
  $Z_y = Z_d \times Z_s$
  answer $= Z_s \times Z_t$

---

**Template 2**

- Prompt:
  $\{Z_{\text{name}}\}$ is playing a $\{Z_{\text{game}}\}$ during $\{Z_{\text{setting}}\}$ for a $\{Z_{\text{prize}}\}$. $\{Z_{\text{name}}\}$ is rolling a $\{Z_s\}$-sided die. By how many percentage points is it more likely that they roll a number greater than $\{Z_{\text{target}}\}$ than that they roll two $\{Z_z\}$ numbers in a row?
- Variables:
  $Z_{\text{name}} \in$ {Jerry, Alex, Casey, Jordan, Taylor, Morgan}
  $Z_{\text{setting}} \in$ {game night, casino night, board game meetup}
  $Z_{\text{game}} \in$ {dice game}
  $Z_{\text{prize}} \in$ {sticker, token, pin}
  $Z_{\text{target}} \in \{2, 4, 6, \ldots, 200\}$
  $Z_d \in \{2, 4, 5, 10, 20, 25, 50, 100\}$
  $Z_z \in$ {even, odd}

- Formula:
  $Z_s = Z_{\text{target}} \times Z_d$
  answer $= \frac{Z_d - 1}{Z_d} \cdot 100 - 25$

---

**Template 3**

- Prompt:
  A $\{Z_{\text{group}}\}$ of $\{Z_n\}$ students has various hobbies. $\{Z_{n1}\}$ like to $\{Z_{\text{hobby1}}\}$, $\{Z_{n2}\}$ like to play $\{Z_{\text{sport}}\}$, and the rest like to either $\{Z_{\text{hobby3}}\}$ or $\{Z_{\text{hobby4}}\}$. How many like to $\{Z_{\text{hobby3}}\}$ if the number that like to $\{Z_{\text{hobby4}}\}$ is $\{Z_{\text{multi}}\}$ times the number that prefer playing $\{Z_{\text{sport}}\}$?

- Variables:
  $Z_{\text{sport}} \in \{\text{basketball, soccer}\}$
  $Z_{\text{hobby1}} \in \{\text{read, paint}\}$
  $Z_{\text{hobby3}} \in \{\text{play music, garden}\}$
  $Z_{\text{hobby4}} \in \{\text{write}\}$
  $Z_{n1} \in \{2, \ldots, 11\}$
  $Z_{n2} \in \{2, \ldots, 11\}$
  $Z_{\text{extra}} \in \{1, \ldots, 10\}$
  $Z_{\text{multi}} \in \{2, \ldots, 11\}$

- Formula:
  $Z_n = Z_{n1} + (Z_{\text{multi}} + 1) \times Z_{n2} + Z_{\text{extra}}$
  answer $= Z_{\text{extra}}$

---

**Template 4**

- Prompt:
  $\{Z_{\text{name}}\}$ is popping popcorn for a snack. As the $\{Z_{\text{p}}\}$ of kernels heats up, the kernels start popping faster. $\{Z_n\}$ pop in the first $\{Z_s\}$ seconds of cooking, then $\{Z_{k2}\}$ times that amount in the next $\{Z_s\}$ seconds. The kernels increase to $\{Z_{k3}\}$ times the initial popping rate in the next $\{Z_s\}$ seconds, but in the final $\{Z_s\}$ seconds, the popping slows down to $\{Z_{\text{r1}}\}$ of the rate as the past $\{Z_s\}$ seconds. After $\{Z_{\text{name}}\}$ takes the $\{Z_{\text{p}}\}$ off the heat, a $\{Z_{\text{r2}}\}$ of the number of kernels that popped in the final $\{Z_s\}$ seconds of cooking also pop from the residual heat. How many pieces of popcorn does $\{Z_{\text{name}}\}$ have to eat?

- Variables:
  $Z_{\text{name}} \in \{\text{Garrett, Alex, Casey}\}$
  $Z_{\text{p}} \in \{\text{pan, pot, skillet}\}$
  $Z_{\text{r1}} \in \{\text{half, quarter}\}$
  $Z_{\text{r2}} \in \{\text{half, quarter}\}$
  $Z_n \in \{16, 32, 48, \ldots, 160\}$
  $Z_s \in \{2, 3, 4\}$
  $Z_{k2} \in \{2, \ldots, 11\}$
  $Z_{k3} \in \{12, \ldots, 21\}$

- Formula:
  second $= Z_{k2} \cdot Z_n$
  third $= Z_{k3} \cdot Z_n$
  fourth $=$ third$/Z_{\text{r1num}}$
  residual $=$ third$/(Z_{\text{r1num}} \cdot Z_{\text{r2num}})$
  answer $= Z_n +$ second $+$ third $+$ fourth $+$ residual

---

**Template 5**

- Prompt:
  $\{Z_{\text{name}}\}$ makes $\{Z_{\text{drink}}\}$ using teaspoons of sugar and cups of water in the ratio of $\{Z_m\}$:$\{Z_n\}$. If she used a total of $\{Z_x\}$ teaspoons of sugar and cups of water, calculate the number of teaspoonfuls of sugar she used.

- Variables:
  $Z_{\text{name}} \in \{\text{Katy, Sophie, Emily}\}$
  $Z_{\text{drink}} \in \{\text{coffee, tea, drink}\}$
  $Z_m \in \{1, \ldots, 15\}$
  $Z_n \in \{1, \ldots, 15\}$
  $Z_k \in \{1, \ldots, 50\}$

- Formula:
  $Z_{\text{total}} = Z_m + Z_n$
  $Z_x = Z_{\text{total}} \cdot Z_k$
  answer $= \frac{Z_m \cdot Z_x}{Z_{\text{total}}}$

---

**Template 6**

- Prompt:
  $\{Z_{\text{name}}\}$ has $\{Z_x\}$ $\{Z_{\text{area}}\}$s of $\{Z_{\text{fruit}}\}$s field. There are $\{Z_k\}$ $\{Z_{\text{fruit}}\}$s per $\{Z_{\text{area}}\}$. $\{Z_{\text{name}}\}$ can harvest his $\{Z_{\text{fruit}}\}$s every $\{Z_n\}$ months. How many $\{Z_{\text{fruit}}\}$s can $\{Z_{\text{name}}\}$ harvest within a year?

- Variables:
  $Z_{\text{name}} \in \{\text{John, Michael, David, James, Robert}\}$
  $Z_{\text{fruit}} \in \{\text{pineapple, apple, mango}\}$
  $Z_{\text{area}} \in \{\text{hectare, square yard}\}$
  $Z_x \in \{10, 15, 20, \ldots, 50\}$
  $Z_k \in \{2, \ldots, 61\}$
  $Z_n \in \{2, 3, 4, 6, 12\}$

- Formula:
  $Z_{\text{cycles}} = 12/Z_n$
  answer $= Z_x \cdot Z_k \cdot Z_{\text{cycles}}$

---

**Template 7**

- Prompt:
  $\{Z_{\text{name}}\}$ is juggling at a $\{Z_{\text{location}}\}$. $\{Z_{\text{name}}\}$ can juggle $\{Z_n\}$ balls. $\{Z_{\text{frac1}}\}$ of the balls are $\{Z_{\text{ball_type}}\}$ balls, and $\{Z_{\text{frac2}}\}$ of the $\{Z_{\text{ball_type}}\}$ balls are $\{Z_{\text{color}}\}$. How many $\{Z_{\text{color}}\}$ $\{Z_{\text{ball_type}}\}$ balls are there?

- Variables:
  $Z_{\text{name}} \in \{\text{Jamie, Alex}\}$
  $Z_{\text{location}} \in \{\text{circus, street show}\}$
  $Z_{\text{ball_type}} \in \{\text{golf, tennis}\}$
  $Z_{\text{color}} \in \{\text{blue}\}$
  $Z_{\text{frac1}}, Z_{\text{frac2}} \in \{1/2, 1/3, 2/3, 1/4, 3/4, 1/5, 2/5, 3/5, 4/5, 1/6, 5/6, 1/7, 2/7, 3/7, 4/7, 5/7, 6/7, 1/8, 3/8, 5/8, 7/8, 1/9, 2/9, 4/9, 5/9, 7/9, 8/9, 1/10, 3/10, 7/10, 9/10\}$
  $Z_k \in \{1, \ldots, 30\}$

- Formula:
  $Z_n = Z_k \times \text{denom}(Z_{\text{frac1}}) \times \text{denom}(Z_{\text{frac2}})$
  answer $= Z_n \times (\text{num}(Z_{\text{frac1}}) \times \text{num}(Z_{\text{frac2}}))$
  $/(\text{denom}(Z_{\text{frac1}}) \times \text{denom}(Z_{\text{frac2}}))$

---

**Template 8**

- Prompt:
  $\{Z_{\text{name}}\}$ places $\{Z_{\text{obj}}\}$s on some $\{Z_{\text{surface}}\}$s. Each $\{Z_{\text{surface}}\}$ can hold $\{Z_x\}$ $\{Z_{\text{obj}}\}$s. If he has $\{Z_n\}$ $\{Z_{\text{obj}}\}$s and $\{Z_k\}$ $\{Z_{\text{surface}}\}$s, how many $\{Z_{\text{obj}}\}$s won't he be able to place on the $\{Z_{\text{surface}}\}$?

- Variables:
  $Z_{\text{name}} \in \{\text{Jaime, John, Mike}\}$
  $Z_{\text{obj}} \in \{\text{egg, olive, almond}\}$
  $Z_{\text{surface}} \in \{\text{tray, plate, table}\}$
  $Z_x \in \{2, \ldots, 21\}$
  $Z_k \in \{2, \ldots, 21\}$
  $Z_{\text{leftover}} \in \{1, \ldots, 10\}$

- Formula:
  $Z_n = Z_k \cdot Z_x + Z_{\text{leftover}}$
  answer $= Z_{\text{leftover}}$

## H. CI Width vs. Number of Inferences

This section presents CI width versus number of inferences for all three models across varying templates and majority-vote sizes ($K$). Each figure corresponds to a specific model and $K$, with each column representing a different template. Within each subplot, the upper panel shows CI width vs. number of inferences, and the lower panel shows sorted CIs over 100 independent sample sets at the evaluation size of the bottom-right Pareto-frontier dot. Points with coverage in the range $99 \pm 1\%$ are shown as solid markers, while points with coverage $\geq 95\%$ are shown as semi-transparent markers. Efficiency gains are computed only for results achieving $99 \pm 1\%$ coverage.

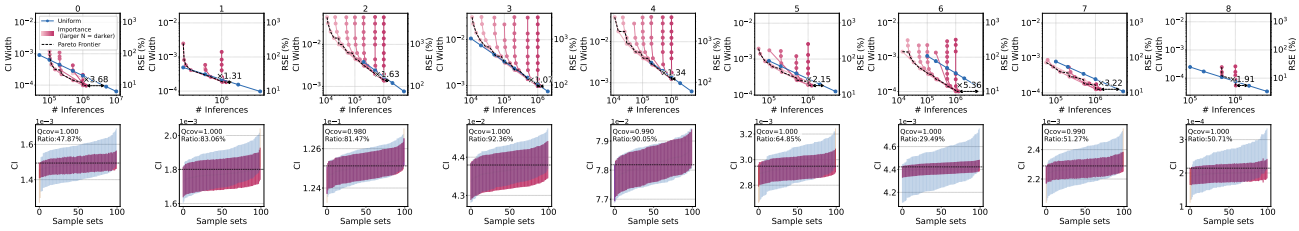


*Figure 7.* Qwen2.5-Math-7B-Instruct, $K = 4$, $\lambda = 0.1$

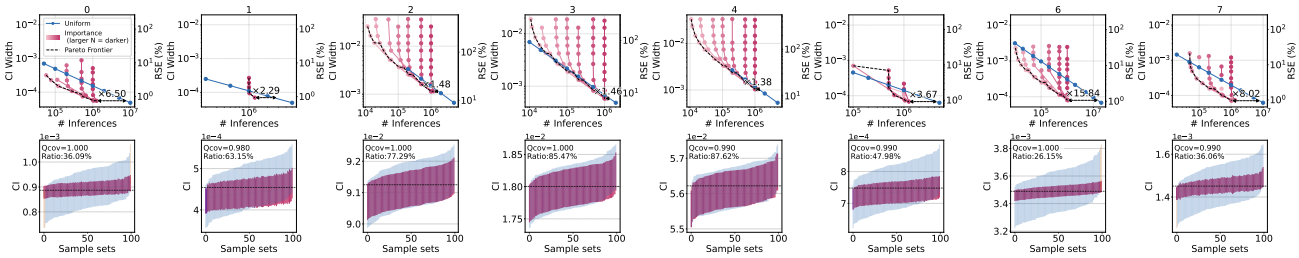


*Figure 8.* Qwen2.5-Math-7B-Instruct, $K = 8$, $\lambda = 0.1$

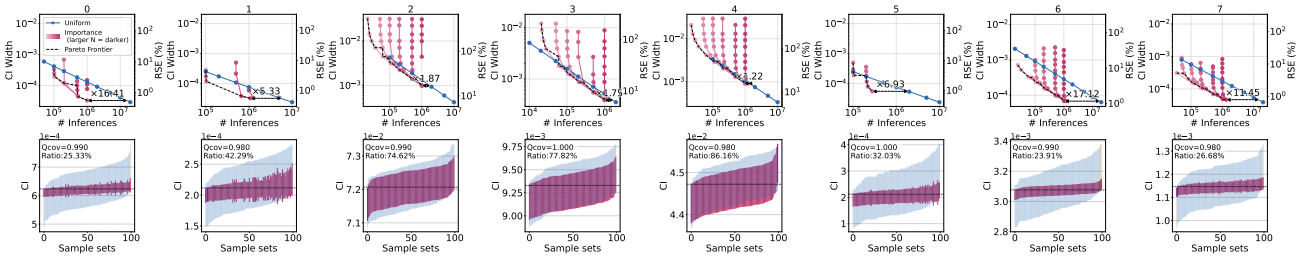


*Figure 9.* Qwen2.5-Math-7B-Instruct, $K = 16$, $\lambda = 0.1$

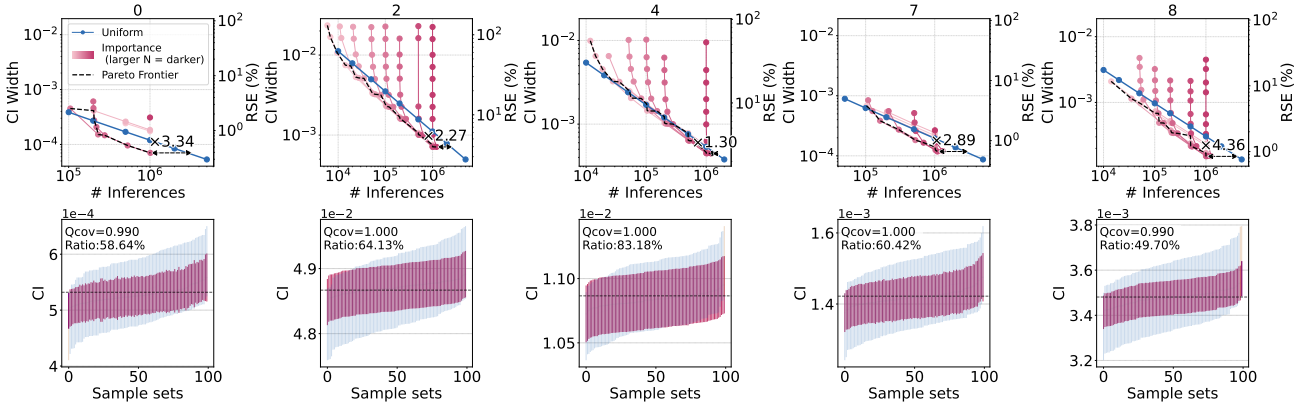


*Figure 10.* gpt-oss-20b-low, $K = 8$, $\lambda = 0.1$

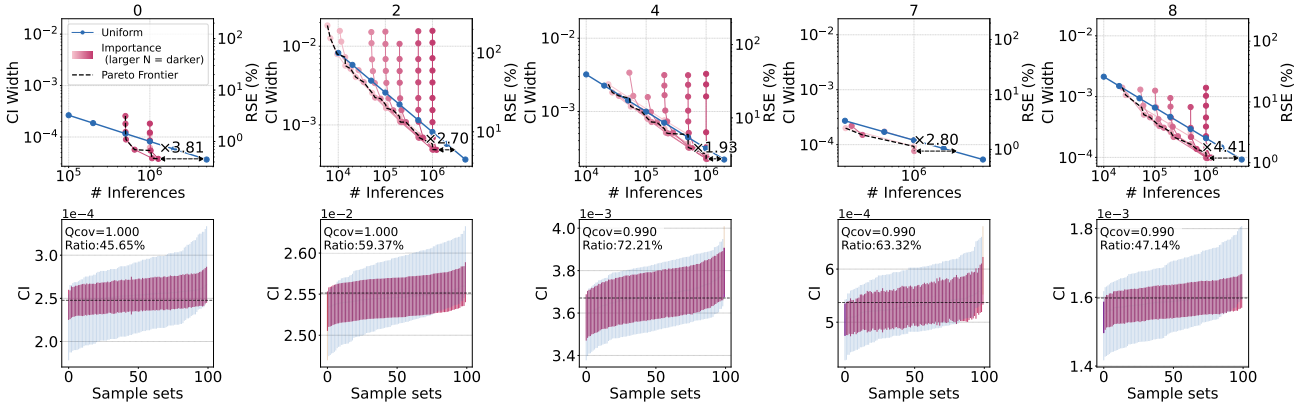


*Figure 11.* gpt-oss-20b-low, $K = 16$, $\lambda = 0.1$

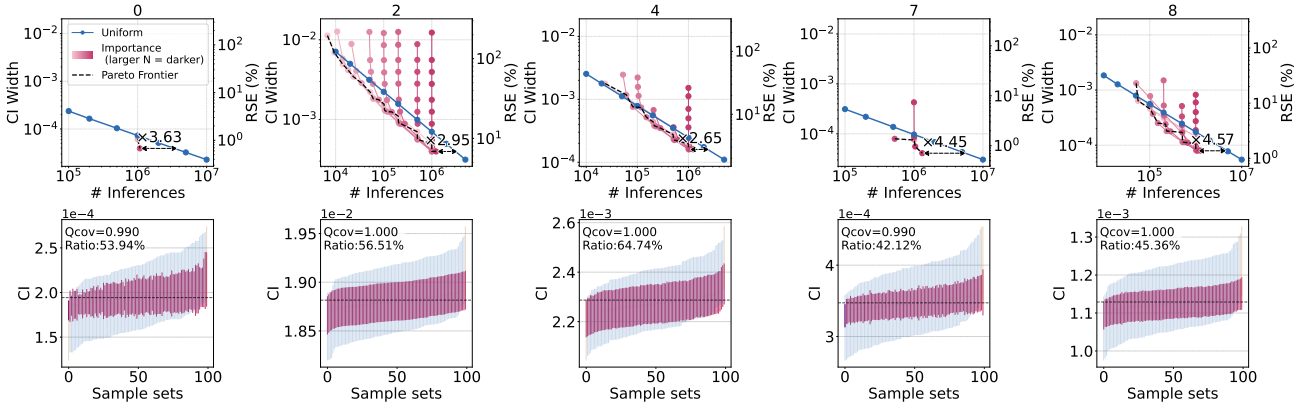


*Figure 12.* gpt-oss-20b-low, $K = 24$, $\lambda = 0.1$

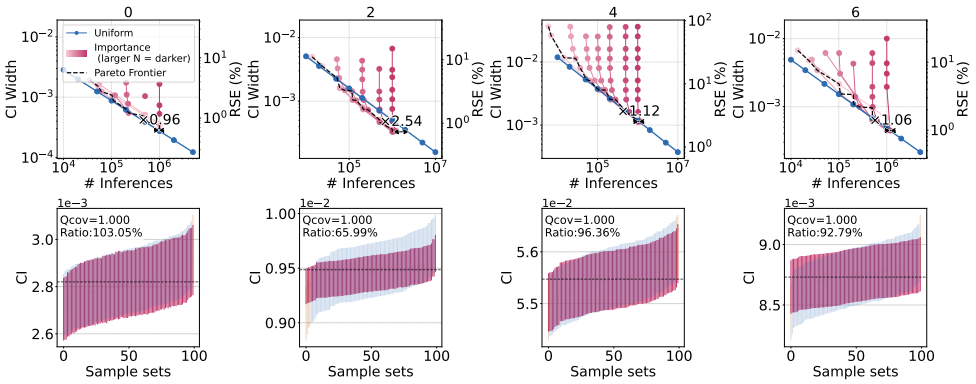


*Figure 13.* Gemini 2.5 Flash Lite, $K = 4$, $\lambda = 0.1$

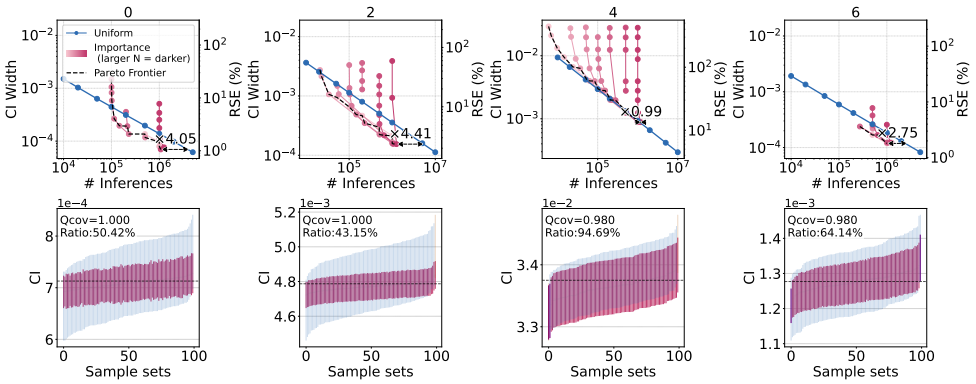


*Figure 14.* Gemini 2.5 Flash Lite, $K = 8$, $\lambda = 0.1$

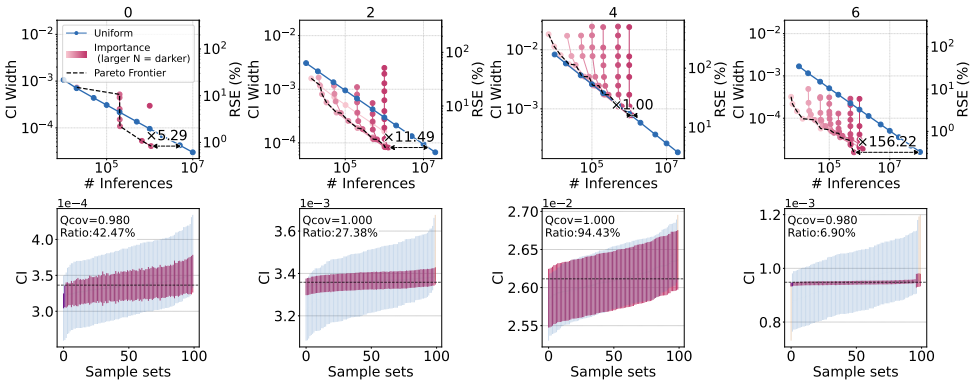


*Figure 15.* Gemini 2.5 Flash Lite, $K = 16$, $\lambda = 0.1$

# I. Additional examples for repetitive failures by LLMs

To supplement the qualitative examples in Figure 3 (Section 4), we provide models' responses that consistently produce erroneous outputs for certain prompts in Figures 19 and 24. The examples illustrate that Qwen2.5-Math-7B-Instruct exhibits repetitive errors for certain ratios of sugar teaspoons to water cups, frequently invoking a spurious "standard US measurement" heuristic to convert cups into tablespoons or teaspoons. For gpt-oss-20b-low on Template 7, the model shows striking failures on the basic arithmetic $84 \times (2/7)$, while being generally robust to other fraction values. We additionally provide dominant failure cases for all model-template pairs and observe various error types. These include simple arithmetic errors in addition/subtraction (18, 26), multiplication/division (16, 20, 17, 23, 25, 27), transcription (21), unnecessary approximation (22), and problem misinterpretation (24). Failure-inducing parameter values and the onset of erroneous reasoning in the generation traces are highlighted in red.

**Prompt (Template 0)**
A {dolphin} is {1276} feet long. {11} {remoras} attach to it, each measuring {348} inches in length. What percentage of the {dolphin}'s body length is the combined length of the {remoras}?

---

**First Output (Qwen2.5-Math-7B-Instruct) (Incorrect)**
To determine what percentage of the dolphin's body length is the combined length of the remoras, we need to follow these steps: ... Now, we perform the division:

$$\frac{319}{1276} \approx 0.2475$$

Then, we multiply by 100 to convert the decimal to a percentage:

$$0.2475 \times 100 = 24.75$$

Therefore, the combined length of the remoras is $\boxed{24.75}$ percent of the dolphin's body length.

---

**Second Output (Correct)**
To determine what percentage of the dolphin's body length is the combined length of the remoras, we need to follow these steps: ... Perform the division and multiplication:

$$\frac{319}{1276} \approx 0.25$$

$$0.25 \times 100 = 25$$

Therefore, the combined length of the remoras is $\boxed{25}$ percent of the dolphin's body length.

---

**Third Output (Correct)**
To determine what percentage of the dolphin's body length is the combined length of the remoras, we need to follow these steps: ...

$$\left(\frac{319}{1276}\right) \times 100 \approx 25 \text{ percent}$$

Thus, the combined length of the remoras is $\boxed{25}$ percent of the dolphin's body length.

---

**Fourth Output (Incorrect)**
To determine what percentage of the dolphin's body length is the combined length of the remoras, we need to follow these steps: ... Now, we perform the division and multiplication:

$$\frac{319}{1276} \approx 0.247$$

$$0.247 \times 100 = 24.7$$

Therefore, the combined length of the remoras is approximately 24.7% of the dolphin's body length. The final answer is:

$$\boxed{24.7}$$

---

**Fifth Output (Incorrect)**
To determine what percentage of the dolphin's body length is the combined length of the remoras, we need to follow these steps: ...

$$\left(\frac{319 \text{ feet}}{1276 \text{ feet}}\right) \times 100 = \frac{31900}{1276} \approx 24.73\%$$

Therefore, the combined length of the remoras is $\boxed{24.73}$% of the dolphin's body length.

---

**Sixth Output (Correct)**
To determine what percentage of the dolphin's body length is the combined length of the remoras, we need to follow these steps: ...

$$\text{Percentage} = \left(\frac{319 \text{ feet}}{1276 \text{ feet}}\right) \times 100 \approx 25\%$$

Therefore, the combined length of the remoras is $\boxed{25}$% of the dolphin's body length.

---

**Seventh Output (Incorrect)**
To determine what percentage of the dolphin's body length is the combined length of the remoras, we need to follow these steps: ...

$$\text{Percentage} = \left(\frac{319 \text{ feet}}{1276 \text{ feet}}\right) \times 100 \approx 24.72\%$$

Therefore, the combined length of the remoras is $\boxed{24.72}$% of the dolphin's body length.
Answer:**25**

*Figure 16.* Seven of eight independently generated outputs from Qwen2.5-Math-7B-Instruct evaluated on Template 0. Error type: wrong division.

**Prompt (Template 1)**
A {coastal mist} rolls in from the ocean to cover {Harborview}. It takes {33} minutes to cover every {4} miles of the city. If the city is {28} miles across from the oceanfront to the opposite inland edge, how many minutes will it take for the {coastal mist} to cover the whole city?

---

**First Output (Qwen2.5-Math-7B-Instruct) (Correct)**
To determine how many minutes it will take for the marine layer to cover the entire city, we start by identifying the rate at which the marine layer rolls in. The marine layer covers 4 miles in 33 minutes. We need to find out how many minutes it will take to cover 28 miles. ...

$$8.25 \times 28 = 8.25 \times (30 - 2) = 8.25 \times 30 - 8.25 \times 2$$

$$8.25 \times 30 = 247.5$$

$$8.25 \times 2 = 16.5$$

$$247.5 - 16.5 = 231$$

Therefore, it will take the marine layer $\boxed{231}$ minutes to cover the whole city.

---

**Second Output (Incorrect)**
To determine how many minutes it will take for the marine layer to cover the entire city, we start by identifying the rate at which the marine layer rolls in. The marine layer covers 4 miles in 33 minutes. We need to find out how many minutes it will take to cover 28 miles. ...

$$\text{Time to cover 28 miles} = 8.25 \text{ minutes per mile} \times 28 \text{ miles} = 229.5 \text{ minutes}$$

Therefore, it will take the marine layer $\boxed{229.5}$ minutes to cover the whole city.

---

**Third Output (Correct)**
To determine how many minutes it will take for the marine layer to cover the entire city, we need to follow these steps: ... Since it takes 33 minutes to cover each 4-mile segment, we multiply the number of segments by the time per segment:

$$\text{Total time} = 7 \text{ segments} \times 33 \text{ minutes/segment} = 231 \text{ minutes}$$

Therefore, it will take $\boxed{231}$ minutes for the marine layer to cover the whole city.

---

**Fourth Output (Incorrect)**
To determine how many minutes it will take for the marine layer to cover the entire city, we start by identifying the rate at which the marine layer rolls in. The marine layer covers 4 miles in 33 minutes. We need to find out how long it will take to cover 28 miles. ...

$$8.25 \text{ minutes per mile} \times 28 \text{ miles} = 229.5 \text{ minutes}$$

Therefore, it will take the marine layer $\boxed{229.5}$ minutes to cover the whole city.

---

**Fifth Output (Incorrect)**
To determine how many minutes it will take for the marine layer to cover the entire city, we start by finding the rate at which the marine layer rolls in. The marine layer covers 4 miles in 33 minutes. We need to find out how many minutes it takes to cover 28 miles. ...

$$8.25 \text{ minutes per mile} \times 28 \text{ miles} = 229.5 \text{ minutes}$$

Therefore, it will take the marine layer $\boxed{229.5}$ minutes to cover the whole city.

---

**Sixth Output (Incorrect)**
To determine how many minutes it will take for the marine layer to cover the entire city, we start by identifying the rate at which the marine layer rolls in. The marine layer covers 4 miles in 33 minutes. We need to find out how many minutes it will take to cover 28 miles. ...

$$8.25 \text{ minutes per mile} \times 28 \text{ miles} = 229.5 \text{ minutes}$$

Therefore, it will take the marine layer $\boxed{229.5}$ minutes to cover the whole city.

---

**Seventh Output (Correct)**
To determine how many minutes it will take for the marine layer to cover the entire city, we start by finding the rate at which the marine layer covers the city. The problem states that it takes 33 minutes to cover every 4 miles. ... Since it takes 33 minutes to cover each 4-mile segment, we multiply the number of segments by the time it takes to cover each segment:

$$\text{Total time} = 7 \text{ segments} \times 33 \text{ minutes/segment} = 231 \text{ minutes}$$

Thus, the total time it will take for the marine layer to cover the whole city is:

$$\boxed{231}$$

Answer:**231**

*Figure 17.* Seven of eight independently generated outputs from Qwen2.5-Math-7B-Instruct evaluated on Template 1. Error type: wrong multiplication.

**Prompt (Template 3)**
A group of {71} students has various hobbies. {4} like to read, {6} like to play basketball, and the rest like to either play music or write. How many like to play music if the number that like to write is {10} times the number that prefer playing basketball?

---

**First Output (Qwen2.5-Math-7B-Instruct) (Incorrect)**
To determine how many students like to play music, we need to follow these steps: ...

$$\text{Total} = 4 + 6 + 60 = 64$$

$$\text{Number of students who like to play music} = 71 - 64 = 7$$

Thus, the number of students who like to play music is $\boxed{7}$.

---

**Second Output (Incorrect)**
To determine how many students like to play music, we start by identifying the number of students who like to write. According to the problem, the number of students who like to write is 10 times the number of students who like to play basketball. ...

$$71 - 4 - 60 = 7$$

Thus, the number of students who like to play music is $\boxed{7}$.

---

**Third Output (Correct)**
To determine how many students like to play music, we start by identifying the number of students who like to write. ... Next, let $m$ represent the number of students who like to play music. Then:

$$4 + 6 + m + 60 = 71$$

$$70 + m = 71$$

$$m = 1$$

Therefore, the number of students who like to play music is $\boxed{1}$.

---

**Fourth Output (Incorrect)**
To determine how many students like to play music, we need to follow these steps: ...

$$\text{Total} = 4 + 6 + 60 = 66$$

$$\text{Number of students who like to play music} = 71 - 66 = 5$$

Thus, the number of students who like to play music is $\boxed{5}$.

---

**Fifth Output (Incorrect)**
To determine how many students like to play music, we start by identifying the number of students who like to write. ...

$$4 + 6 + 60 = 66$$

Since the total number of students is 71, the number of students who like to play music is:

$$71 - 66 = 5$$

Thus, the number of students who like to play music is:

$$\boxed{5}$$

---

**Sixth Output (Incorrect)**
To determine how many students like to play music, we need to follow these steps: ...

$$\text{Total} = 4 + 6 + 60 = 60$$

$$\text{Number of students who like to play music} = 71 - 60 = 11$$

Thus, the number of students who like to play music is $\boxed{11}$.

---

**Seventh Output (Correct)**
To determine how many students like to play music, we start by identifying the number of students who like to write. ...

$$71 - (4 + 6 + 60) = 71 - 70 = 1$$

Thus, the number of students who like to play music is $\boxed{1}$.
Answer:**1**

*Figure 18.* Seven of eight independently generated outputs from Qwen2.5-Math-7B-Instruct evaluated on Template 3. Error type: wrong addition/subtraction.

**Prompt (Template 5)**
{Katy} makes {coffee} using teaspoons of sugar and cups of water in the ratio of {15}:{1}. If she used a total of {736} teaspoons of sugar and cups of water, calculate the number of teaspoonfuls of sugar she used.

---

**First Output (Qwen2.5-Math-7B-Instruct) (Incorrect)**
To determine the number of teaspoonfuls of sugar Katy used, we start by understanding the given ratio of sugar to water, which is 15:1. ... Since 1 cup of water is equivalent to 16 teaspoons (assuming standard US measurements where 1 cup = 16 tablespoons and 1 tablespoon = 3 teaspoons), we convert the cups of water to teaspoons:

$$w \text{ cups} = w \times 16 \text{ teaspoons}$$

... Thus, the number of teaspoonfuls of sugar Katy used is:

$$\boxed{360}$$

---

**Second Output (Incorrect)**
To determine the number of teaspoonfuls of sugar Katy used, we start by defining the ratio of sugar to water, which is given as 15:1. ...
We are also given that the total amount of sugar and water used is 736 teaspoons. Since 1 cup of water is equivalent to 16 teaspoons (assuming standard US measurements), ... Thus, the number of teaspoonfuls of sugar Katy used is:

$$\boxed{360}$$

---

**Third Output (Incorrect)**
To determine the number of teaspoonfuls of sugar Katy used, we start by understanding the given ratio of sugar to water, which is 15:1. ... We are also given that the total amount of sugar and water used is 736 teaspoons. Since 1 cup of water is equivalent to 16 teaspoons (assuming standard US cups), we can express the total amount of water in teaspoons as $16W$. ... Therefore, the number of teaspoonfuls of sugar Katy used is $\boxed{360}$.

---

**Fourth Output (Correct)**
To determine the number of teaspoonfuls of sugar Katy used, we start by understanding the given ratio of sugar to water, which is 15:1. ... First, let's denote the number of teaspoons of sugar by $S$ and the number of cups of water by $W$. According to the ratio, we can write:

$$\frac{S}{W} = 15$$

... Thus, the number of teaspoonfuls of sugar Katy used is:

$$\boxed{690}$$

Answer: **690**

*Figure 19.* First four of eight independently generated outputs from Qwen2.5-Math-7B-Instruct evaluated on Template 5. Error type: unjustified unit conversion.

**Prompt (Template 6)**
{John} has {15} {hectares} of {pineapples} field. There are {47} {pineapples} per {hectare}. {John} can harvest his {pineapples} every {2} months. How many {pineapples} can {John} harvest within a year?

---

**First Output (Qwen2.5-Math-7B-Instruct) (Incorrect)**
To determine how many pineapples John can harvest within a year, we need to follow these steps: ...

$$\text{Total number of pineapples} = 15 \text{ hectares} \times 47 \text{ pineapples/hectare} = 655 \text{ pineapples}$$

Therefore, the total number of pineapples John can harvest within a year is $\boxed{3930}$.

---

**Second Output (Incorrect)**
To determine how many pineapples John can harvest within a year, we need to follow these steps: ...

$$\text{Total number of pineapples} = 15 \text{ hectares} \times 47 \text{ pineapples/hectare} = 695 \text{ pineapples}$$

Therefore, the total number of pineapples John can harvest within a year is $\boxed{4170}$.

---

**Third Output (Incorrect)**
To determine how many pineapples John can harvest within a year, we need to follow these steps: ...

$$15 \text{ hectares} \times 47 \text{ pineapples/hectare} = 655 \text{ pineapples}$$

Therefore, the number of pineapples John can harvest within a year is $\boxed{3930}$.

---

**Fourth Output (Incorrect)**
To determine how many pineapples John can harvest within a year, we need to follow these steps: ...

$$15 \text{ hectares} \times 47 \text{ pineapples/hectare} = 655 \text{ pineapples}$$

Therefore, the total number of pineapples John can harvest within a year is:

$$\boxed{3930}$$

---

**Fifth Output (Incorrect)**
To determine how many pineapples John can harvest within a year, we need to follow these steps: ...

$$\text{Total number of pineapples} = 15 \text{ hectares} \times 47 \text{ pineapples/hectare} = 695 \text{ pineapples}$$

Therefore, the number of pineapples John can harvest within a year is $\boxed{4170}$.

---

**Sixth Output (Incorrect)**
To determine how many pineapples John can harvest within a year, we need to follow these steps: ...

$$\text{Total number of pineapples} = 15 \text{ hectares} \times 47 \text{ pineapples/hectare} = 655 \text{ pineapples}$$

Therefore, the total number of pineapples John can harvest within a year is $\boxed{3930}$.

---

**Seventh Output (Correct)**
To determine how many pineapples John can harvest within a year, we need to follow these steps: ...

$$15 \text{ hectares} \times 47 \text{ pineapples/hectare} = 705 \text{ pineapples}$$

Therefore, the number of pineapples John can harvest within a year is $\boxed{4230}$.
Answer: **4230**

*Figure 20.* Seven of eight independently generated outputs from Qwen2.5-Math-7B-Instruct evaluated on Template 6. Error type: wrong multiplication.

**Prompt (Template 0)**
{Benny} saw a {1920}-foot whale with {12} remoras, each {192} inches long, attached to it. What percentage of the whale's body length is the combined length of the remoras?

---

**First Output (Qwen2.5-Math-7B-Instruct) (Incorrect)**
analysisWe need combined length of remoras: 12 * 192 inches. Convert to feet: 192 inches = 16 feet. So each 16 ft, 12*16 = 192 ft. Whale length 192 ft. Percentage = (192/192)*100 = 100... The whale's body length is given as 192 feet. Thus, the combined remora length equals the whale's length:

$$\frac{192 \text{ ft}}{192 \text{ ft}} = 1 \quad \Longrightarrow \quad 100\%$$

$$\boxed{100\%}$$

---

**Second Output (Correct)**
analysisCompute remoras total length: 12 * 192 inches = 2304 inches. Convert whale length: 1920 ft = 1920*12=23040 inches. Percentage = 2304/23040=0.1 =10%. ... Thus the percentage is:

$$\frac{2304}{23040} \times 100\% = 10\%$$

$$\boxed{10\%}$$

---

**Third Output (Correct)**
analysisNeed remora combined length: 12 * 192 inches = 2304 inches = 192 inches? Wait 192 inches = 16 ft. 12*16=192 ft. Whale length 1920 ft. Percentage = 192/1920=0.10=10%. ... As a percentage,

$$0.10 \times 100\% = 10\%$$

$$\boxed{10\%}$$

---

**Fourth Output (Incorrect)**
analysisConvert remora length to feet: 192 inches /12=16 ft. 12 remoras -¿192 ft. Whale 192 ft. Percentage =192/192*100=100%.
...

$$\boxed{100\%}$$

---

**Fifth Output (Incorrect)**
analysisWe need combined length of remoras: 12 * 192 inches. Convert to feet: 192 inches = 16 feet (since 12 inches =1 ft, 192/12=16). So each remora 16 ft. 12*16=192 ft. Whale length 192 ft. So percentage = (192/192)*100 =100%. Box. ...

$$\text{Combined remora length} = 12 \times 192 \text{ in} = 12 \times 16 \text{ ft} = 192 \text{ ft}$$

Whale length $= 192$ ft.

$$\text{Percentage} = \frac{192}{192} \times 100\% = 100\%$$

$$\boxed{100\%}$$

---

**Sixth Output (Correct)**
analysisCompute total remora length: 12 * 192 inches. Convert to feet: 12*192 = 2304 inches. 12 feet? Wait 2304/12 = 192 feet. Whale length 1920 feet. Percentage = 192 / 1920 *100 = 10%. ... The whale is 1920 ft long. The percentage of the whale's length that the remoras occupy is:

$$\frac{192 \text{ ft}}{1920 \text{ ft}} \times 100\% = 10\%$$

$$\boxed{10\%}$$

---

**Seventh Output (Incorrect)**
analysisConvert remora length to feet:192 in=16 ft. Each 16 ft. 12 remoras total 12*16=192 ft. Whale length 192 ft. Percentage =192/192*100=100%. ... The whale is also 192 ft long, so

$$\frac{192 \text{ ft}}{192 \text{ ft}} \times 100\% = 100\%$$

$$\boxed{100\%}$$

Answer: **10**

*Figure 21.* Seven of eight independently generated outputs from gpt-oss-20b-low evaluated on Template 0. Error type: transcription.

**Prompt (Template 2)**
{Jerry} is playing a dice game during game night for a sticker. {Jerry} is rolling a {1100}-sided die. By how many percentage points is it more likely that they roll a number greater than {22} than that they roll two {even} numbers in a row?

---

**First Output (gpt-oss-20b-low) (Correct)**
analysisWe need probabilities.
Die 1100 sides, numbers 1-1100. Probability roll >22: numbers 23-1100: count = 1100-22=1078. Probability =1078/1100.
... Converting to percentage points: $0.73 \times 100 = 73$ percentage points.

$$\boxed{73}$$

---

**Second Output (Incorrect)**
analysisWe need probability of >22 vs probability of two even numbers in a row. 1100-sided die uniform 1-1100.
Probability greater than 22: numbers 23 to 1100 inclusive. Count =1100-22=1078? Wait 1100-22=1078. Yes. So P1=1078/1100 = simplify? 1078/1100 = 539/550 ≈0.980909.
... Thus it is about

$$\boxed{73.09\%}$$

more likely that a roll exceeds 22 than that two consecutive even numbers are rolled.

---

**Third Output (Correct)**
analysisWe need probabilities. Die sides 1100 fair. Probability roll >22: numbers 23 to 1100 inclusive. Count =1100-22=1078. Probability p1=1078/1100.
... The difference in probabilities is

$$0.98 - 0.25 = 0.73.$$

Expressed as a percentage, this is $0.73 \times 100 = 73$ percentage points.

$$\boxed{73}$$

---

**Fourth Output (Incorrect)**
analysisWe need probability of >22 from 1100 sides: numbers 23-1100 inclusive count = 1100-22=1078. So P1=1078/1100.
Probability of two even numbers in a row in a single roll of the die twice? Four outcomes?
Compute difference: 1078/1100 = 0.981818... difference = 0.981818-0.25=0.731818... multiply by 100 for percentage points difference: 73.1818 percentage points. ...

$$\boxed{73.18 \text{ percentage points}}$$

---

**Fifth Output (Incorrect)**
analysisWe compute P(>22) = numbers 23-1100 inclusive = 1100-22=1078 outcomes out of 1100 ⇒ probability =1078/1100=0.9790909 approx.
... Expressed as percentage points, this is

$$0.72909 \times 100 \approx 72.909\%.$$

$$\boxed{72.909\% \text{ (approximately)}}$$

---

**Sixth Output (Correct)**
analysisWe need P(roll >22) vs P(two even in a row). Die has 1100 sides labeled presumably 1-1100. Probability of >22: there are 1100-22=1078 numbers >22. So P1=1078/1100.
... Difference in percentage points:

$$98\% - 25\% = 73 \text{ percentage points.}$$

$$\boxed{73}$$

---

**Seventh Output (Correct)**
analysisWe need probability of roll >22: numbers 23..1100 inclusive. Count =1100-22=1078. Probability =1078/1100≈0.980. Two even numbers in a row: probability even = 550/1100=0.5. Two in a row: 0.5^2=0.25. Difference in percent points: (0.980 -0.25)*100 = 73%? Let's compute: 1078/1100 = 0.98 exactly (since 1100*0.98=1078). So difference = 0.98-0.25=0.73 ⇒ 73 percentage points. ... The difference in likelihood, in percentage points, is

$$(0.98 - 0.25) \times 100 = 0.73 \times 100 = 73 \text{ percentage points.}$$

$$\boxed{73 \text{ percentage points}}$$

Answer: **73**

*Figure 22.* Seven of eight independently generated outputs from gpt-oss-20b-low evaluated on Template 2. Error type: unnecessary approximation.

**Prompt (Template 7)**
{Jamie} is juggling at a circus. {Jamie} can juggle {928} balls. {3/4} of the balls are golf balls, and {7/8} of the golf balls are blue. How many blue golf balls are there?

---

**First Output (gpt-oss-20b-low) (Correct)**
To determine the number of blue golf balls Jamie has, we need to follow these steps: ...

$$696 \times \frac{7}{8} = 696 \div 8 \times 7 = 87 \times 7 = 609$$

So, Jamie has 609 blue golf balls. Therefore, the number of blue golf balls is $\boxed{609}$.

---

**Second Output (Incorrect)**
To determine the number of blue golf balls Jamie has, we need to follow these steps: ...

$$696 \times \frac{7}{8} = 696 \times 0.875 = 612$$

So, Jamie has 612 blue golf balls. The final answer is:

$$\boxed{612}$$

---

**Third Output (Incorrect)**
To determine the number of blue golf balls Jamie is juggling, we need to follow these steps: ...

$$696 \times \frac{7}{8} = 696 \times 0.875 = 612$$

So, Jamie has 612 blue golf balls. Therefore, the number of blue golf balls is $\boxed{612}$.

---

**Fourth Output (Incorrect)**
To determine the number of blue golf balls Jamie is juggling, we need to follow these steps: ...

$$696 \times \frac{7}{8} = 696 \times 0.875 = 612$$

So, Jamie has 612 blue golf balls. The number of blue golf balls is $\boxed{612}$.

---

**Fifth Output (Correct)**
To determine the number of blue golf balls Jamie is juggling, we need to follow these steps: ...

$$696 \times \frac{7}{8} = 696 \div 8 \times 7 = 87 \times 7 = 609$$

So, Jamie has 609 blue golf balls. The final answer is:

$$\boxed{609}$$

---

**Sixth Output (Incorrect)**
To determine the number of blue golf balls Jamie has, we need to follow these steps: ...

$$696 \times \frac{7}{8} = 696 \times 0.875 = 612$$

Therefore, Jamie has 612 blue golf balls. The final answer is:

$$\boxed{612}$$

---

**Seventh Output (Correct)**
To determine the number of blue golf balls Jamie is juggling, we need to follow these steps: ...

$$7 \times 87 = 609$$

So, there are 609 blue golf balls. Therefore, the number of blue golf balls is $\boxed{609}$.
Answer: **609**

*Figure 23.* Seven of eight independently generated outputs from gpt-oss-20b-low evaluated on Template 7. Error type: wrong multiplication.

**Prompt (Template 8)**
{Jaime} places eggs on some tables. Each table can hold {7} eggs. If he has {57} eggs and {7} tables, how many eggs won't he be able to place on the table?

---

**First Output (gpt-oss-20b-low) (Incorrect)**
analysisWe compute 57 mod 7. 7*8=56, remainder 1. So answer 1 egg.

$$\boxed{1}$$

---

**Second Output (Incorrect)**
analysisWe need remainder 57 mod 7. 7*8=56, remainder1. So 1 egg can't be placed.

$$\boxed{1}$$

---

**Third Output (Correct)**
analysisWe need to see: 7 tables each hold 7 eggs ⇒ max capacity 49? Wait 7 tables *7 eggs =49. He has 57 eggs. So leftover =57-49=8. So answer 8. 1. **Capacity of the tables**: Each table holds 7 eggs. With 7 tables, the total capacity is

$$7 \text{ tables} \times 7 \text{ eggs/table} = 49 \text{ eggs}.$$

2. **Eggs he has**: Jaime has 57 eggs. 3. **Eggs that cannot be placed**: Subtract the capacity from the total eggs:

$$57 - 49 = 8$$

$$\boxed{8}$$

---

**Fourth Output (Incorrect)**
analysisCompute 57 mod 7. 7*8=56 remainder 1.

$$\boxed{1}$$

---

**Fifth Output (Incorrect)**
analysisWe need remainder of 57 divided by 7. 7*8=56, remainder 1. So 1 egg won't fit. The number of eggs that cannot be placed is the remainder when 57 is divided by 7.

$$57 \div 7 = 8 \text{ with a remainder of } 1.$$

So $\boxed{1}$ egg cannot be placed on the tables.

---

**Sixth Output (Correct)**
analysisWe need 57 eggs, each table holds 7, 7 tables. Max capacity 7*7 =49. Eggs unused =57-49=8. So answer 8.

$$\boxed{8}$$

---

**Seventh Output (Incorrect)**
analysisWe compute 57 mod 7. 7*8=56, remainder 1. So 1 egg left. To find the number of eggs that cannot be placed, divide the total eggs by the capacity of each table and look at the remainder:

$$57 \div 7 = 8 \text{ with a remainder of } 1$$

So after using 8 tables for 56 eggs, there is **1 egg** that cannot be placed.

$$\boxed{1}$$

Answer: **8**

*Figure 24.* Seven of eight generated outputs from gpt-oss-20b-low evaluated on Template 8. Error type: problem misinterpretation.

**Prompt (Template 0)**
A dolphin is {2090} feet long. {19} remoras, each {264} inches long, attach to it. What percentage of the dolphin's body length is the combined length of the remoras?

---

**First Output (Qwen2.5-Math-7B-Instruct) (Incorrect)**
Here's how to solve this problem step-by-step: **...**
**Combined length of remoras in inches:**

- 19 remoras * 264 inches/remora = 4960 + 792 = 4960 + 800 - 8 = 5760 - 8 = 5752 inches

**2. Calculate the combined length of the remoras as a percentage of the dolphin's length. ...**
**Answer:** The combined length of the remoras is approximately 22.93% of the dolphin's body length.

---

**Second Output (Incorrect)**
Here's how to solve this problem step-by-step: **...**
**2. Calculate the combined length of the remoras.**

- Number of remoras = 19
- Length of each remora = 264 inches
- Combined length of remoras = 19 remoras * 264 inches/remora = 4956 inches

**...**
The final answer is $\boxed{19.8\%}$.

---

**Third Output (Incorrect)**
Here's how to solve this problem step-by-step:
**2. Calculate the combined length of the remoras.**

- Combined remora length = 19 remoras * 264 inches/remora = 4996 inches

The final answer is $\boxed{19.92\%}$.

---

**Fourth Output (Incorrect)**
Here's how to solve this problem step-by-step:
**1. Convert all measurements to the same unit.** It's easiest to work with inches since the remoras are already measured in inches. **...**
**2. Calculate the combined length of the remoras.**

- There are 19 remoras.
- 19 remoras * 264 inches/remora = 4968 inches

**...**
The final answer is $\boxed{19.81\%}$.

---

**Fifth Output (Correct)**
Here's how to solve this problem: **...**
**5. Calculate the Combined Length of the Remoras:**

$$\text{Combined remora length} = 19 \times 264 \text{ inches} = 5016 \text{ inches}$$

**...**
**9. Final Answer:** The combined length of the remoras is 20% of the dolphin's body length. The final answer is $\boxed{20\%}$.

---

**Sixth Output (Incorrect)**
Here's how to solve this problem: **...**
**2. Calculate the combined length of the remoras.**

- Number of remoras: 19
- Length of each remora: 264 inches
- Combined remora length: 19 remoras * 264 inches/remora = 4996 inches

**...**
The final answer is $\boxed{19.9\%}$.

---

**Seventh Output (Incorrect)**
Here's how to solve this problem step-by-step: **...**
**2. Calculate the combined length of the remoras.**

- There are 19 remoras, and each is 264 inches long.
- Combined remora length = 19 remoras * 264 inches/remora = 4916 inches

**...**
The final answer is $\boxed{19.6\%}$.
Answer: **20**

*Figure 25.* Seven of eight generated outputs from Gemini 2.5 Flash Lite evaluated on Template 0. Error type: wrong multiplication.

**Prompt (Template 2)**
{Jerry} is playing a dice game during game night for a sticker. {Jerry} is rolling a {3040}-sided die. By how many percentage points is it more likely that they roll a number greater than {152} than that they roll two {even} numbers in a row?

---

**First Output (Incorrect)**
Let $N = 3040$ be the number of sides on the die. The possible outcomes when rolling the die are integers from 1 to $N$. ... The number of outcomes greater than 152 is 3040 - 152 = 1888. ... $P(\text{number} > 152) = \frac{59}{95} \approx 0.6210526$, $P(\text{two even numbers}) = \frac{1}{4} = 0.25$. Difference $= 0.6210526 - 0.25 = 0.3710526$. In percentage points, this is $0.3710526 \times 100\% = 37.10526\%$.

The result is $\frac{705}{19}$ percentage points. The final answer is $\boxed{\frac{705}{19}}$.

---

**Second Output (Incorrect)**
Let $N = 3040$ be the number of sides on the die. We want to compare the probability of two events: Event A: Rolling a number greater than 152. Event B: Rolling two even numbers in a row. The number of outcomes in Event A is $3040 - 153 + 1 = 3040 - 152 = 1888$. ... ...

$$P(A) - P(B) = \frac{1888}{3040} - \frac{1}{4} = \frac{1888}{3040} - \frac{760}{3040} = \frac{1128}{3040}$$

The final answer is $\boxed{37.105}$.

---

**Third Output (Incorrect)**
Let $N = 3040$ be the number of sides on the die. We want to compare the probability of rolling a number greater than 152 with the probability of rolling two even numbers in a row. The number of outcomes greater than 152 is $3040 - 153 + 1 = 1888$. $P(\text{number} > 152) = \frac{1888}{3040} = \frac{59}{95}$. ... $P(\text{two even numbers in a row}) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. ... ... The result $\frac{705}{19}$ is approximately 37.105. The final answer is $\boxed{\frac{705}{19}}$.

---

**Fourth Output (Incorrect)**
Let $N = 3040$ be the number of sides on the die. The possible outcomes when rolling the die are the integers from 1 to $N$. The number of outcomes in event $A$ is $3040 - 153 + 1 = 1888$. ... $P(A) = \frac{59}{95} \approx 0.6210526$, $P(B) = \frac{1}{4} = 0.25$. ... The difference is $P_1 - P_2 = \frac{1888}{3040} - \frac{1}{4} = \frac{1128}{3040} = \frac{141}{380}$. The percentage point difference is $\frac{141}{380} \times 100\% = \frac{705}{19}\% = 37\frac{2}{19}\%$.

The final answer is $\boxed{\frac{705}{19}}$.

---

**Fifth Output (Incorrect)**
Let $N = 3040$ be the number of sides on the die. The possible outcomes are the integers from 1 to $N$. We assume the die is fair, so each outcome has a probability of $\frac{1}{N}$. The number of such outcomes is $3040 - 153 + 1 = 1888$. ... $P(B) = P(E \text{ on first roll}) \times P(E \text{ on second roll}) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. ... Rounding to two decimal places, the difference is 37.11 percentage points.

The final answer is $\boxed{37.11}$.

---

**Sixth Output (Incorrect)**
Let $N = 3040$ be the number of sides on the die. We are comparing two probabilities: The number of outcomes in event $A$ is $3040 - 153 + 1 = 3040 - 152 = 1888$. ... $P(B) = P(\text{even on 1st roll}) \times P(\text{even on 2nd roll}) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. ... The final answer is $\frac{705}{19}$. The final answer is $\boxed{37.105}$.

---

**Seventh Output (Correct)**
Let $N$ be the number of sides on the die, so $N = 3040$. $P(\text{greater than } 152) = \frac{2888}{3040}$. $P(\text{greater than } 152) = \frac{361}{380}$. We can also write this as a decimal: $\frac{2888}{3040} = 0.95$. $P(\text{two even numbers in a row}) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4} = 0.25$. Difference in probability $= 0.95 - 0.25 = 0.70$. Difference in percentage points $= 0.70 \times 100 = 70$.
The final answer is $\boxed{70}$.
Answer:**70**

*Figure 26.* Seven of eight generated outputs from Gemini 2.5 Flash Lite evaluated on Template 2. Error type: wrong subtraction.

**Prompt (Template 6)**
{John} has {35} {square yard}s of {pineapple}s field. There are {43} {pineapple}s per {square yard}. {John} can harvest his {pineapple}s every {2} months. How many {pineapple}s can {John} harvest within a year?"

---

**First Output (Gemini 2.5 Flash Lite) (Correct)**
Here's how to solve the problem step-by-step:

1. **Calculate the total number of pineapples:**
* John has 35 square yards of field.
* There are 43 pineapples per square yard.
* Total pineapples = 35 square yards * 43 pineapples/square yard = 1505 pineapples.

...
The final answer is $\boxed{9030}$.

---

**Second Output (Incorrect)**
Here's how to solve the problem step-by-step:

1. **Calculate the total number of pineapples:**
* John has 35 square yards of field.
* There are 43 pineapples per square yard.
* Total pineapples = 35 square yards * 43 pineapples/square yard
* Total pineapples = 1455 pineapples

...
The final answer is $\boxed{8730}$.

---

**Third Output (Incorrect)**
Here's how to solve the problem step-by-step:

1. **Calculate the total number of pineapples John has:**
* He has 35 square yards.
* There are 43 pineapples per square yard.
* Total pineapples = 35 square yards * 43 pineapples/square yard

* 35 * 43 = 1495 pineapples

...
The final answer is $\boxed{8970}$.

---

**Fourth Output (Incorrect)**
Here's how to solve the problem step-by-step:

1. **Calculate the total number of pineapples:**
* John has 35 square yards of field.
* There are 43 pineapples per square yard.
* Total pineapples = 35 square yards * 43 pineapples/square yard = 1495 pineapples.

...
The final answer is $\boxed{8970}$.

---

**Fifth Output (Incorrect)**

Here's how to solve the problem step-by-step:

1. **Calculate the total number of pineapples:**
* John has 35 square yards of field.
* There are 43 pineapples per square yard.
* Total pineapples = 35 square yards * 43 pineapples/square yard = 1495 pineapples.

...
The final answer is $\boxed{8970}$.
Answer: **9030**

*Figure 27.* First five of eight independently generated outputs from Gemini 2.5 Flash Lite evaluated on Template 6. Error type: wrong multiplication.