Zitong Yang

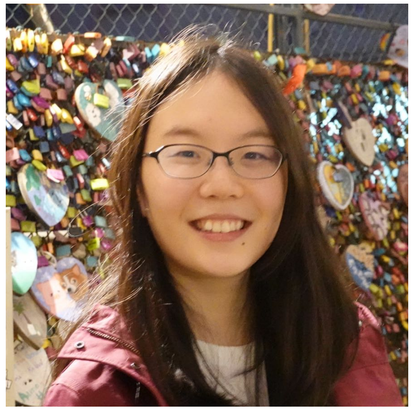
BigML Tech Talk,
Feb. 14, 2025

https://arxiv.org/abs/2501.19393

Collaborators

Niklas Muennighoff*

Weijia Shi*



Xiang Lisa Li*

Li Fei-Fei

Hannaneh Hajishirzi

Luke Zettlemoyer

Percy Liang

Emmanuel Candès

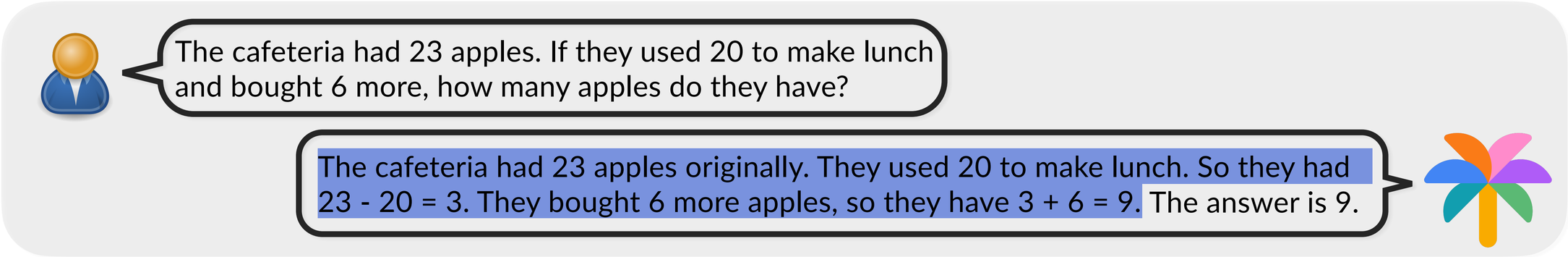
Tatsunori Hashimoto

** Equal contribution*

“History” (before o1) of test-compute scaling

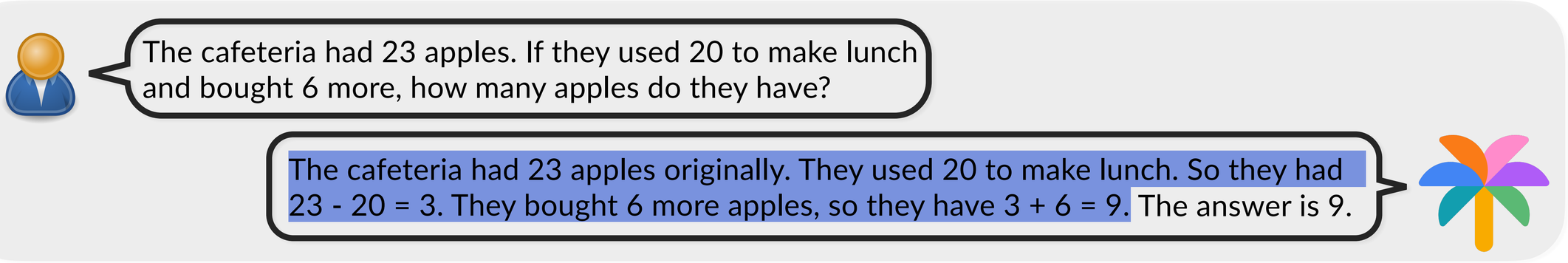
“History” (before o1) of test-compute scaling

- Scratch space: Chain of Thought (Nye et al., 2021; Wei et al., 2022), STaR (Zelikman et al, 2022)

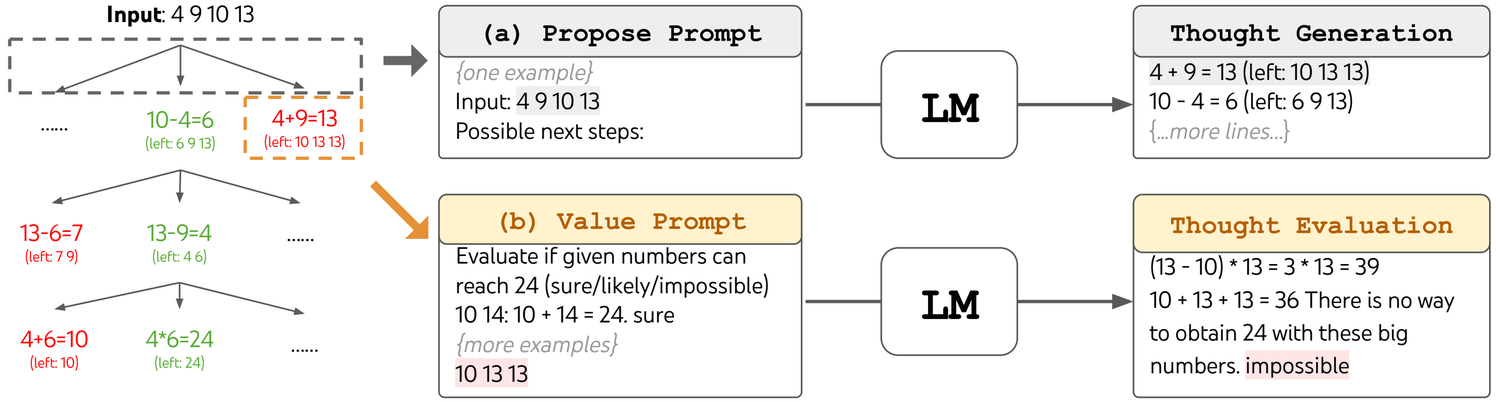


“History” (before o1) of test-compute scaling

- Scratch space: Chain of Thought (Nye et al., 2021; Wei et al., 2022), STaR (Zelikman et al, 2022)

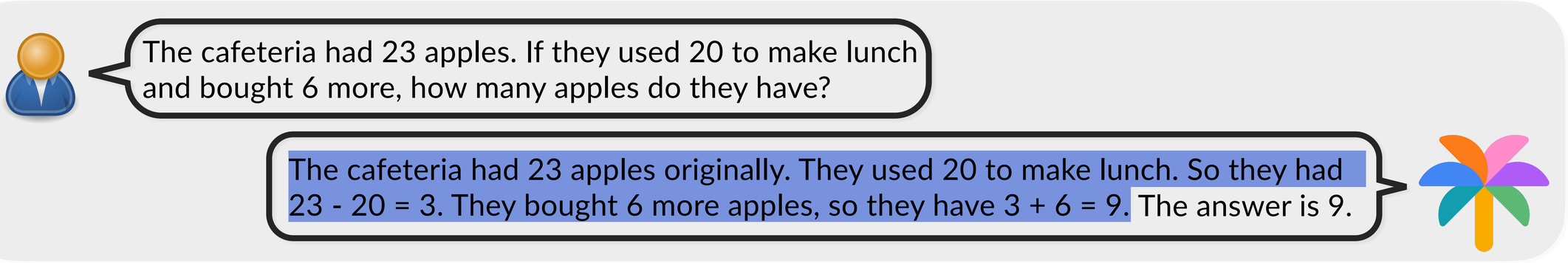


- Verifiers: Tree of thoughts (Yao et al., 2023), Self-critic (2022+)

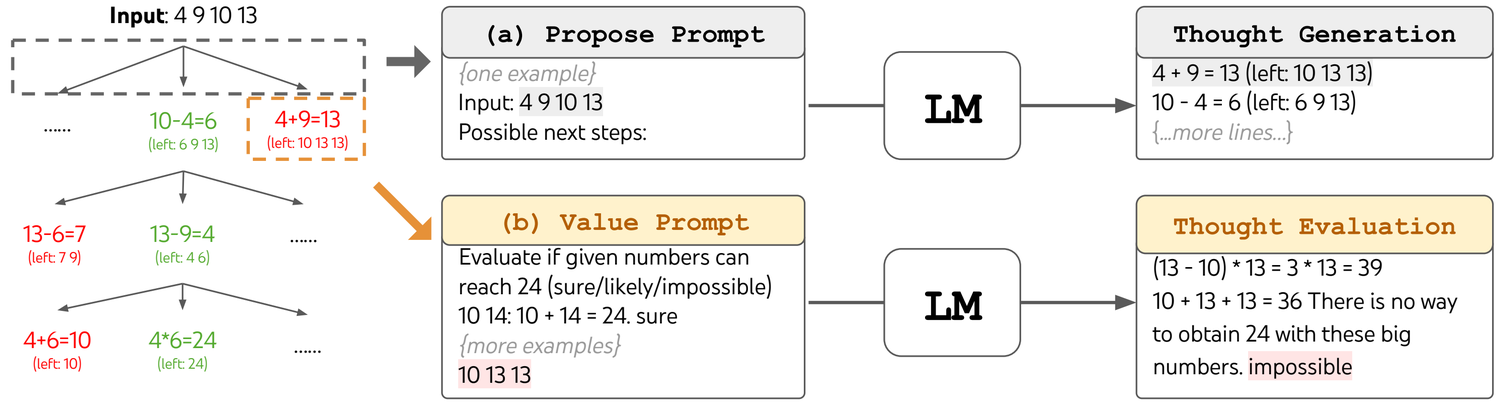


“History” (before o1) of test-compute scaling

- Scratch space: Chain of Thought (Nye et al., 2021; Wei et al., 2022), STaR (Zelikman et al, 2022)



- Verifiers: Tree of thoughts (Yao et al., 2023), Self-critic (2022+)



- Process supervision: PRM800K (Lightman et al., 2023)

OpenAI o1-preview

- On Sep. 12, 2024, OpenAI announced o1-preview

September 12, 2024 Product

Introducing OpenAI o1-preview

A new series of reasoning models for solving hard problems. Available now.

OpenAI o1-preview

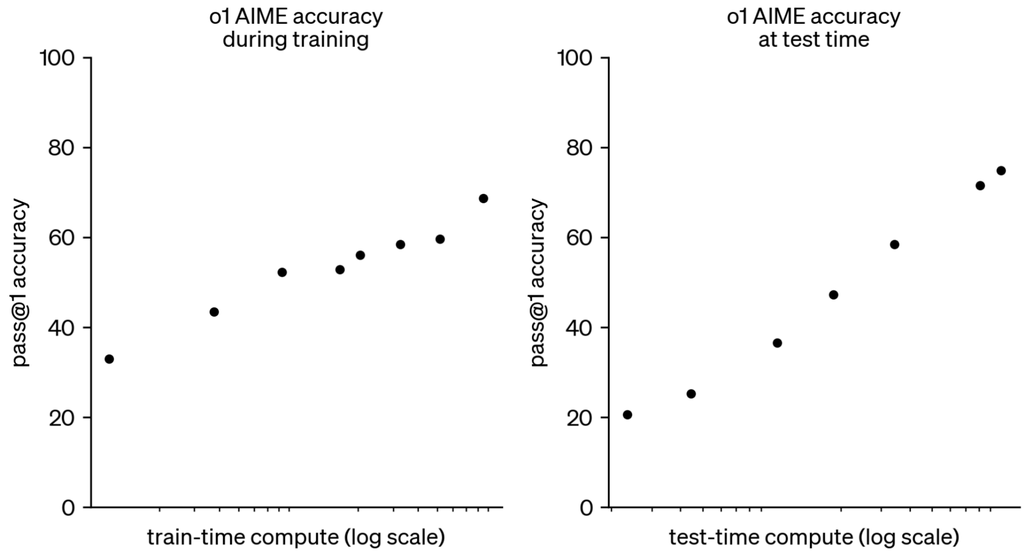
- On Sep. 12, 2024, OpenAI announced o1-preview

September 12, 2024 Product

Introducing OpenAI o1-preview

A new series of reasoning models for solving hard problems. Available now.

- Test-compute scaling: “o1 performance smoothly improves with test-time compute”



OpenAI o1-preview

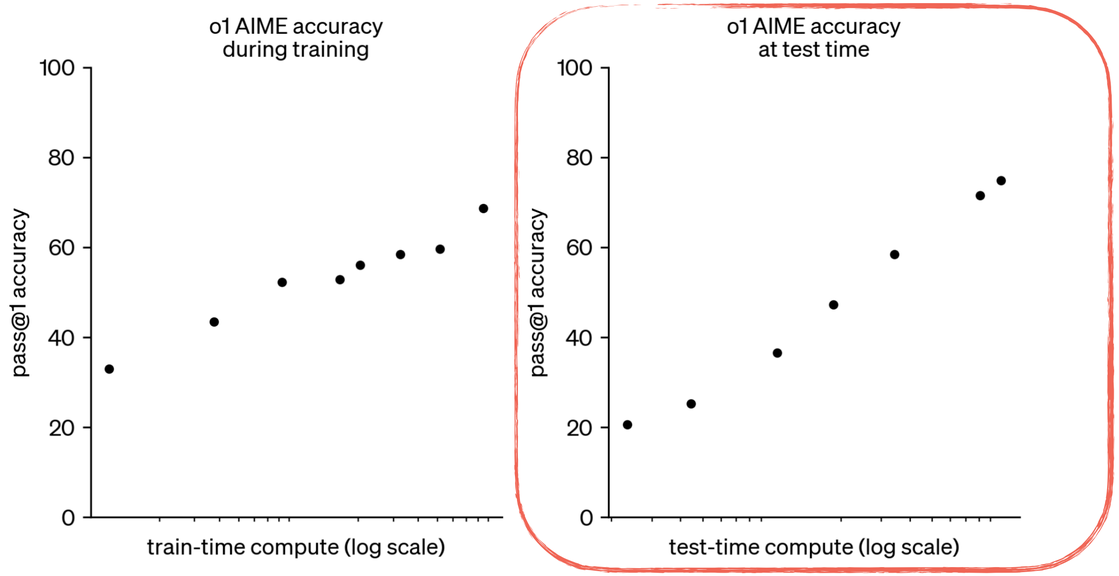
- On Sep. 12, 2024, OpenAI announced o1-preview

September 12, 2024 Product

Introducing OpenAI o1-preview

A new series of reasoning models for solving hard problems. Available now.

- Test-compute scaling: “o1 performance smoothly improves with test-time compute”

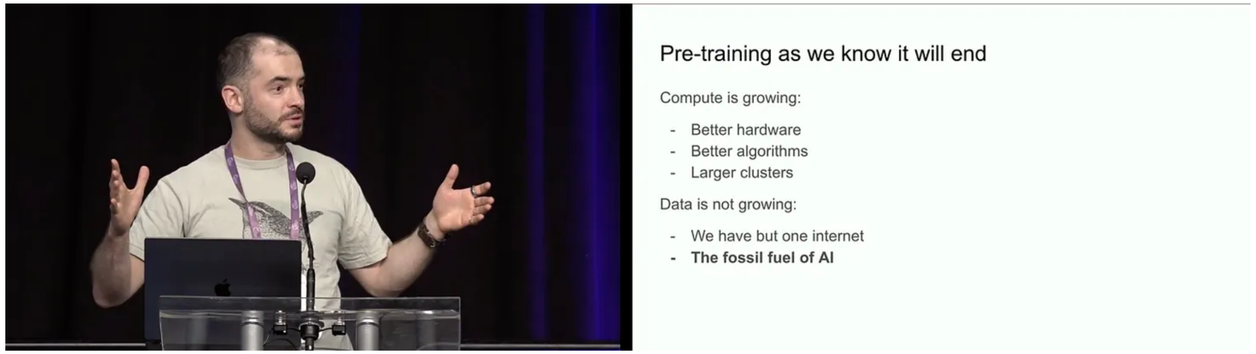


Tatsu Hashimoto Sep 27th, 2024

i think my measurable definition of success is being able to replicate the right side of this plot

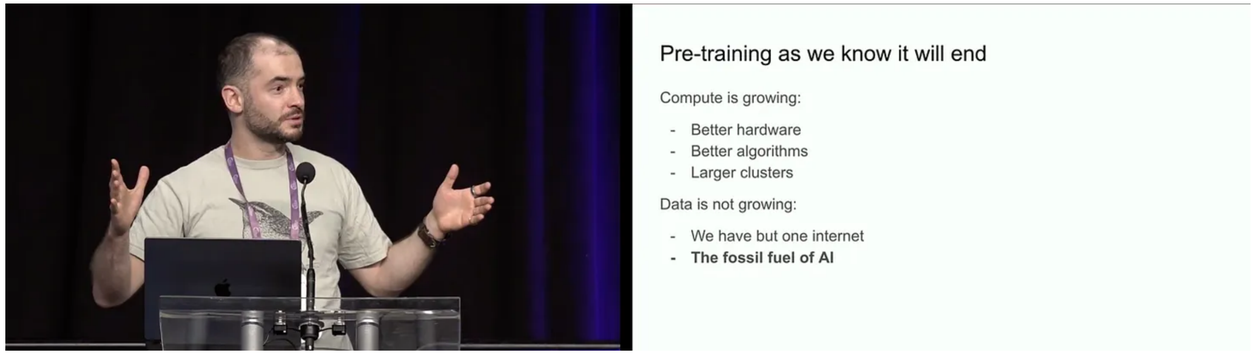
Where does the popularity came from?

- Limitations of data scaling: “we have but one internet”



Where does the popularity came from?

- Limitations of data scaling: “we have but one internet”



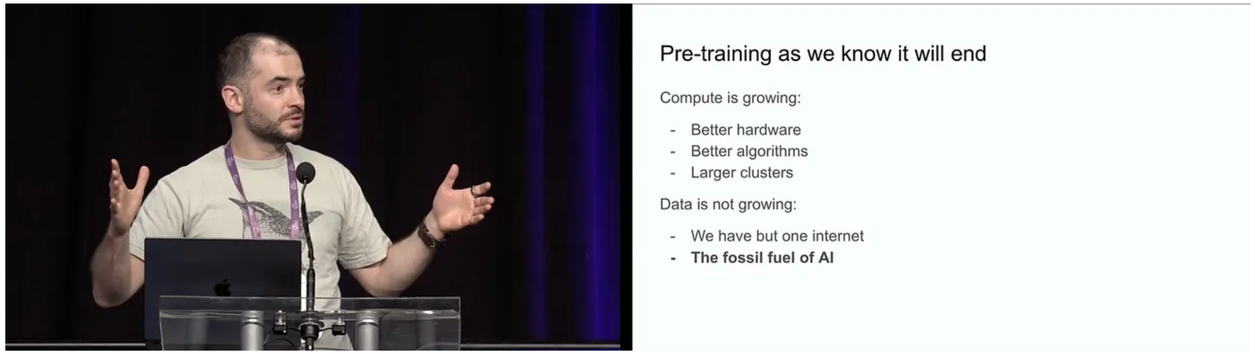
- Dramatic performance improvement on certain benchmarks

On the 2024 AIME exams, GPT-4o only solved on average **12%** (1.8/15) of problems. o1 averaged **74%** (11.1/15) with a single sample per problem…

On MMLU, GPT-4o scores **88.0%** while o1 scores **90.8%**.

Where does the popularity came from?

- Limitations of data scaling: “we have but one internet”



- Dramatic performance improvement on certain benchmarks

On the 2024 AIME exams, GPT-4o only solved on average **12%** (1.8/15) of problems. o1 averaged **74%** (11.1/15) with a single sample per problem…

On MMLU, GPT-4o scores **88.0%** while o1 scores **90.8%**.

- Intriguing CoT patterns: planning, backtracking, self-evaluation, etc.

Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

Our **large-scale reinforcement learning** algorithm teaches the model how to think productively using its chain of thought in a highly **data-efficient** training process.

Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

Our **large-scale reinforcement learning** algorithm teaches the model how to think productively using its chain of thought in a highly **data-efficient** training process.

How large is large-scale? How efficient is data-efficient?

Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

Our **large-scale reinforcement learning** algorithm teaches the model how to think productively using its chain of thought in a highly **data-efficient** training process.

How large is large-scale? How efficient is data-efficient?

- Can we scale the test-time compute indefinitely to solve more and more problems?

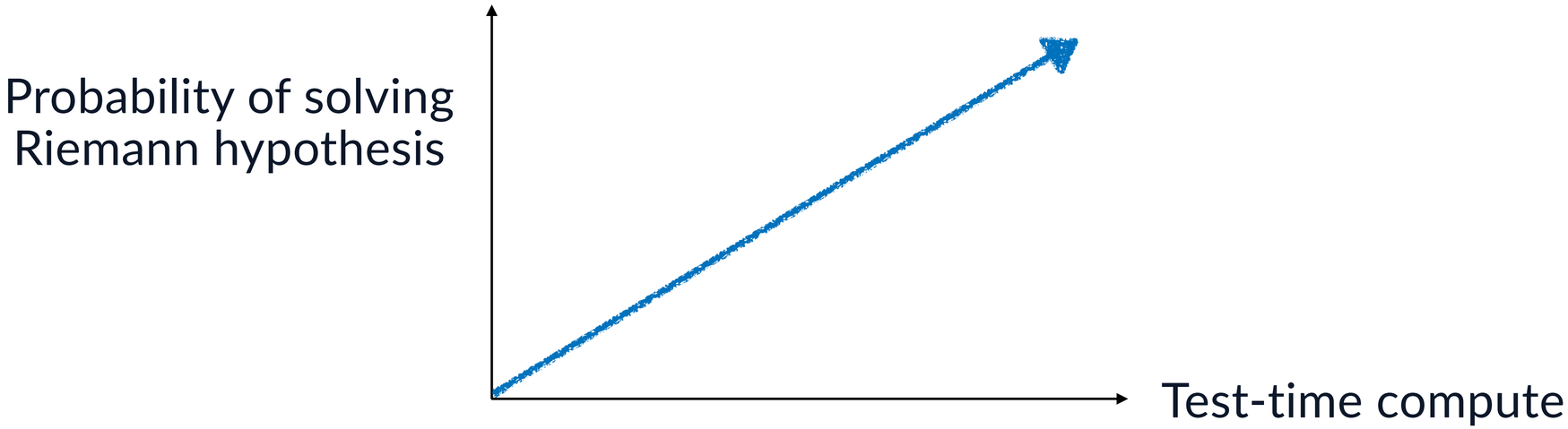
Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

Our **large-scale reinforcement learning** algorithm teaches the model how to think productively using its chain of thought in a highly **data-efficient** training process.

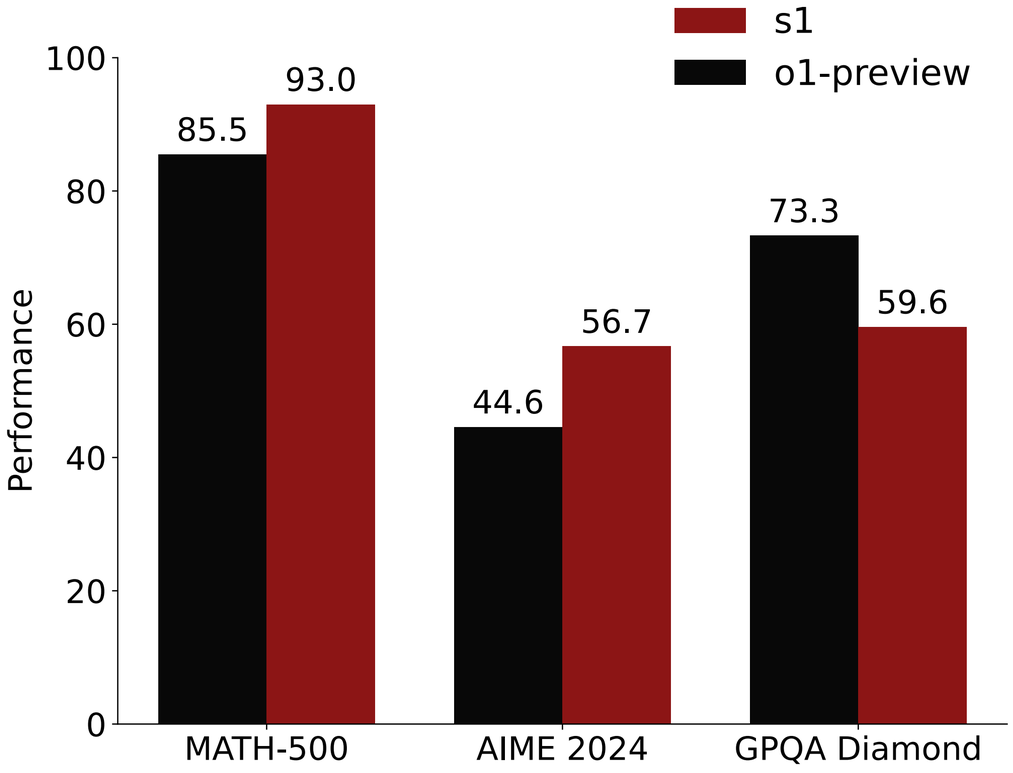
How large is large-scale? How efficient is data-efficient?

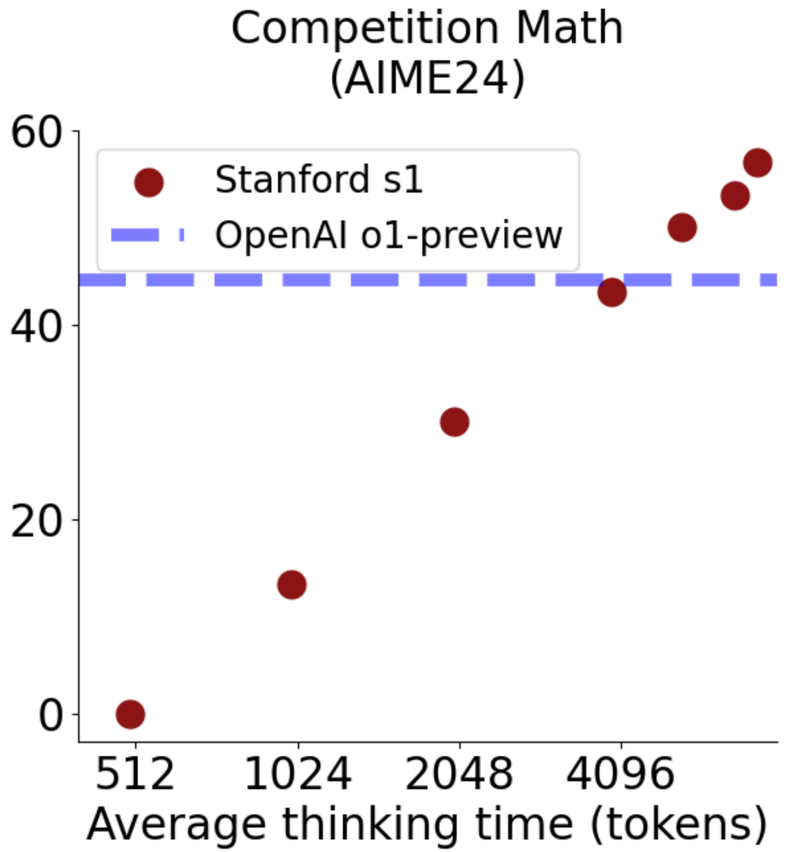
- Can we scale the test-time compute indefinitely to solve more and more problems?



s1

Supervised finetuning on **1,000** chains-of-thoughts delivers o1-preview level capability





Gemini Thinking: source of chains of thoughts

Jeff Dean reposted

Zitong Yang
@ZitongYang0

OMG, this model got the three gambler's problem right (a problem I reserved for testing these reasoning models), it's the first model that got this problem correct, out of o1-preview, o1, r1, QwQ.

Problem:
Consider three gamblers initially having (a, b, c\) dollars. Each trial consists of choosing two players uniformly at random and having them flip a fair coin; they transfer $1 in the usual way. Once players are ruined, they drop out. Let (S_1\) be the number of games required for one player to be ruined. Let (S_2\) be the number of games required for two players to be ruined. Find (\mathbb{E}\{S_1\}\) and (\mathbb{E}\{S_2\}\).

It can first consider the well-known case of two gamblers' ruin, where we consider the martingale X_n^2, and then say X_n^2 is essentially considering X_nY_n where Y_n = N - X_n. From this observation, it generalizes the 3-player game to the construction of X_n*Y_n*Zn.

This intuition is precisely how I solved this problem myself when I worked on it for the first time when preparing for the Stanford statistics' qualifying exam.

Consider the quantity (X(N-X)\) for two players.
Expected increase is 1.
Consider the quantity (ab+ac+bc\). Expected decrease is 1/3.
Consider the quantity (abc\). Expected decrease is ((a+b+c)/3\).

Let's reconsider the problem setup. The game ends when one player is ruined for (S_1\). The game ends when two players are ruined for (S_2\).
The time until one player is ruined is (S_1\).
The time from when one player is ruined until a second player is ruined is the remaining time for (S_2\).

Consider the process until one player is ruined. At this point, we have two players remaining.
Suppose player 3 is ruined first. The amounts are (x, y, 0\) where (x+y = N\).
The expected time for one of these to be ruined is (xy\).
We have (\mathbb{E}[S_2]\) = Expected time until two are ruined.
This involves the cases where (1,2), (1,3), (2,3) are ruined.

Let's use linearity of expectation.
Consider the expected number of games until a specific pair of players interacts such that one of them gets ruined.

Let the amounts be a, b, c.
Consider the quantity (f(x, y, z) = xyz\).
Consider the process until one player is ruined.
(\mathbb{E}[f(X_{S_1}, Y_{S_1}, Z_{S_1})) - f(a, b, c)] = \mathbb{E}\left[\sum_{t=0}^{S_1-1} \Delta f_t \right]\)
(0 - abc = \mathbb{E}\left[\sum_{t=0}^{S_1-1} -\frac{1}{3} (X_t+Y_t+Z_t) \right] = -\frac{N}{3} \mathbb{E}[S_1]\).
This gives (\mathbb{E}[S_1] = \frac{3abc}{N}\).

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

question	gemini_thoughts
..for martingale..	..use Doob's..
..triangle ABC..	..AB is parallel..
.....	.....
..potential wall..	..engien-state..

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

question	gemini_thoughts	is_qwen32b_correct	gemini_length	domain
..for martingale..	..use Doob's..	Yes	8257	probability
..triangle ABC..	..AB is parallel..	No	4320	geometry
.....	.....	.....	.....	.....
..potential wall..	..engien-state..	No	5697	physics

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

Step 3. Remove easy questions.

question	gemini_thoughts	is_qwen32b_correct	gemini_length	domain
..for martingale..	..use Doob's..	Yes	8257	probability
..triangle ABC..	..AB is parallel..	No	4320	geometry
.....	.....	.....	.....	.....
..potential wall..	..engien-state..	No	5697	physics

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

Step 3. Remove easy questions.

question	gemini_thoughts	is_qwen32b_correct	gemini_length	domain
..triangle ABC..	..AB is parallel..	No	4320	geometry
.....	.....	.....	.....	.....
..potential wall..	..engien-state..	No	5697	physics

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

Step 3. Remove easy questions.

- Since we will finetune Qwen2.5-32B-Instruct, we want questions model can't already solve to maximize the 1K budget.

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

Step 3. Remove easy questions. Uniform sampling over domain; Power law sampling in length.

- Since we will finetune Qwen2.5-32B-Instruct, we want questions model can't already solve to maximize the 1K budget.

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

Step 3. Remove easy questions. Uniform sampling over domain; Power law sampling in length.

- Since we will finetune Qwen2.5-32B-Instruct, we want questions model can't already solve to maximize the 1K budget.
- Many questions crowded in a single domain is perhaps less helpful, so touch as many domain as possible.

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

Step 3. Remove easy questions. Uniform sampling over domain; Power law sampling in length.

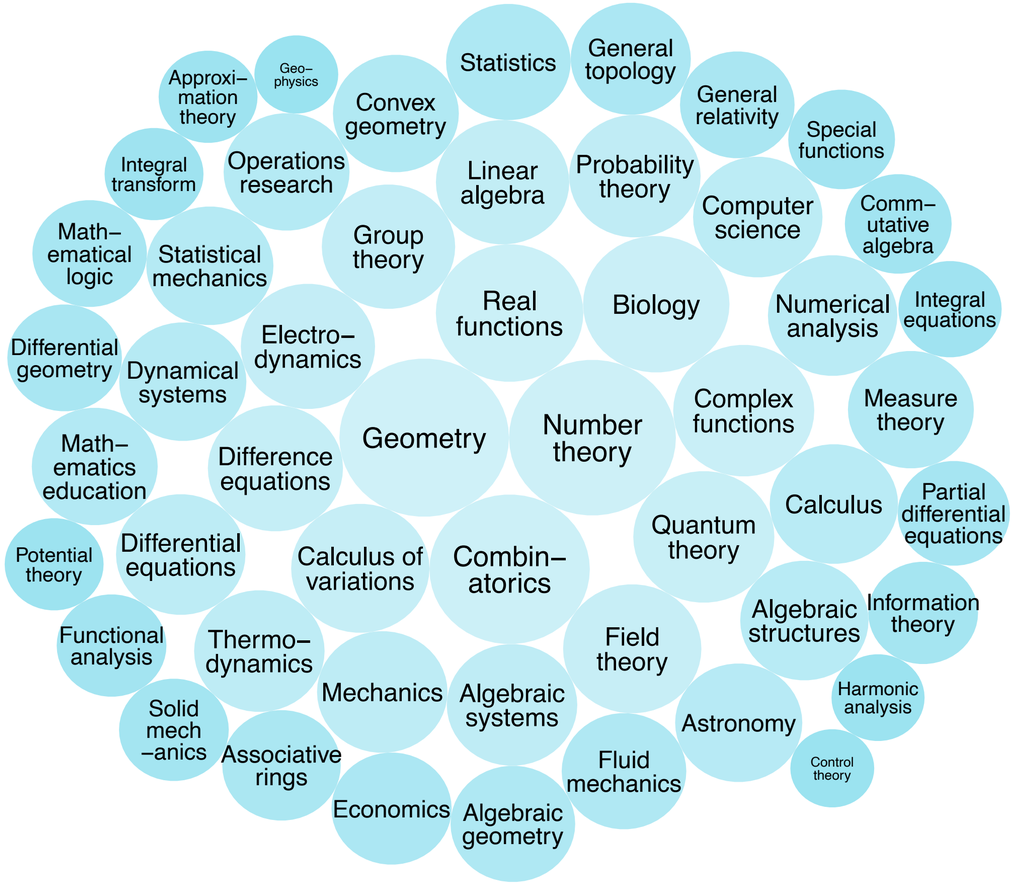
- Since we will finetune Qwen2.5-32B-Instruct, we want questions model can't already solve to maximize the 1K budget.
- Many questions crowded in a single domain is perhaps less helpful, so touch as many domain as possible.
- Longer chains contain more diverse problem solving strategy, so favor longer chains.

s1K: dataset construction

Step 1. Question collection and Gemini thoughts generation.

Step 2. Question featurization: model correctness, Gemini thinking length, problem domain.

Step 3. Remove easy questions. Uniform sampling over domain; Power law sampling in length.



- Since we will finetune Qwen2.5-32B-Instruct, we want questions model can't already solve to maximize the 1K budget.
- Many questions crowded in a single domain is perhaps less helpful, so touch as many domain as possible.
- Longer chains contain more diverse problem solving strategy, so favor longer chains.

Ablation studies on s1K

Experiment	Definition	AIME 2024	MATH 500	GPQA Diamond
1K-random	Randomly chosen 1K questions	36.7%	90.6%	52.0%
1K-diverse	Random sampling over domains	26.7%	91.2%	54.6%
1K-longest	Longest 1K questions	33.3%	90.4%	59.6%
59K-Full	All questions	53.3%	92.8%	58.1%
s1K	The final s1K	50.0%	93.0%	57.6%

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let’s analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let’s analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

- Forcing thinking to be more than 1000 tokens.

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let’s analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

Without BF `<thinking>` `...first 100 tokens...`

- Forcing thinking to be more than 1000 tokens.

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let’s analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

Without BF `<thinking>` `...first 100 tokens...` `the 101-th token thinking`

- Forcing thinking to be more than 1000 tokens.

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let’s analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

Without BF `<thinking>` `...first 100 tokens...` `the 101-th token thinking`

With BF `<thinking>` `...first 100 tokens...`

- Forcing thinking to be more than 1000 tokens.

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let’s analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

Without BF `<thinking>` `...first 100 tokens...` `the 101-th token thinking`

With BF `<thinking>` `...first 100 tokens...` `</thinking> Final answer:`

- Forcing thinking to be more than 1000 tokens.

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let's analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

Without BF `<thinking>` `...first 100 tokens...` `the 101-th token thinking`

With BF `<thinking>` `...first 100 tokens...` `</thinking> Final answer:`

- Forcing thinking to be more than 1000 tokens.

Without BF `<thinking>` `...first 529 tokens...` `</thinking>...`

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let's analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

Without BF `<thinking>` `...first 100 tokens...` `the 101-th token thinking`

With BF `<thinking>` `...first 100 tokens...` `</thinking> Final answer:`

- Forcing thinking to be more than 1000 tokens.

Without BF `<thinking>` `...first 529 tokens...` `</thinking>...`

With BF `<thinking>` `...first 529 tokens...`

Budget forcing

- s1K training template: Introduce two additional special tokens: end of thinking and start of thinking.

```
<user>
How many r in raspberry?
</user>
<assistant>
<thinking>
Let's analyze the problem...
...
Therefore...
</thinking>
The final answer is…
</assistant>
```

- Forcing thinking to be less than 100 tokens.

Without BF `<thinking>` `...first 100 tokens...` `the 101-th token thinking`

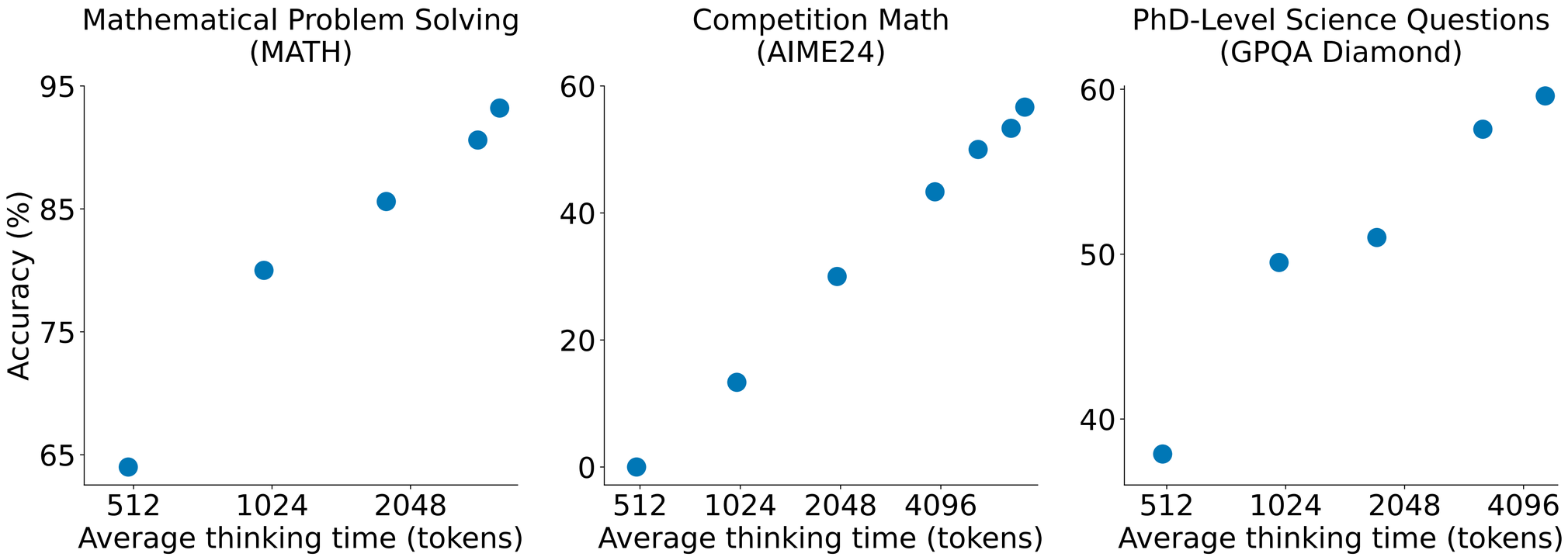
With BF `<thinking>` `...first 100 tokens...` `</thinking> Final answer:`

- Forcing thinking to be more than 1000 tokens.

Without BF `<thinking>` `...first 529 tokens...` `</thinking>...`

With BF `<thinking>` `...first 529 tokens...` `Wait,` `...continues...`

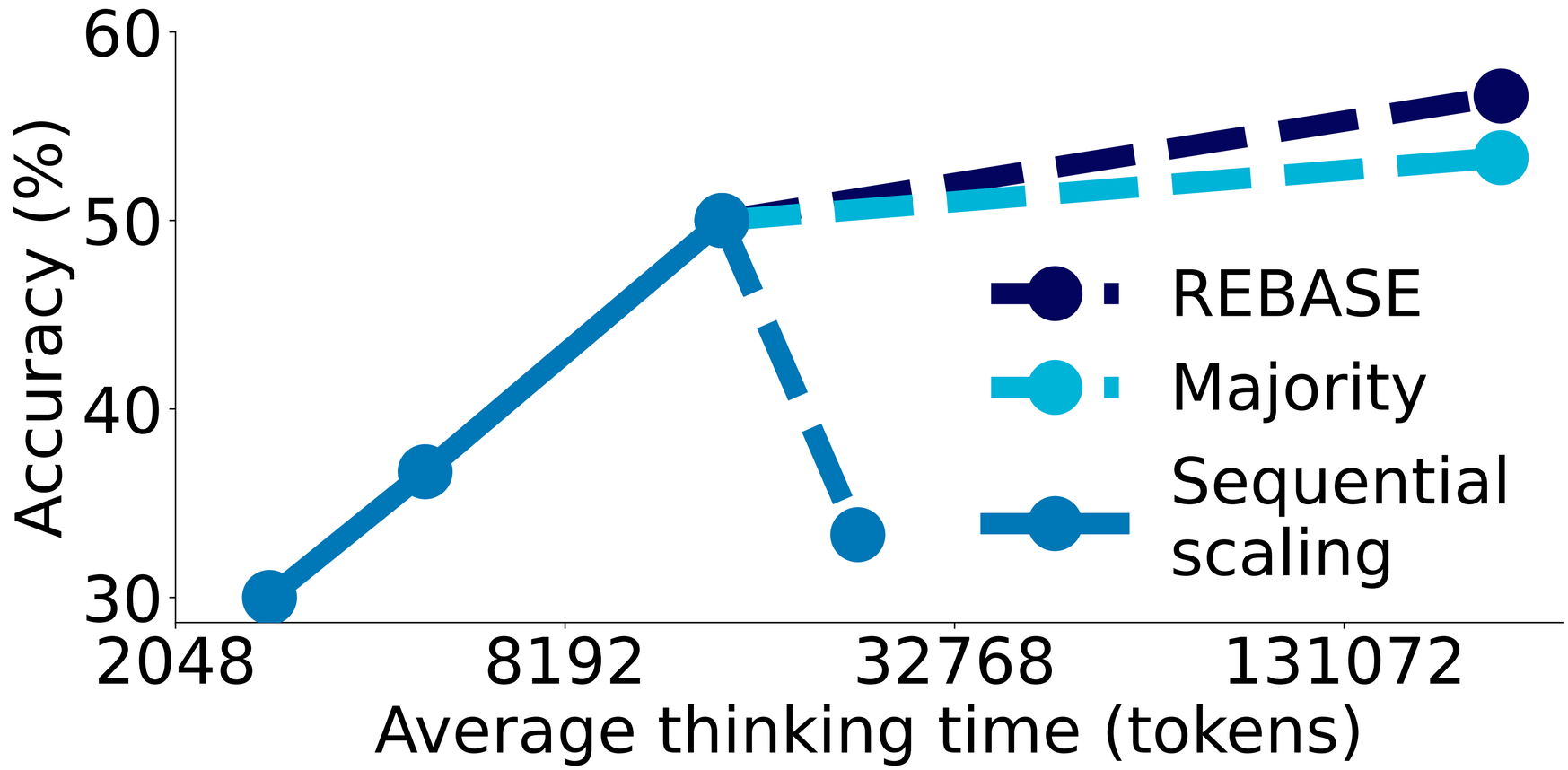
Budget forcing on s1 delivers test-compute scaling



Extrapolation: On AIME24, s1 achieves 50% accuracy, budget forcing boosts it to 57%.

Bonus: continued scaling beyond chains of thoughts

Combining internal chains of thoughts with external verifier to perform deeper search.



Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

- Can we scale the test-time compute indefinitely to solve more and more problems?

Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

```
100%|IIIIIIIIIIIIIIIIIIII| 315/315 [26:46<00:00,  5.10s/it]
```

Answer: 26 mins on 16 H100.

- Can we scale the test-time compute indefinitely to solve more and more problems?

Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

```
100%|IIIIIIIIIIIIIIIIIIII| 315/315 [26:46<00:00,  5.10s/it]
```

  Answer: 26 mins on 16 H100.

  Implication: o1-like reasoning ability is already present in the pretrained base model.

- Can we scale the test-time compute indefinitely to solve more and more problems?

Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

```
100%|IIIIIIIIIIIIIIIIIIII| 315/315 [26:46<00:00,  5.10s/it]
```

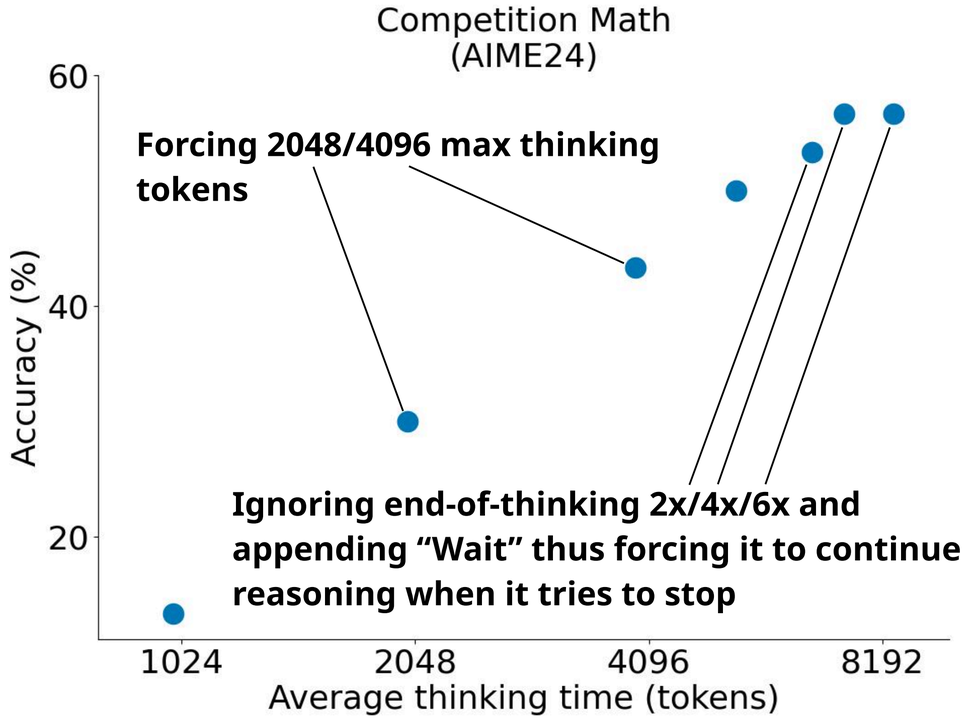
Answer: 26 mins on 16 H100.

Implication: o1-like reasoning ability is already present in the pretrained base model.

- Can we scale the test-time compute indefinitely to solve more and more problems?

Answer: Undetermined.

- Aggressive budget forcing eventually hurts accuracy.
- At least with BF, we can't scale test-time compute forever.



Scientific questions spurred o1

- How much resource does it take to create o1-like capability?

```
100%|IIIIIIIIIIIIIIIIIIII| 315/315 [26:46<00:00,  5.10s/it]
```

Answer: 26 mins on 16 H100.

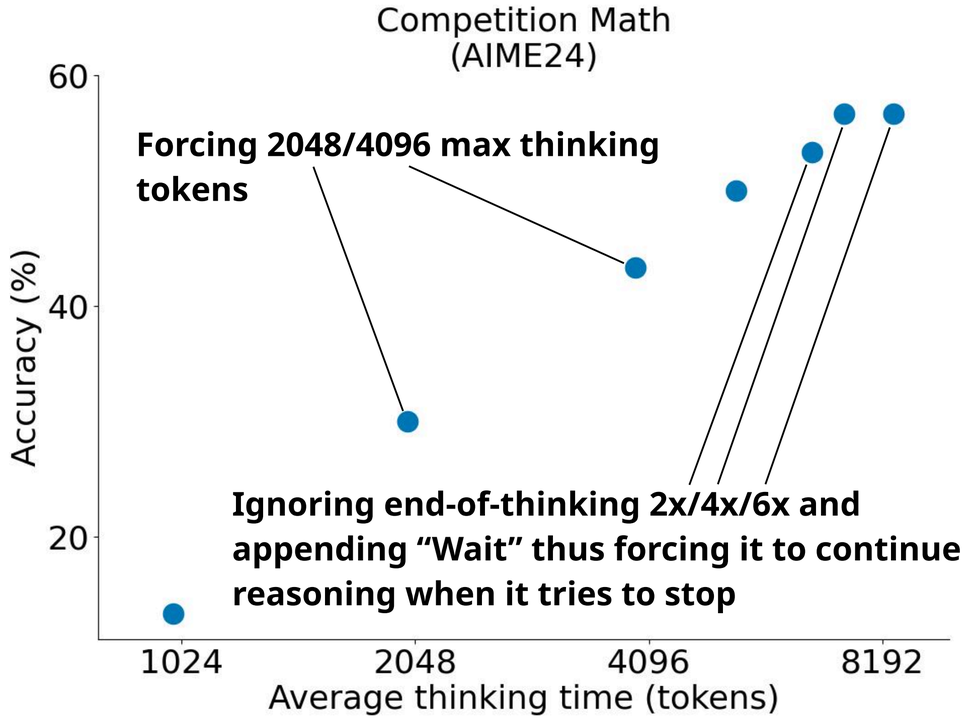
Implication: o1-like reasoning ability is already present in the pretrained base model.

- Can we scale the test-time compute indefinitely to solve more and more problems?

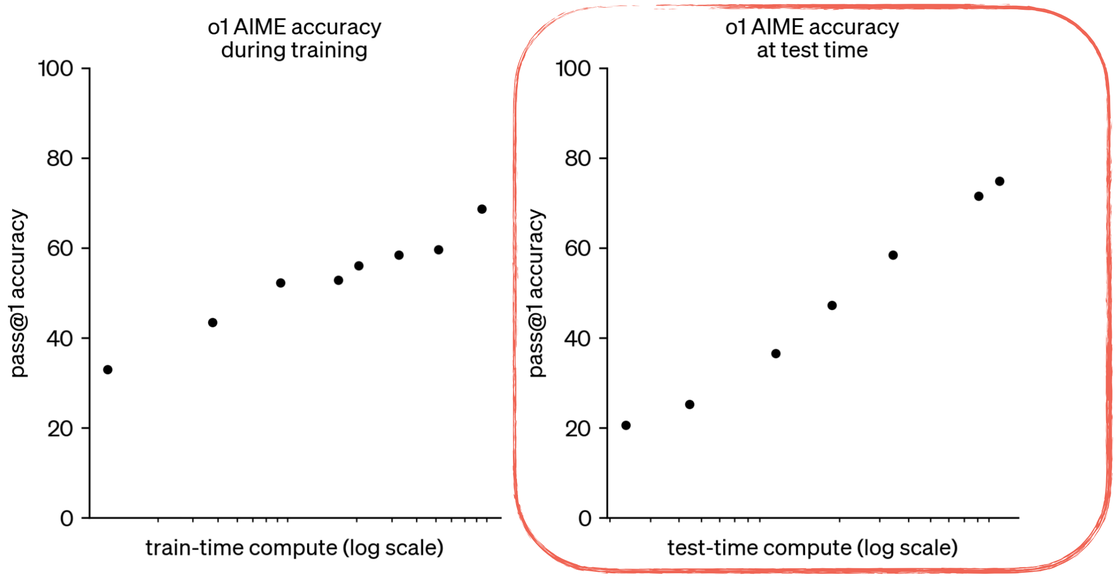
Answer: Undetermined.

- Aggressive budget forcing eventually hurts accuracy.
- At least with BF, we can't scale test-time compute forever.

Implication: Be cautious with test-time scaling claims.



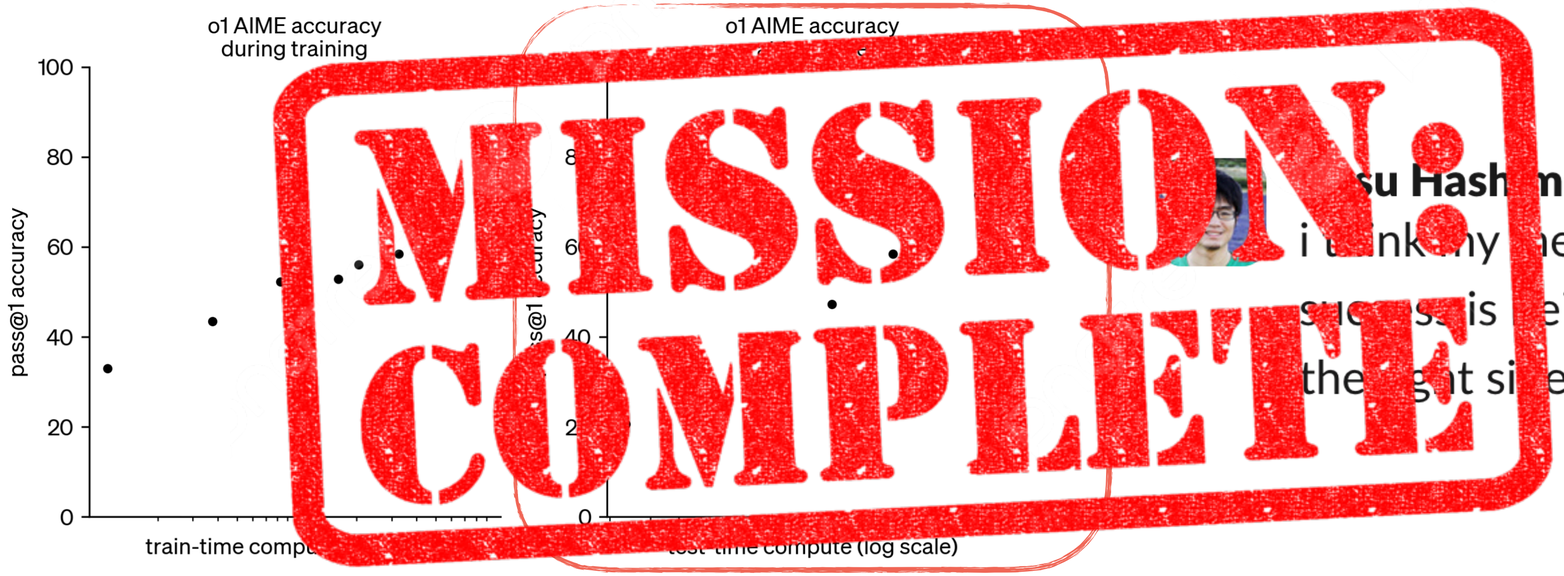
Looking back



Tatsu Hashimoto Sep 27th, 2024

i think my measurable definition of success is being able to replicate the right side of this plot

Looking back



su Hash moto Sep 27th, 2024

i t nk ny easurable definition of
s s is eing able to replicate
the t si e of this plot