Test-time scaling

CS 5624: Natural Language Processing

Spring 2025

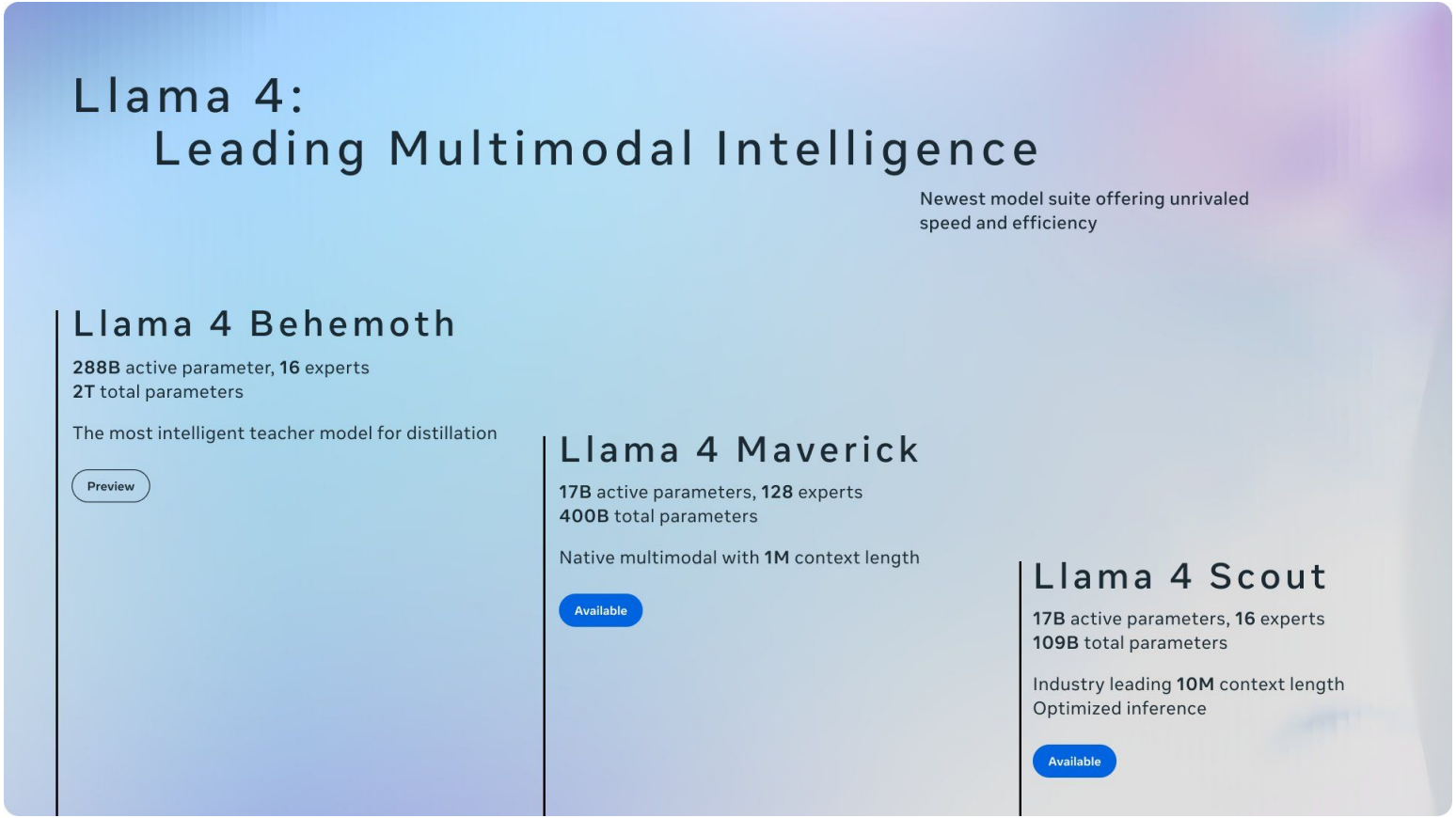
https://tuvllms.github.io/nlp-spring-2025

Tu Vu

VIRGINIA TECH

LLM News:

All available on together.ai

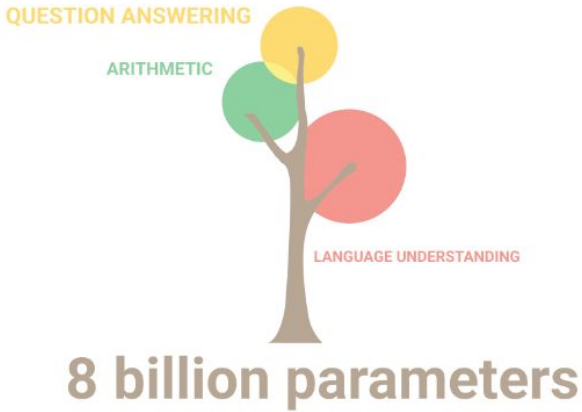


lmarena.ai

Meta's Llama 4 Maverick jumps to #2! the 4th org to score > 1400

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	1	Gemini-2.5-Pro-Exp-03-25	1439	+7/-10	5858	Google	Proprietary
2	5	Llama-4-Maverick-03-26-experimental	1417	+13/-12	2520	Meta	Llama
2	1	ChatGPT-4o-latest (2025-03-26)	1410	+8/-10	4899	OpenAI	Proprietary
2	4	Grok-3-Preview-02-24	1403	+6/-6	12391	xAI	Proprietary
3	2	GPT-4.5-Preview	1398	+5/-7	12312	OpenAI	Proprietary
6	7	Gemini-2.0-Flash-Thinking-Exp-01-21	1380	+4/-4	24298	Google	Proprietary
6	4	Gemini-2.0-Pro-Exp-02-05	1380	+4/-4	20289	Google	Proprietary
6	4	DeepSeek-V3-0324	1369	+10/-10	3526	DeepSeek	MIT
8	5	DeepSeek-R1	1358	+5/-5	14259	DeepSeek	MIT
9	14	Gemini-2.0-Flash-001	1354	+5/-5	20028	Google	Proprietary
9	4	o1-2024-12-17	1351	+5/-4	26722	OpenAI	Proprietary
12	14	Gemma-3-27B-it	1341	+5/-5	8420	Google	Gemma
12	14	Qwen2.5-Max	1340	+6/-3	18906	Alibaba	Proprietary
12	10	o1-preview	1335	+3/-4	33182	OpenAI	Proprietary
15	14	o3-mini-high	1325	+4/-4	15927	OpenAI	Proprietary
15	17	DeepSeek-V3	1318	+4/-5	22835	DeepSeek	DeepSeek
15	21	QwQ-32B	1315	+7/-9	5662	Alibaba	Apache 2.0

Scaling-up train-time compute



From "PaLM: Scaling Language Modeling with Pathways" by Chowdhery et al. (2022)

Test-time scaling

- Uses extra test-time compute to improve performance

Discussion: pros and cons of test-time scaling

-

Test-time scaling methods

- **Parallel (repeated sampling)**
  - multiple solution attempts (run independently)
  - chooses the most frequent or the best response
  - Brown et al. (2024); Irvine et al. (2023); Levi (2024)
- **Sequential**
  - later computations depend on earlier ones (e.g., a long reasoning trace)
  - allows it to refine each attempt based on previous outcomes
  - Muennighoff et al.(2025); Snell et al. (2024); Hou et al. (2025); Lee et al. (2025)

Large Language Monkeys: Scaling Inference Compute with Repeated Sampling

Bradley Brown*†‡, Jordan Juravsky*†, Ryan Ehrlich*†, Ronald Clark‡, Quoc V. Le§, Christopher Ré†, and Azalia Mirhoseini†§

†Department of Computer Science, Stanford University
‡University of Oxford
§Google DeepMind

bradley.brown@cs.ox.ac.uk, jbj@stanford.edu, ryanehrlich@cs.stanford.edu, ronald.clark@cs.ox.ac.uk, qvl@google.com, chrismre@stanford.edu, azalia@stanford.edu

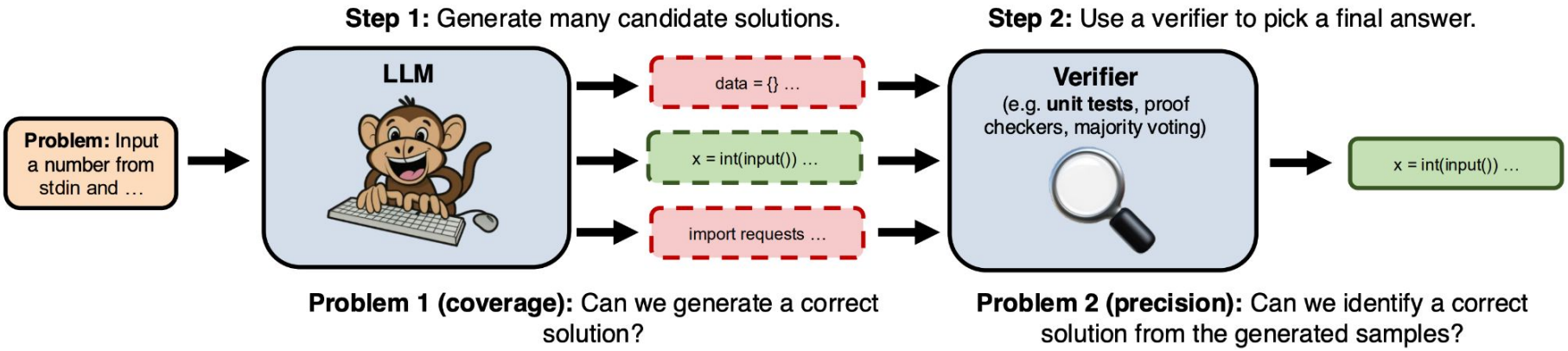


Figure 1: The repeated sampling procedure that we follow in this paper. 1) We generate many independent candidate solutions for a given problem by sampling from an LLM with a positive temperature. 2) We use a domain-specific verifier (ex. unit tests for code) to select a final answer from the generated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

Measuring coverage

$$\text{pass@}k = \frac{1}{\#\text{ of problems}} \sum_{i=1}^{\#\text{ of problems}} \left[1 - \frac{\binom{N-C_i}{k}}{\binom{N}{k}} \right]$$

$$\text{pass@}k = \frac{1}{\#\text{ of problems}} \sum_{i=1}^{\#\text{ of problems}} \left[1 - \frac{\binom{N-C_i}{k}}{\binom{N}{k}}\right]$$

- N: Total number of completions (outputs) generated per problem.
- C_i: Number of **correct** completions among the N for problem i.
- $\binom{N}{k}$: Number of ways to choose k outputs from N.
- $\binom{N-C_i}{k}$: Number of ways to choose k outputs *all of which are incorrect* (i.e., chosen only from the $N - C_i$ incorrect ones).
- So, $\frac{\binom{N-C_i}{k}}{\binom{N}{k}}$ is the **probability** that you choose **only incorrect outputs** when sampling k outputs from the N total.
- Subtracting this from 1 gives the probability that **at least one** correct solution is in the top-k.

Coverage increases as we scale the number of samples

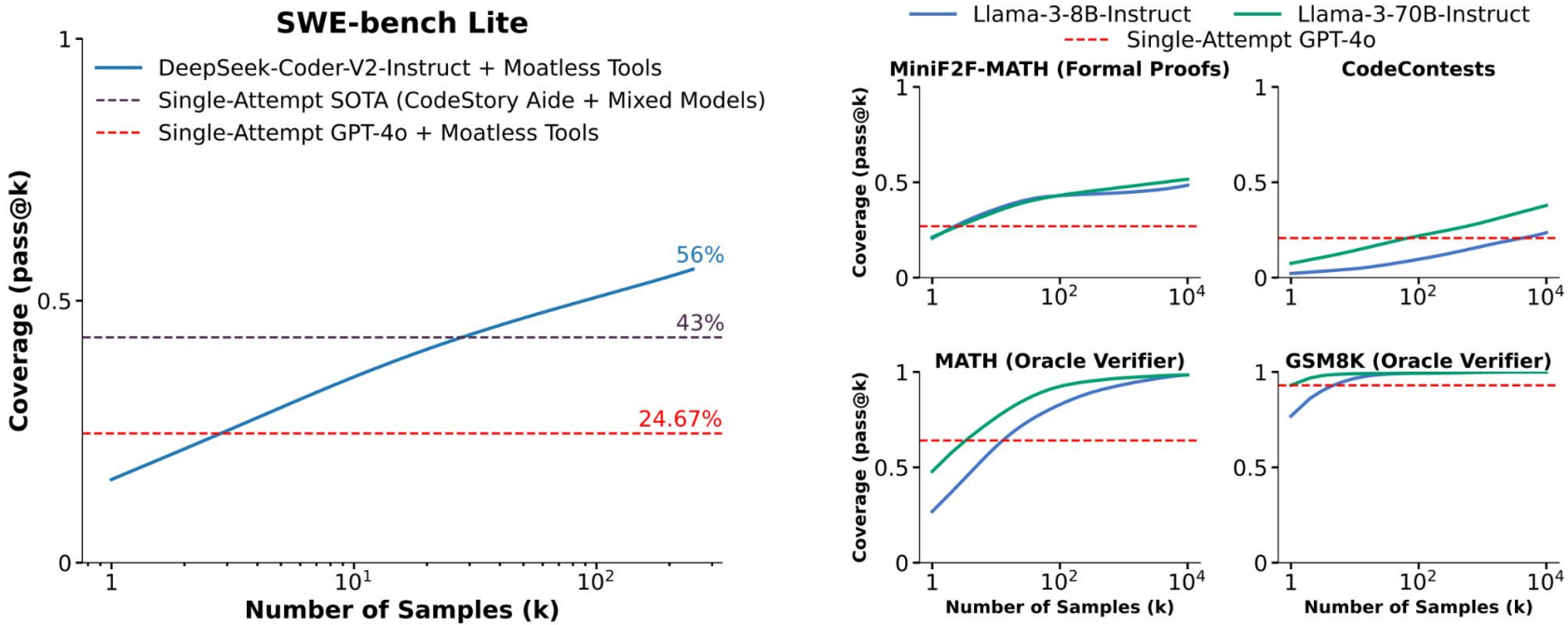


Figure 2: Across five tasks, we find that coverage (the fraction of problems solved by at least one generated sample) increases as we scale the number of samples. Notably, using repeated sampling, we are able to increase the solve rate of an open-source method from 15.9% to 56% on SWE-bench Lite.

Scaling inference time compute via repeated sampling leads to consistent coverage gains

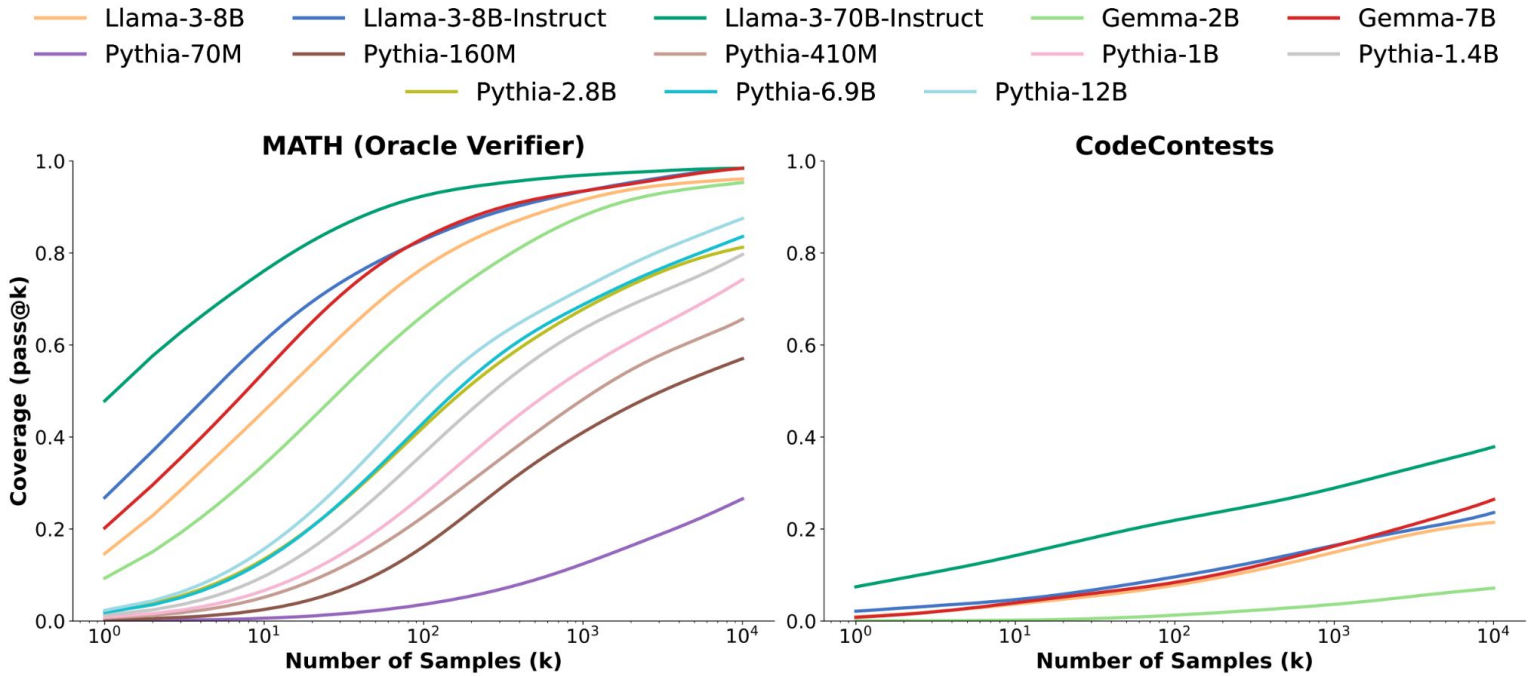


Figure 3: Scaling inference time compute via repeated sampling leads to consistent coverage gains across a variety of model sizes (70M-70B), families (Llama, Gemma and Pythia) and levels of post-training (Base and Instruct models).

Inference FLOPs

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

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

Ideal model size depends on the task, compute budget, and coverage requirements

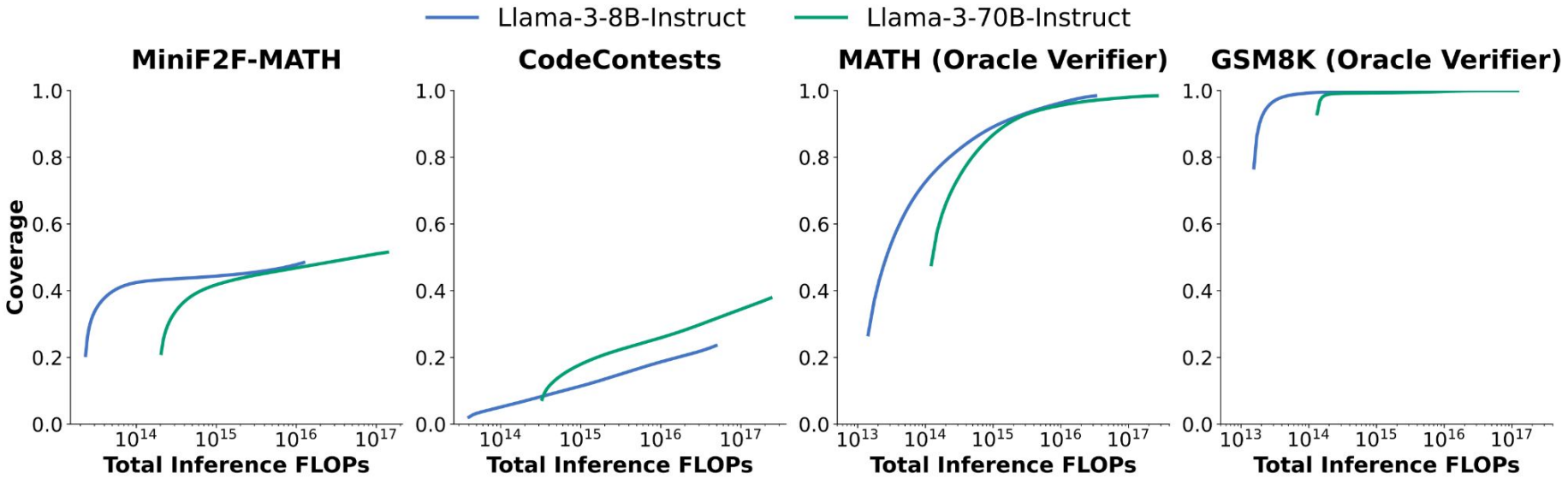


Figure 4: Comparing cost, measured in number of inference FLOPs, and coverage for Llama-3-8B-Instruct and Llama-3-70B-Instruct. We see that the ideal model size depends on the task, compute budget, and coverage requirements. Note that Llama-3-70B-Instruct does not achieve 100% coverage on GSM8K due to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

API cost

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.

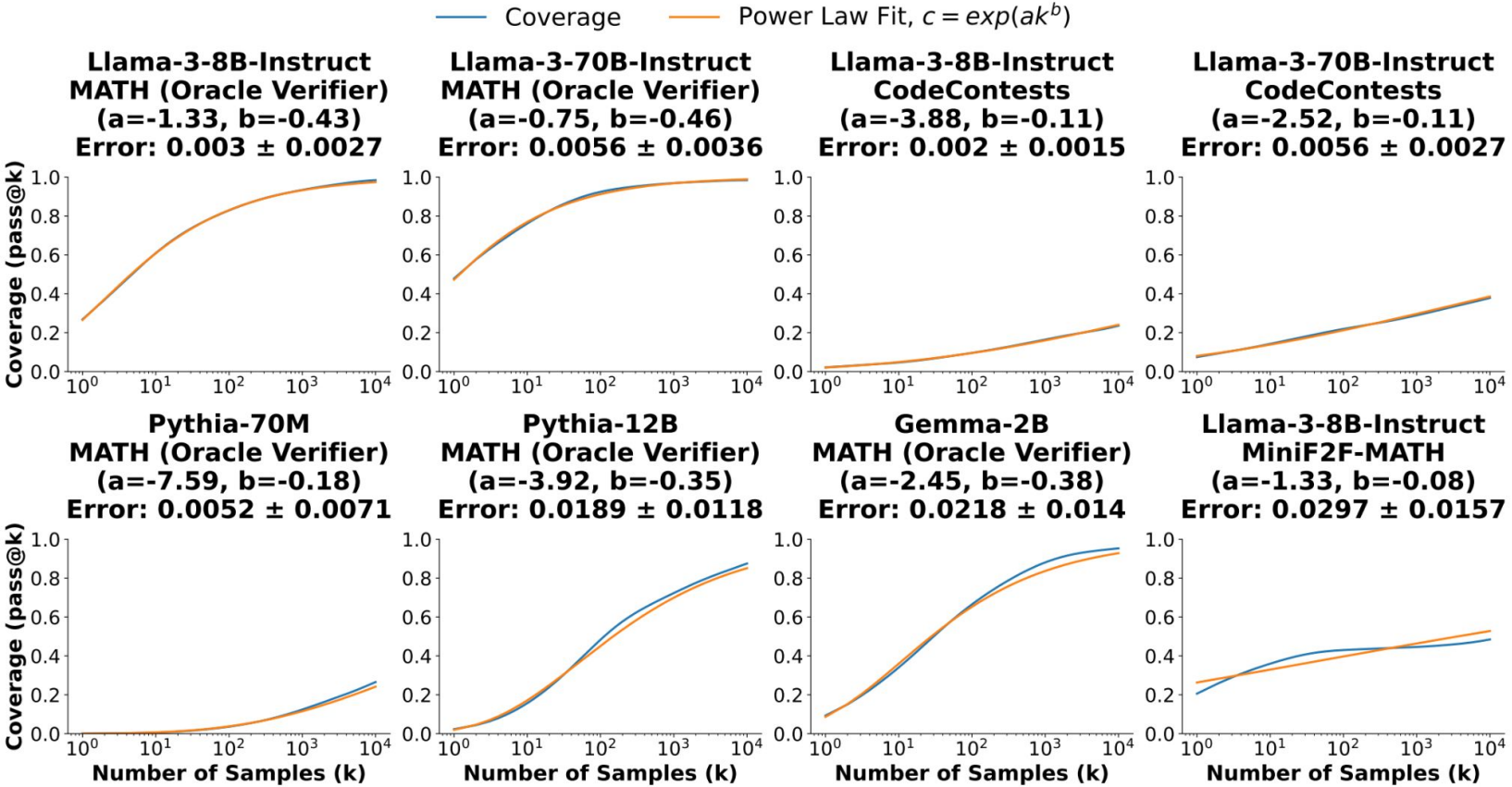


Figure 5: The relationship between coverage and the number of samples can be modelled with an exponentiated power law for most tasks and models. We highlight that some curves, such as Llama-3-8B-Instruct on MiniF2F-MATH, do not follow this trend closely. We show the mean and standard deviation of the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

We model the logarithm of coverage c as a function of the number of samples k:

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

where $a, b \in \mathbb{R}$ are parameters to be fitted.

To directly predict coverage c, we exponentiate both sides:

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

The relationship between coverage and the number of samples modelled with an exponentiated power law

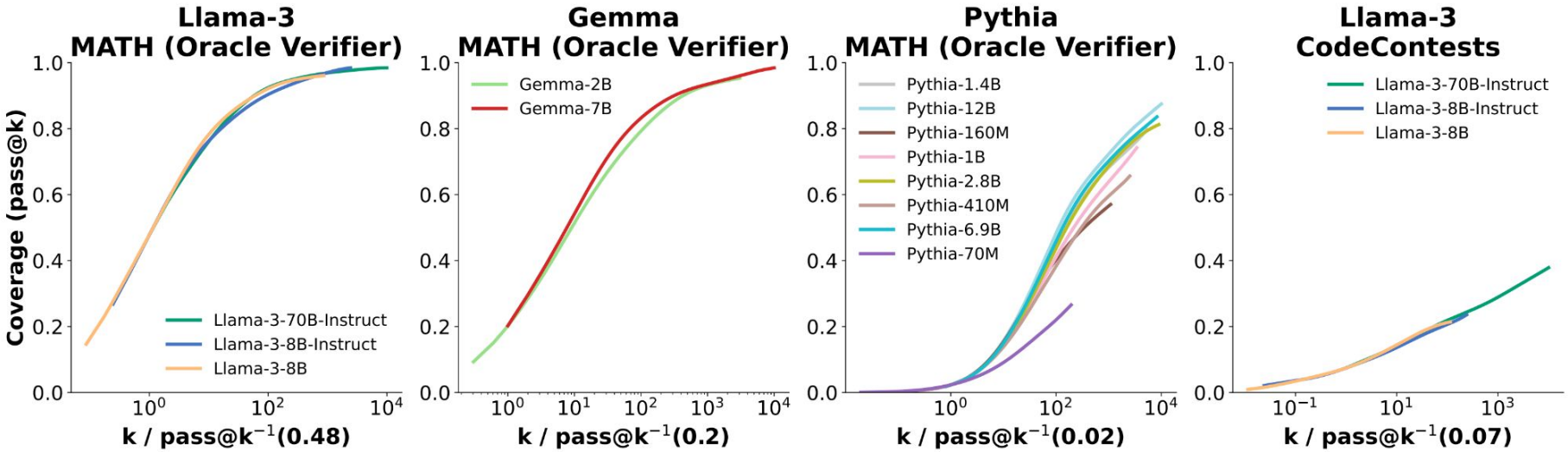


Figure 6: Overlaying the coverage curves from different models belonging to the same family. We perform this overlay by horizontally shifting every curve (with a logarithmic x-axis) so that all curves pass through the point $(1, c)$. We pick c to be the maximum pass@1 score over all models in the plot.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

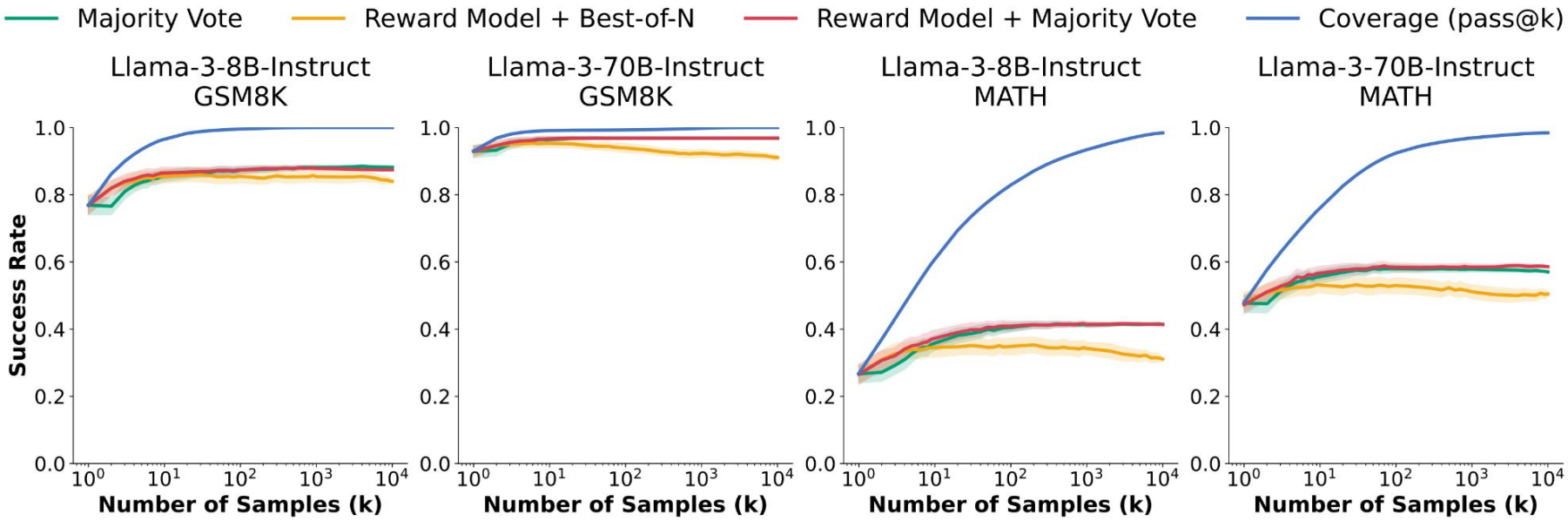


Figure 7: Comparing coverage (performance with an oracle verifier) to mainstream methods available for picking the correct answer (majority voting, reward model selection and reward model majority voting) as we increase the number of samples. Although near-perfect coverage is achieved, all sample selection methods fail to reach the coverage upper bound and saturate before reaching 100 samples. For every k value, we calculate the metric on 100 subsets of size k then plot the mean and one standard deviation across subsets.

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.

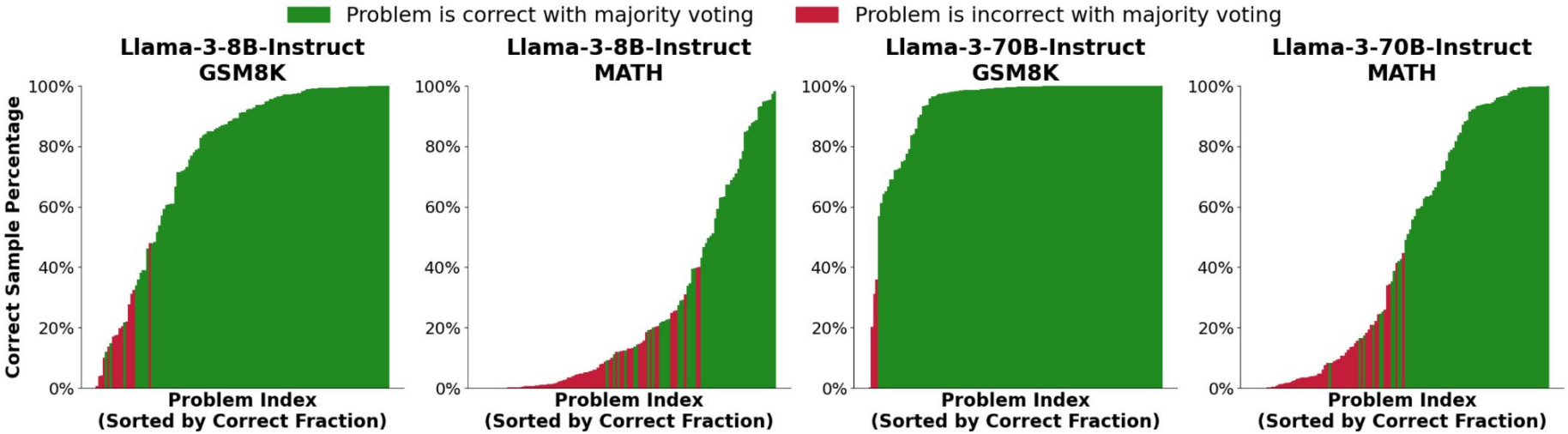


Figure 8: Bar charts showing the fraction of samples (out of 10,000 samples) that are correct, for each problem in the subsets of GSM8K and MATH we evaluate on. There is one bar per problem, and the height of the bar corresponds to the fraction of samples that arrive at the correct answer. Bars are green 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.

s1: Simple test-time scaling

Niklas Muennighoff [*1 3 4] **Zitong Yang** [*1] **Weijia Shi** [*2 3] **Xiang Lisa Li** [*1] **Li Fei-Fei** [1] **Hannaneh Hajishirzi** [2 3]
Luke Zettlemoyer [2] **Percy Liang** [1] **Emmanuel Candès** [1] **Tatsunori Hashimoto** [1]

Test-time scaling with s1-32B

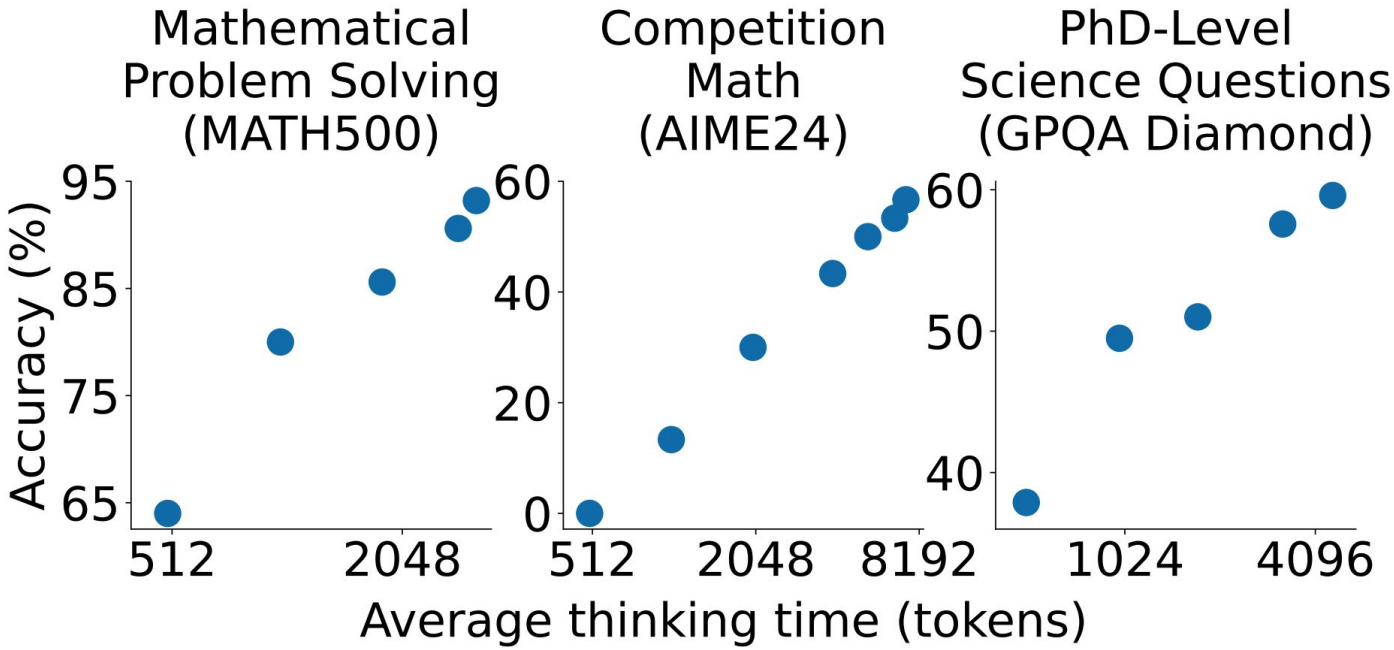
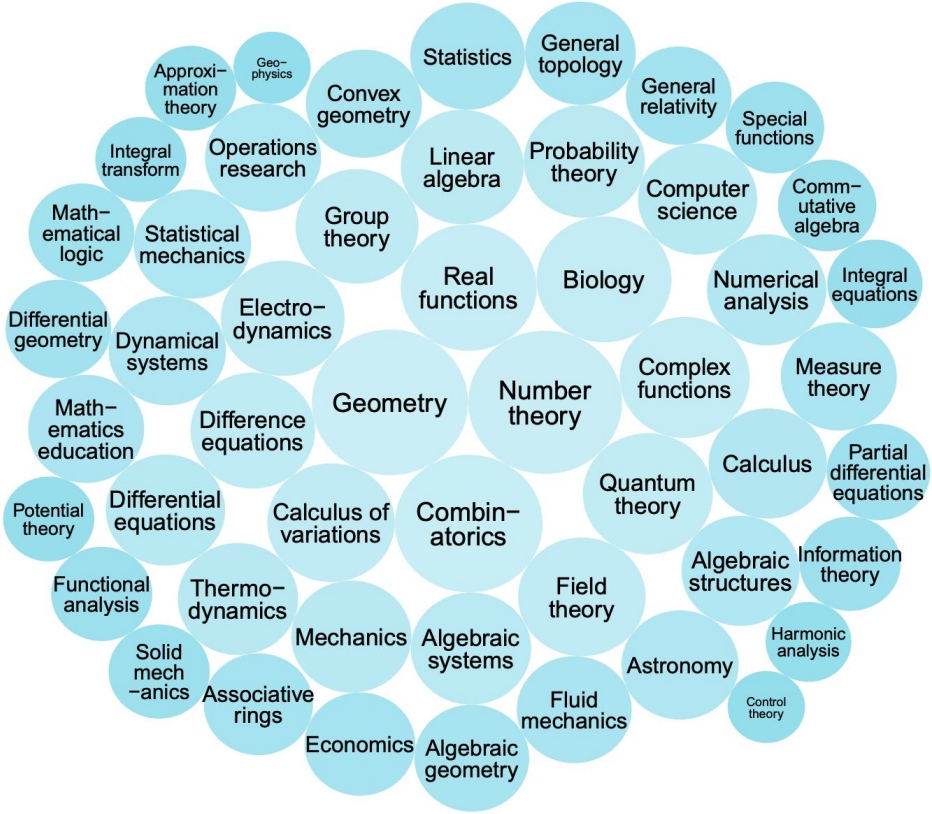
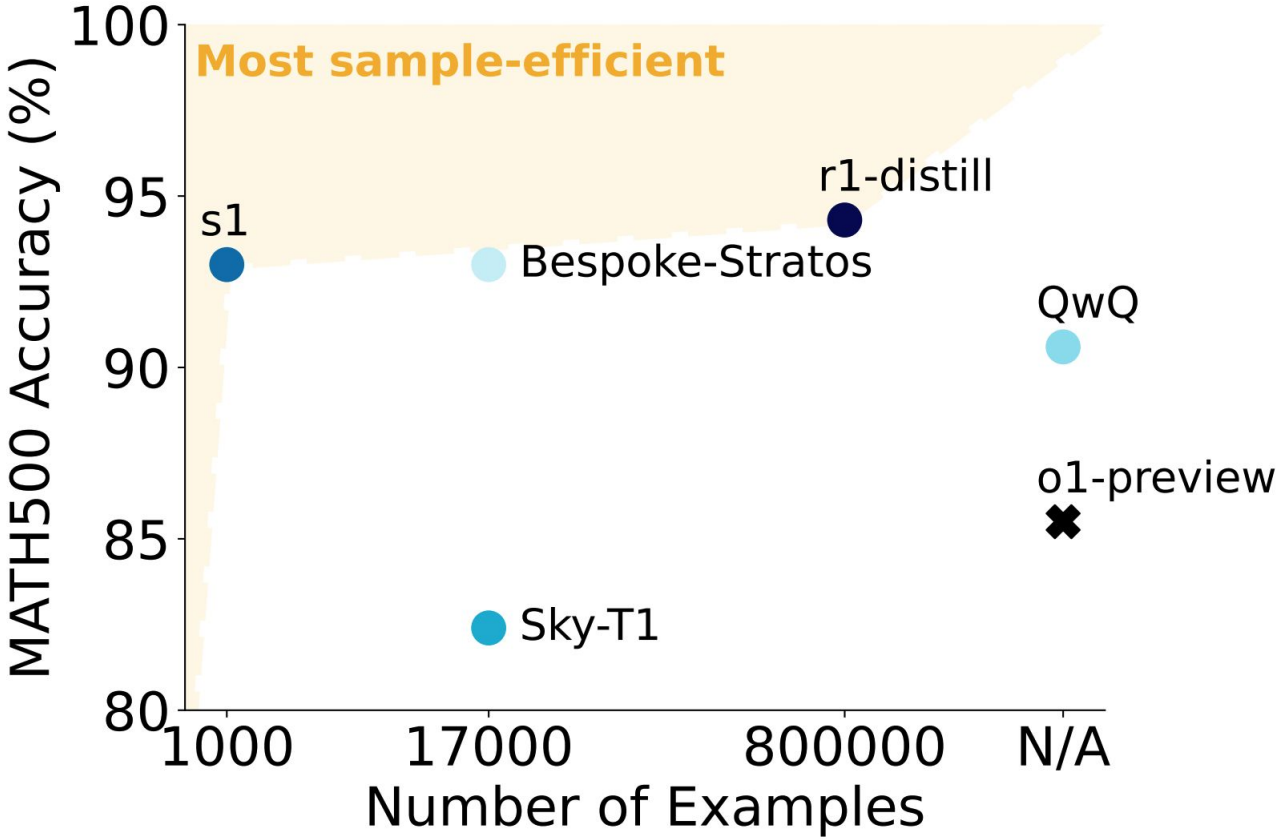


Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

s1K is a dataset of 1,000 high-quality, diverse, and difficult questions with reasoning traces.



s1-32B is on the sample-efficiency frontier



Budget forcing

How many r in raspberry? **Question**

Let's break down the process of counting the letter 'r' in the word "raspberry" ...
* First letter: 'r' - This is an 'r', count = 1.
* Second letter: 'a' - Not an 'r', count remains 1 ...
* Sixth letter: 'e' - Not an 'r', count remains 1.
* Seventh letter: 'r' - This is an 'r', count = 2.
* Eighth letter: 'y' - Not an 'r', count remains 2 ...
The number of 'r's in "raspberry" is 2.
Wait, let's re-read the question carefully. It asks "How many r in raspberry?" ... * r - a - s - p - b - e - r - r - y ... * First 'r' ... * Second 'r' ... * Third 'r' ... Count = 3 ... **Reasoning trace**

My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is $\boxed{3}$ **Response**

For each method under consideration, we evaluate a set of tasks $a \in \mathcal{A}$ by varying the test-time compute on a fixed benchmark (e.g., AIME24). This yields a piecewise linear function f, where the x-axis represents compute (measured in thinking tokens) and the y-axis corresponds to accuracy.

Control

$$\text{Control} = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \mathbb{I}(a_{\min} \leq a \leq a_{\max})$$

This measures the proportion of evaluations whose test-time compute (in thinking tokens) falls within a specified range $[a_{\min}, a_{\max}]$. In practice, we often constrain only $a_{\max}$. As thinking tokens reflect test-time compute, this metric quantifies how well a method supports controllability over computational cost. We report Control as a percentage, with 100% indicating perfect control.

Scaling

$$\text{Scaling} = \frac{1}{\binom{|\mathcal{A}|}{2}} \sum_{\substack{a,b \in \mathcal{A} \\ b > a}} \frac{f(b) - f(a)}{b - a}$$

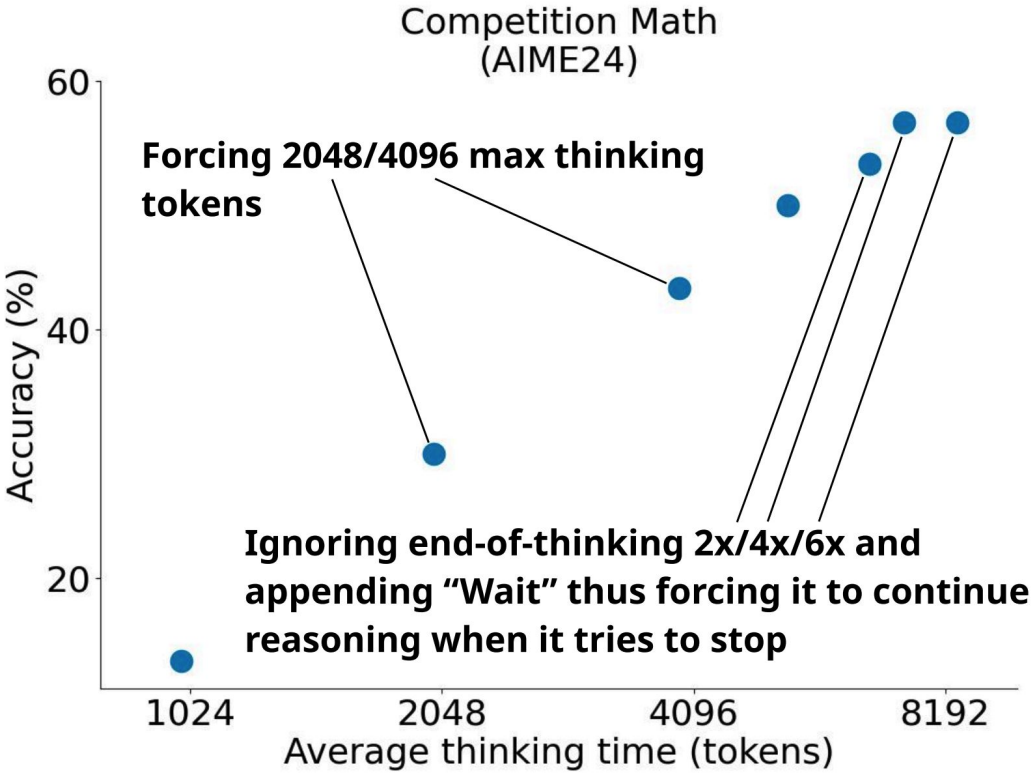
This captures the average slope of the piecewise linear function f, representing the relationship between compute and accuracy. A positive slope is necessary, and higher values indicate better scaling behavior.

Performance

$$\text{Performance} = \max_{a \in \mathcal{A}} f(a)$$

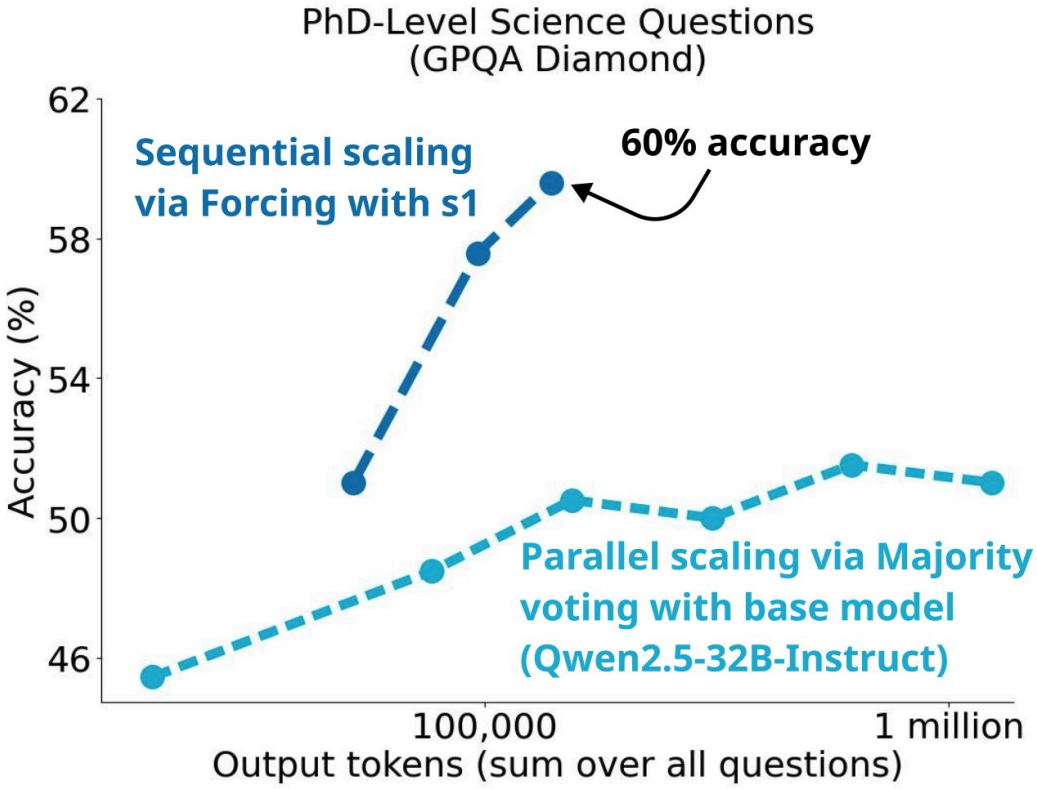
This is the highest accuracy achieved across all levels of test-time compute. In the limit, a method with consistently positive scaling will approach 100% performance.

Budget forcing shows clear scaling trends and extrapolates to some extent



(a) Sequential scaling via budget forcing

Parallel scaling via majority voting



(b) Parallel scaling via majority voting

s1-32B is a strong open reasoning model

Model	# ex.	AIME 2024	MATH 500	GPQA Diamond
API only				
o1-preview	N.A.	44.6	85.5	73.3
o1-mini	N.A.	70.0	90.0	60.0
o1	N.A.	**74.4**	**94.8**	**77.3**
Gemini 2.0 Flash Think.	N.A.	60.0	N.A.	N.A.
Open Weights				
Qwen2.5-32B-Instruct	N.A.	26.7	84.0	49.0
QwQ-32B	N.A.	50.0	90.6	54.5
r1	≫800K	**79.8**	**97.3**	**71.5**
r1-distill	800K	72.6	94.3	62.1
Open Weights and Open Data				
Sky-T1	17K	43.3	82.4	56.8
Bespoke-32B	17K	**63.3**	93.0	58.1
s1 w/o BF	**1K**	50.0	92.6	56.6
s1-32B	**1K**	56.7	**93.0**	**59.6**

s1K data ablations

Model	AIME 2024	MATH 500	GPQA Diamond
1K-random	36.7 [-26.7%, -3.3%]	90.6 [-4.8%, 0.0%]	52.0 [-12.6%, 2.5%]
1K-diverse	26.7 [-40.0%, -10.0%]	91.2 [-4.0%, 0.2%]	54.6 [-10.1%, 5.1%]
1K-longest	33.3 [-36.7%, 0.0%]	90.4 [-5.0%, -0.2%]	59.6 [-5.1%, 10.1%]
59K-full	53.3 [-13.3%, 20.0%]	92.8 [-2.6%, 2.2%]	58.1 [-6.6%, 8.6%]
s1K	50.0	93.0	57.6

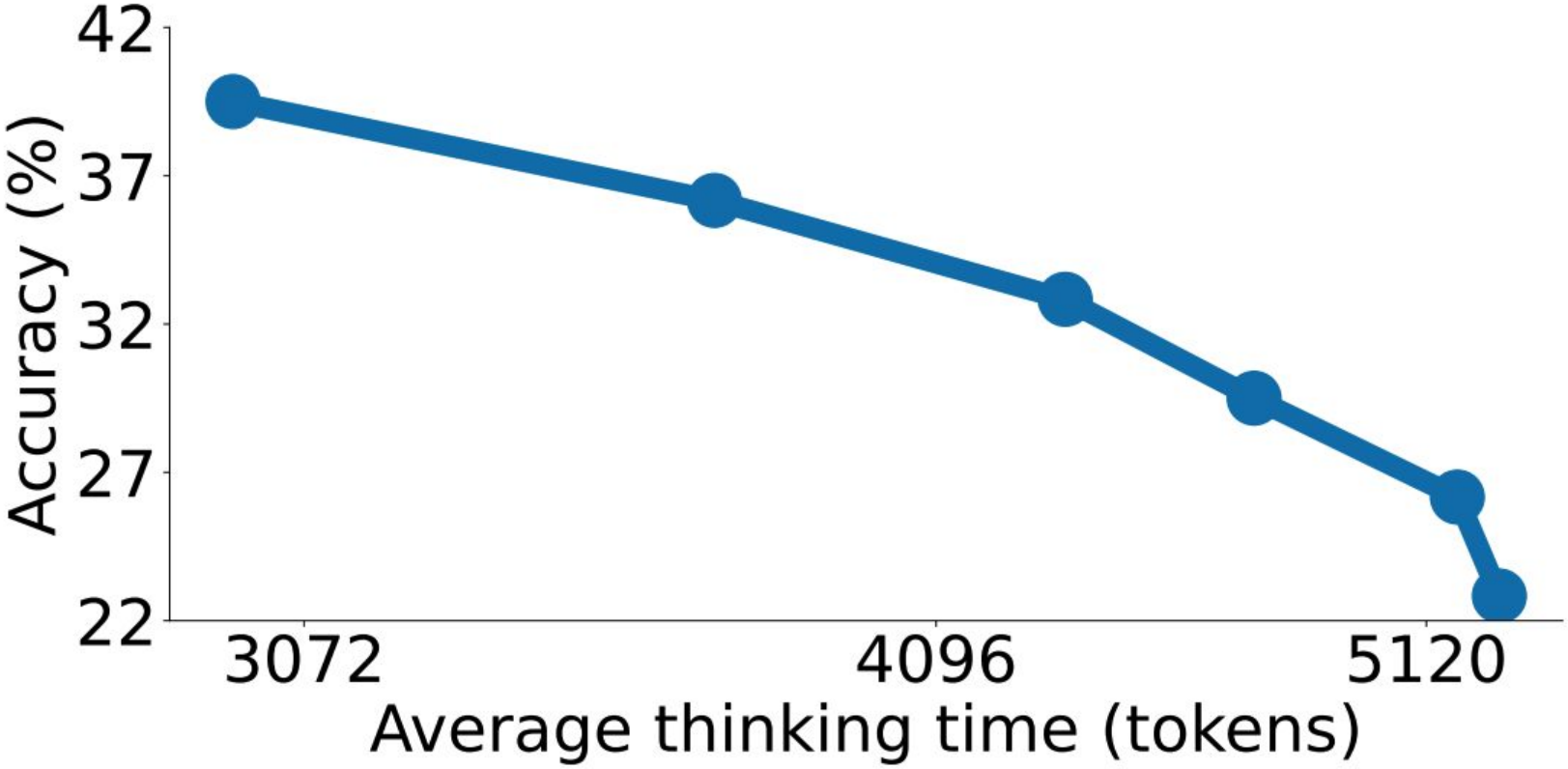
Ablations on methods to scale test-time compute

Method	Control	Scaling	Performance	$\lvert\mathcal{A}\rvert$
BF	**100%**	15	**56.7**	5
TCC	40%	-24	40.0	5
TCC + BF	**100%**	13	40.0	5
SCC	60%	3	36.7	5
SCC + BF	**100%**	6	36.7	5
CCC	50%	**25**	36.7	2
RS	**100%**	-35	40.0	5

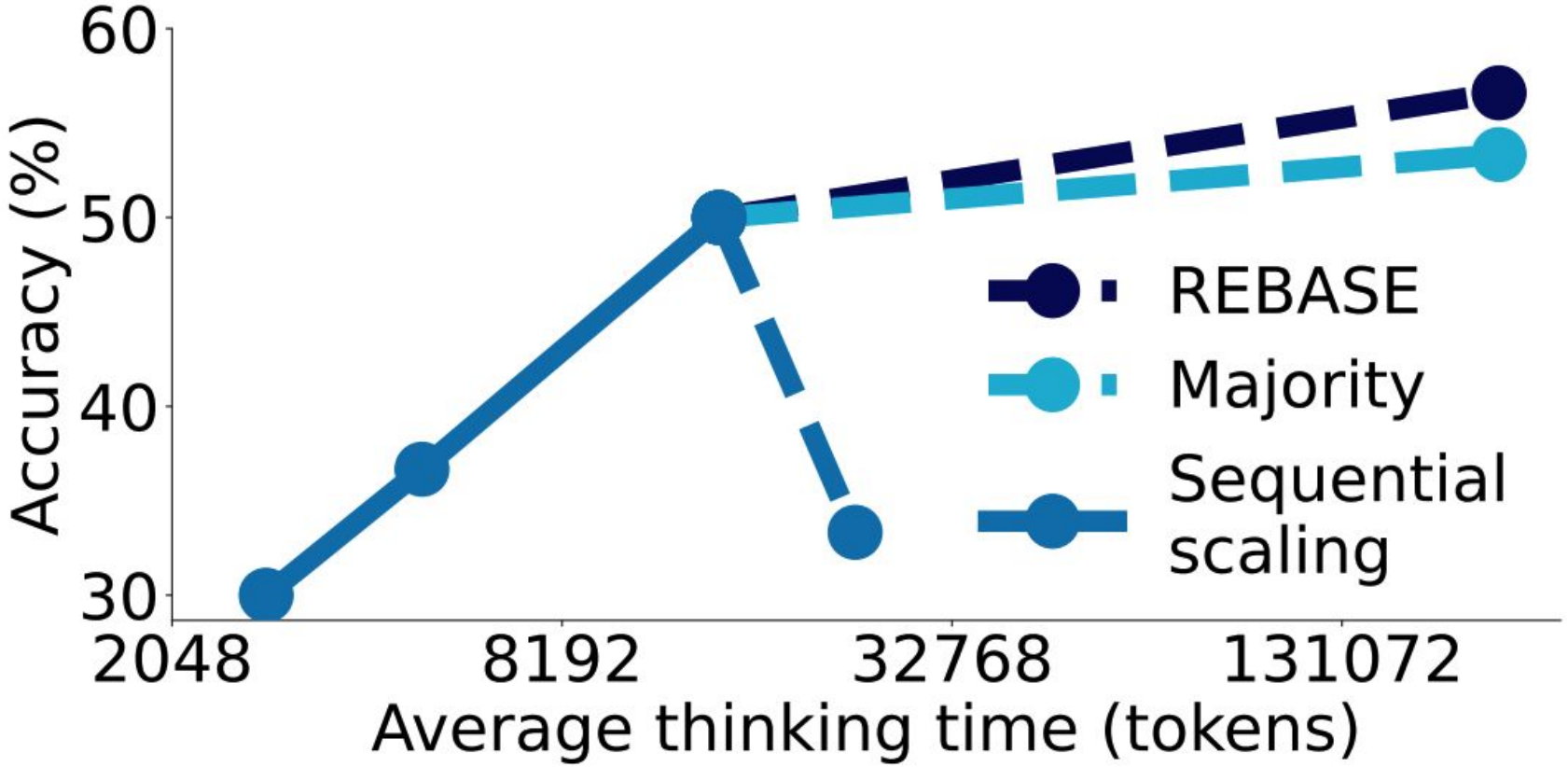
Budget forcing extrapolation ablations

Model	AIME 2024	MATH 500	GPQA Diamond
No extrapolation	50.0	**93.0**	57.6
2x without string	50.0	90.2	55.1
2x “Alternatively”	50.0	92.2	**59.6**
2x “Hmm”	50.0	**93.0**	**59.6**
2x “Wait”	**53.3**	**93.0**	**59.6**

Rejection sampling



Title goes here



Why does supervised fine-tuning on just 1,000 samples lead to such performance gains?

- We hypothesize that the model is already exposed to large amounts of reasoning data during pretraining which spans trillions of tokens.
- Thus, the ability to perform reasoning is already present in our model.
- Our sample-efficient fine-tuning stage just activates it and we scale it further at test time with budget forcing.

Superficial Alignment Hypothesis

- LIMA: Less is more for alignment (Zhou et al., 2023)
  - 1,000 examples can be sufficient to align a model to adhere to user preferences

Thank you!