

Inference Time Scaling

Sharon Biju and Suraj Vaddi

Inference Time Scaling - Background

- Inference Time: time it takes model to make predictions (perform inference)
- Larger models typically have longer inference times because they need more computations for each prediction
- Performance scales with repeated sampling and inference compute
- Inference Compute can Outperform Model Scaling

Paper 1

“Self-Consistency Improves Chain of Thought Reasoning in Language Models”

Chain-of-Thought (CoT) Prompting - Background

- encourages language models to break down reasoning tasks into intermediate steps rather than predicting a final answer directly

Greedy Decoding and Its Issues

- standard approach for generating model outputs, where the model selects the most probable token at each step
- Issue 1- Lack of Diversity: Picks only one reasoning path, which may be incorrect
- Issue 2 - Local Optimality: It may not explore other valid paths that lead to a more accurate answer

Alternative Decoding Strategy: Self-Consistency

- Relies on sampling multiple reasoning paths and selects most consistent final answer instead of relying on a single path
- Entirely unsupervised: works with pre-trained language models
- Needs no additional human annotation
- Requires additional training, auxiliary models, or fine-tuning
- How it works:
 1. Samples a diverse set of reasoning paths by sampling from the language model's decoder using stochastic decoding methods (paper mentions temperature and top-k sampling)
 2. Selects most consistent answer by marginalizing sampled reasoning paths

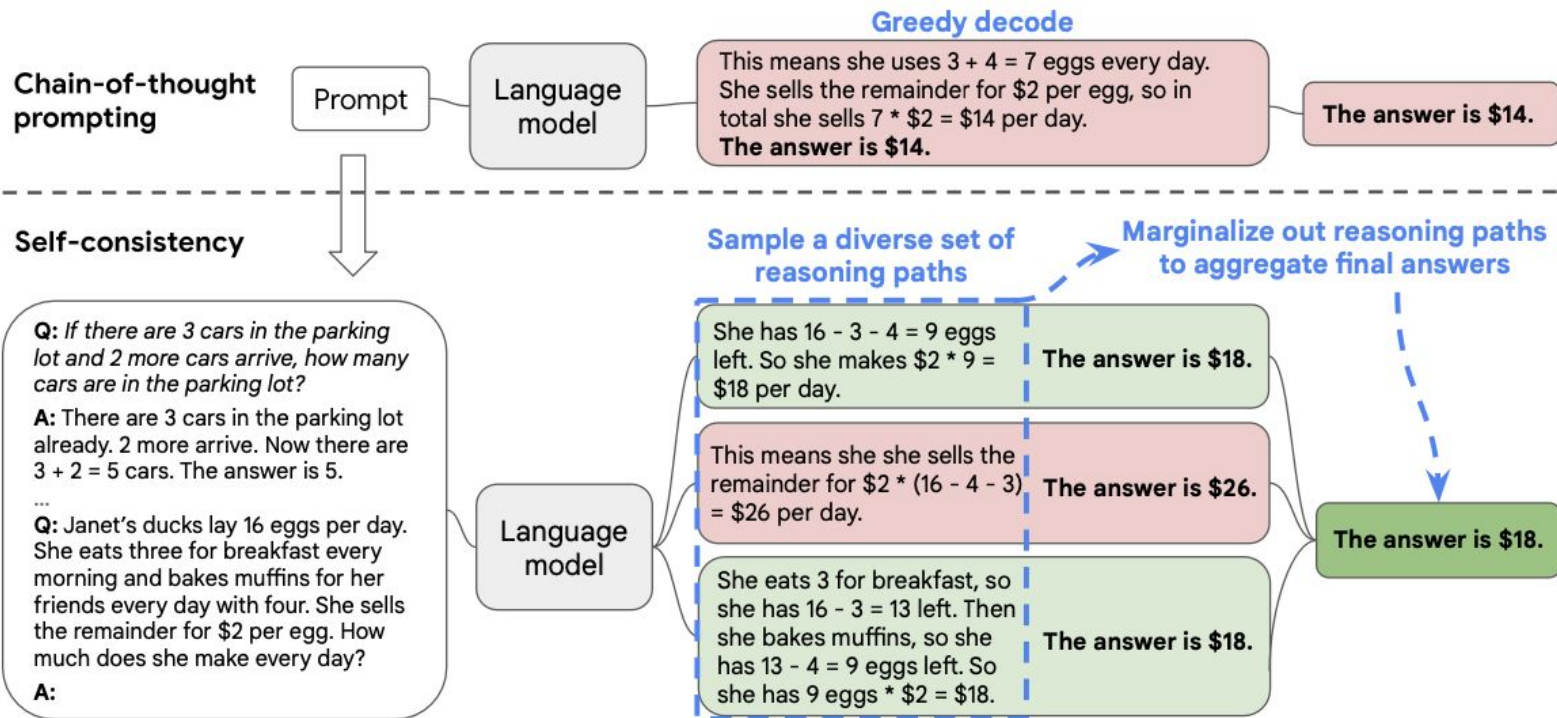


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the “greedy decode” in CoT prompting by sampling from the language model’s decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

Self-Consistency vs Greedy Decoding

Method	How It Works	Pros	Cons
Greedy Decoding	Picks the highest probability token at each step	Fast, simple	Can get stuck in local optima
Self-Consistency	Samples multiple reasoning paths and picks the most consistent answer	More robust, reduces errors	Requires more compute

Experimental Setup - Benchmarks

- Arithmetic reasoning: Math Word Problem Repository, AQUA-RAT
- Commonsense reasoning: CommonsenseQA, StrategyQA, AI2 Reasoning Challenge
- Symbolic reasoning: last letter concatenation and Coinflip

Table 14: Few-shot exemplars for AQUA-RAT.

Q: John found that the average of 15 numbers is 40. If 10 is added to each number then the mean of the numbers is? Answer Choices: (a) 50 (b) 45 (c) 65 (d) 78 (e) 64

A: If 10 is added to each number, then the mean of the numbers also increases by 10. So the new mean would be 50. The answer is (a).

Table 15: Few-shot exemplars for ARC easy/challenge.

Q: George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? (a) dry palms. (b) wet palms. (c) palms covered with oil. (d) palms covered with lotion.

A: Dry surfaces will more likely cause more friction via rubbing than other smoother surfaces, hence dry palms will produce the most heat. The answer is (a).

Table 16: Few-shot exemplars for HotpotQA (closed-book setting).

Q: Which magazine was started first Arthur's Magazine or First for Women?

A: Arthur's Magazine started in 1844. First for Women started in 1989. So Arthur's Magazine was started first. The answer is Arthur's Magazine.

Experimental Setup - Models Used

- UL2 20B
- LaMBDA 137B
- GPT-3 175B
- PaLM 540B

Applied temperature sampling with $T=0.5$ and truncated top-k ($k=40$) tokens with highest probability

Applied temperature sampling with $T=0.7$ and no top-k truncation

Applied temperature sampling with $T=0.7$ and $k=40$

Experimental Goal: Compare Self-Consistency vs. CoT with Greedy decoding

- Reported results averaged over 10 runs
- Sampled 40 outputs independently from the decoder in each run

Experiment Results - Arithmetic Reasoning Accuracy

	Method	AddSub	MultiArith	ASDiv	AQuA	SVAMP	GSM8K
	Previous SoTA	94.9^a	60.5 ^a	75.3 ^b	37.9 ^c	57.4 ^d	35 ^e / 55 ^g
UL2-20B	CoT-prompting	18.2	10.7	16.9	23.6	12.6	4.1
	Self-consistency	24.8 (+6.6)	15.0 (+4.3)	21.5 (+4.6)	26.9 (+3.3)	19.4 (+6.8)	7.3 (+3.2)
LaMDA-137B	CoT-prompting	52.9	51.8	49.0	17.7	38.9	17.1
	Self-consistency	63.5 (+10.6)	75.7 (+23.9)	58.2 (+9.2)	26.8 (+9.1)	53.3 (+14.4)	27.7 (+10.6)
PaLM-540B	CoT-prompting	91.9	94.7	74.0	35.8	79.0	56.5
	Self-consistency	93.7 (+1.8)	99.3 (+4.6)	81.9 (+7.9)	48.3 (+12.5)	86.6 (+7.6)	74.4 (+17.9)
GPT-3 Code-davinci-001	CoT-prompting	57.2	59.5	52.7	18.9	39.8	14.6
	Self-consistency	67.8 (+10.6)	82.7 (+23.2)	61.9 (+9.2)	25.6 (+6.7)	54.5 (+14.7)	23.4 (+8.8)
GPT-3 Code-davinci-002	CoT-prompting	89.4	96.2	80.1	39.8	75.8	60.1
	Self-consistency	91.6 (+2.2)	100.0 (+3.8)	87.8 (+7.6)	52.0 (+12.2)	86.8 (+11.0)	78.0 (+17.9)

Experiment Results - Commonsense and Symbolic Reasoning Accuracy

	Method	CSQA	StrategyQA	ARC-e	ARC-c	Letter (4)	Coinflip (4)
	Previous SoTA	91.2^a	73.9 ^b	86.4 ^c	75.0 ^c	N/A	N/A
UL2-20B	CoT-prompting	51.4	53.3	61.6	42.9	0.0	50.4
	Self-consistency	55.7 (+4.3)	54.9 (+1.6)	69.8 (+8.2)	49.5 (+6.8)	0.0 (+0.0)	50.5 (+0.1)
LaMDA-137B	CoT-prompting	57.9	65.4	75.3	55.1	8.2	72.4
	Self-consistency	63.1 (+5.2)	67.8 (+2.4)	79.3 (+4.0)	59.8 (+4.7)	8.2 (+0.0)	73.5 (+1.1)
PaLM-540B	CoT-prompting	79.0	75.3	95.3	85.2	65.8	88.2
	Self-consistency	80.7 (+1.7)	81.6 (+6.3)	96.4 (+1.1)	88.7 (+3.5)	70.8 (+5.0)	91.2 (+3.0)
GPT-3 Code-davinci-001	CoT-prompting	46.6	56.7	63.1	43.1	7.8	71.4
	Self-consistency	54.9 (+8.3)	61.7 (+5.0)	72.1 (+9.0)	53.7 (+10.6)	10.0 (+2.2)	75.9 (+4.5)
GPT-3 Code-davinci-002	CoT-prompting	79.0	73.4	94.0	83.6	70.4	99.0
	Self-consistency	81.5 (+2.5)	79.8 (+6.4)	96.0 (+2.0)	87.5 (+3.9)	73.4 (+3.0)	99.5 (+0.5)

Experiment Results - Commonsense and Symbolic Reasoning Accuracy

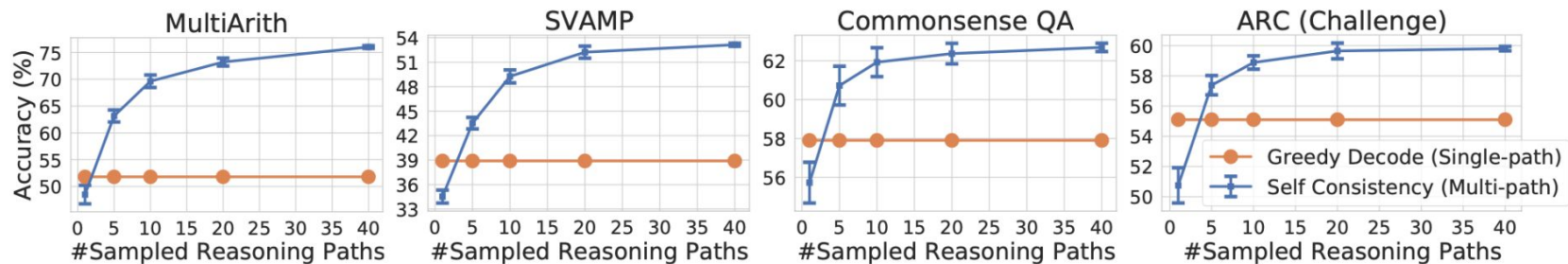


Figure 2: Self-consistency (blue) significantly improves accuracy over CoT-prompting with greedy decoding (orange) across arithmetic and commonsense reasoning tasks, over LaMDA-137B. Sampling a higher number of diverse reasoning paths consistently improves reasoning accuracy.

- Coinflip Reasoning and Last Letter Concatenation
- OOD Test

Comparison to Sample-and-Rank

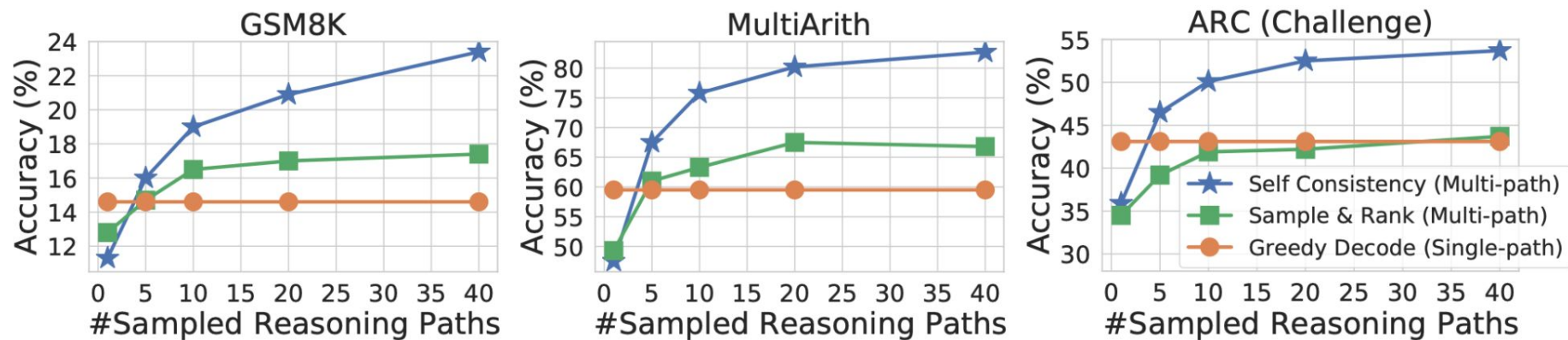


Figure 3: Self-consistency significantly outperforms sample-and-rank with the same # of samples.

- compares self-consistency with sample-and-rank on GPT-3 code-davinci-001
- Sample-and-Rank: ranks sequences based on their log probability and selects what's top ranked

Comparison to Beam Search

	Beam size / Self-consistency paths	1	5	10	20	40
AQuA	Beam search decoding (top beam)	23.6	19.3	16.1	15.0	10.2
	Self-consistency using beam search	23.6	19.8 \pm 0.3	21.2 \pm 0.7	24.6 \pm 0.4	24.2 \pm 0.5
	Self-consistency using sampling	19.7 \pm 2.5	24.9 \pm 2.6	25.3 \pm 1.8	26.7 \pm 1.0	26.9 \pm 0.5
MultiArith	Beam search decoding (top beam)	10.7	12.0	11.3	11.0	10.5
	Self-consistency using beam search	10.7	11.8 \pm 0.0	11.4 \pm 0.1	12.3 \pm 0.1	10.8 \pm 0.1
	Self-consistency using sampling	9.5 \pm 1.2	11.3 \pm 1.2	12.3 \pm 0.8	13.7 \pm 0.9	14.7 \pm 0.3

Table 6: Compare self-consistency with beam search decoding on the UL2-20B model.

- Beam Search: decoding technique generating fixed number of tokens at each step (called beams) and selects best beam based on its cumulative probability at each step
- Self-consistency is capable of adopting beam search to decode each reasoning path

Comparison to Ensemble-based Approaches

	GSM8K	MultiArith	SVAMP	ARC-e	ARC-c
CoT (Wei et al., 2022)	17.1	51.8	38.9	75.3	55.1
Ensemble (3 sets of prompts)	18.6 \pm 0.5	57.1 \pm 0.7	42.1 \pm 0.6	76.6 \pm 0.1	57.0 \pm 0.2
Ensemble (40 prompt permutations)	19.2 \pm 0.1	60.9 \pm 0.2	42.7 \pm 0.1	76.9 \pm 0.1	57.0 \pm 0.1
Self-Consistency (40 sampled paths)	27.7 \pm 0.2	75.7 \pm 0.3	53.3 \pm 0.2	79.3 \pm 0.3	59.8 \pm 0.2

Table 7: Self-consistency outperforms prompt-order and multi-prompt ensembles on LaMDA-137B.

- 2 ensemble methods tested: Prompt Order Permutation and Multiple Sets of Prompts
- “Self-consistency acts more like a ‘self-ensemble’ on top of a single language model”

Additional Studies - Self-Consistency is Robust to Sampling Strategies and Scaling

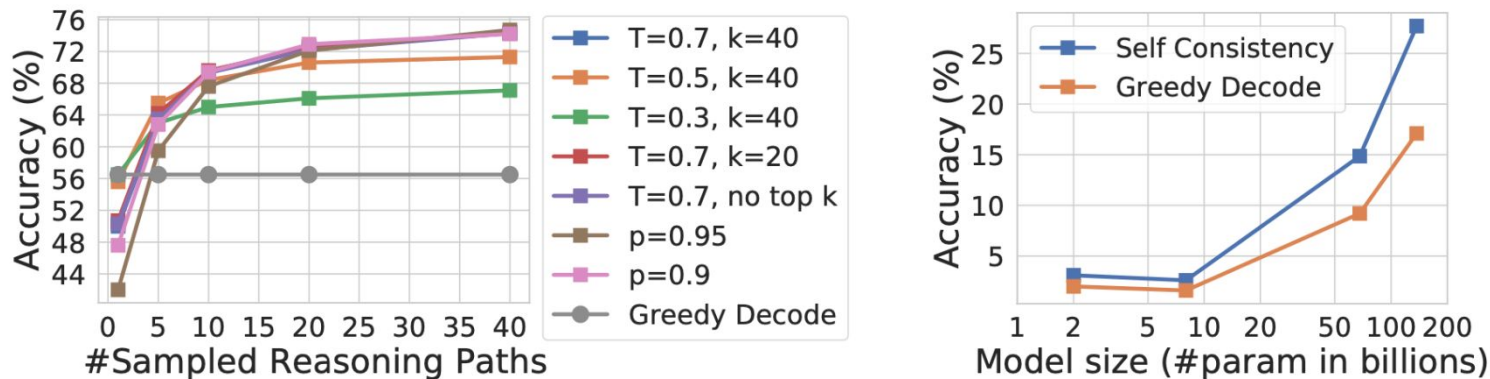


Figure 4: GSM8K accuracy. (Left) Self-consistency is robust to various sampling strategies and parameters. (Right) Self-consistency improves performance across language model scales.

- Varied parameters like T in temperature sampling, k in top- k sampling, and p in nucleus sampling to see how those affect performance of self-consistency

Additional Studies - Other Findings

- Self-consistency Improves Robustness to Imperfect Prompts
- Self-consistency Works for Non-Natural Language Reasoning Paths and Zero-shot CoT

CoT with Greedy Being Outperformed By Self-Consistency

Question	Greedy Decode	Sampled Path 1	Sampled Path 2
[MultiArith] <i>Sam had 10 socks. If he threw away 3 old ones that didn't fit and bought 36 new ones, how many socks would he have?</i>	Sam had 10 socks. He threw away 3 old ones that didn't fit. He bought 36 new ones. That means he has $10 + 36 - 3 = 41$ socks. The answer is 41.	Sam had 10 socks. He threw away 3 old ones. So now he has 7 left. Then he bought 36 new ones. So now he has $7 + 36 = 43$ socks. The answer is 43.	He had 10 socks. He threw away 3 old ones that didn't fit. He bought 36 new ones. So in total he has $10 - 3 + 36 = 43$ socks. The answer is 43.
[SVAMP] <i>In a school there are 697 girls and the rest are boys. If there are 228 more girls than boys. How many boys are there in that school?</i>	There are 697 girls and the rest are boys. That means there are $697 + 228 = 925$ girls. The rest are boys. So the number of boys is $925 - 697 = 228$. The answer is 228.	The number of boys is the difference between 697 and 228. This is $697 - 228 = 469$. The answer is 469.	In the beginning there are 697 girls and the rest are boys. So, the number of boys must be $697 - 228 = 469$ boys. The answer is 469.
[AQuA] <i>There are 10 oranges in a basket. Find the no. of ways in which 2 oranges are chosen from the basket? (a) 45 (b) 90 (c) 120 (d) 150 (e) 180</i>	There are 10 oranges in a basket. So there are 10 ways in which 2 oranges are chosen from the basket.	There are 10 oranges in a basket. We need to choose 2 oranges from the basket. So the number of ways are 10C2. So the answer is (a).	There are 10 oranges in the basket. If you choose 2 oranges, you have $10C2 = 45$ ways of choosing 2 oranges. So the answer is (a).
[CommonsenseQA] <i>The man laid on the soft moss and looked up at the trees, where was the man? (a) niagra falls (b) forest (c) waterfall (d) ground (e) tree</i>	The answer must be something that is soft and mossy. Of the above choices, only ground is soft and mossy. So the answer is (d).	The answer must be somewhere where moss is found. Of the above choices, only forests are known for having moss. So the answer is (b).	The answer must be a place where the man is laying down and looking up at trees. Of the above choices, only forest is a place where people lay down and look at trees. So the answer is (b).
[ARC] <i>Which of these will most likely harm a habitat? (a) planting trees. (b) water pollution. (c) rainfall. (d) sunlight.</i>	Planting trees will not harm a habitat. So the answer is (a).	The answer is (b), water pollution, because it will most likely harm the habitat. The other answers are not likely to harm the habitat.	The answer is (b), water pollution. Trees are good for a habitat, rainfall is necessary for a habitat, and sunlight is necessary for a habitat. So the answer is (b).
[ARC-challenge] <i>Which process in the carbon cycle takes the longest to complete? (a) emission of waste. (b) respiration in animals. (c) photosynthesis in plants. (d) formation of fossil fuels.</i>	The process of photosynthesis in plants takes the longest to complete. So the answer is (c).	The formation of fossil fuels takes the longest time to complete. So the answer is (d).	The process that takes the longest time to complete is the formation of fossil fuels, which happens over millions of years. So the answer is (d).

Table 12: Additional examples where self-consistency helps repair the errors over greedy decode on LaMDA-137B. Two sampled reasoning paths that are consistent with the ground truth are shown.

Future Work

- Can we design adaptive test-time compute methods?
- How do we reduce compute costs while keeping high accuracy?

Paper 2

“Large Language Monkeys: Scaling Inference Compute with Repeated Sampling”

Research Goals

Scaling during training is heavily researched, but scaling during inference has not had the same investment.

- Coverage: As the number of samples increases, what fraction of problems can we solve using any sample that was generated?
- Precision: How often can we identify correct samples from our collection of generations?

Coverage and Precision Implementation

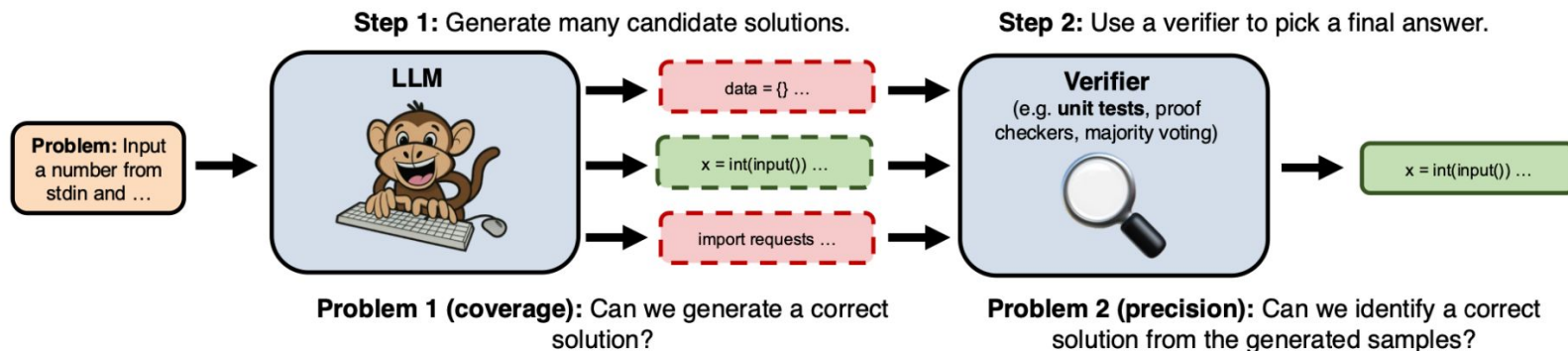


Figure 1: The repeated sampling procedure that we follow in this paper. 1) We generate many independent candidate solutions for a given problem by sampling from an LLM with a positive temperature. 2) We use a domain-specific verifier (ex. unit tests for code) to select a final answer from the generated samples.

Task Datasets

1. **GSM8K:** grade-school level math word problems, using a random subset of 128 problems from test set.
2. **MATH:** harder math word problems that are generally harder than those from GSM8K , using 128 random problems from test set.
3. **MiniF2F-MATH:** mathematics problems formalized into proof checking languages, using Lean4 as our language, and evaluate on the 130 test set problems that are formalized from the MATH dataset.
4. **CodeContests:** competitive programming problems. Each problem has a text description, along with a set of input-output test cases (hidden from the model) that can be used to verify the correctness of a candidate solution. We enforce that models write their solutions using Python3.
5. **SWE-bench Lite:** real-world Github issues, where each problem consists of a description and a snapshot of a code repository. To solve a problem, models must edit files in the codebase (in the Lite subset of SWE-bench that we use, only a single file needs to be changed). Candidate solutions can be automatically checked using the repository's suite of unit tests.

Task Datasets (1. GSM8K)

Grade-school level math word problems, using a random subset of 128 problems from test set.

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

Solution: Beth bakes 4 2 dozen batches of cookies for a total of $4 \times 2 = 8$ dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of $12 \times 8 = 96$ cookies
She splits the 96 cookies equally amongst 16 people so they each eat $96 / 16 = 6$ cookies

Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs \$3.50?

Mrs. Lim got 68 gallons - 18 gallons = 50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = 200 gallons.
She was able to sell 200 gallons - 24 gallons = 176 gallons.
Thus, her total revenue for the milk is $\$3.50/\text{gallon} \times 176 \text{ gallons} = \616 .

Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?

Solution: Tina buys 3 12-packs of soda, for $3 \times 12 = 36$ sodas
6 people attend the party, so half of them is $6 / 2 = 3$ people
Each of those people drinks 3 sodas, so they drink $3 \times 3 = 9$ sodas
Two people drink 4 sodas, which means they drink $2 \times 4 = 8$ sodas
With one person drinking 5, that brings the total drank to $5 + 9 + 8 = 25$ sodas
As Tina started off with 36 sodas, that means there are $36 - 25 = 11$ sodas left

Final Answer: 11

<https://paperswithcode.com/dataset/gsm8k>

Task Datasets (2. MATH)

Harder math word problems that are generally harder than those from GSM8K , using 128 random problems from test set.

MATH Dataset (Ours)

Problem: Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?

Solution: There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.

Problem: If $\sum_{n=0}^{\infty} \cos^{2n} \theta = 5$, what is $\cos 2\theta$?

Solution: This geometric series is

$$1 + \cos^2 \theta + \cos^4 \theta + \dots = \frac{1}{1 - \cos^2 \theta} = 5. \text{ Hence,}$$

$$\cos^2 \theta = \frac{4}{5}. \text{ Then } \cos 2\theta = 2 \cos^2 \theta - 1 = \boxed{\frac{3}{5}}.$$

Problem: The equation $x^2 + 2x = i$ has two complex solutions. Determine the product of their real parts.

Solution: Complete the square by adding 1 to each side.

Then $(x + 1)^2 = 1 + i = e^{\frac{\pi}{4}} \sqrt{2}$, so $x + 1 = \pm e^{\frac{\pi}{8}} \sqrt[4]{2}$.

The desired product is then

$$(-1 + \cos(\frac{\pi}{8}) \sqrt[4]{2})(-1 - \cos(\frac{\pi}{8}) \sqrt[4]{2}) =$$

$$1 - \cos^2(\frac{\pi}{8}) \sqrt{2} = 1 - \frac{(1 + \cos(\frac{\pi}{4}))}{2} \sqrt{2} = \boxed{\frac{1 - \sqrt{2}}{2}}.$$

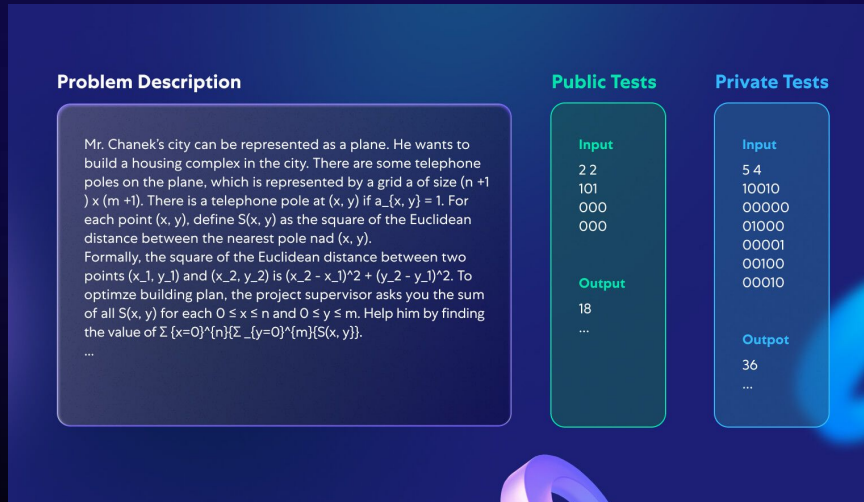
Task Datasets (3. MiniF2F-MATH)

Mathematics problems formalized into proof checking languages, using Lean4 as our language, and evaluate on the 130 test set problems that are formalized from the MATH dataset.

Natural Language	Two non-zero real numbers, a and b , satisfy $ab = a - b$. Which of the following is a possible value of $\frac{a}{b} + \frac{b}{a} - ab$? (A) -2 (B) $-\frac{1}{2}$ (C) $\frac{1}{3}$ (D) $\frac{1}{2}$ (E) 2
Lean	<pre>theorem amc12.2000_p11 (a b : ℝ) (h₀ : a ≠ 0 ∧ b ≠ 0) (h₁ : a * b = a - b) : a / b + b / a - a * b = 2 := begin field_simp [h₀.1, h₀.2], simp only [h₁, mul_comm, mul_sub], ring, end</pre>

Task Datasets (4. CodeContests)

Competitive programming problems. Each problem has a text description, along with a set of input-output test cases (hidden from the model) that can be used to verify the correctness of a candidate solution. We enforce that models write their solutions using Python3.

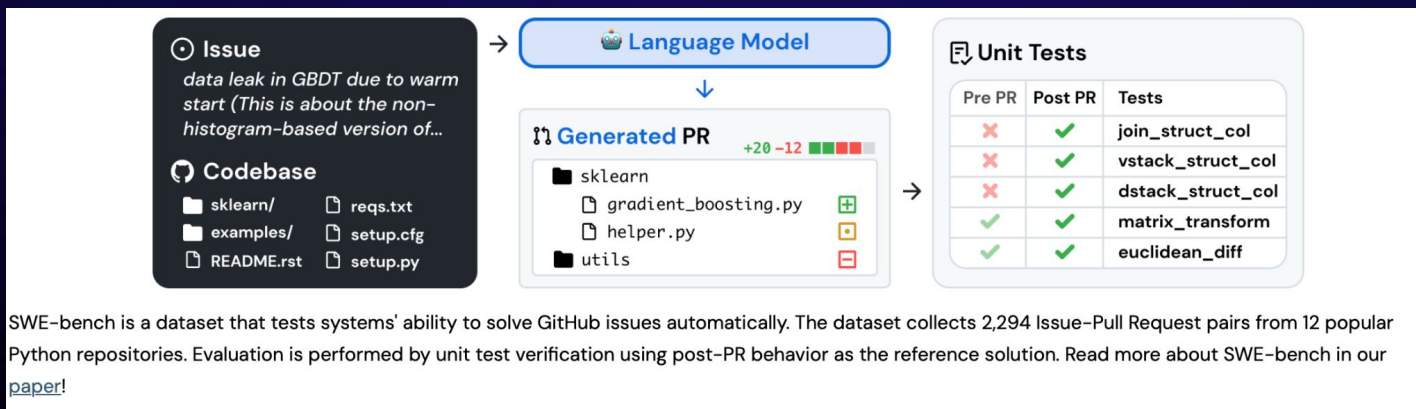


The screenshot displays a problem page with three main sections: Problem Description, Public Tests, and Private Tests. The Problem Description section contains text about Mr. Chanek's city and a grid of telephone poles, defining a function S(x, y) as the square of the Euclidean distance to the nearest pole. The Public Tests section shows an input of a 2x2 grid and an output of 18. The Private Tests section shows a 5x4 grid and an output of 36.

Problem Description	Public Tests	Private Tests
<p>Mr. Chanek's city can be represented as a plane. He wants to build a housing complex in the city. There are some telephone poles on the plane, which is represented by a grid of size $(n + 1) \times (m + 1)$. There is a telephone pole at (x, y) if $a_{x,y} = 1$. For each point (x, y), define $S(x, y)$ as the square of the Euclidean distance between the nearest pole and (x, y).</p> <p>Formally, the square of the Euclidean distance between two points (x_1, y_1) and (x_2, y_2) is $(x_2 - x_1)^2 + (y_2 - y_1)^2$. To optimize building plan, the project supervisor asks you the sum of all $S(x, y)$ for each $0 \leq x \leq n$ and $0 \leq y \leq m$. Help him by finding the value of $\sum_{x=0}^n \sum_{y=0}^m S(x, y)$.</p> <p>...</p>	<p>Input</p> <pre>2 2 101 000 000</pre> <p>Output<pre>18 ...</pre></p>	<p>Input<pre>5 4 10010 00000 01000 00001 00100 00010</pre><p>Output<pre>36 ...</pre></p></p>

Task Datasets (5. SWE-bench Lite)

Real-world Github issues, where each problem consists of a description and a snapshot of a code repository. To solve a problem, models must edit files in the codebase (in the Lite subset of SWE-bench that we use, only a single file needs to be changed). Candidate solutions can be automatically checked using the repository's suite of unit tests.



<https://www.swebench.com/index.html#test>

Repeated Sampling

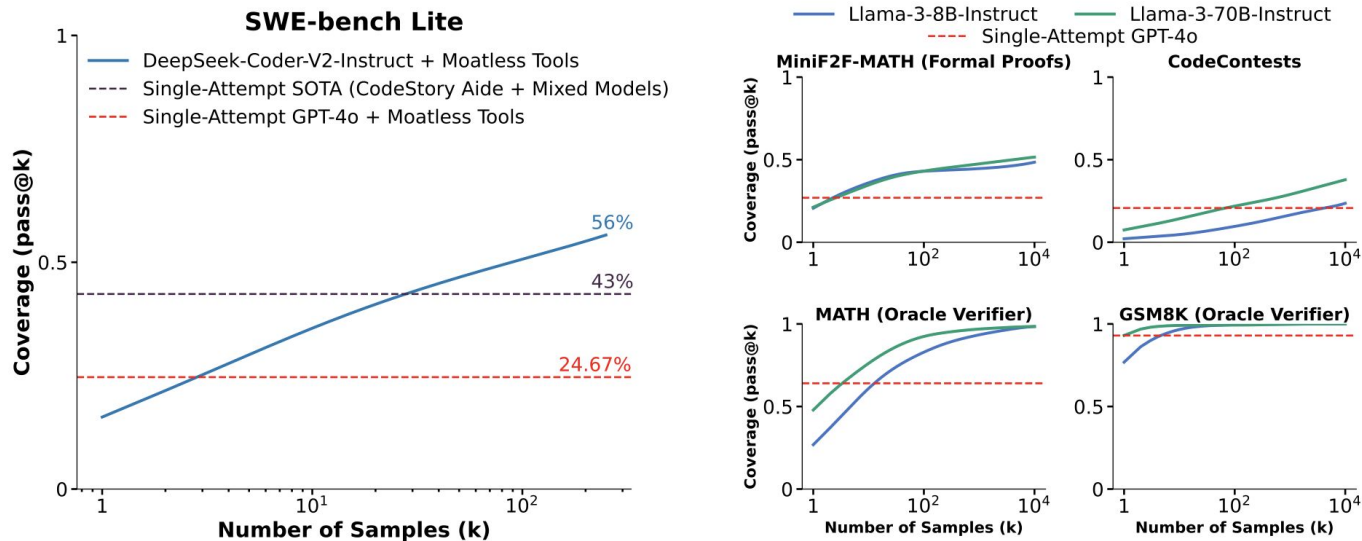


Figure 2: Across five tasks, we find that coverage (the fraction of problems solved by at least one generated sample) increases as we scale the number of samples. Notably, using repeated sampling, we are able to increase the solve rate of an open-source method from 15.9% to 56% on SWE-bench Lite.

Broader Set of Models

1. Llama 3: Llama-3-8B, Llama-3-8B-Instruct, Llama-3-70B-Instruct.
2. Gemma: Gemma-2B, Gemma-7B [52].
3. Pythia: Pythia-70M through Pythia-12B (eight models in total) [9].

Repeated Sampling across Model Sizes, Families, and Post-Training

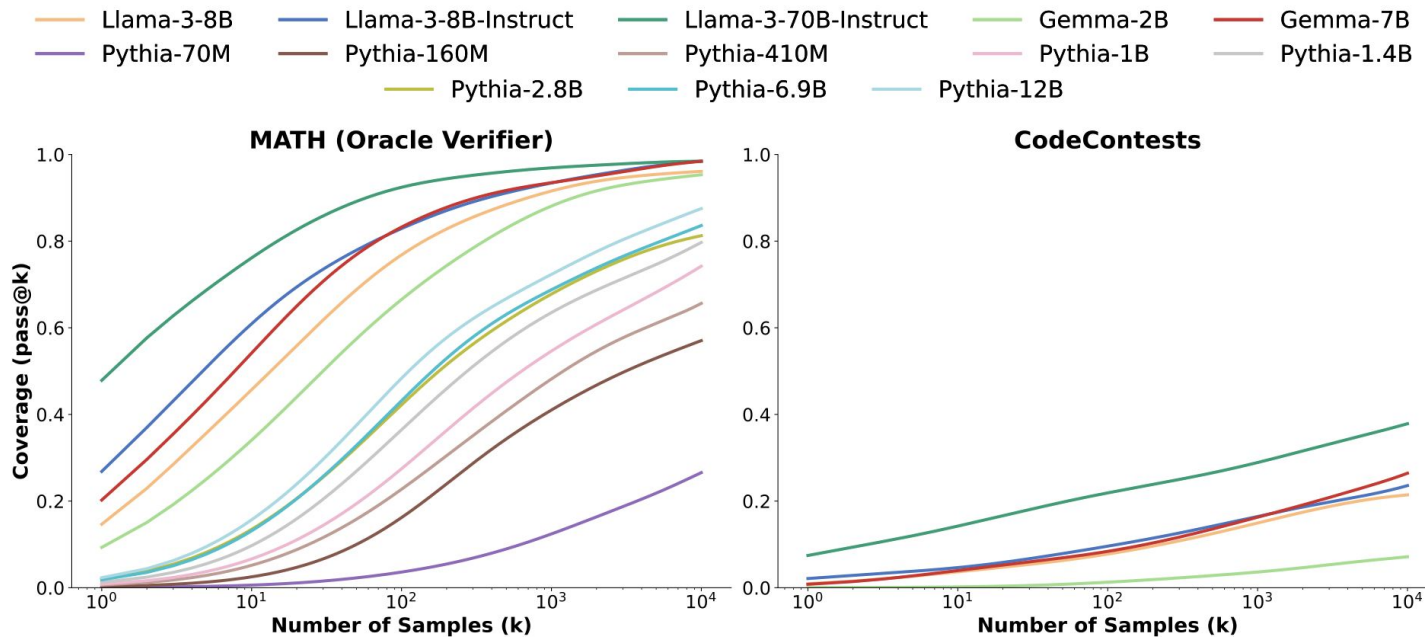


Figure 3: Scaling inference time compute via repeated sampling leads to consistent coverage gains across a variety of model sizes (70M-70B), families (Llama, Gemma and Pythia) and levels of post-training (Base and Instruct models).

FLOPs as a Cost Metric

$$\text{FLOPsPerToken}(\text{ContextLen}) \approx 2 * (\text{NumParameters} + 2 * \text{NumLayers} * \text{TokenDim} * \text{ContextLen})$$

$$\text{TotalInferenceFLOPs} \approx \left(\sum_{t=1}^{\text{NumPromptTokens}} \text{FLOPsPerToken}(t) \right) + \left(\sum_{t=1}^{\text{NumDecodeTokens}} \text{FLOPsPerToken}(t + \text{NumPromptTokens}) * \text{NumCompletions} \right)$$

Repeated Sampling to Balance Performance and Cost

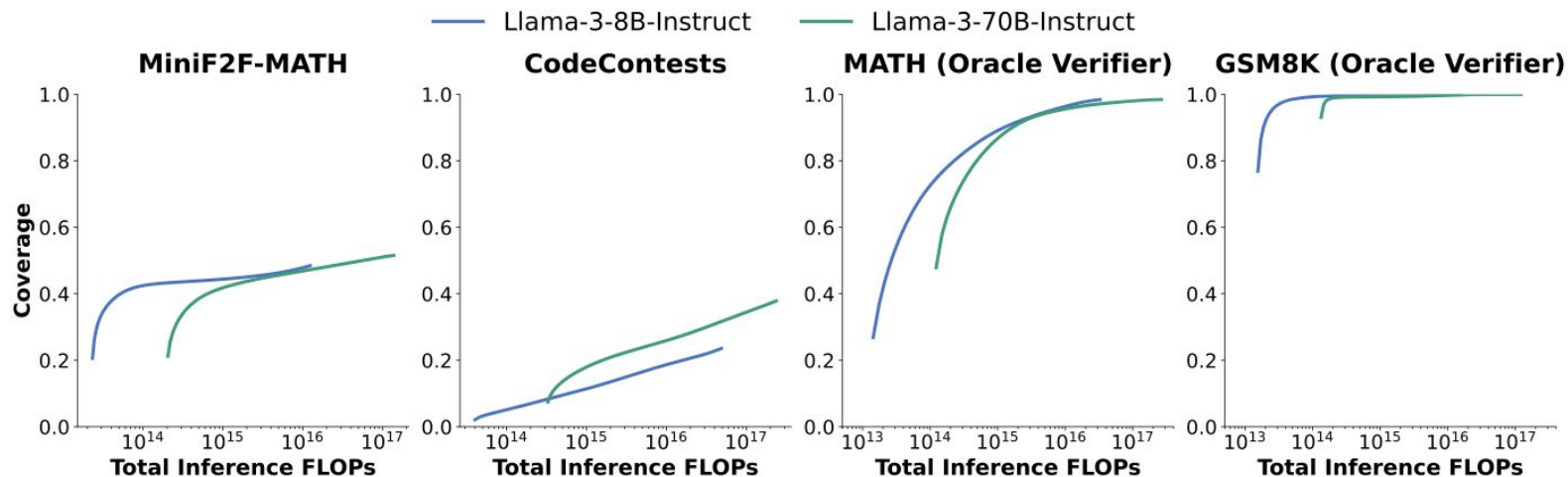


Figure 4: Comparing cost, measured in number of inference FLOPs, and coverage for Llama-3-8B-Instruct and Llama-3-70B-Instruct. We see that the ideal **model size** depends on the task, compute budget, and coverage requirements. Note that Llama-3-70B-Instruct does not achieve 100% coverage on GSM8K due to an incorrectly labelled ground truth answer: see Appendix [E](#).

Model	Cost per attempt (USD)	Number of attempts	Issues solved (%)	Total cost (USD)	Relative total cost
DeepSeek-Coder-V2-Instruct	0.0072	5	29.62	10.8	1x
GPT-4o	0.13	1	24.00	39	3.6x
Claude 3.5 Sonnet	0.17	1	26.70	51	4.7x

Table 1: Comparing API cost (in US dollars) and performance for various models on the SWE-bench Lite dataset using the Moatless Tools agent framework. When sampled more, the open-source DeepSeek-Coder-V2-Instruct model can achieve the same issue solve-rate as closed-source frontier models for under a third of the price.

Coverage versus Sample Budget

1. The relationship between coverage and the number of samples can often be modelled with an exponentiated power law.
2. For a given task, the coverage curves of different models from the same family resemble S-curves with similar slopes but distinct horizontal offsets.

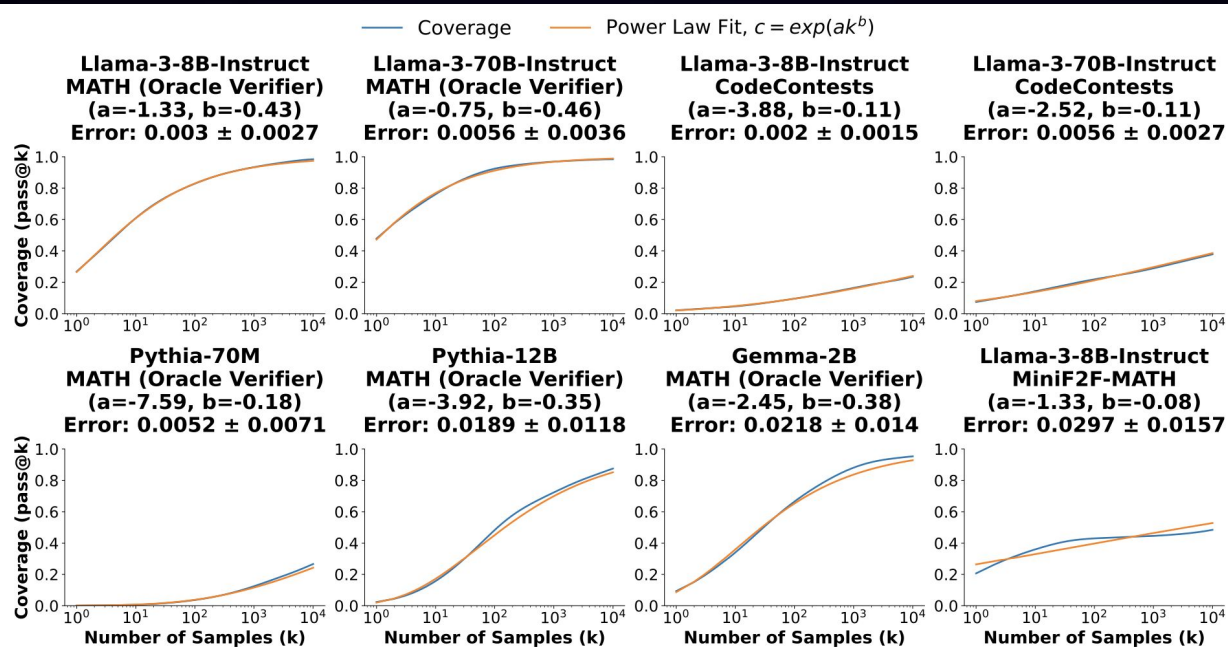


Figure 5: The relationship between coverage and the number of samples can be modelled with an exponentiated power law for most tasks and models. We highlight that some curves, such as Llama-3-8B-Instruct on MiniF2F-MATH, do not follow this trend closely. We show the mean and standard deviation of the error between the coverage curve and the power law fit across 100 evenly sampled points on the log scale.

$$\log(c) \approx ak^b \quad c \approx \exp(ak^b)$$

Scaling Laws and Coverage Curves

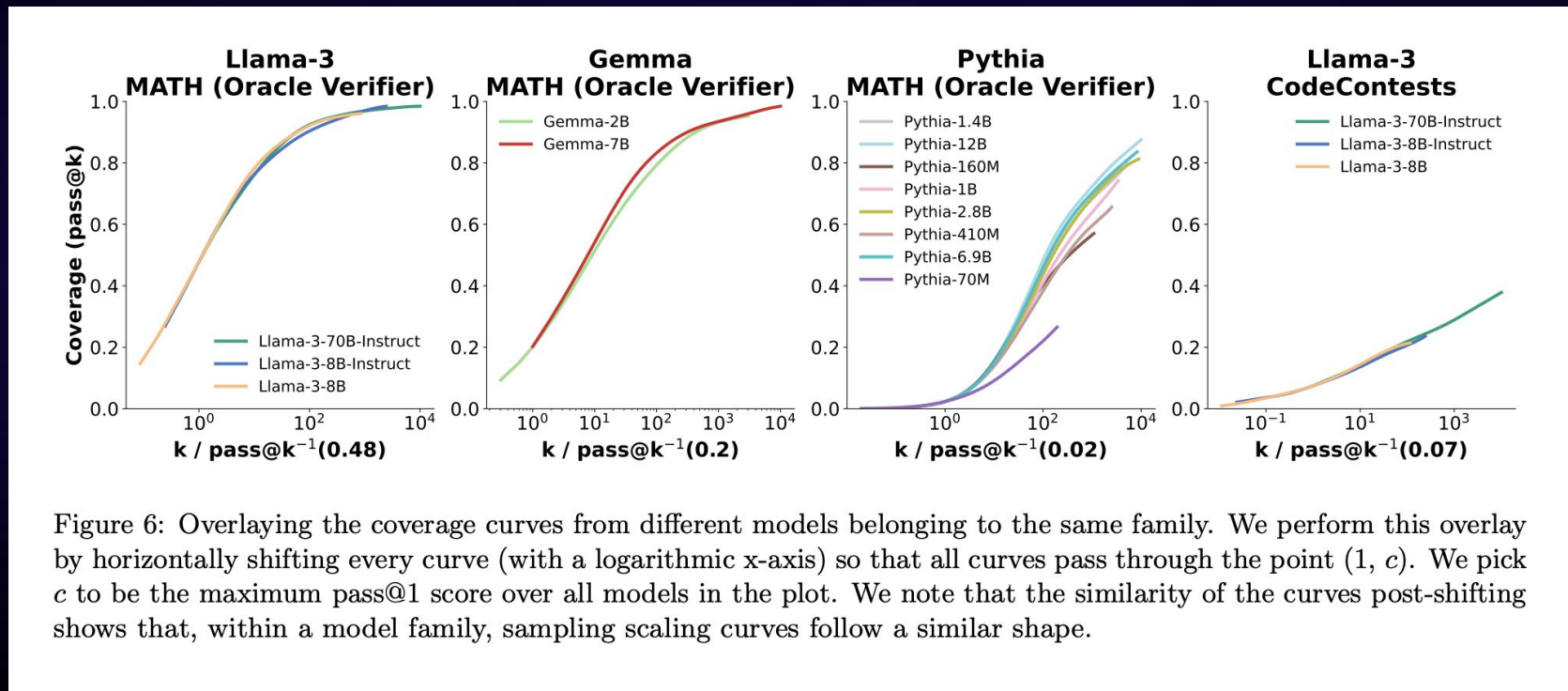


Figure 6: Overlaying the coverage curves from different models belonging to the same family. We perform this overlay by horizontally shifting every curve (with a logarithmic x-axis) so that all curves pass through the point $(1, c)$. We pick c to be the maximum pass@1 score over all models in the plot. We note that the similarity of the curves post-shifting shows that, within a model family, sampling scaling curves follow a similar shape.

Scalability of Verification Methods

GSM8K and MATH lack automatic verification tools

1. *Majority Vote*: We pick the most common final answer
2. *Reward Model + Best-of-N*: We use a reward model to score each solution, and pick the answer from the highest-scoring sample.
3. *Reward Model + Majority Vote*: We calculate a majority vote where each sample is weighted by its reward model score.

Pass@1	# Problems	# CoT Graded	Correct CoT	Incorrect CoT	Incorrect Ground Truth
0-10%	5	15	11	1	1 problem, 3 CoTs
10-25%	10	30	27	3	0 problems
25-75%	29	30	28	2	0 problems
75-100%	84	30	30	0	0 problems

Table 2: Human evaluation of the validity of the Chain-of-Thought reasoning in Llama-3-8B-Instruct answers to GSM8K problems. 3 chains of thought were graded per problem. Even for difficult questions, where the model only gets $\leq 10\%$ of samples correct, the CoTs almost always follow valid logical steps. For the model generations and human labels, [see here](#).

Software Task Verification

“However, tools like unit tests take a black-box approach to verifying a piece of code and are not as comprehensive as methods like proof checkers. These imperfections in the verification process can lead to false positives and/or false negatives that are important to consider.”

when applying repeated sampling

1. Flaky Tests in SWE-bench Lite
2. False Negatives in CodeContests

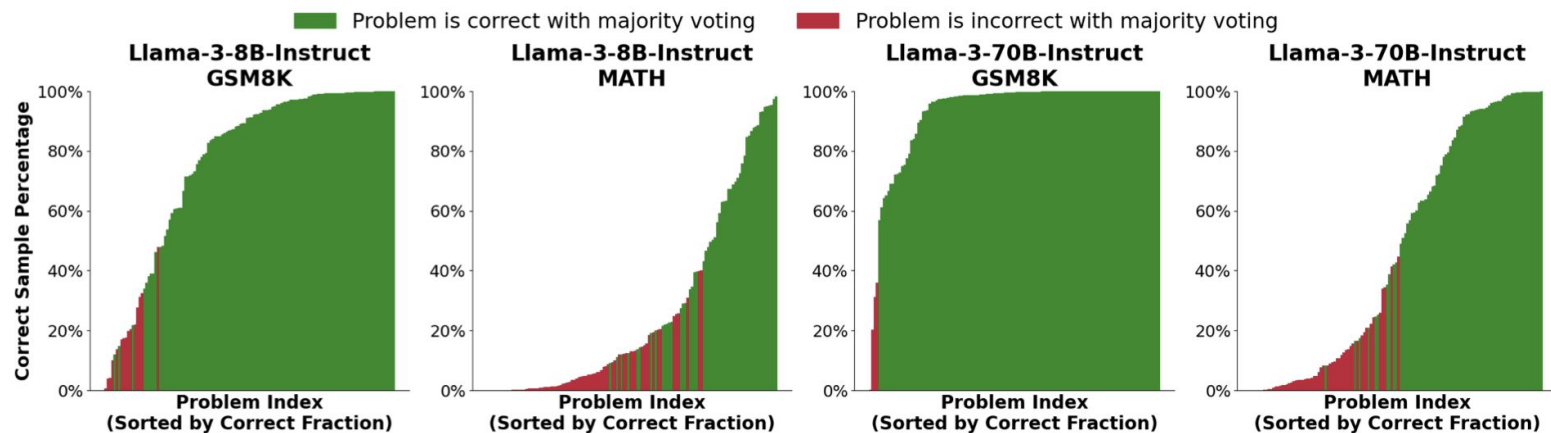


Figure 8: Bar charts showing the fraction of samples (out of 10,000 samples) that are correct, for each problem in the subsets of GSM8K and MATH we evaluate on. There is one bar per problem, and the height of the bar corresponds to the fraction of samples that arrive at the correct answer. Bars are green if self-consistency picked the correct answer and are red otherwise. We highlight that there are many problems with correct solutions, where the correct solutions are sampled infrequently.

Future and Limitations

1. **Solution Diversity:** relying on a positive sampling temperature as the sole mechanism for creating diversity among samples; combining this token-level sampling with other, higher-level approaches may be able to further increase diversity (AlphaCode metadata tags).
2. **Multi-Turn Interactions:** Providing models with execution feedback from these tools should improve solution quality.
3. **Learning From Previous Attempts:** Currently, our experiments fully isolate attempts from each other. Access to existing samples, particularly if verification tools can provide feedback on them, may be helpful when generating future attempts.

Paper 3

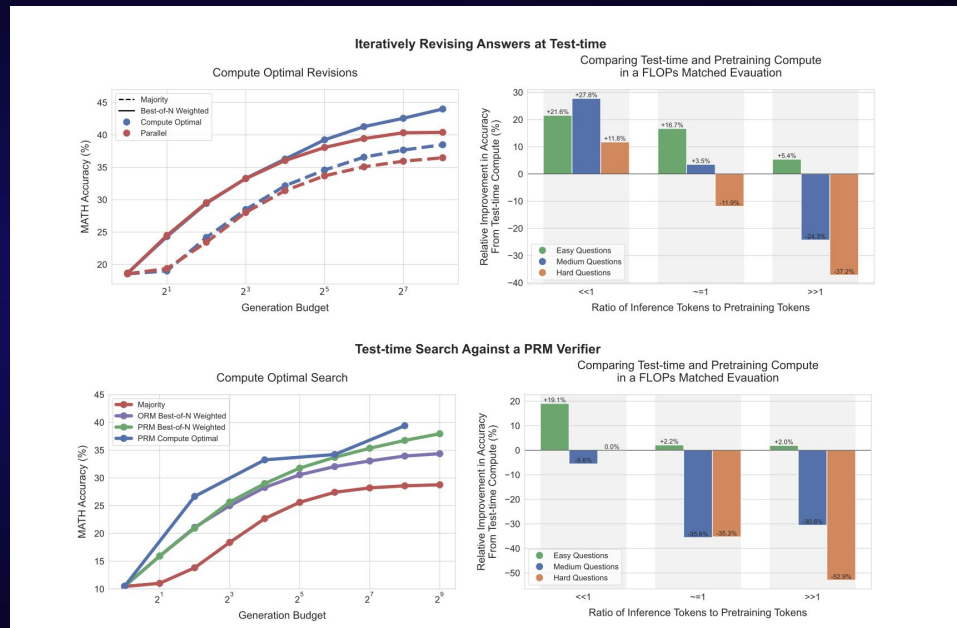
“Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters”

Can we enable language models to most effectively make use of additional computation at test time so as to improve the accuracy of their response?

Two Key Approaches to Scaling Test-Time Compute

1. Searching Against a Verifier (Process-Based Reward Model - PRM): reward model that evaluates and reinforces intermediate reasoning steps rather than just final outputs

2. Updating Model's Distribution Over Response Adaptively



Compute-Optimal Strategy Background

Test-Time Compute Optimal Scaling Strategy: A method for adaptively allocating test-time computation resources to maximize model performance on a given prompt, selecting the most effective strategy based on prompt difficulty.

Model-Predicted Difficulty: A method for estimating the difficulty of a given prompt by using a learned verifier's predicted correctness scores across multiple model-generated responses, rather than relying on ground-truth correctness checks.

Oracle Difficulty: A difficulty estimation approach that categorizes questions based on a model's actual pass rate using ground-truth correctness checks, serving as a more precise but less practical method for determining test-time compute allocation.

Compute-Optimal Strategy

- Adapting Test-Time Compute per Prompt
- Dynamically allocates test-time compute depending on difficulty
- Key finding: Using this adaptive strategy, the model achieves the same performance as a best-of-N approach while using 4× less compute.

Best-of-N: sampling N outputs in parallel from a base LLM and choosing whichever scores highest from a verifier

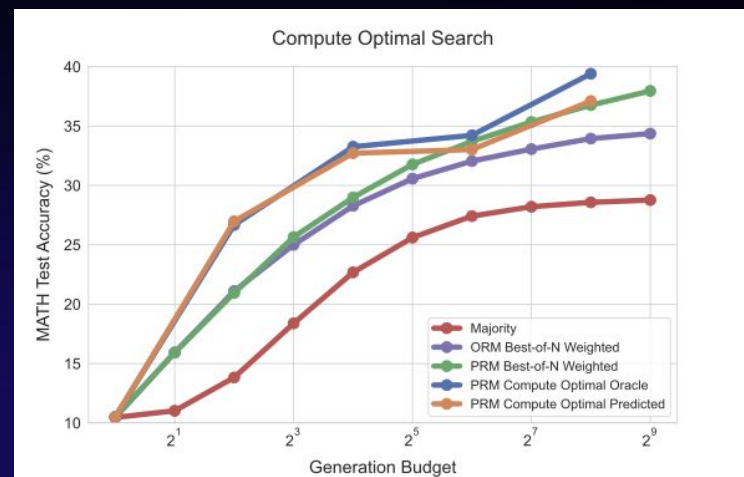


Figure 4 | *Comparing compute-optimal test-time compute allocation against baselines with PRM search.* By scaling test time compute per the notion of question difficulty, we find that we can nearly outperform PRM best-of-N using up to 4× less test-time compute (e.g. 16 verses 64 generations). “**Compute-optimal oracle**” refers to using oracle difficulty bins derived from the groundtruth correctness information, and “**compute-optimal predicted**” refers to using the PRM’s predictions to generate difficulty bins. Observe that the curves with either type of difficulty bins largely overlap with each other.

Experimental Setup

1. Datasets: focus on MATH benchmark
2. Models: Conduct analysis on PaLM-2S*
3. Evaluation: Compare Test-time Compute Scaling vs. Pretraining Scaling

Experimental Results

1. Scaling Test-Time Compute Can Outperform a 14× Larger Model
2. Different Strategies Work Better for Different Difficulty Levels
3. Search-Based Approaches Can Improve Answer Quality
4. Combining Both Methods is Most Effective

Implications

1. Smaller Models and More test-time compute can cause a cheaper deployment without sacrificing the model performance
2. Adaptive test time compute has a more efficient resource use
3. LLMs could iteratively refine their outputs and self-improve with minimal reliance on human supervision

Scaling Search-Based Methods

1. Process-based reward models (PRMs) evaluate solutions step-by-step
2. 3 search methods were compared
 - Best-of-N search
 - Beam Search
 - Lookahead Search: An enhanced beam search that simulates up to k future steps to improve value estimation before selecting the best current step
3. Finding: beam search works well at low compute but over optimization hurts performance at a high compute

Optimizing PRM Via Search Methods

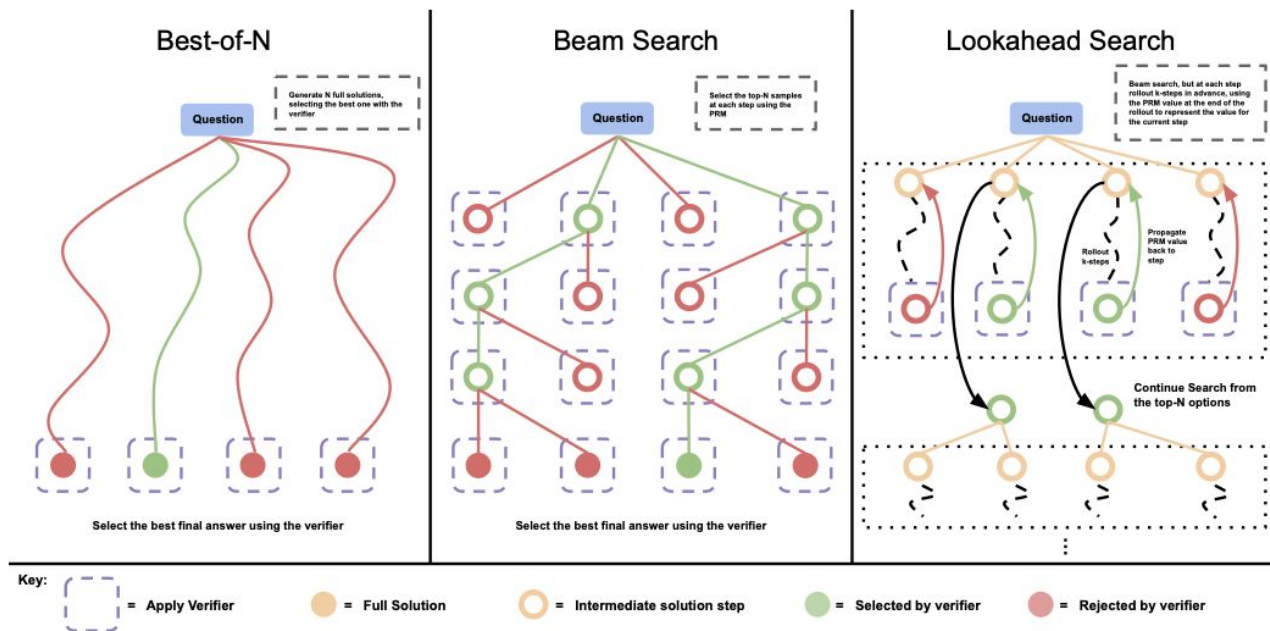


Figure 2 | *Comparing different PRM search methods.* **Left:** Best-of-N samples N full answers and then selects the best answer according to the PRM final score. **Center:** Beam search samples N candidates at each step, and selects the top M according to the PRM to continue the search from. **Right:** lookahead-search extends each step in beam-search to utilize a k-step lookahead while assessing which steps to retain and continue the search from. Thus lookahead-search needs more compute.

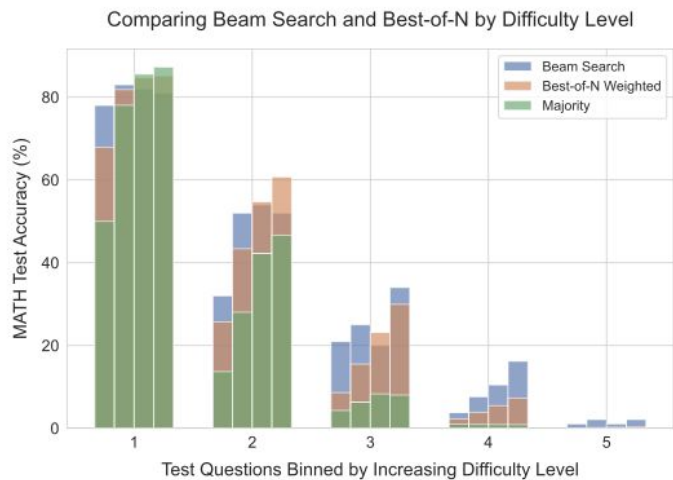
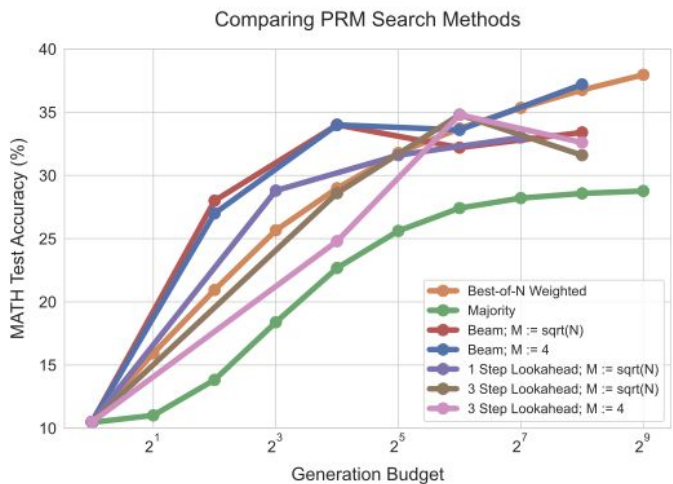


Figure 3 | Left: Comparing different methods for conducting search against PRM verifiers. We see that at low generation budgets, beam search performs best, but as we scale the budget further the improvements diminish, falling below the best-of-N baseline. Lookahead-search generally underperforms other methods at the same generation budget. **Right: Comparing beam search and best-of-N binned by difficulty level.** The four bars in each difficulty bin correspond to increasing test-time compute budgets (4, 16, 64, and 256 generations). On the easier problems (bins 1 and 2), beam search shows signs of over-optimization with higher budgets, whereas best-of-N does not. On the medium difficulty problems (bins 3 and 4), we see beam search demonstrating consistent improvements over best-of-N.

Scaling Iterative Revisions

1. Instead of searching for the best answer, the model refines its own response
2. Finding: Sequential revisions outperform parallel sampling in most cases

Tradeoff between Parallel and Sequential Compute

1. Easy Problems: Sequential revisions are better (small fixes improve accuracy)
2. Hard Problems: Parallel sampling helps (exploring different approaches)

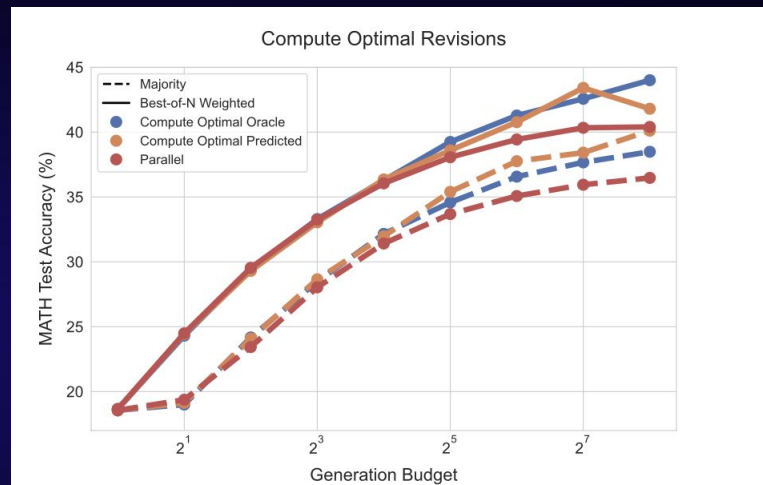


Figure 8 | *Comparing compute-optimal test-time compute allocation against the parallel compute baseline with our revision model.* By optimally scaling test-time compute according to question difficulty, we find that we can outperform best-of-N using up to **4x** less test-time compute (e.g. 64 samples versus 256). “**Compute-optimal oracle**” refers to using the oracle difficulty bins derived from the ground truth correctness information, and “**compute optimal predicted**” refers to using the PRM’s predictions to produce model-predicted difficulty bins.

FLOPs Analysis

- Matching FLOPs for a 14x Smaller Training Model Requires Significant Pre Training Cost

$$X = 6ND_{\text{pretrain}}$$

Model Size

Number of Pre-training Data
Tokens

$$Y = 2ND_{\text{inference}}$$

Number of Tokens Generated at
Inference Time

Pretraining FLOPs vs Test-Time FLOPs

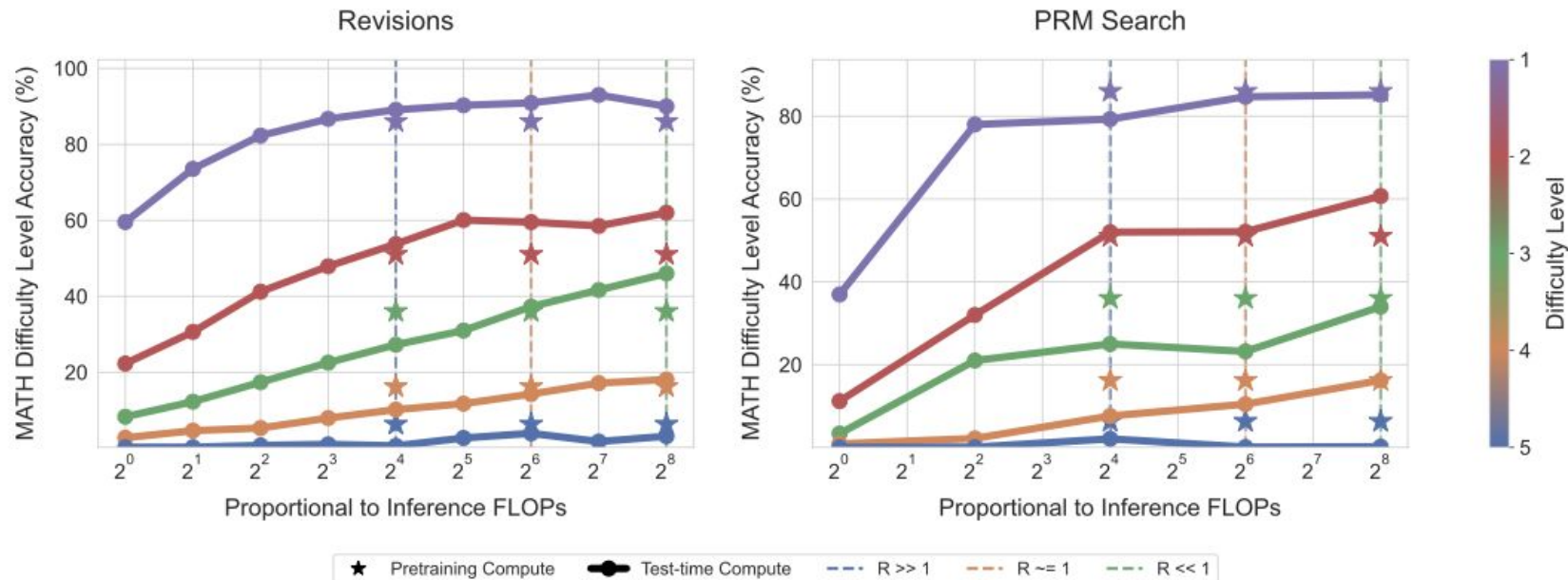
- Pretraining FLOPs: compute used to train a model
- Test-Time FLOPs: compute used during inference
- If pretraining FLOPs are increased -> get a larger model with potentially better raw performance
- If inference FLOPs increased -> model can refine its answers via self-revisions, search, or verification to improve accuracy

FLOPs Tradeoff Between Pre Training and Test-Time Compute

3 Cases:

- $R \gg 1$ (More Inference Tokens) \rightarrow Pre Training is better
- $R \approx 1$ (Balanced inference and pretraining) \rightarrow Test-time compute is effective
- $R \ll 1$ (Few Inference tokens) \rightarrow Test-time compute is best

Comparing Test-time and Pretraining Compute



Limitations and Future Work

- Hard problems still need pretraining
- Better difficulty estimation necessary for adaptive compute
- Combining test-time compute with pre training strategies may yield better results

Conclusions

Questions?