10-423/10-623 Generative AI

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Reasoning Models
+
Mechanistic Interpretability

Matt Gormley & Pat Virtue

Lecture 26
Apr. 23, 2025

Reminders

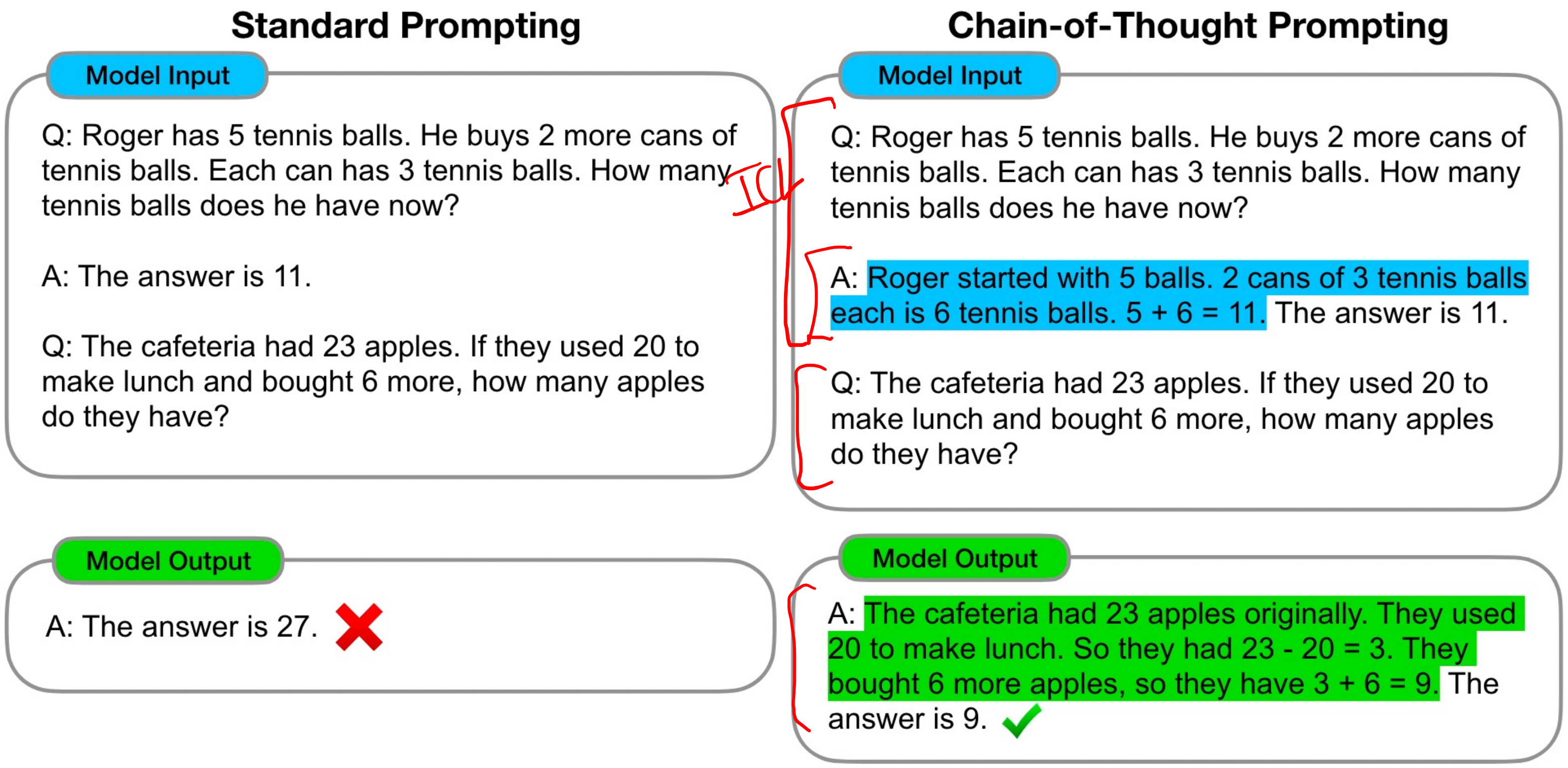
- Project "Poster"
  - Upload Due: Sun, Apr-27 at 11:59pm
  - Presentations: Tue, Apr-29 at 8:30am-11:30am
- Project Final Report
  - Due: Thu, May 1 at 11:59pm
- Project Code Upload
  - Due: Thu, May 1 at 11:59pm

REASONING MODELS

Chain-of-Thought Prompting

Recall…

- Asking the model to reason about its answer can improve its performance for few-shot in-context learning
- **Chain-of-thought prompting** provides such reasoning in the in-context examples



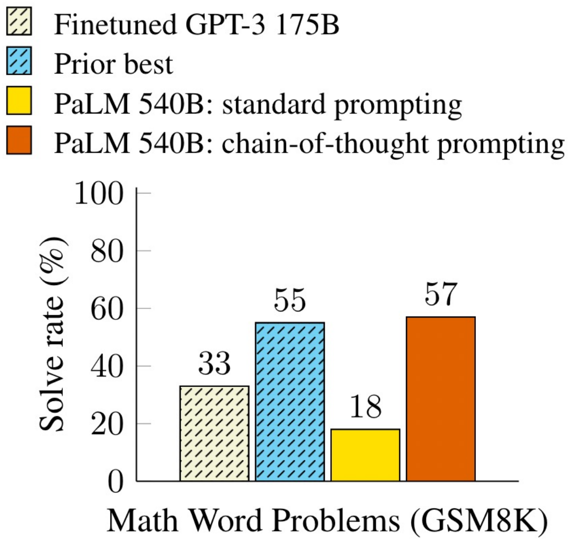


Figure 2: PaLM 540B uses chain-of-thought prompting to achieve new state-of-the-art performance on the GSM8K benchmark of math word problems. Finetuned GPT-3 and prior best are from Cobbe et al. (2021).

Figure from http://arxiv.org/abs/2201.11903

Recall...

Chain-of-Thought Prompting

- Asking the model to reason about its answer can improve its performance for few-shot in-context learning
- **Chain-of-thought prompting** provides such reasoning in the in-context examples

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

- But the model does better even if you just prompt it to reason step-by-step

Figure from https://arxiv.org/pdf/2205.11916.pdf

Example Reasoning Problem

Question:

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz

Thinking... (5 sec)

Answer:

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz

Thinking:

First, what is going on here?

We are given:

First, an example:

"oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step"

We can see that when "oyfjdnisdr rtqwainr acxz mynzbhhx" is transformed to "Think step by step"

Our task is: Use the example above to decode: "oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz"

So the first part is to figure out how "oyfjdnisdr rtqwainr acxz mynzbhhx" is decoded into "Think step by step"

...

Example from https://openai.com/index/learning-to-reason-with-llms/

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz

Thinking:

…

Observation 1: The phrase "Think step by step" is 18 letters (excluding spaces) . Let's count the letters.

Plaintext: Think step by step

First word: Think (5 letters)

Second word: step (4 letters)

Third word: by (2 letters)

Fourth word: step (4 letters)

Total letters in plaintext: 5 + 4 + 2 + 4 = 15 letters

Hmm.

….

Example from https://openai.com/index/learning-to-reason-with-llms/

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz

Thinking:

…

Similarly, let's count the letters in "oyfjdnisdr rtqwainr acxz mynzbhhx"

First, break down the letters:

Ciphertext letters: o y f j d n i s d r r t q w a i n r a c x z m y n z b h h x

Total letters: Let's count the letters in the ciphertext.

First word: oyfjdnisdr (10 letters)

Second word: rtqwainr (8 letters)

Third word: acxz (4 letters)

Fourth word: mynzbhhx (8 letters)

Total letters in ciphertext: 10 + 8 + 4 + 8 = 30 letters

…

Example from https://openai.com/index/learning-to-reason-with-llms/

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz

Thinking:

…

Given the time constraints, perhaps the easiest way is to try to see patterns.

Option 1: Try to find mapping from letters to letters.

Do any letters match?

First, let's write down the ciphertext and plaintext letters on top of each other.

Ciphertext: o y f j d n i s d r r t q w a i n r a c x z m y n z b h h x

Plaintext: T h i n k s t e p b y s t e p

Wait a minute.

I think maybe there is an anagram or substitution cipher here.

…

Example from https://openai.com/index/learning-to-reason-with-llms/

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz

Thinking:

…

Check the number of letters.

First ciphertext word: oyfjdnisdr (10 letters)

First plaintext word: Think (5 letters)

Hmm.

…

Interesting.

It seems that the ciphertext words are exactly twice as long as the plaintext words.

(10 vs 5, 8 vs 4, 4 vs 2, 8 vs 4)

Idea: Maybe we need to take every other letter or rebuild the plaintext from the ciphertext accordingly.

Let's test this theory.

…

Example from https://openai.com/index/learning-to-reason-with-llms/

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz

Thinking:
…

Now, what's the mapping from "oy" to 'T'?

Is there a way to map 'oy' to 'T'?
…
Option 1: Sum of the letters.

Convert letters to numbers (A=1, B=2, ..., Z=26)

So let's try to get to 'T' from 'o' and 'y'

o = 15

y = 25

Sum: 15 +25 = 40

But 'T' is 20.
…

Example from https://openai.com/index/learning-to-reason-with-llms/

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz

1276 lines later

Thinking:

…

So the message so far is

'THERE ARE THREE RS IN'

And then 'STRAWBERRY'

Wait, let's put the words together:

'THERE ARE THREE Rs IN STRAWBERRY'

I think 'RS' is 'R's', meaning 'Rs'

So the message is:

'THERE ARE THREE R'S IN STRAWBERRY'

Which is true.

So the decoded message is:

"THERE ARE THREE R'S IN STRAWBERRY"

Example from https://openai.com/index/learning-to-reason-with-llms/

Example Reasoning Problem

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz

Decoding Method:

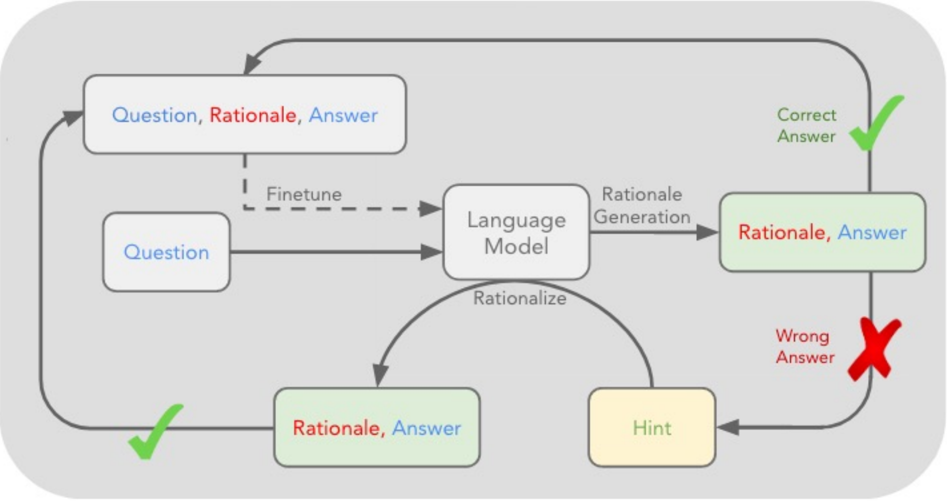
1. **Pair the letters** in the ciphertext.
2. **Convert each letter to its numerical position** in the alphabet (A=1, B=2, ..., Z=26).
3. **Sum the numerical values** of each pair.
4. **Compute the average** of the sum (divide by 2).
5. **Convert the average back to a letter** (1=A, 2=B, ..., 26=Z).

- OpenAI o1 chose not to release the "Thinking" output
- Instead, they released a summary of that output.

Example from https://openai.com/index/learning-to-reason-with-llms/

Self-Taught Reasoner (STaR)

- Data:
  - rationale examples (human annotated, small quantity)
  - problems without rationales (large quantity)
- Repeat:
  - bootstrap rationale training data:
    - use ICL with a few rationale examples
    - generate rationales for problems without
    - if generated answer is wrong, then try to regenerate a rationale that leads to a correct answer
  - fine-tune on all rationales that led to correct answers



Q: What can be used to carry a small dog?
Answer Choices:
(a) swimming pool
(b) basket
(c) dog show
(d) backyard
(e) own home
A: The answer must be something that can be used to carry a small dog. Baskets are designed to hold things. Therefore, the answer is basket (b).

GSM8k
↳ 8% SOTA 2022
↳ 10% STaR
⋮
↳ 90'+% SOTA 2025

Figure from http://arxiv.org/abs/2203.14465

AIME Dataset

Dataset Description

This dataset contains problems from the American Invitational Mathematics Examination (AIME) 2024. AIME is a prestigious high school mathematics competition known for its challenging mathematical problems.

Search this dataset

<table>
<tr><th>ID
string · lengths
9↔10 50%</th><th>Problem
string · lengths
333↔405 23.3%</th><th>Solution
string · lengths
284↔657 43.3%</th><th>Answer
int64
23↔110 33.3%</th></tr>
<tr><td>2024-II-4</td><td>Let x,y and z be positive real numbers that satisfy the following system of equations:
\[\log_2\left({x \over yz}\right) = {1 \over 2}\]
\[\log_2\left({y \over xz}\right) = {1 \over 3}\]
\[\log_2\left({z \over xy}\right) = {1 \over 4}\]
Then the value of $\left|\log_2(x^4y^3z^2)\right|$ is $\tfrac{m}{n}$ where m and n are relatively prime positive integers. Find $m+n$.</td><td>Denote $\log_2(x) = a$, $\log_2(y) = b$, and $\log_2(z) = c$.

Then, we have:
$a-b-c = \frac{1}{2}$,
$-a+b-c = \frac{1}{3}$,
$-a-b+c = \frac{1}{4}$.

Now, we can solve to get $a = \frac{-7}{24}, b = \frac{-9}{24}, c = \frac{-5}{12}$.
Plugging these values in, we obtain $|4a + 3b + 2c| = \frac{25}{8} \implies \boxed{033}$.</td><td>33</td></tr>
<tr><td>2024-II-12</td><td>Let $O(0,0), A(\tfrac{1}{2}, 0),$ and $B(0, \tfrac{\sqrt{3}}{2})$ be points in the coordinate plane. Let $\mathcal{F}$ be the famil…</td><td>Begin by finding the equation of the line $\overline{AB}$: $y = -\sqrt{3}x + \frac{\sqrt{3}}{2}$. Now, consider the general…</td><td>23</td></tr>
<tr><td>2024-I-4</td><td>Jen enters a lottery by picking 4 distinct numbers from $S=\{1,2,3,\cdots,9,10\}.$ 4 numbers are randomly chosen from $S.$…</td><td>This is a conditional probability problem. Bayes' Theorem states that \[P(A|B)=\dfrac{P(B|A)\cdot P(A)}{P(B)}\] in other words, th…</td><td>116</td></tr>
<tr><td>2024-I-3</td><td>Alice and Bob play the following game. A stack of n tokens lies before them. The players take turns with Alice going first. On…</td><td>Let's first try some experimentation. Alice obviously wins if there is one coin. She will just take it and win. If there are 2…</td><td>809</td></tr>
</table>

Figure a nd text from https://huggingface.co/datasets/Maxwell-Jia/AIME_2024/viewer/default/train?views%5B%5D=train&row=0

OpenAI o1

- The o1 model was **trained with Reinforcement Learning** to generate **chain-of-thought style rationales** for its answers
- These rationales (referred to as *Thinking* tokens) were hidden from the user, and a summary of the *Thinking* tokens was presented instead
- **At train time**: the compute could be increased by performing more reinforcement learning
- **At test time**: the compute could be increased by spending more time thinking

- **Result 1:** more train time compute leads to higher accuracy on reasoning problems

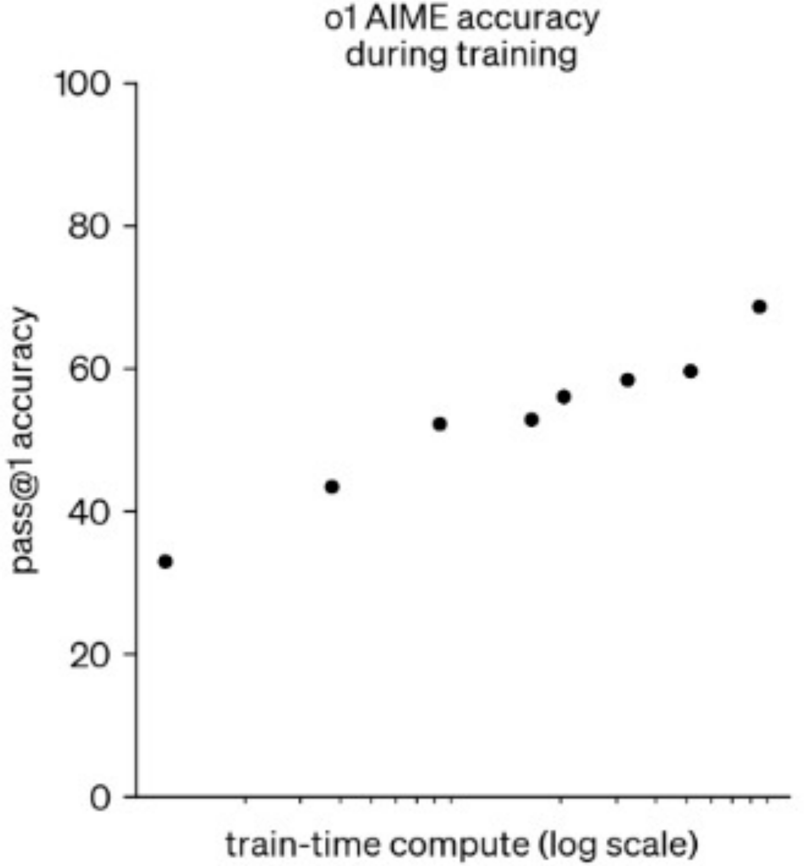


Figure from https://openai.com/index/learning-to-reason-with-llms/

OpenAI o1

- The o1 model was **trained with Reinforcement Learning** to generate **chain-of-thought style rationales** for its answers
- These rationales (referred to as *Thinking* tokens) were hidden from the user, and a summary of the *Thinking* tokens was presented instead
- **At train time**: the compute could be increased by performing more reinforcement learning
- **At test time**: the compute could be increased by spending more time thinking

- **Result 2:** more test time compute leads to higher accuracy on reasoning problems

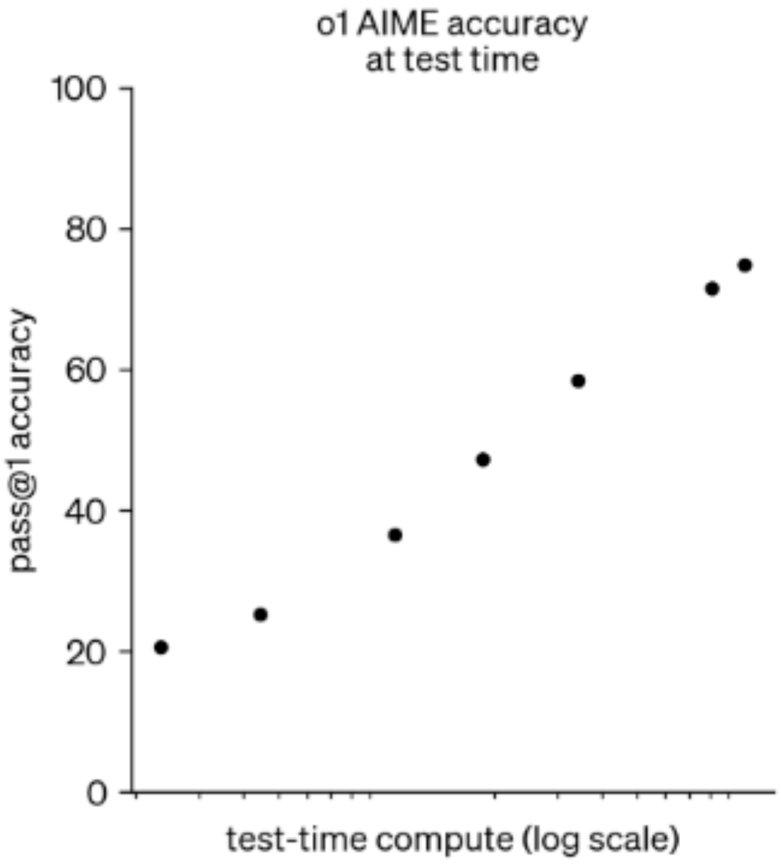


Figure from https://openai.com/index/learning-to-reason-with-llms/

OpenAI o1

- The o1 model was **trained with Reinforcement Learning** to generate **chain-of-thought style rationales** for its answers
- These rationales (referred to as *Thinking* tokens) were hidden from the user, and a summary of the *Thinking* tokens was presented instead
- **At train time:** the compute could be increased by performing more reinforcement learning
- **At test time:** the compute could be increased by spending more time thinking

Q: Why is this description so vague and non-technical?

A: Because **Open**AI only released a blog post, and this is about the sum total of what it said

Figure from https://openai.com/index/learning-to-reason-with-llms/

OpenAI o1

Across a variety of math, reasoning, commonsense, coding, etc. problems o1 improves over gpt4o

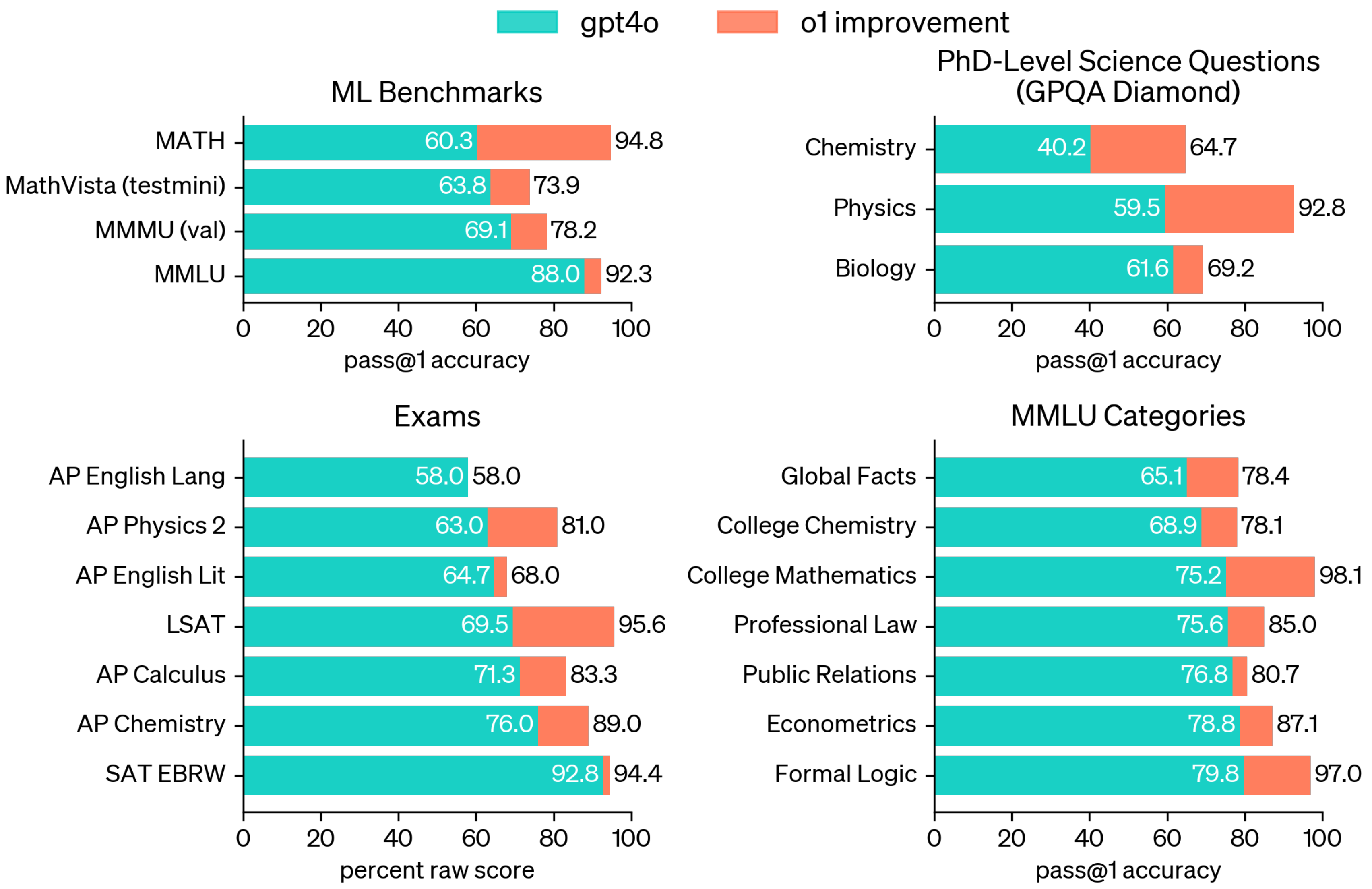
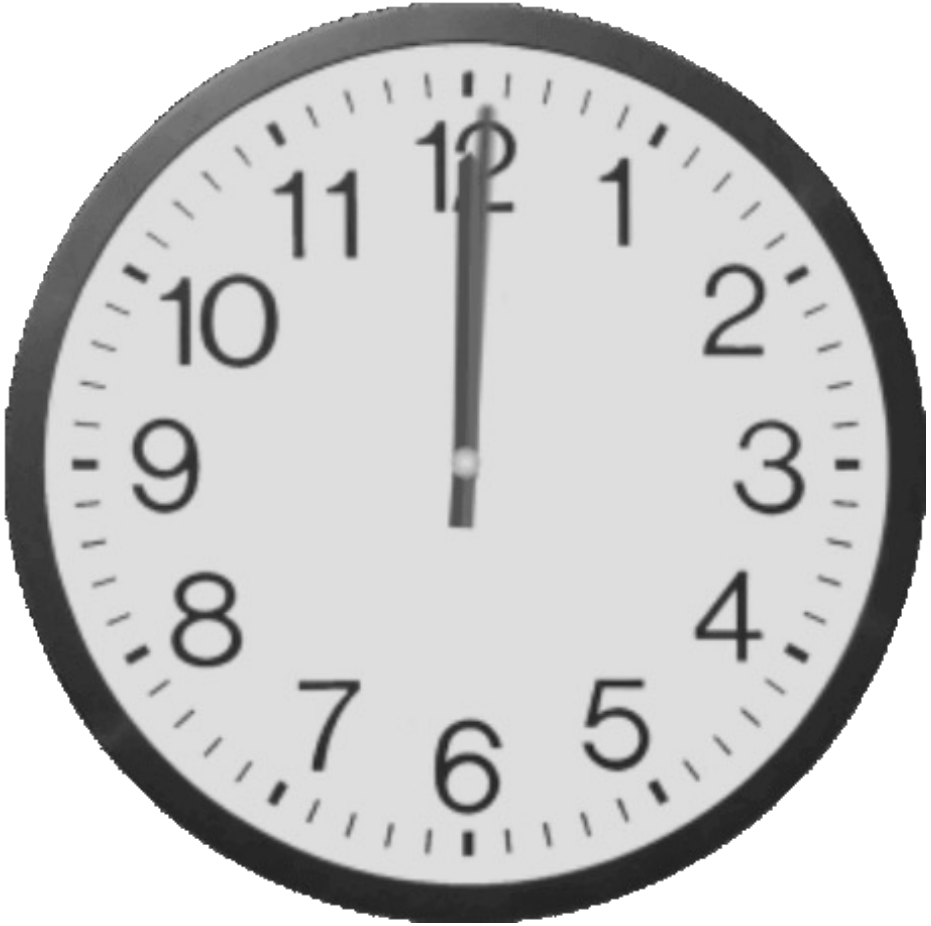


Figure from https://openai.com/index/learning-to-reason-with-llms/

- The closed source model (o1) was clearly superior than any open source models
- So we waited for the open source models to catch up…



DeepSeek-R1-Zero and DeepSeek-R1

- Enter DeepSeek-R1…
- This open source and open weight model is 671B parameters
- It is a carefully tuned version of the base model DeepSeek-V3
- And it achieves comparable performance to o1

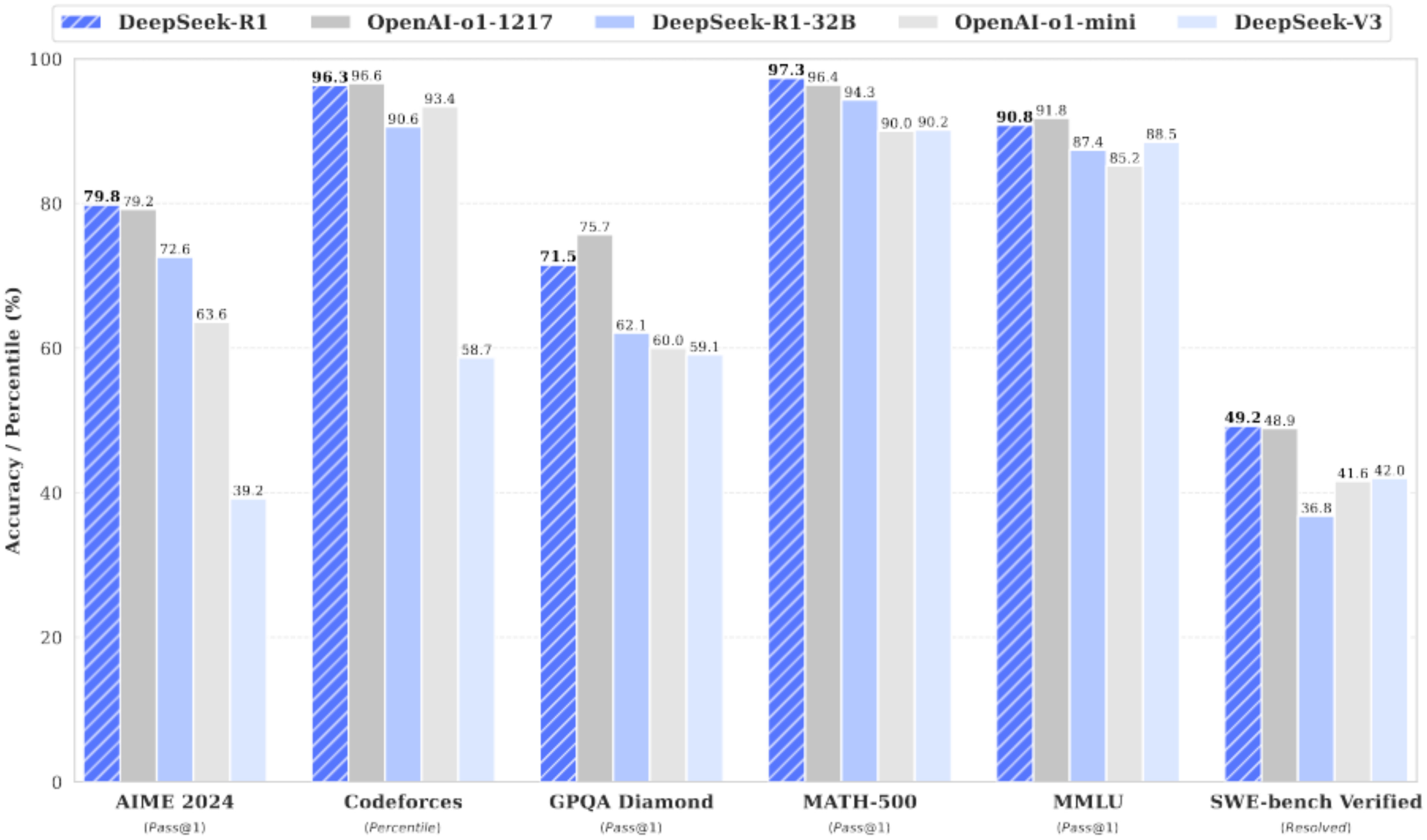


Figure 1 | Benchmark performance of DeepSeek-R1.

Figure from http://arxiv.org/abs/2501.12948

PPO vs. GRPO

- DeepSeek-Math came before DeepSeek-R1 and introduced the idea of GRPO
- GRPO is an RL algorithm akin to PPO, but it greatly reduces the memory requirements by removing the need for a Value Model

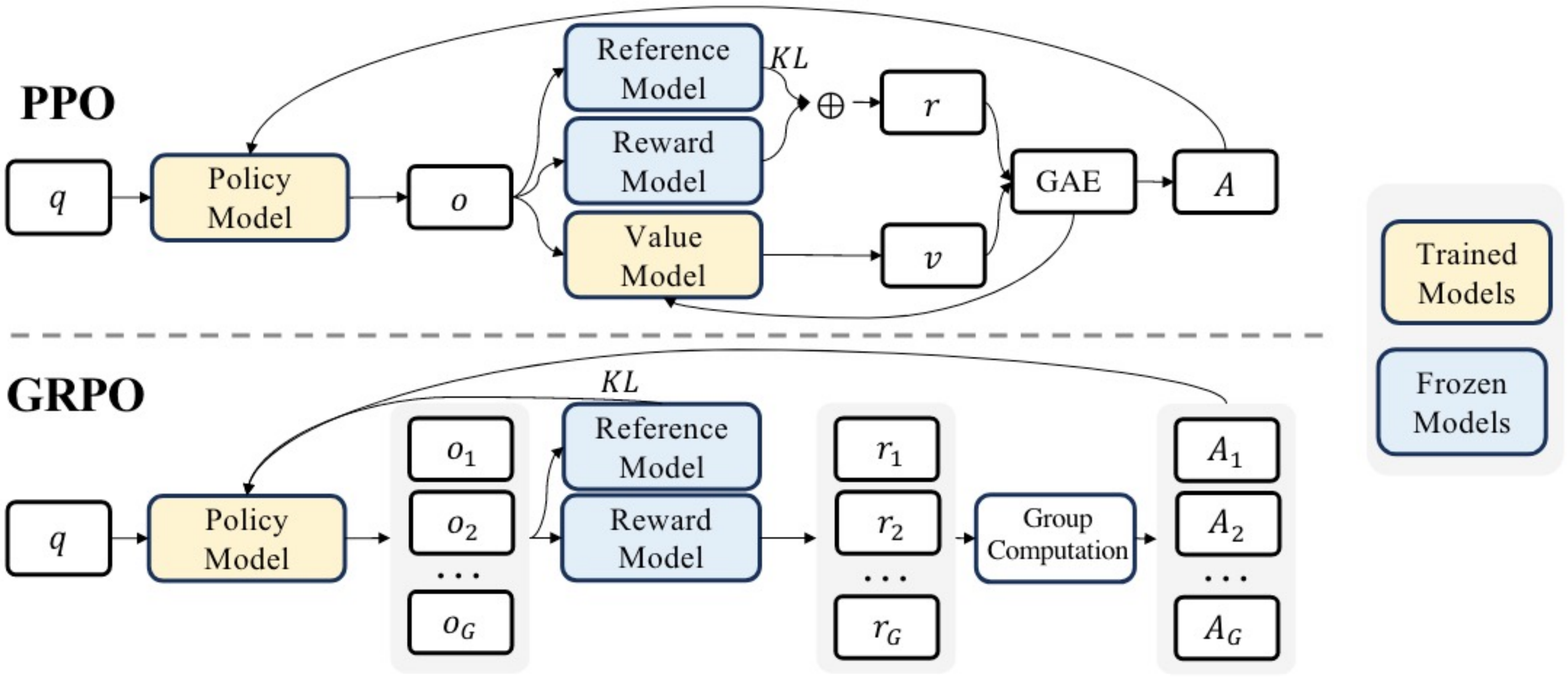


Figure from http://arxiv.org/abs/2402.03300

PPO vs. GRPO

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is an actor-critic RL algorithm that is widely used in the RL fine-tuning stage of LLMs (Ouyang et al., 2022). In particular, it optimizes LLMs by maximizing the following surrogate objective:

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min \left[\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})}, 1 - \varepsilon, 1 + \varepsilon \right) A_t \right], \quad (1)$$

where π_θ and $\pi_{\theta_{old}}$ are the current and old policy models, and q, o are questions and outputs sampled from the question dataset and the old policy $\pi_{\theta_{old}}$, respectively. ε is a clipping-related hyper-parameter introduced in PPO for stabilizing training. A_t is the advantage, which is computed by applying Generalized Advantage Estimation (GAE) (Schulman et al., 2015), based on the rewards $\{r_{\geq t}\}$ and a learned value function V_ψ. Thus, in PPO, a value function needs to be trained alongside the policy model and to mitigate over-optimization of the reward model, the standard approach is to add a per-token KL penalty from a reference model in the reward at each token (Ouyang et al., 2022), i.e.,

$$r_t = r_\varphi(q, o_{\leq t}) - \beta \log \frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{ref}(o_t|q, o_{<t})}, \quad (2)$$

where r_φ is the reward model, π_{ref} is the reference model, which is usually the initial SFT model, and β is the coefficient of the KL penalty.

Figure from http://arxiv.org/abs/2402.03300

PPO vs. GRPO

accurate at each token. To address this, as shown in Figure 4, we propose Group Relative Policy Optimization (GRPO), which obviates the need for additional value function approximation as in PPO, and instead uses the average reward of multiple sampled outputs, produced in response to the same question, as the baseline. More specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}, \quad (3)$$

where ε and β are hyper-parameters, and $\hat{A}_{i,t}$ is the advantage calculated based on relative rewards of the outputs inside each group only, which will be detailed in the following subsections. The group relative way that GRPO leverages to calculate the advantages, aligns well with the comparative nature of rewards models, as reward models are typically trained on datasets of comparisons between outputs on the same question. Also note that, instead of adding KL penalty in the reward, GRPO regularizes by directly adding the KL divergence between the trained policy and the reference policy to the loss, avoiding complicating the calculation of $\hat{A}_{i,t}$.

Figure from http://arxiv.org/abs/2402.03300

DeepSeek-R1-Zero

Training method:

- Trained entirely via Reinforcement Learning (RL) without any supervised fine-tuning (SFT).
- Started with a pretrained base model (DeepSeek-V3-Base), then used RL with human preferences to drive learning.
- Relied on a pure RL pipeline, making it one of the first large-scale demonstrations of RL-only training in LLMs.
- Aimed to explore whether reasoning abilities can emerge solely through RL, without labeled datasets.

Figure from http://arxiv.org/abs/2501.12948

DeepSeek-R1-Zero

Reward model:

- Did *not* use a neural reward model
- Instead just defined a rule-based reward model consisting of two parts:
  - Accuracy rewards: did the model answer the question correctly?
  - Format rewards: did the model adhere to the prompt template?

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: prompt. Assistant:

Table 1 | Template for DeepSeek-R1-Zero. prompt will be replaced with the specific reasoning question during training.

Figure from http://arxiv.org/abs/2501.12948

DeepSeek-R1-Zero

Results:

- On AIME, the longer the model is trained with RL, the better the performance becomes
- Eventually it surpasses o1 performance

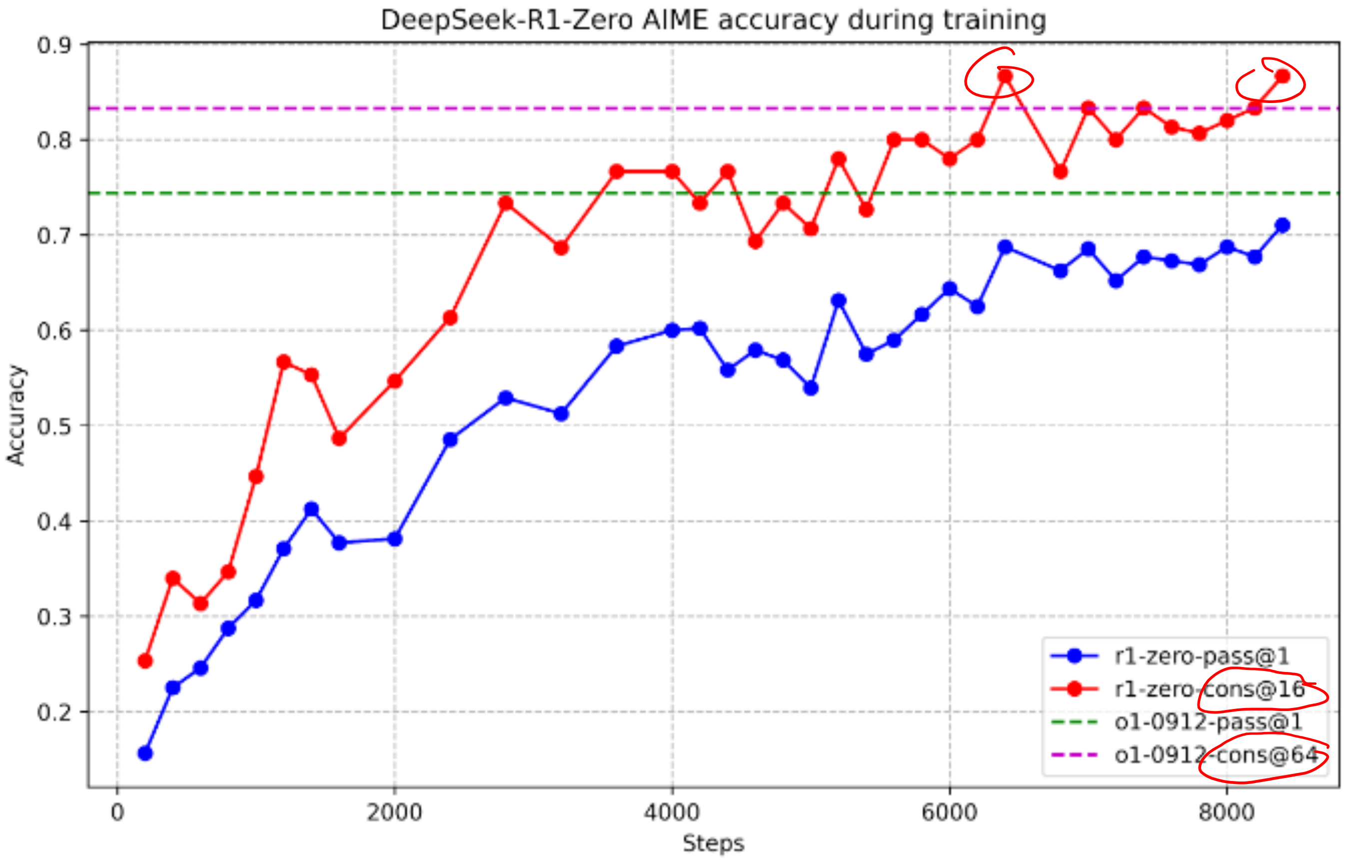


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

Figure from http://arxiv.org/abs/2501.12948

DeepSeek-R1-Zero

Results:

- Gradually the model learns to use longer and longer sequences of *Thinking* tokens
- This is accomplished purely through the RL objective
- There is no direct action taken to increase reasoning length

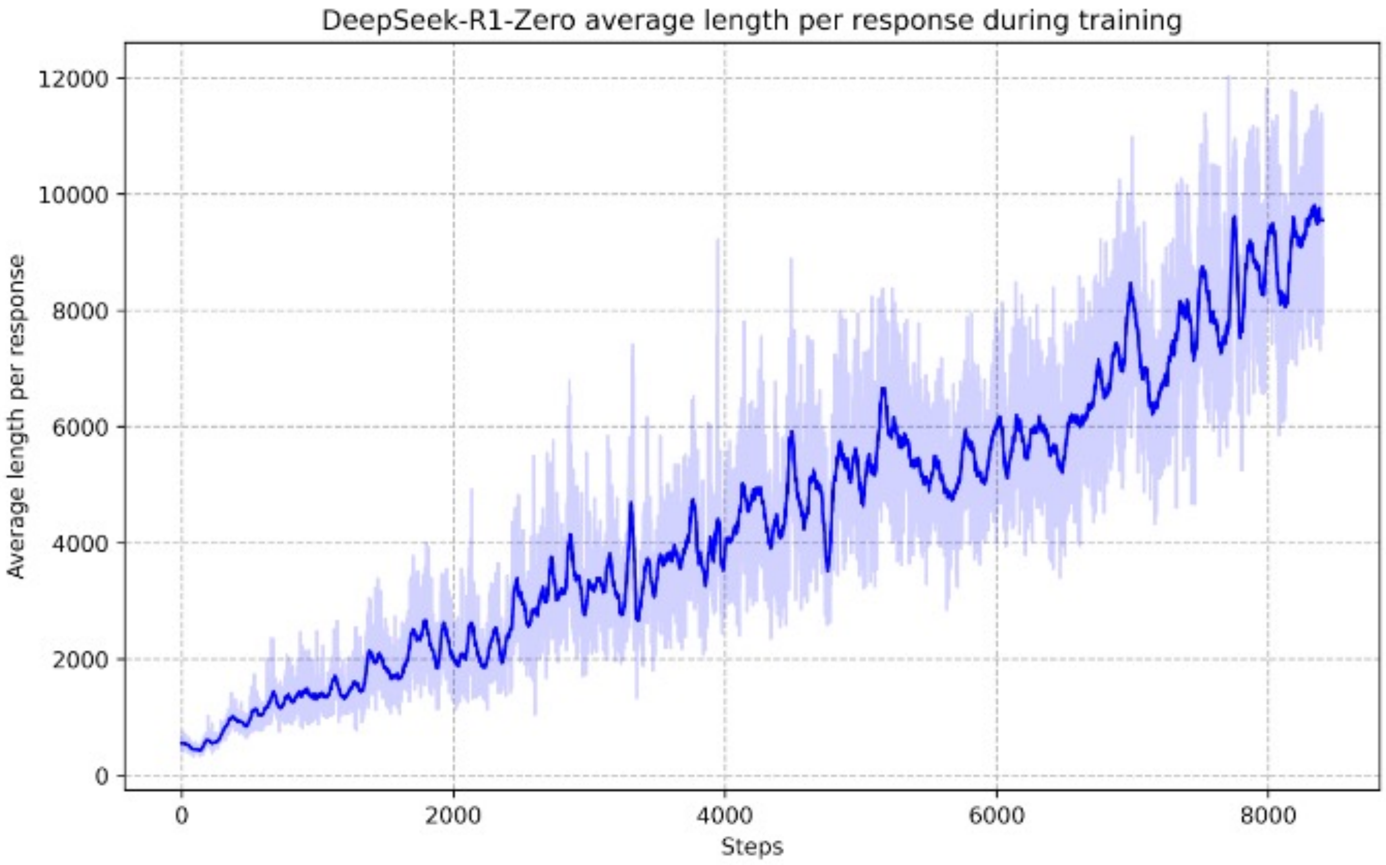


Figure 3 | The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time.

Figure from http://arxiv.org/abs/2501.12948

DeepSeek-R1-Zero

Problems:

- Poor readability (e.g. humans don't really understand what it's saying)
- Language mixing (e.g. English and Chinese muddled into a pigeon language)

Figure from http://arxiv.org/abs/2501.12948

DeepSeek-R1

Training method:

- Built on R1-Zero with a hybrid training strategy:

    1. **Cold Start:** Fine-tune a base model on a few thousand curated, human-friendly long CoTs.
    2. **Reasoning-Focused RL:** Scale up RL with math, coding, and logic tasks. This time, add *language-consistency rewards* to push the model into staying coherent in a single language.
    3. **Rejection Sampling + SFT:** Sample correct, well-structured chains-of-thought from the RL model, augment them with general capabilities data (writing, Q&A, self-cognition), and train a new base checkpoint.
    4. **RL Across Scenarios:** A second RL stage includes both reasoning tasks *and* general tasks for "helpfulness" and "harmlessness."

- This two-stage pipeline addressed the shortcomings (e.g. repetition, language mixing) observed in R1-Zero.
- Emphasized improved readability, coherence, and task accuracy due to the incorporation of SFT before RL.

Figure from https://huggingface.co/blog/NormalUhr/deepseek-r1-explained