Learning to Reason with LLMs

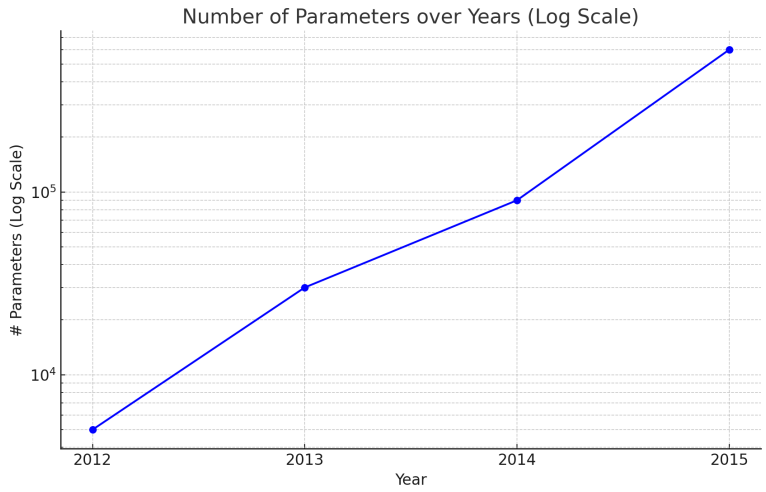
Noam Brown
OpenAI

It takes a village to grow a 🍓

Annual Computer Poker Competition

- Each year, research labs would make poker bots and play them against each other.

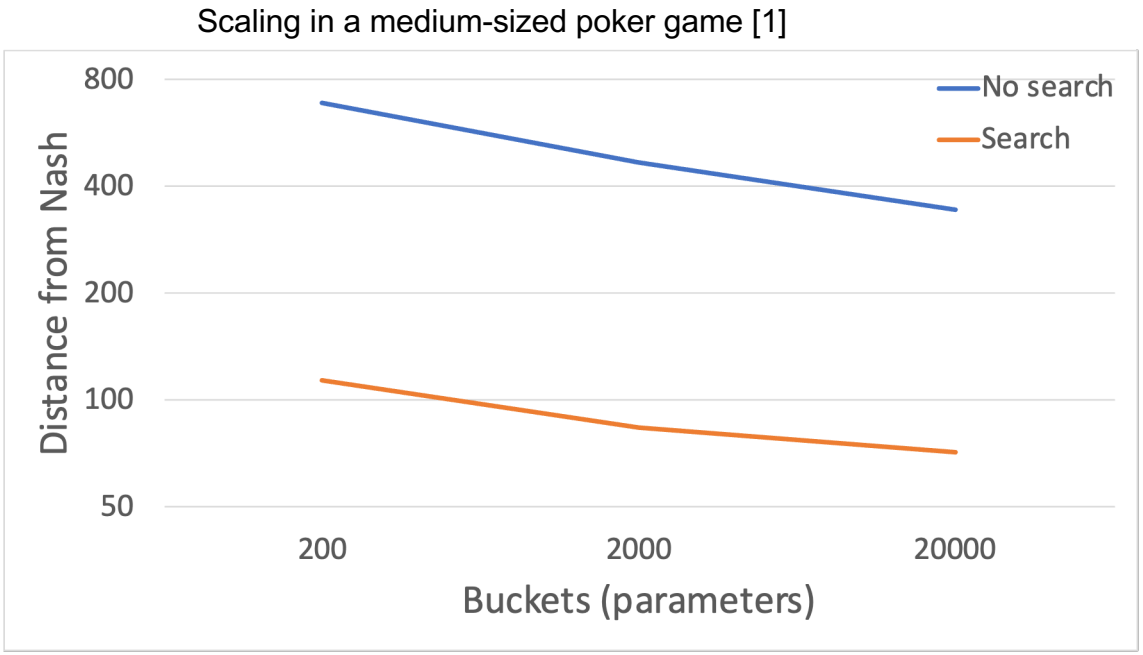
- It turned into a competition of scaling models:



2015 Brains vs. AI Poker Competition

- In 2015 we (CMU) challenged 4 top poker pros to an 80,000-hand poker competition
- $120,000 in prize money to incentivize them
- Our bot (Claudico) lost by 9.1 bb/100

The importance of search in poker



[1] “Safe and Nested Subgame Solving in Imperfect-Information Games.”
Brown & Sandholm. NeurIPS 2017 Best Paper.

2017 Brains vs AI Two-Player Poker AI

[Brown & Sandholm Science-17]

- Libratus against 4 top poker pros

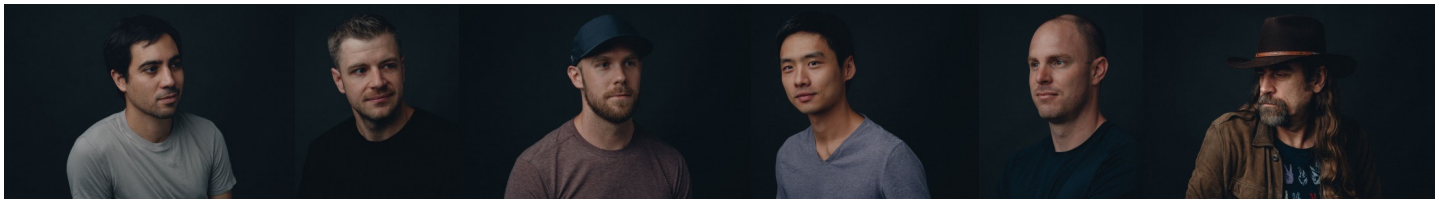


- 120,000 hands of poker
- $200,000 in prize money
- **Won by 15 bb/100** (Claudico had lost by 9 bb/100)
  - P-value ≈ 0.0002
- Each human lost individually to Libratus

2019 Pluribus Six-Player Poker AI

[Brown & Sandholm Science-19]

- Pluribus against 15 top pros in ***six-player*** no-limit Texas Hold'em



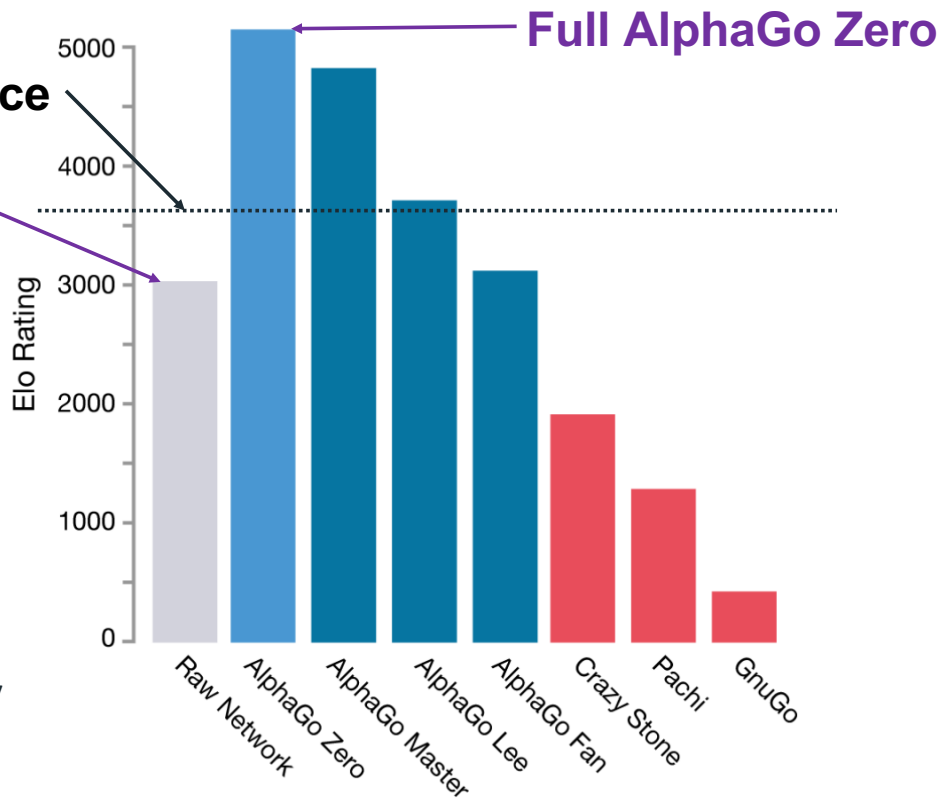
- 10,000 hands over 12 days in June 2019
  - Used variance-reduction techniques to decrease luck
  - One bot playing with five humans
- Won with >95% statistical significance
- **Cost under $150 to train**, runs on 28 CPU cores (no GPUs)

Why wasn’t search considered important in poker before?

- Cultural factors: researchers wanted the solution for the entire game upfront
- Scaling test-time compute makes experiments more expensive
- Incentives:
  - People were always thinking about winning the next Annual Computer Poker Competition (ACPC)
  - The ACPC limited test-time compute to 2 CPU cores
- **Most important:** people underestimated the difference it would make

Search in Go

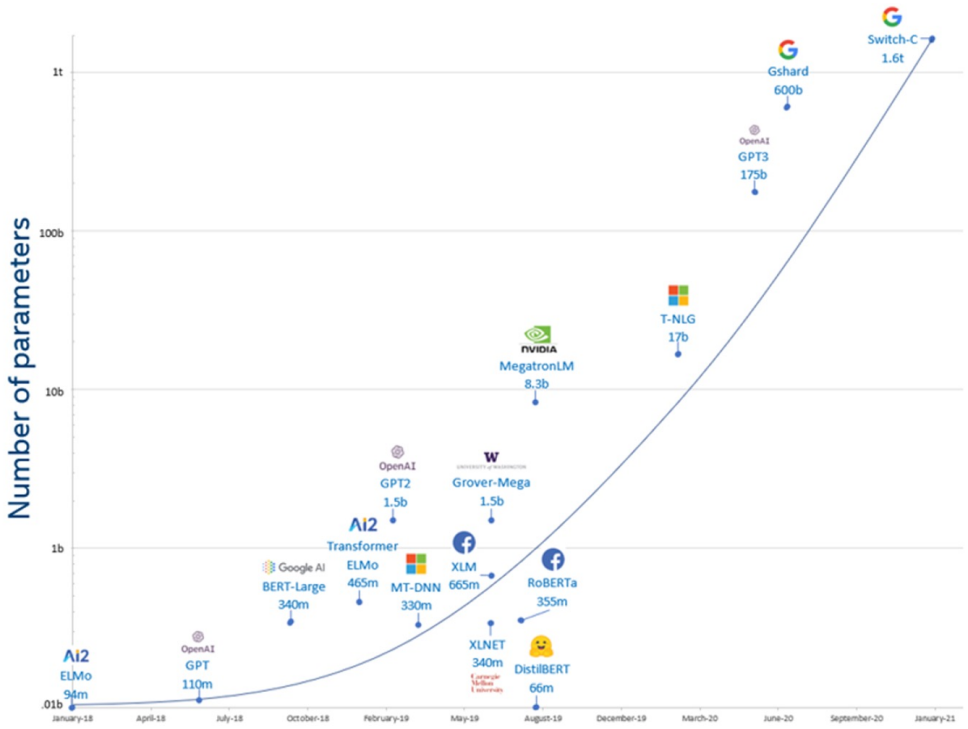
[Silver et al. Science-17]



- Increasing Elo by 120 points requires either:
  - ~2x model size and training
  - ~2x test-time search
- To get the raw policy net from 3000 Elo to 5200 Elo, you would need to scale by **~100,000x**

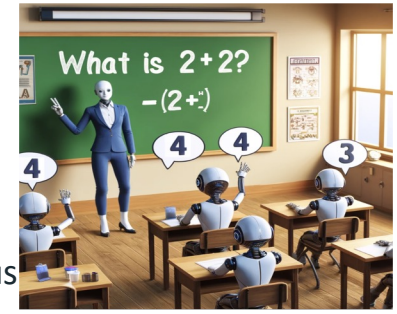
The state of AI today

- Every year, the models become bigger, trained on more data, for longer
- Every year, they do better than the previous year's models
- But the inference costs are still quite low
  - Pretraining costs >$100 million
  - Inference costs pennies

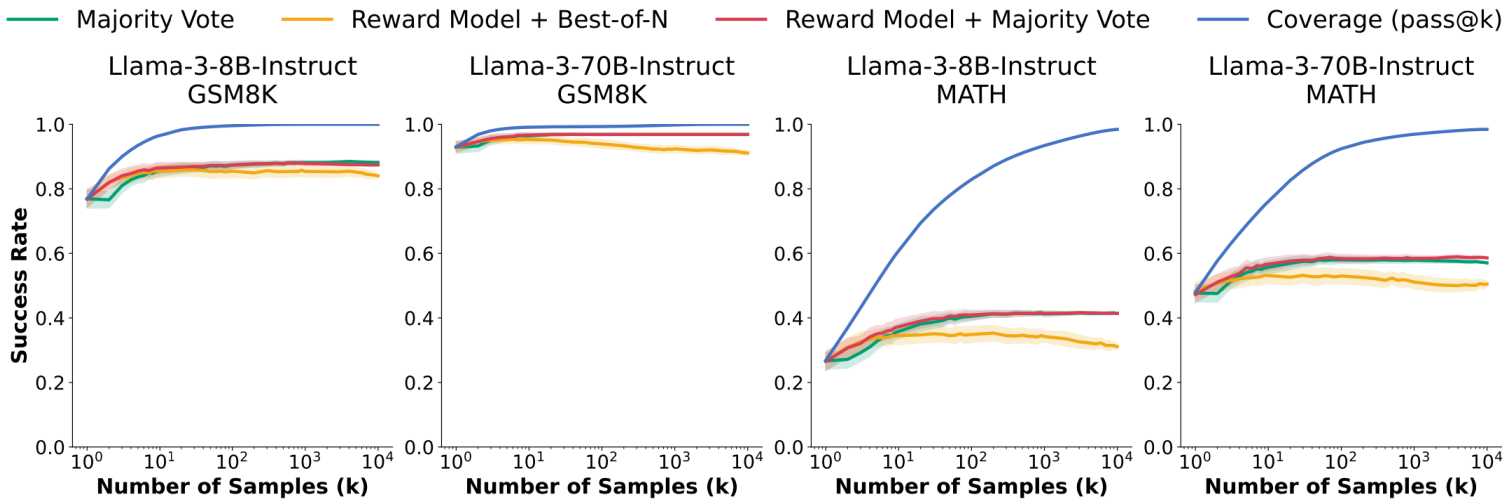


Is there a general way to scale inference compute in LLMs?

Verification in LLMs via Consensus / Majority Vote



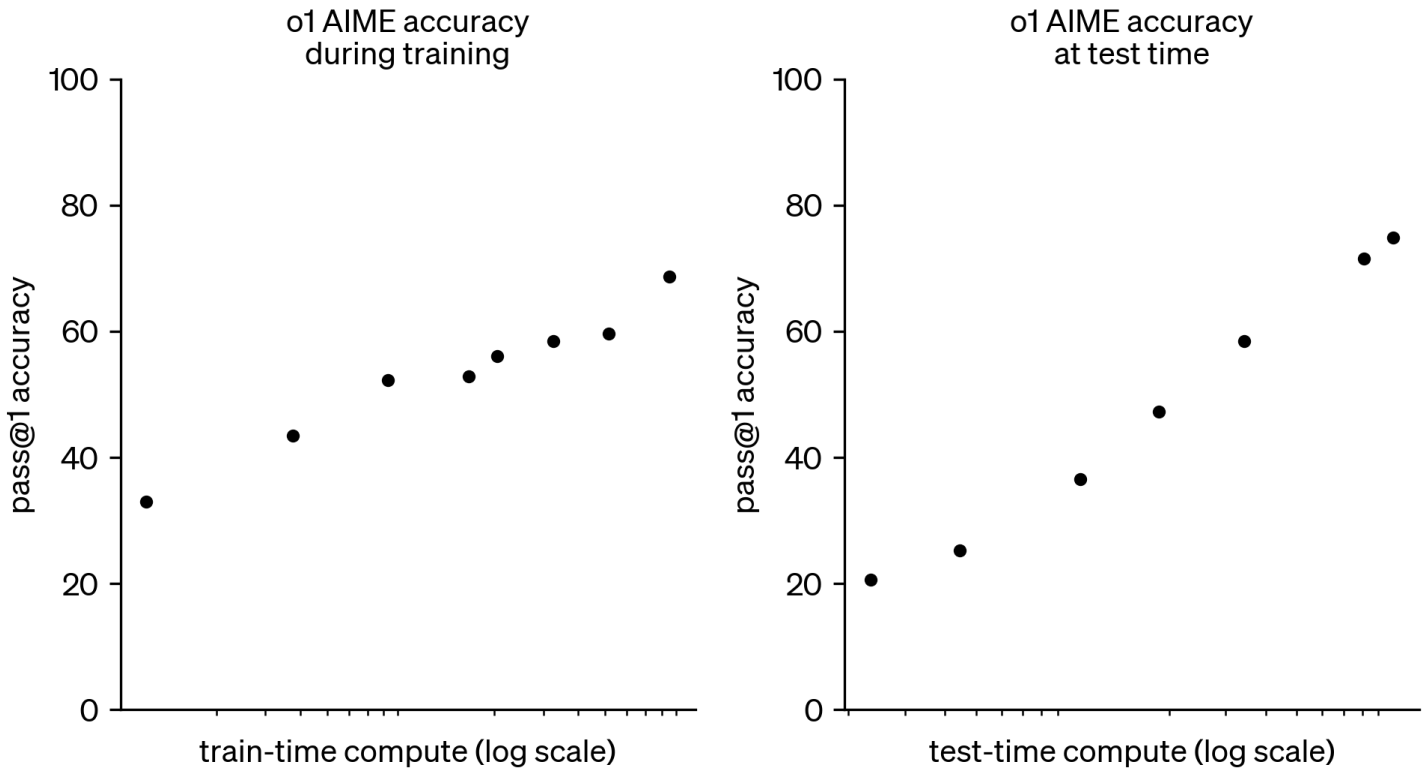
- Generate a bunch of solutions and take the most common one
  - Can be thought of as similar to sampling at low temperature
  - Minerva [Lewkowycz et al.] goes from 33.6% to 50.3% on MATH due to consensus
  - Consensus flatlines before 100 samples



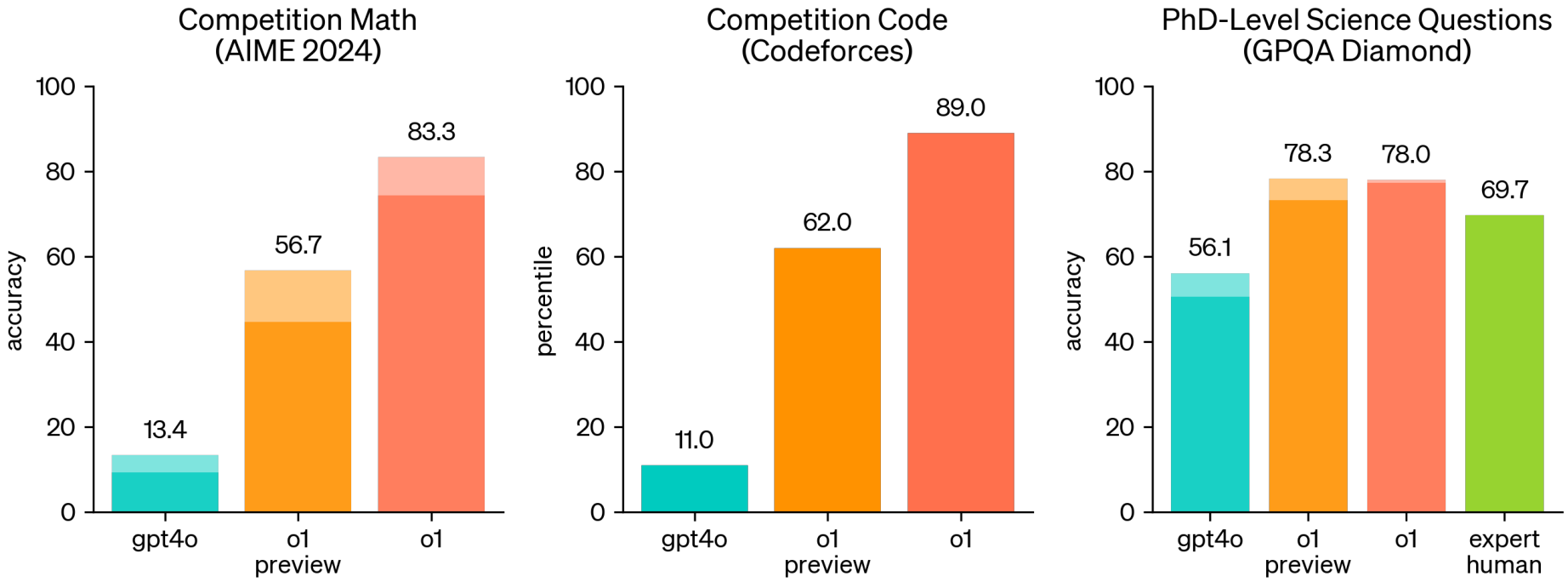
Large Language Monkeys: Scaling Inference Compute with Repeated Sampling

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

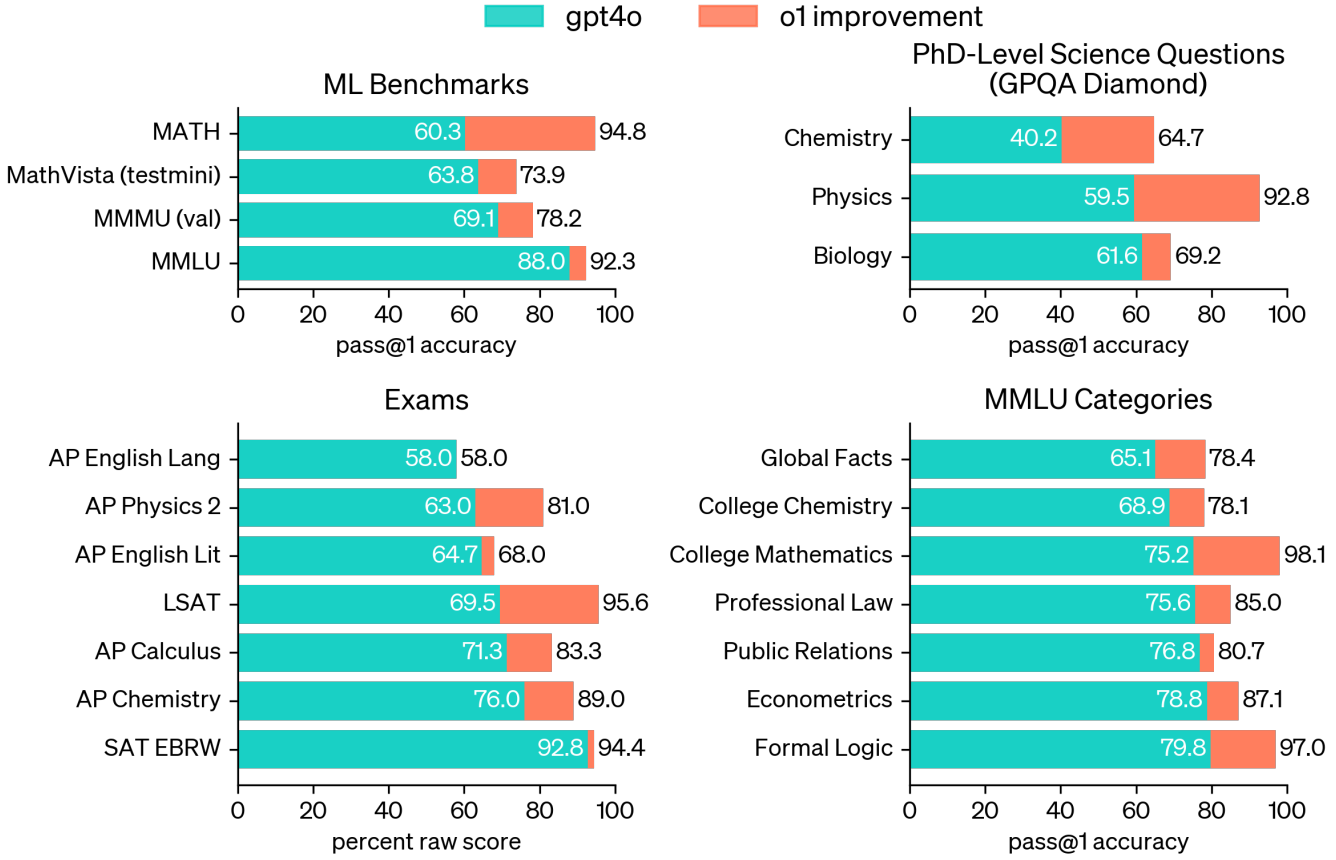
OpenAI o1


o1 AIME accuracy
during training
100
80
60
40
20
0
pass@1 accuracy
train-time compute (log scale)
o1 AIME accuracy
at test time
100
80
60
40
20
0
pass@1 accuracy
test-time compute (log scale)

Evals of OpenAI o1-preview and o1



Evals of OpenAI o1-preview and o1

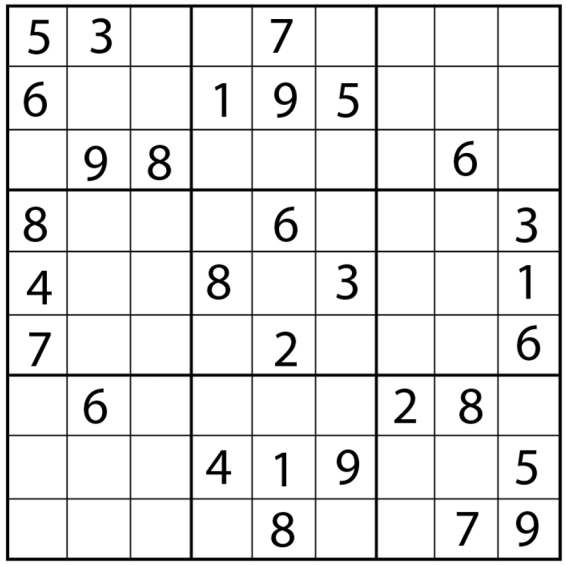


How does OpenAI o1 work?

- Uses large-scale RL to generate a chain of thought (CoT) [Wei et al. NeurIPS-2022] before answering
- CoT is longer and high-quality than what is attained via prompting
- CoT contains behavior like:
  - Error correction
  - Trying multiple strategies
  - Breaking down problems into smaller steps
- Example CoTs on the research blog post! https://openai.com/index/learning-to-reason-with-llms/

The Generator-Verifier Gap

- For some problems, verifying a good solution is easier than generating one
- Examples where verification is easier than generation:
  - Many puzzles (Sudoku, for example)
  - Math
  - Programming



The Generator-Verifier Gap

- For some problems, verifying a good solution is easier than generating one
- Examples where verification is easier than generation:
  - Many puzzles (Sudoku, for example)
  - Math
  - Programming
- Examples where verification isn't much easier
  - Information retrieval (What's the capital of Bhutan?)
  - Image recognition

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

The Generator-Verifier Gap

- For some problems, verifying a good solution is easier than generating one
- Examples where verification is easier than generation:
  - Many puzzles (Sudoku, for example)
  - Math
  - Programming
- Examples where verification isn't much easier
  - Information retrieval (What's the capital of Bhutan?)
  - Image recognition
- When a generator-verifier gap exists *and we have a good verifier*, we can spend more compute on inference to achieve better performance

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Scaling Scaling Laws with Board Games (Hex)

[Andy L. Jones, arXiv-2021]

Knowing now that compute can be spent in two places, at train time and test time, the immediate question is: how do these two budgets trade off? This is illustrated in Fig. 9, which shows that the trade-off is linear in log-compute: for each additional 10× of train-time compute, about 15× of test-time compute can be eliminated, down to a floor of a single-node tree search.

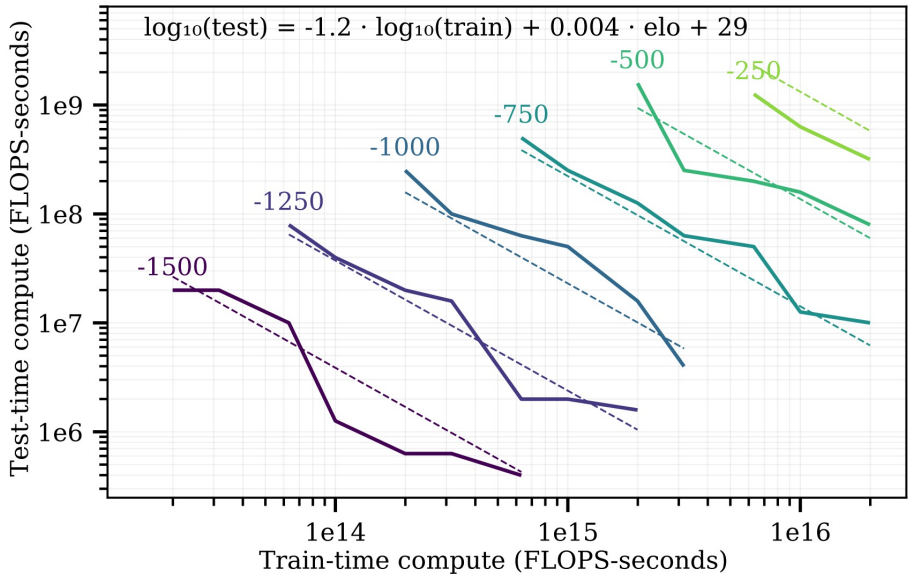


Fig. 9. The trade-off between train-time compute and test-time compute. Each dotted line gives the minimum train-test compute required for a certain Elo on a 9 × 9 board

Where does this go?

- Much **higher inference compute**, but much **more capable models**
  - What inference cost are you willing to pay for a proof of the Riemann Hypothesis?
  - What inference cost are you willing to pay for new life-saving drugs?
- There is still room to push inference compute much further
- AI can be more than chatbots

The Bitter Lesson by Richard Sutton

> “The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective… The two methods that seem to scale arbitrarily in this way are ***search*** and ***learning***.”

What's next for me?

- We are launching a new **multi-agent** reasoning team
- Looking for strong engineers that are interested in research
  - Prior multi-agent research experience not required
  - Tool use and distributed systems experience would be helpful
- If you are interested, apply at https://jobs.ashbyhq.com/openai/form/oai-multi-agent