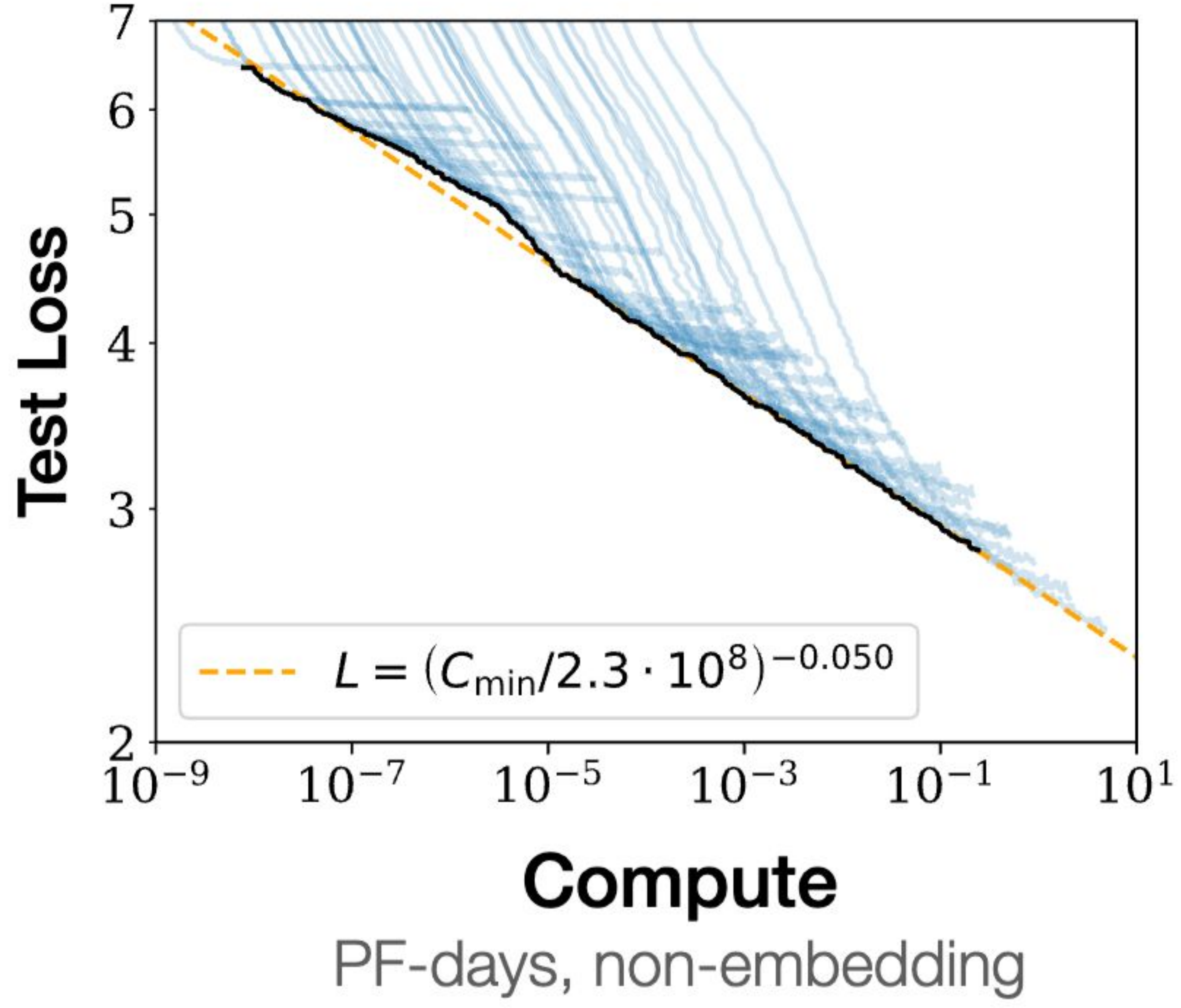
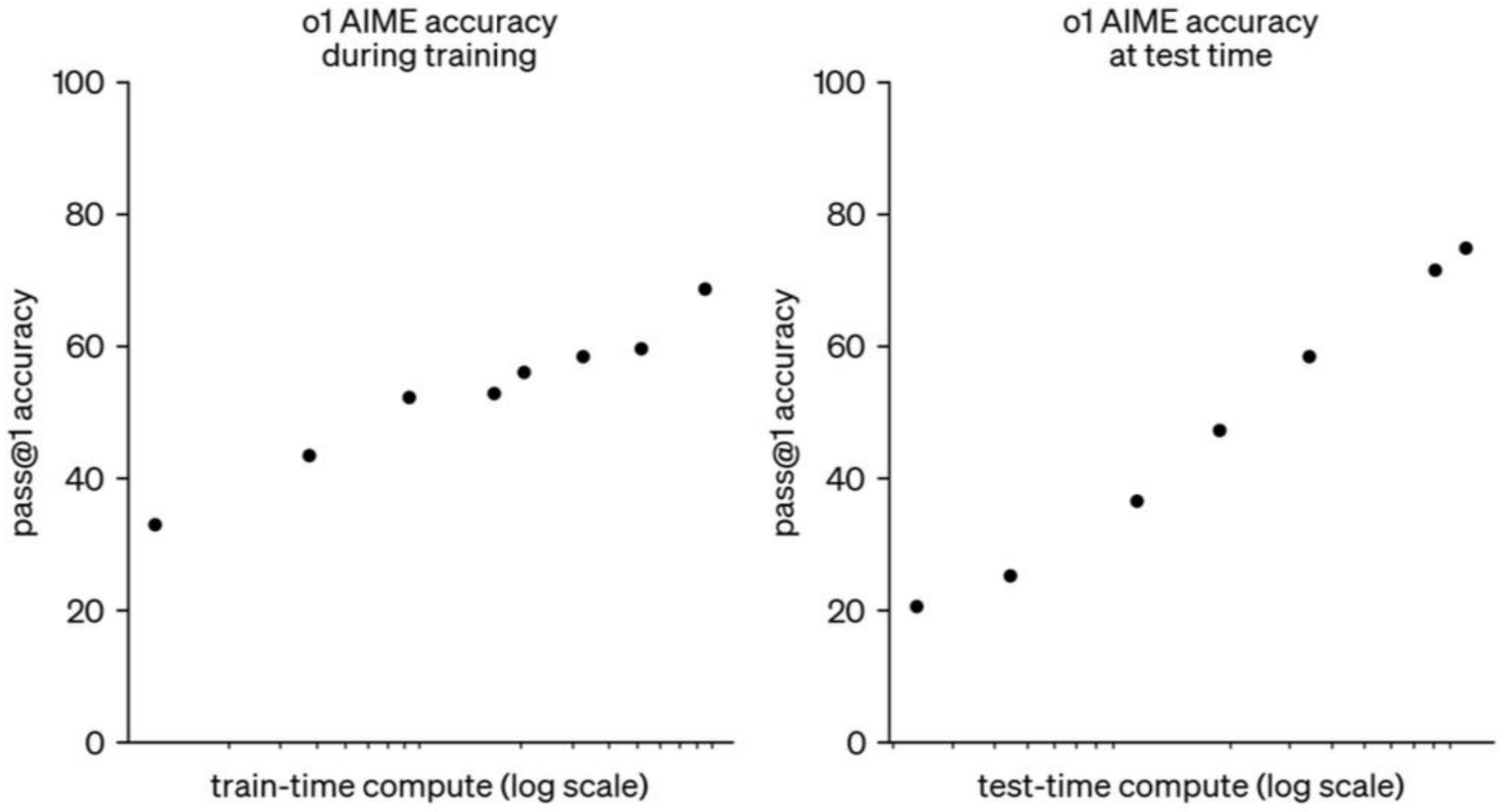
从 TinyZero 到 APR：语言模型推理能力的探索与自适 应并行化

Jiayi Pan, Xiuyu Li
UC Berkeley

Berkeley AI Research

推理模型，范式转变





o1 performance smoothly improves with both train-time and test-time compute

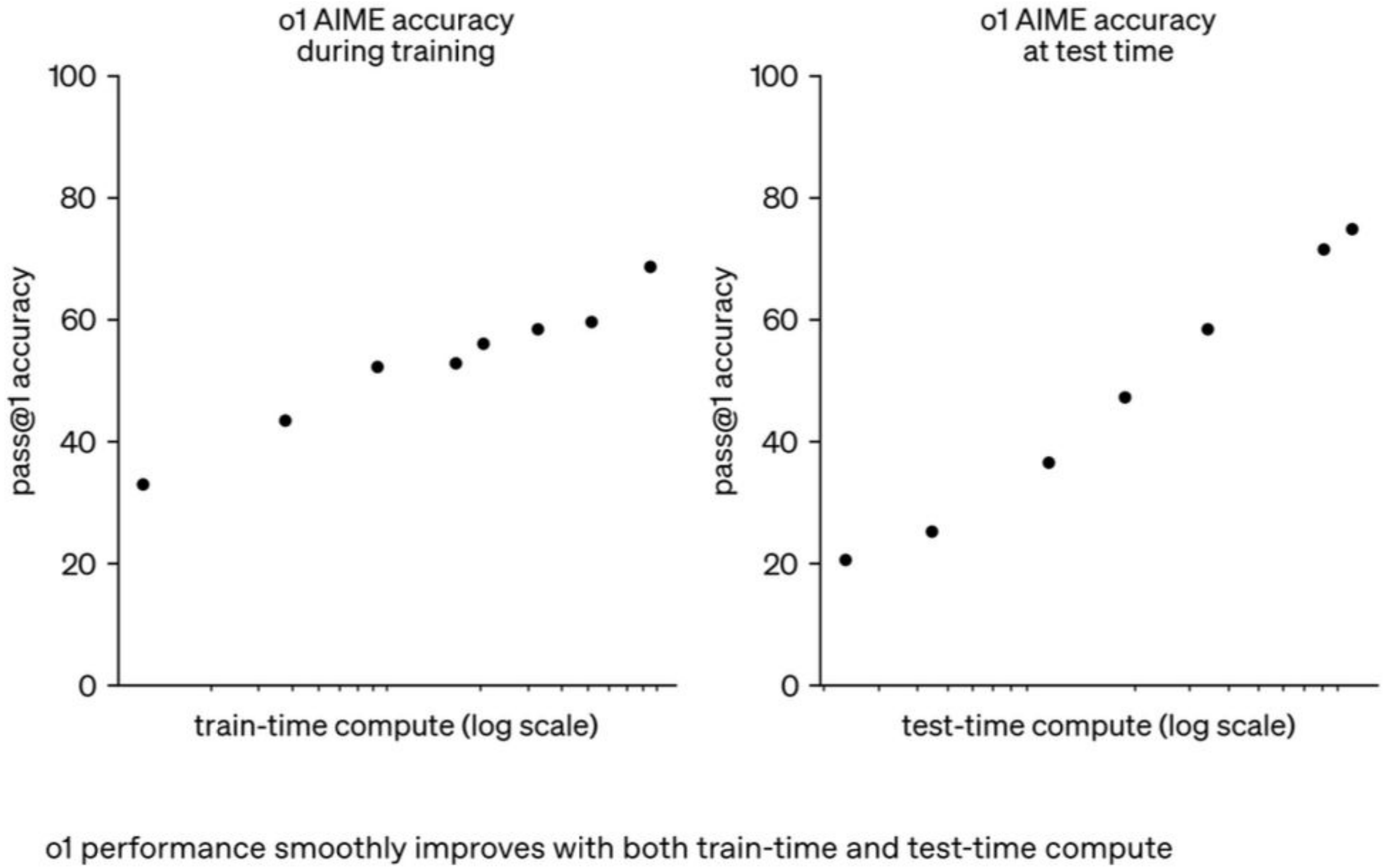
Pretraining Scaling Law, OpenAI, 2020

RL Scaling Curve, OpenAI, 2024

推理模型，范式转变

Scaling RL, test-time compute

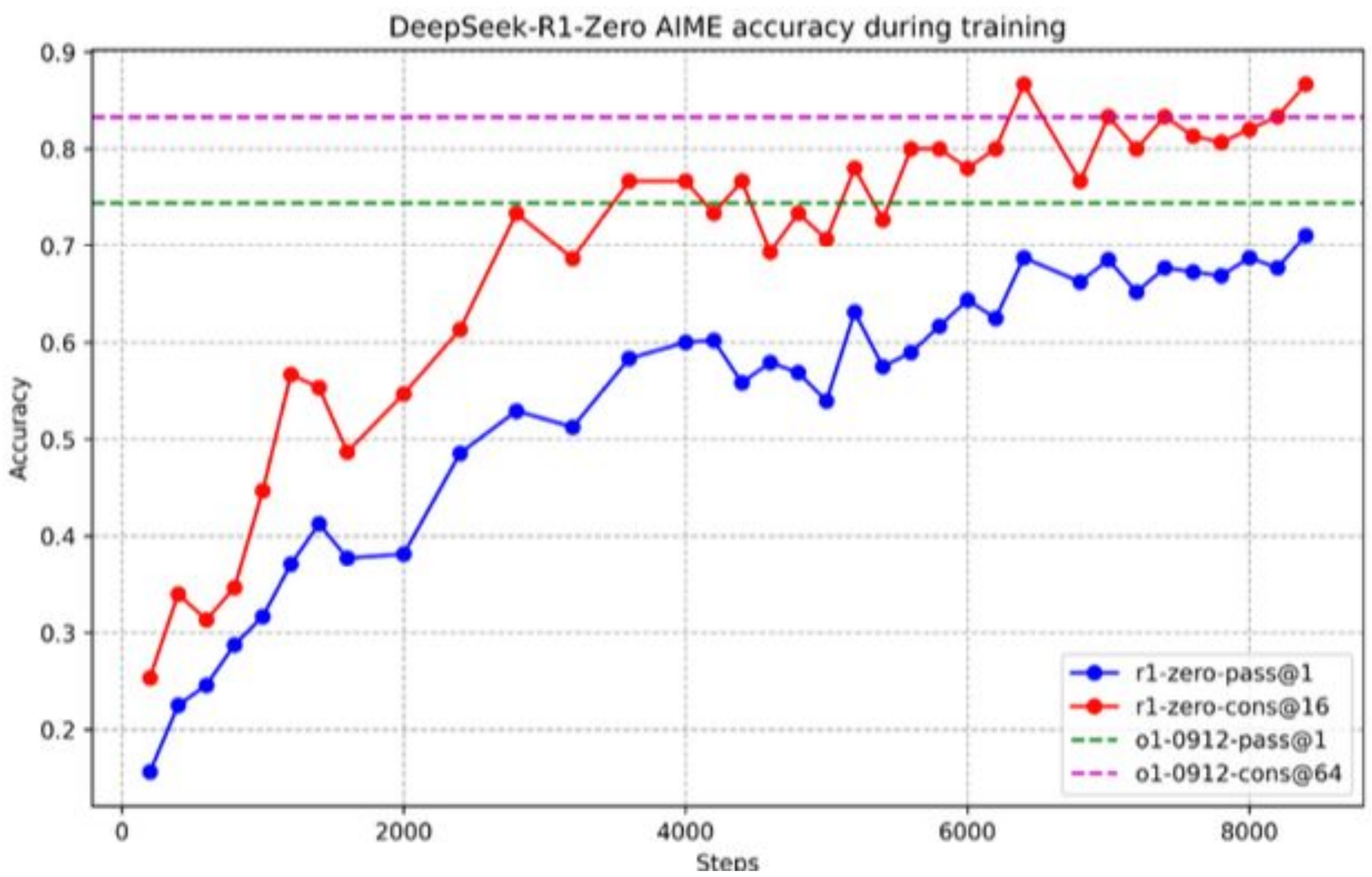
- RL is Scalable
  - 越多RL算力，越高性能
- Think before responde
  - 思维链越长，回答越好

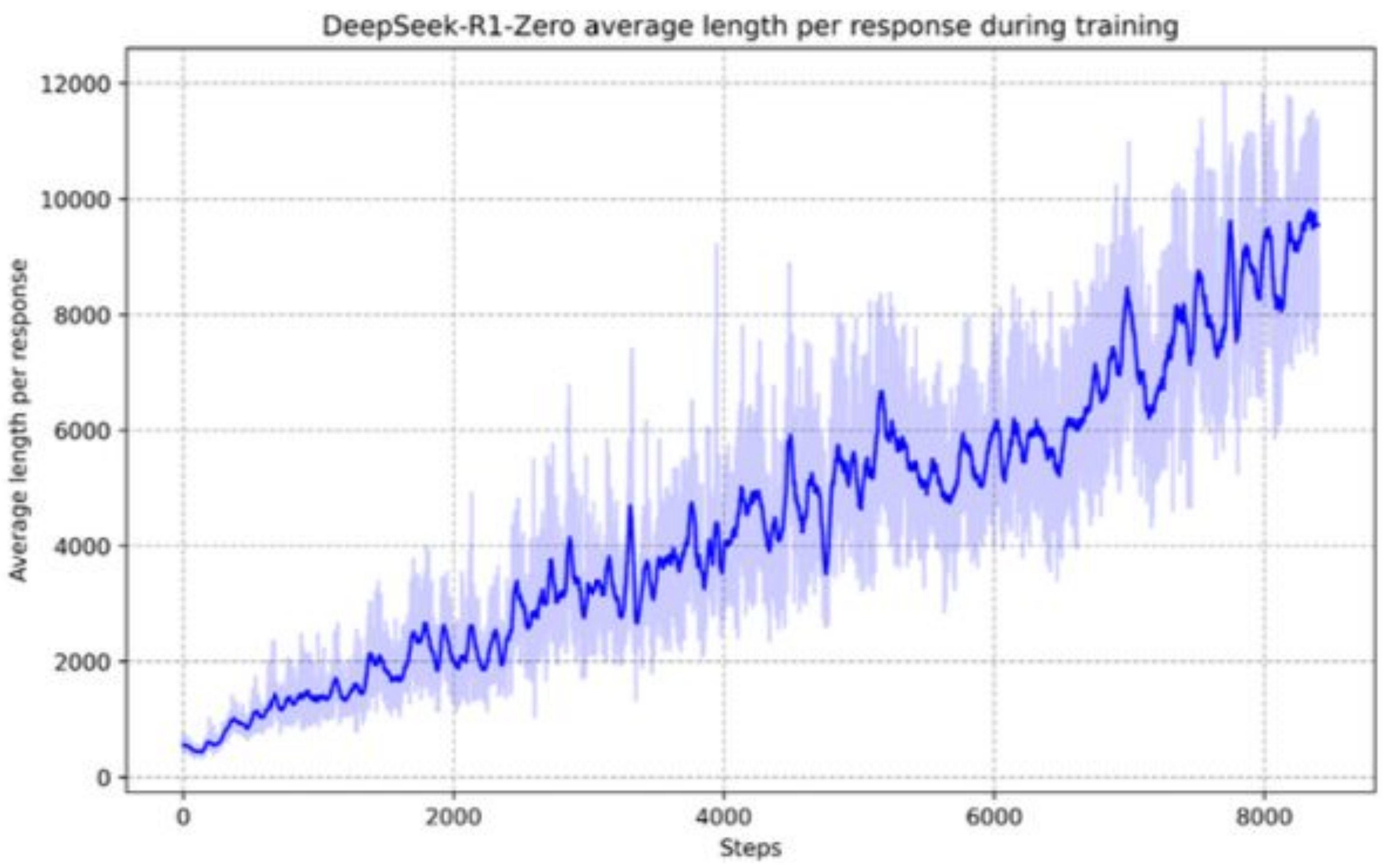


RL Scaling Curve, OpenAI, 2024

DeepSeek R1 (Zero) 揭秘推理模型

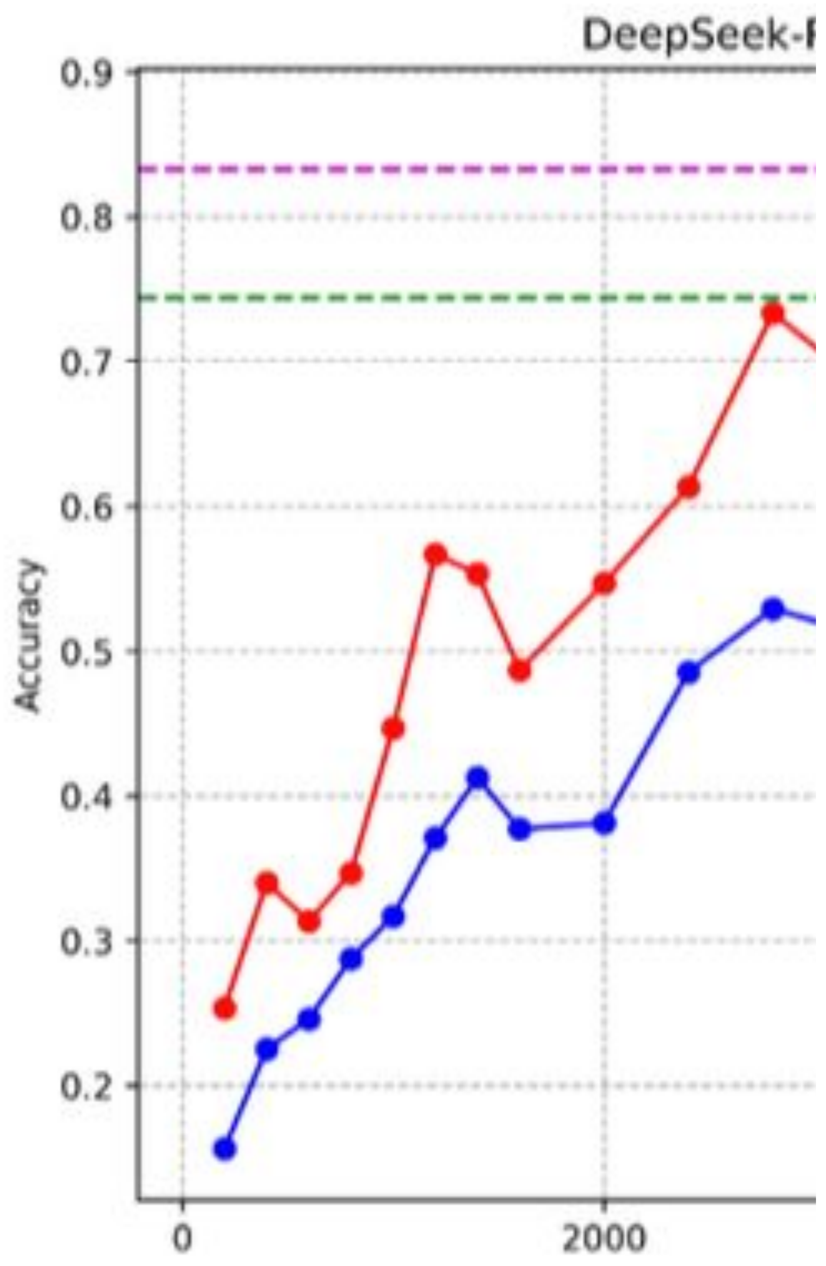
- DeepSeek v3 + GRPO against rule-based reward
  - 长思维链自动涌现（无需SFT）





DeepSeek R1 (Zero) 揭秘推理模型

- DeepSeek v3 + GRPO against rule-based reward
  - 长思维链自动涌现（无需SFT）



Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both $\cdots$

$\left(\sqrt{a - \sqrt{a + x}}\right)^2 = x^2 \implies a - \sqrt{a + x} = x^2.$

Rearrange to isolate the inner square root term:

$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$

$\ldots$

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$

We started with the equation:

$\sqrt{a - \sqrt{a + x}} = x$

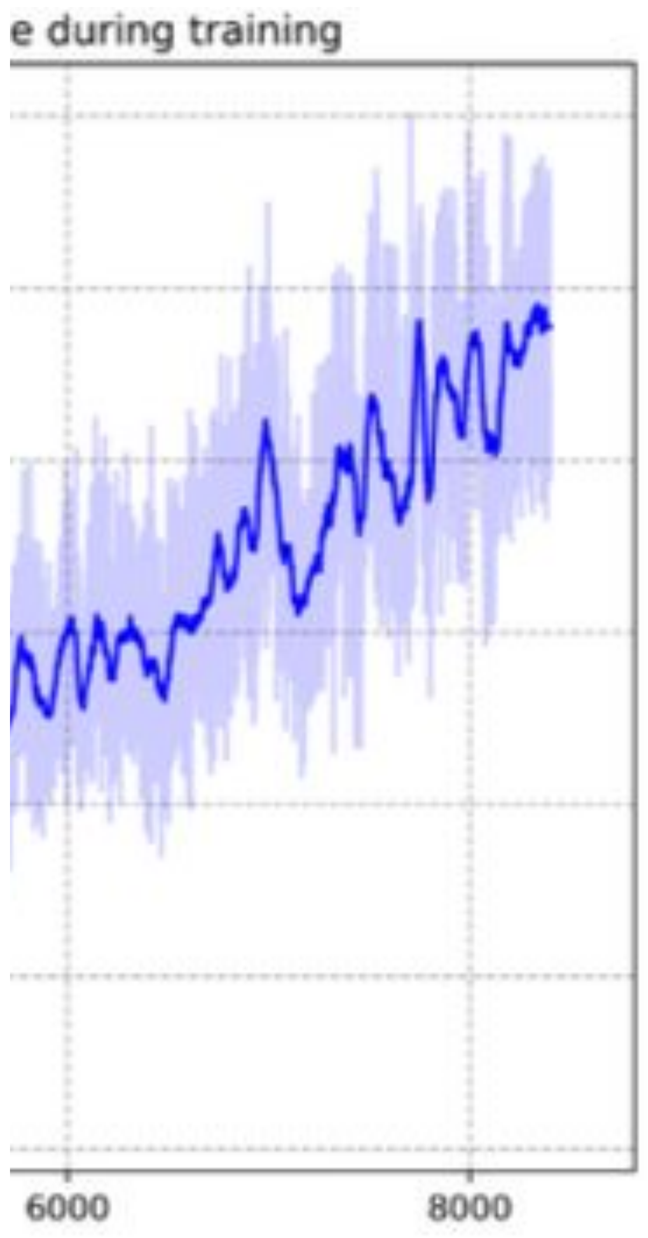
First, let's square both sides:

$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$

Next, I could square both sides again, treating the equation: $\cdots$

$\ldots$

Table 3 | An interesting "aha moment" of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us, allowing us to witness the power and beauty of reinforcement learning.



TinyZero – 推理模型的最小化 实现

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, Alane Suhr
UC Berkeley, UIUC, Independent

Berkeley AI Research

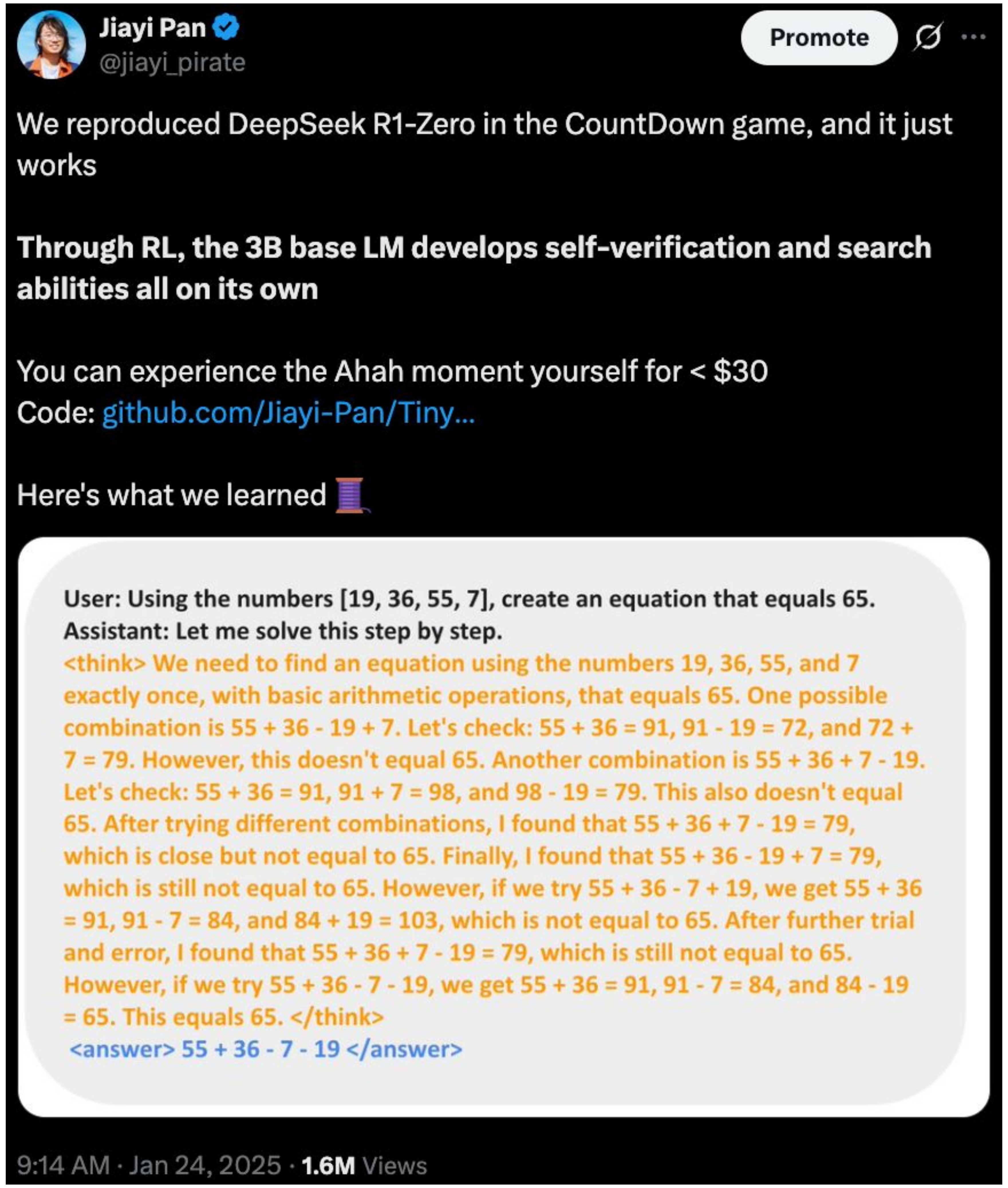
TinyZero – 推理模型的最小化 实现

推理模型的最小化实现

- 用3B模型，$30复现推理模型的核心结果
- 模型从简单的回复出发，慢慢涌现出长思维链与复杂的推理模式觎 – 如Search, Backtracking

核心观察：

- 降低模型能力的同时降低任务复杂度，我们仍然可以观测到推理模型涌现的各种有趣性质 – 便于实验、迭代



Jiayi Pan
@jiayi_pirate
Promote

We reproduced DeepSeek R1-Zero in the CountDown game, and it just works

Through RL, the 3B base LM develops self-verification and search abilities all on its own

You can experience the Ahah moment yourself for < $30
Code: github.com/Jiayi-Pan/Tiny...

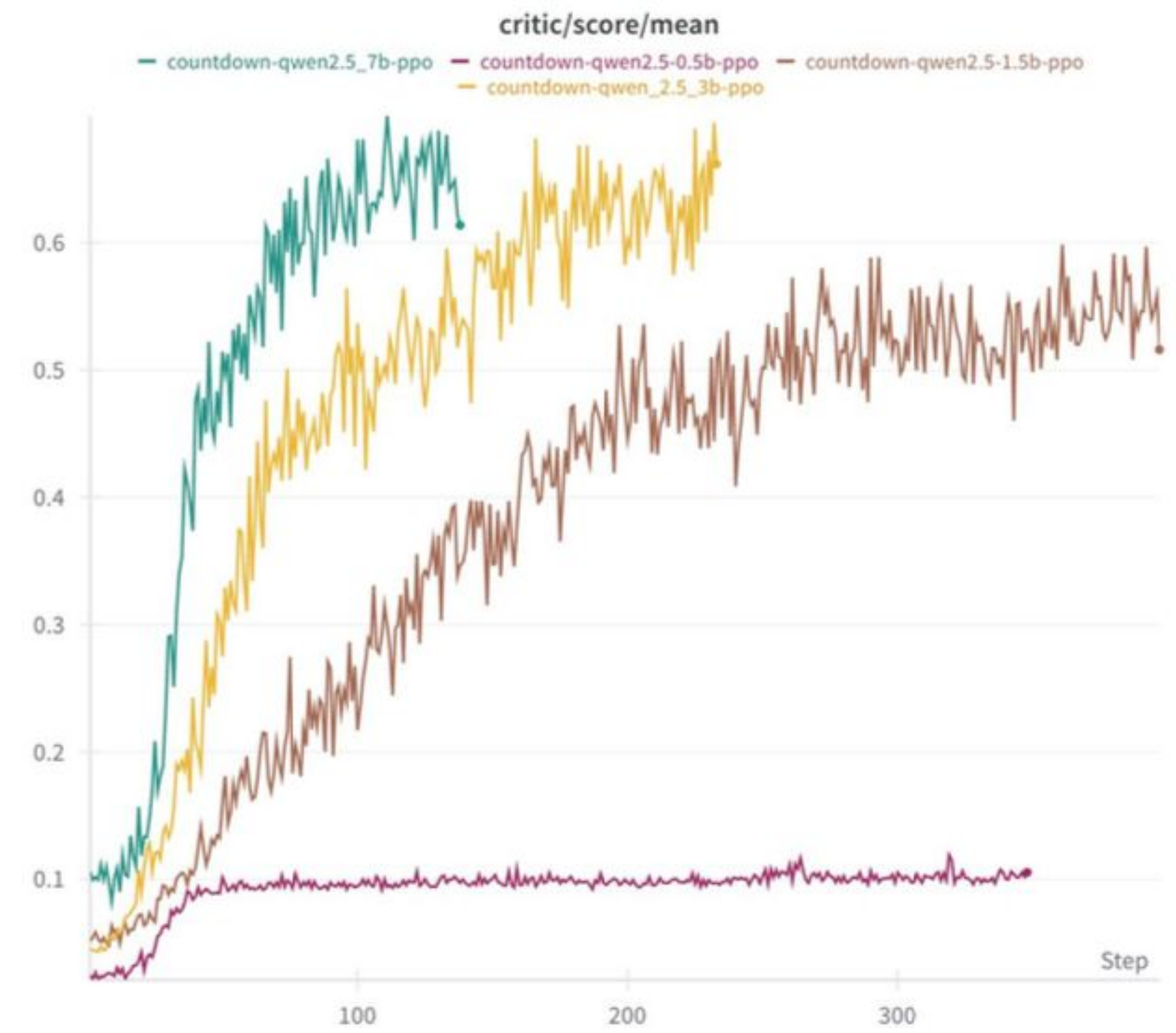
Here's what we learned

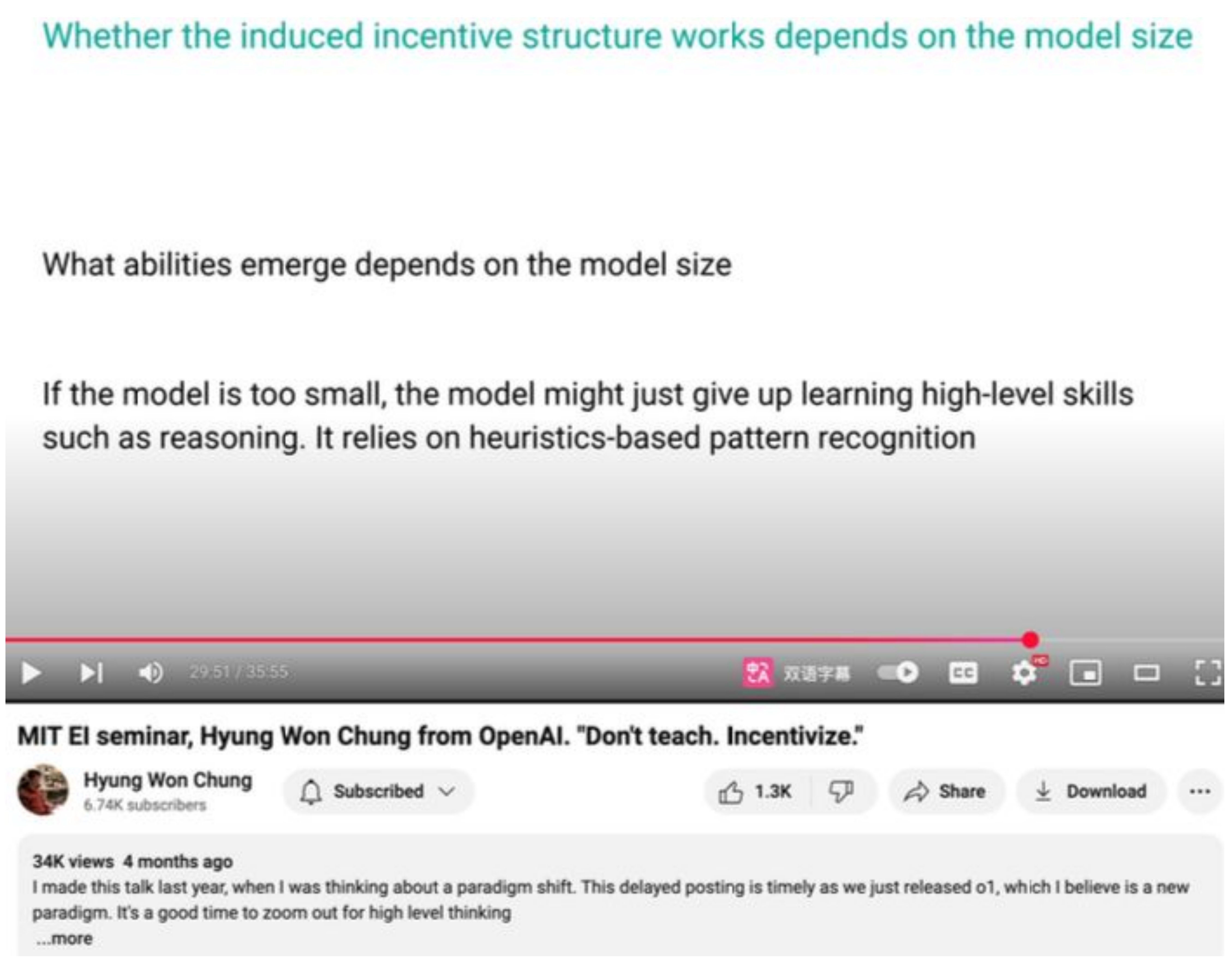
User: Using the numbers [19, 36, 55, 7], create an equation that equals 65.
Assistant: Let me solve this step by step.
<think> We need to find an equation using the numbers 19, 36, 55, and 7 exactly once, with basic arithmetic operations, that equals 65. One possible combination is 55 + 36 - 19 + 7. Let's check: 55 + 36 = 91, 91 - 19 = 72, and 72 + 7 = 79. However, this doesn't equal 65. Another combination is 55 + 36 + 7 - 19. Let's check: 55 + 36 = 91, 91 + 7 = 98, and 98 - 19 = 79. This also doesn't equal 65. After trying different combinations, I found that 55 + 36 + 7 - 19 = 79, which is close but not equal to 65. Finally, I found that 55 + 36 - 19 + 7 = 79, which is still not equal to 65. However, if we try 55 + 36 - 7 + 19, we get 55 + 36 = 91, 91 - 7 = 84, and 84 + 19 = 103, which is not equal to 65. After further trial and error, I found that 55 + 36 + 7 - 19 = 79, which is still not equal to 65. However, if we try 55 + 36 - 7 - 19, we get 55 + 36 = 91, 91 - 7 = 84, and 84 - 19 = 65. This equals 65. </think>
<answer> 55 + 36 - 7 - 19 </answer>

9:14 AM · Jan 24, 2025 · **1.6M** Views

长思维链的 涌现和 预训练模型能力 强相关

只有在足够强的基座模型上，强化模型才会激发长思维链





强化学习让模型自发学会不同策略

TinyZero模型在不同任务上自发找出了不同的解题策略，利用test time compute的方法

User: Using the numbers [19, 36, 55, 7], create an equation that equals 65.
Assistant: Let me solve this step by step.
<think> We need to find an equation using the numbers 19, 36, 55, and 7 exactly once, with basic arithmetic operations, that equals 65. One possible combination is 55 + 36 - 19 + 7. Let's check: 55 + 36 = 91, 91 - 19 = 72, and 72 + 7 = 79. However, this doesn't equal 65. Another combination is 55 + 36 + 7 - 19. Let's check: 55 + 36 = 91, 91 + 7 = 98, and 98 - 19 = 79. This also doesn't equal 65. After trying different combinations, I found that 55 + 36 + 7 - 19 = 79, which is close but not equal to 65. Finally, I found that 55 + 36 - 19 + 7 = 79, which is still not equal to 65. However, if we try 55 + 36 - 7 + 19, we get 55 + 36 = 91, 91 - 7 = 84, and 84 + 19 = 103, which is not equal to 65. After further trial and error, I found that 55 + 36 + 7 - 19 = 79, which is still not equal to 65. However, if we try 55 + 36 - 7 - 19, we get 55 + 36 = 91, 91 - 7 = 84, and 84 - 19 = 65. This equals 65. </think>
<answer> 55 + 36 - 7 - 19 </answer>

User: Give me the answer of the following equation: 129 * 14 [...]
Assistant: Let me solve this step by step.
<think> To solve the equation 129 * 14, you can use the distributive property of multiplication over addition to break down the problem into smaller, more manageable parts. Here's the step-by-step reasoning:
1. Break down 14 into 10 + 4. Now the equation looks like 129 * (10 + 4).
2. Use the distributive property to multiply 129 by each part of the sum separately: 129 *10 + 129* 4.
3. Calculate each term: 1290 (for 129 *10) and 516 (for 129* 4).
4. Add the results together: 1290 + 516 = 1806.
So, the answer to the equation 129 * 14 is
<answer> 1806 </answer>

推理模型的下一步

- 探索推理模型的训练科学，找出更优算法
- 大规模LLM RL的框架、基础设施
- 加入多轮的工具使用 (Agent / O3)
- 构建更多的训练环境 (Hard tasks with reward)
- 如何将RL推向更难验证的任务上（创意写作, ...）
- 比长思维链更高效的Test-time Scaling算法

Learning Adaptive Parallel Reasoning with Language Models

Jiayi Pan*, Xiuyu Li*, Long Lian*, Charlie Victor Snell, Yifei Zhou,
Adam Yala, Trevor Darrell, Kurt Keutzer, Alane Suhr

UC Berkeley and UCSF   * Equal Contribution

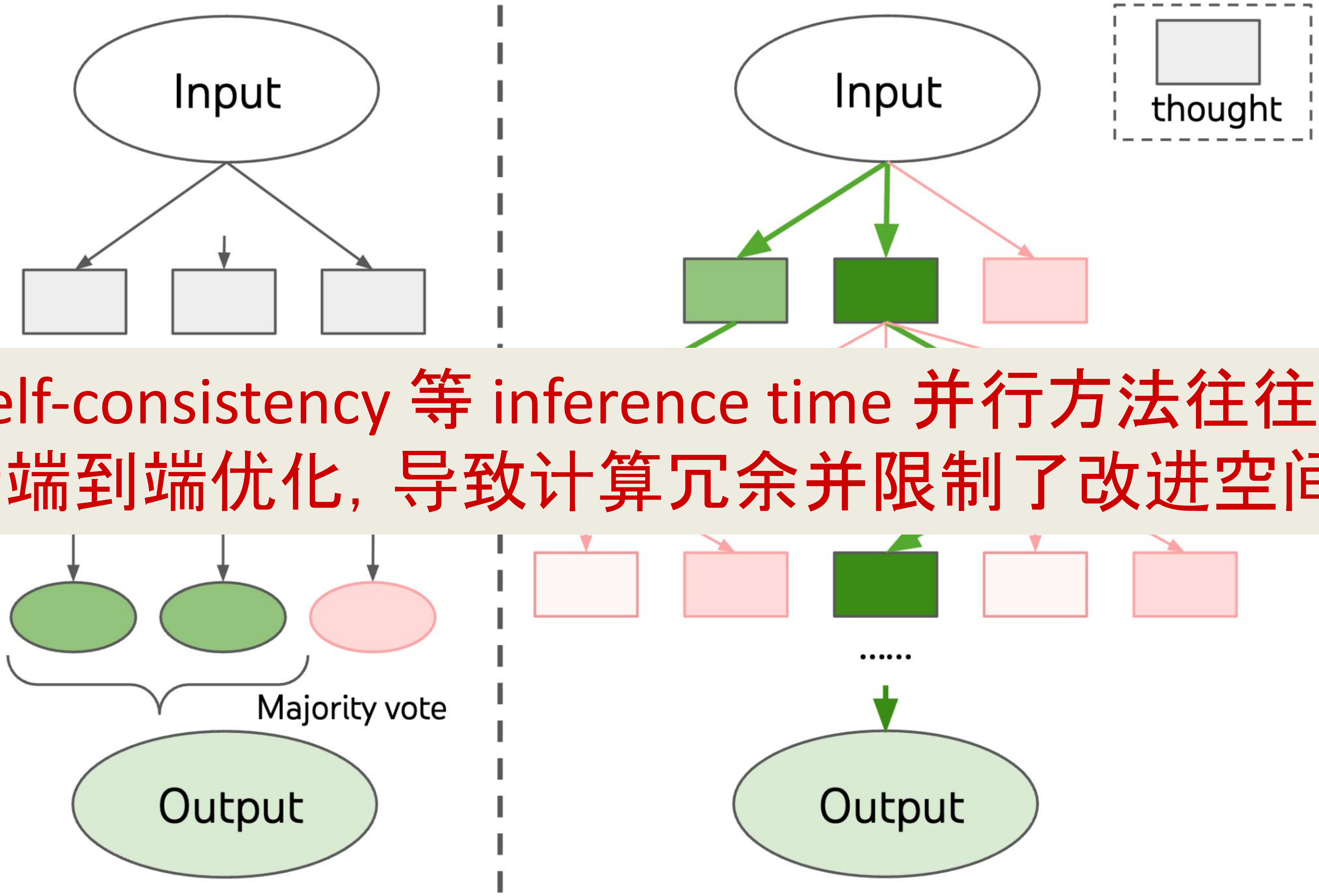
Berkeley AI Research

长思维链的挑战：上下文窗口和延 迟

- 思维链长度可达数万 tokens (Claude 3.7 Sonnet-thinking has up to 32k CoT budget)
- 等待时长可达数分钟
- 未来只会需要更多的 test-time compute

想法：并行调用语言模型

即使本质上并非多步骤任务，我们仍会手动组合多个 LLM 调用来完成下游任务，以提升搜索/推理能力。

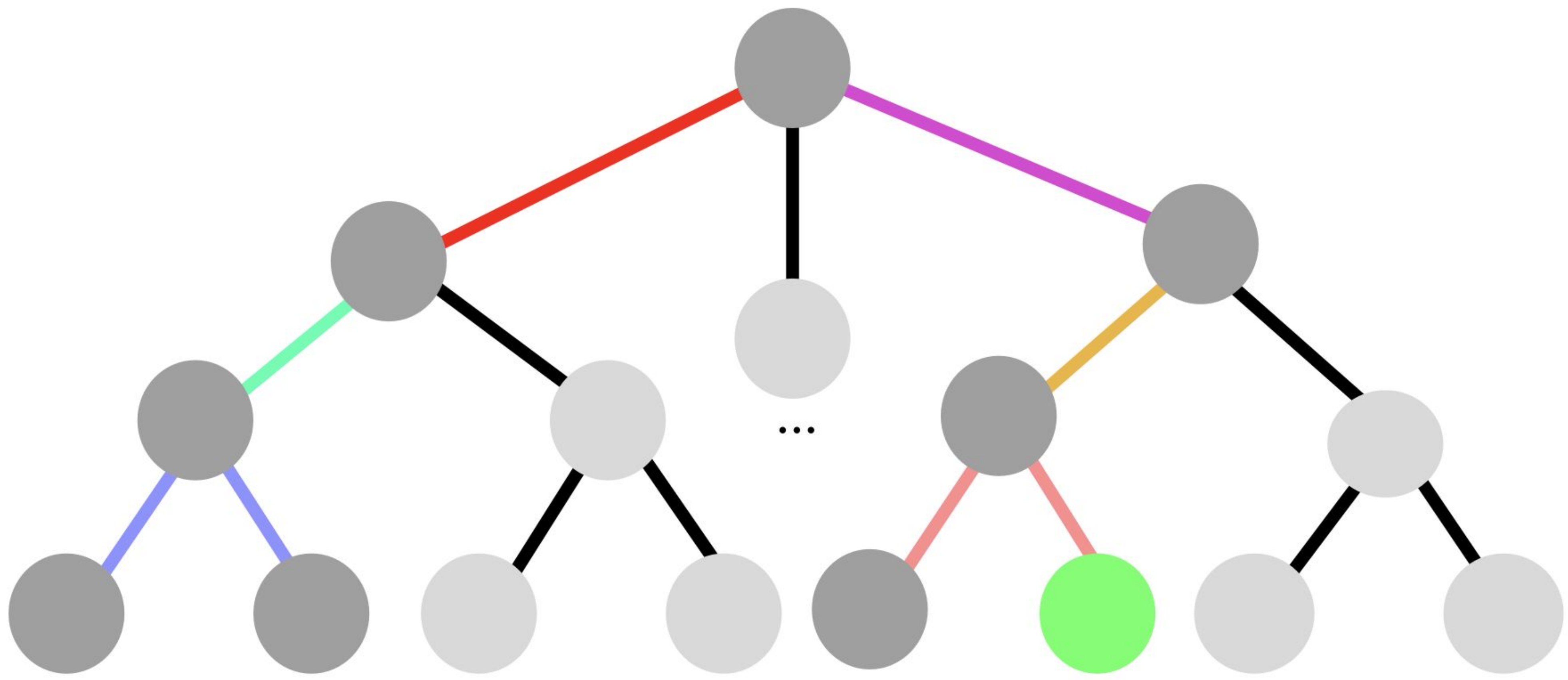


Self Consistency　　Tree of Thoughts

Limitation：诸如 self-consistency 等 inference time 并行方法往往缺乏推理路径间的协调，且未进行端到端优化，导致计算冗余并限制了改进空间。

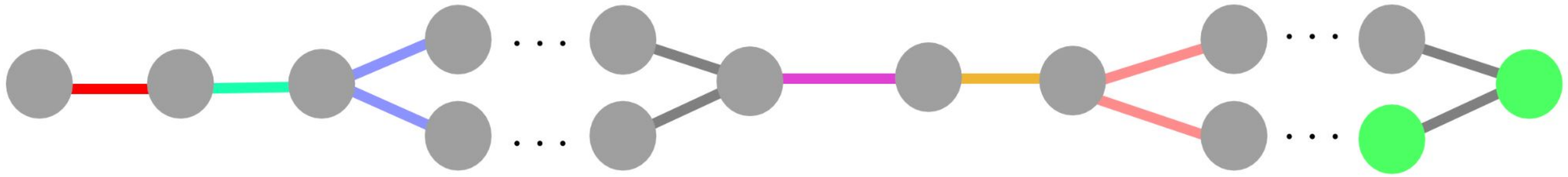
想法:并行调用语言模型

自适应并行推理 (APR):教导语言模型端到端自适应地编排串行和并行计算



推理可被视为遍历一系列中间步骤的树状结构，以寻找最终解决方案。

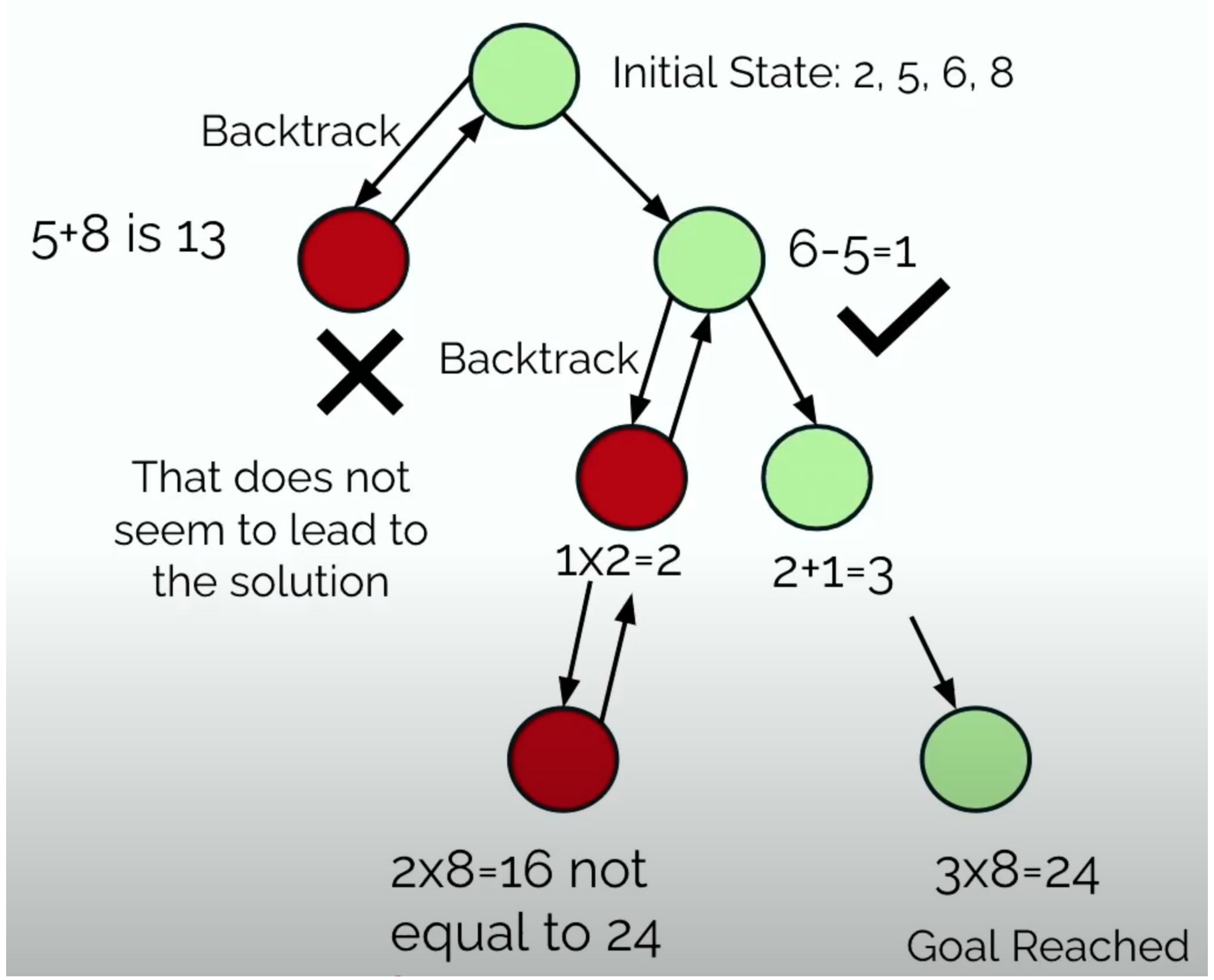
以往的思维链方法即便存在**可并行组件**，仍对推理树进行序列化处理。



我们的方法训练模型**生成子线程**，这些子线程**并行处理子任务**。一旦完成，它们将被**合并回主线程**。

案例研究：Countdown

教授 LLMs 通过展示搜索流 (stream of search) ——一种扁平化的字符串，代表包含探索、验证与回溯的搜索路径——来学习和解决复杂问题

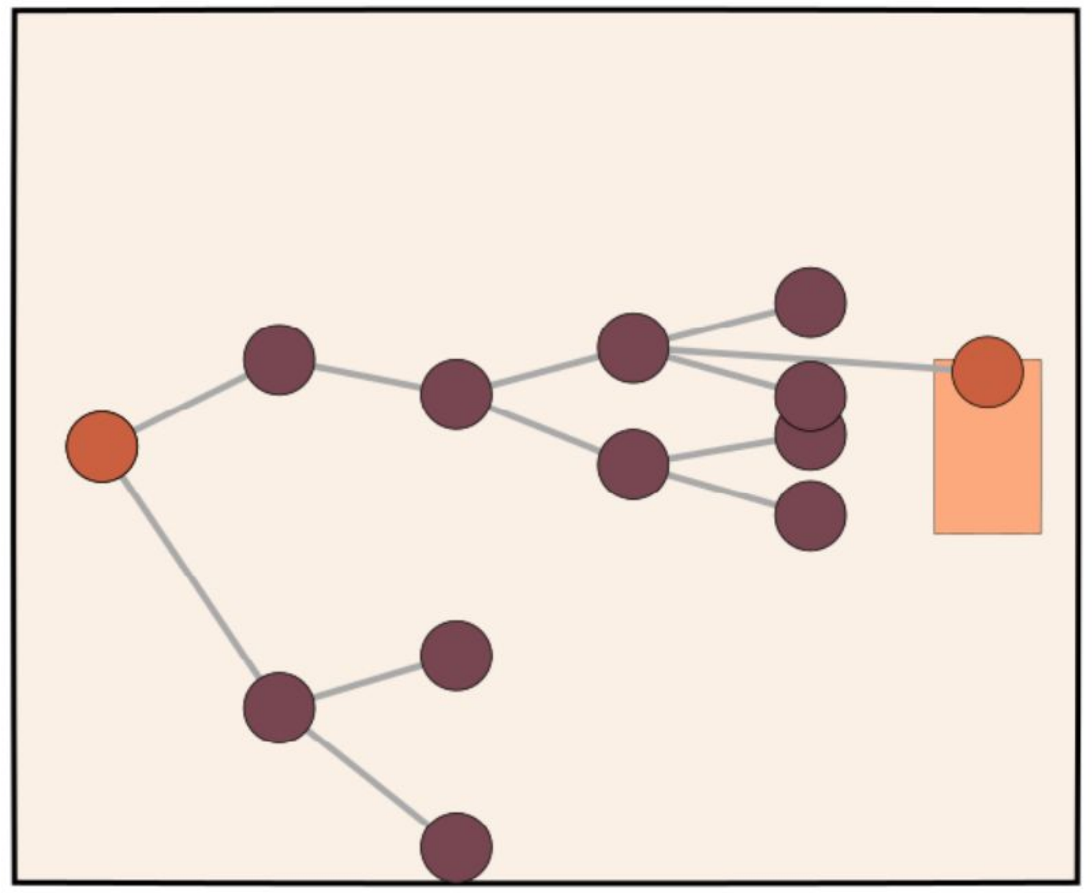


Initial State: 2, 5, 6, 8
Let's Think Step by Step:
Step 1: Let's try 5+8=13. This does not seem promising.
Going back to the start and trying again.
Step 1: 6-5=1. This seems good.
Step 2: 1x2=2. This seems good.
Step 3: 2x8=16. This is not the target.
Going back to step 2 and trying again.
Step 2: 2+1=3.
Step 3: 3x8=24. This is the target. Goal Reached!

Gandhi et al. Stream of Search (SoS): Learning to Search in Language. COLM 2024.

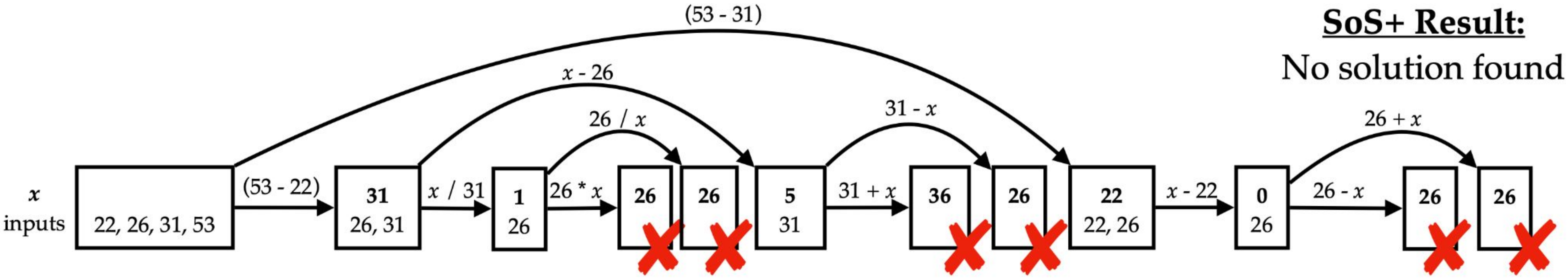
案例研究：Countdown

- 树状搜索通过广度优先/深度优先探索多条路径
- 转换成线性序列（流）
- 允许模型在流中犯错



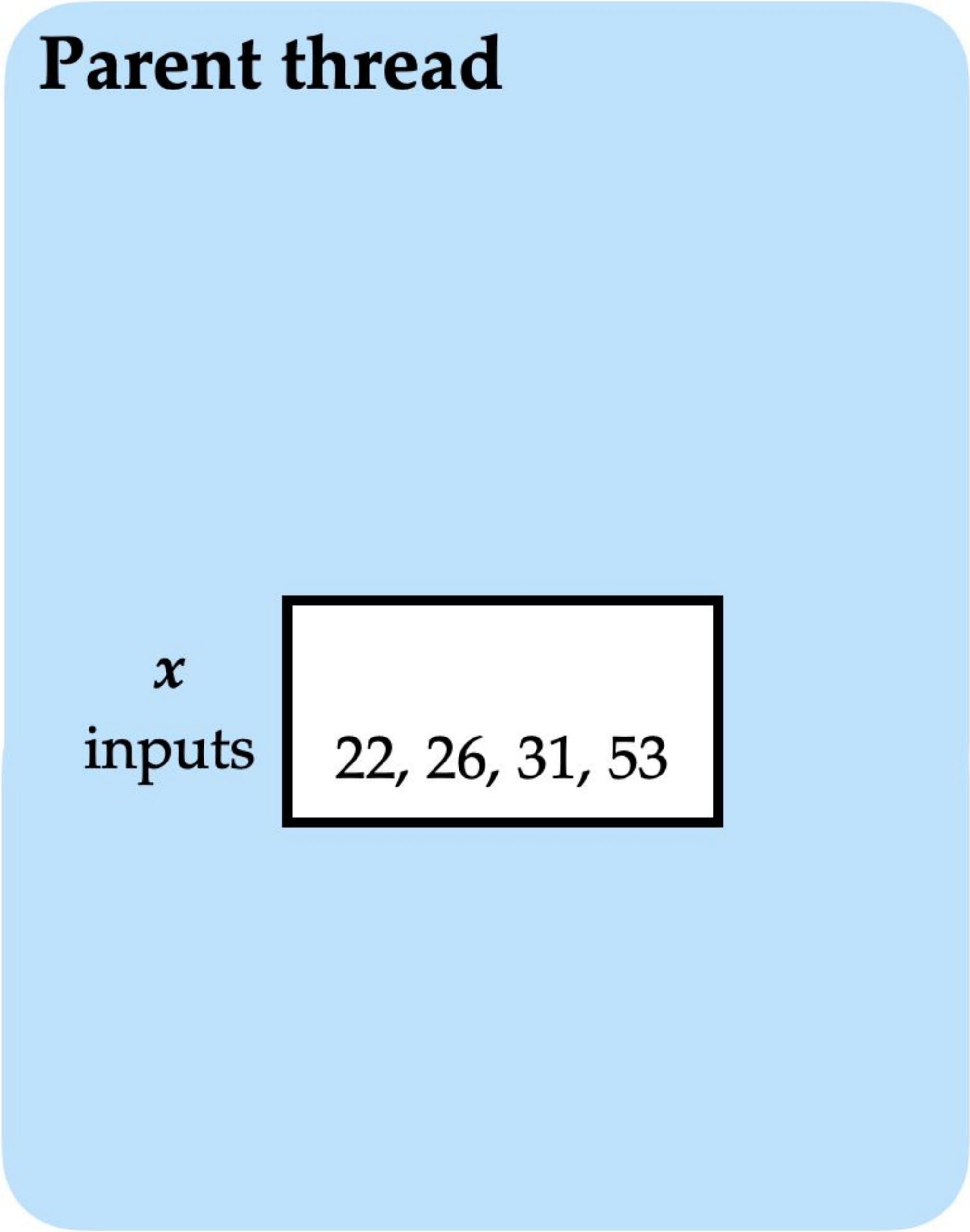
长思维链的挑战：上下文窗口

序列化的 CoT 生成过长的输出，导致耗尽上下文窗口仍无法得到结果，同时延迟增加



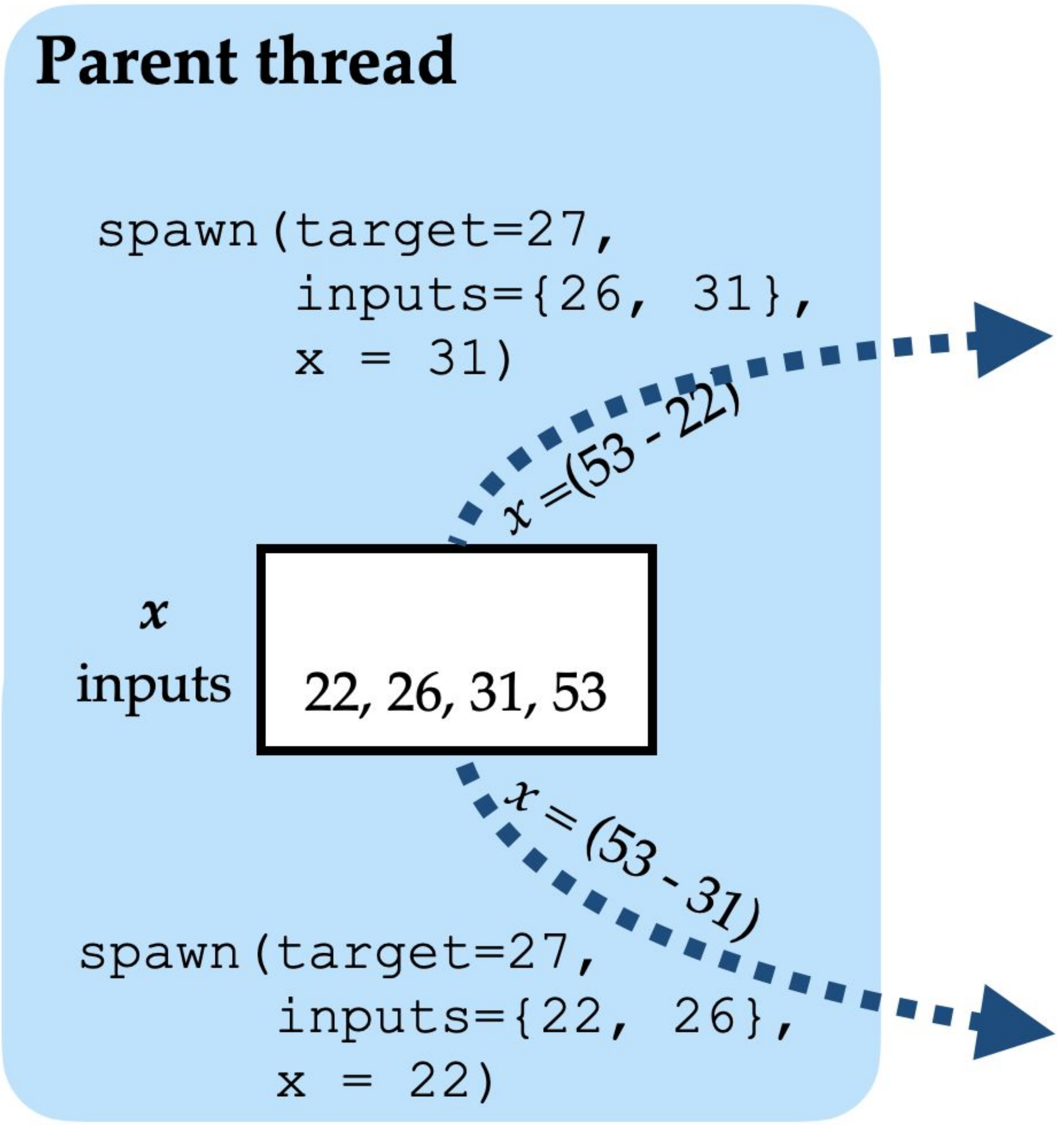
`spawn()` 和 `join()`

`spawn()`：当 `spawn(msgs)` 被调用时，将生成多个并行执行的子线程。子线程之间的协调是通过主线程为每个子线程传递不同的上下文来实现的。



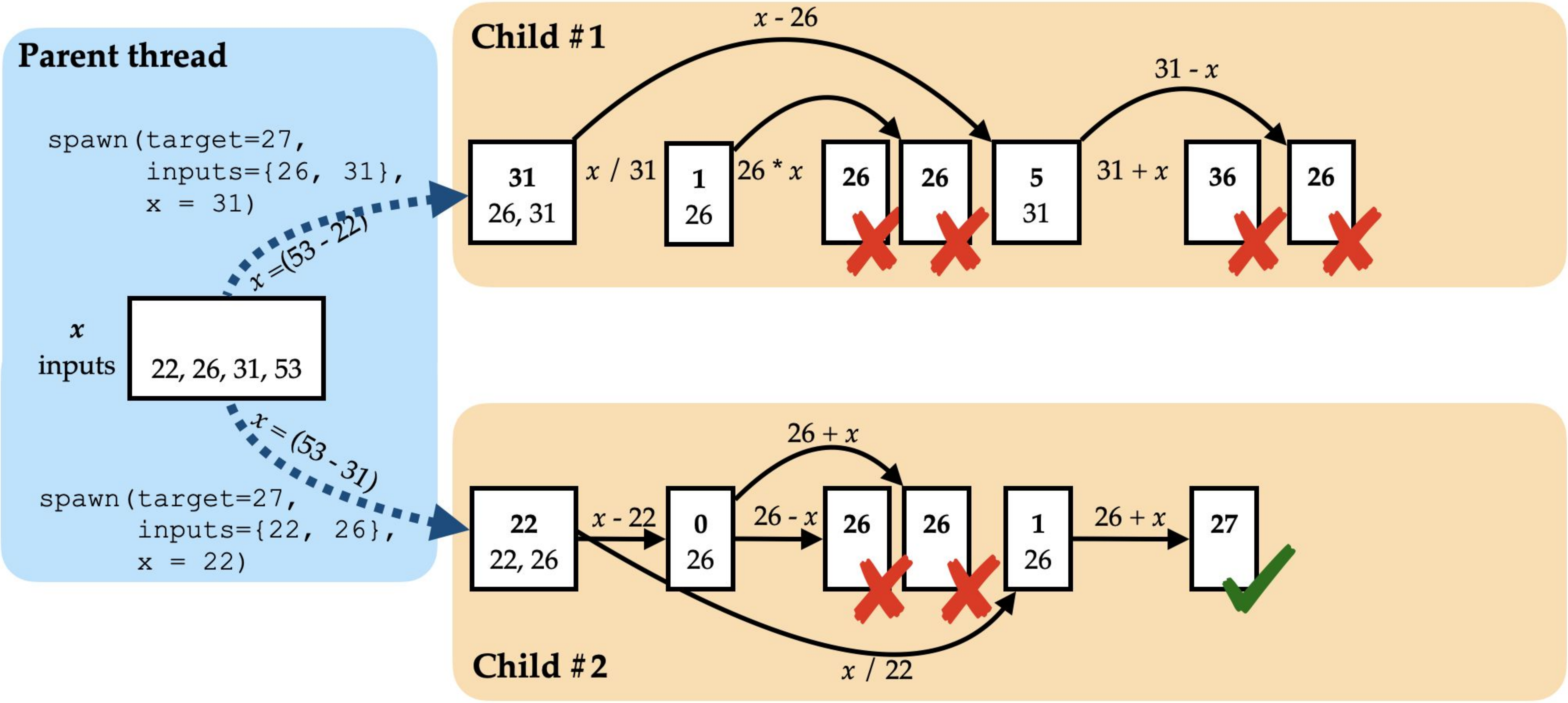
spawn() 和 join()

spawn()：当 spawn(msgs) 被调用时，将生成多个并行执行的子线程。子线程之间的协调是通过主线程为每个子线程传递不同的上下文来实现的。



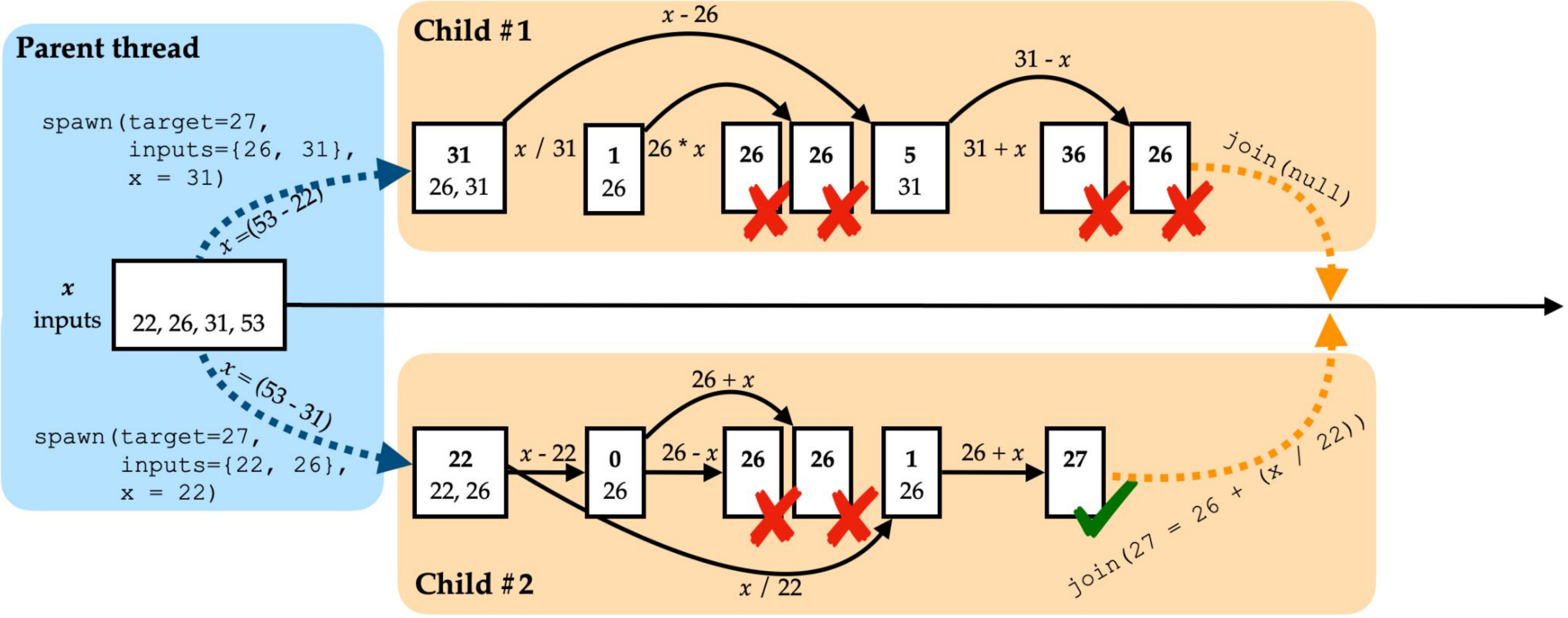
spawn() 和 join()

子线程独立但同步地使用与主线程相同的语言模型执行推理，每个线程的上下文仅限于主线程通过消息传递的 token。



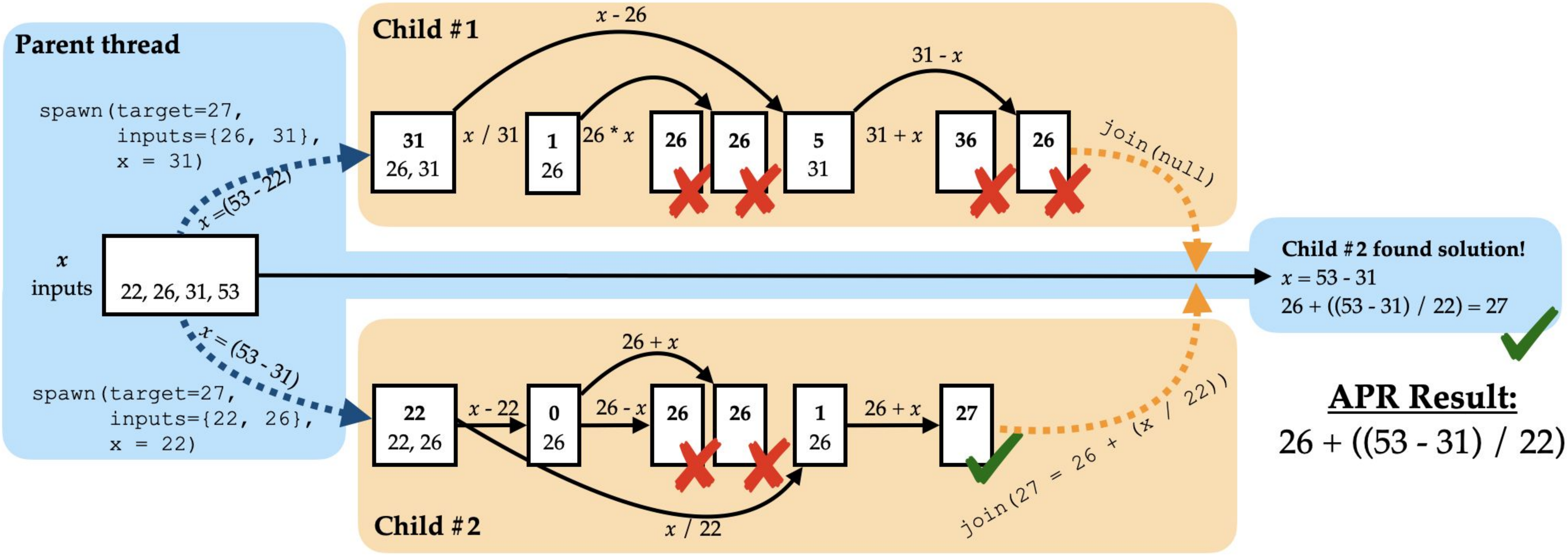
spawn() 和 join()

join()：当子线程执行 join(msg) 操作时，它将终止自身的推理过程，并指定一系列 tokens 作为 msg 返回给主线程。子线程仅返回成功的解决路径，从而实现与主线程间简洁且有针对性的通信。对于搜索失败的子线程，则不会返回任何解决路径。

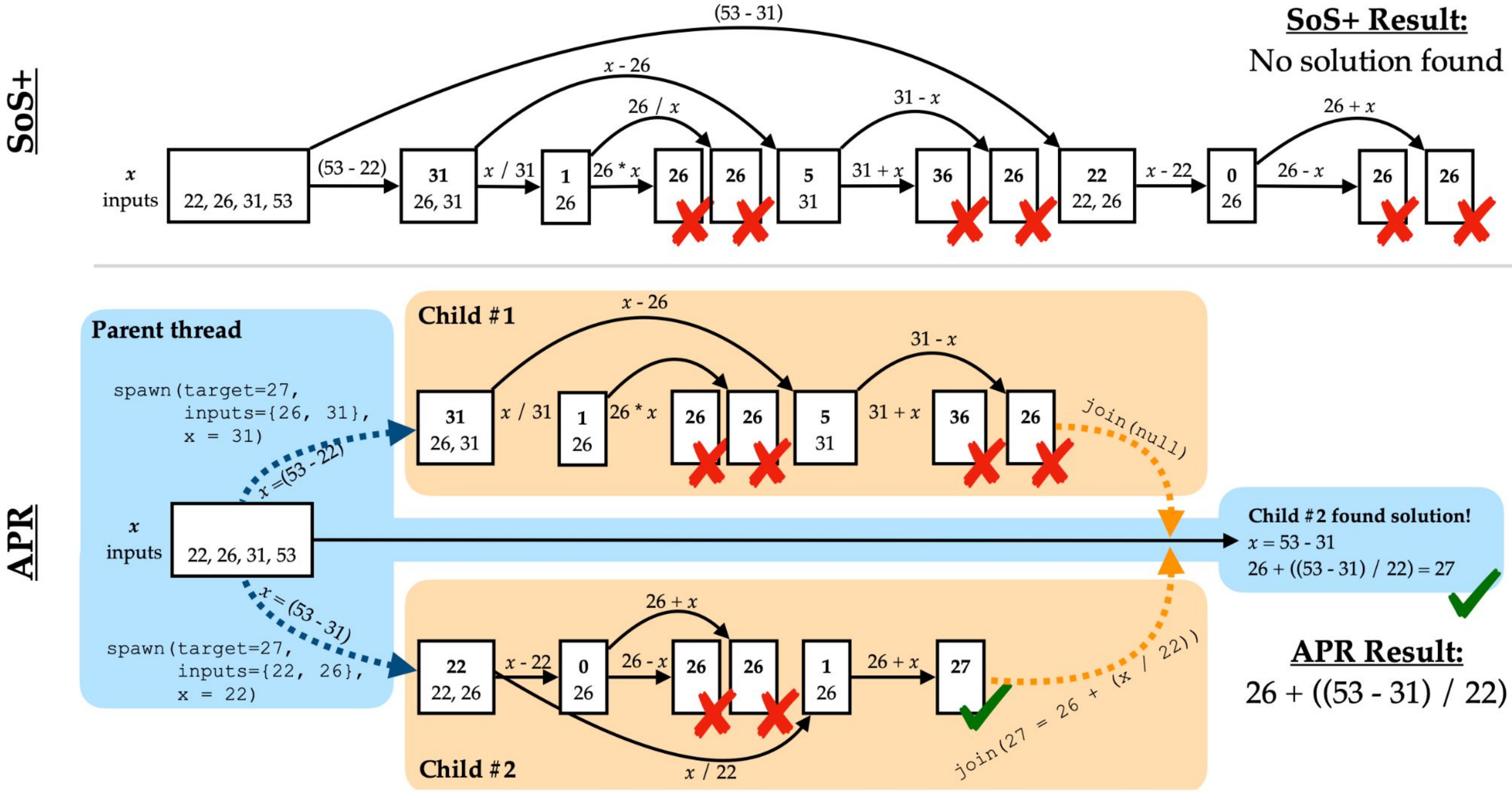


spawn() 和 join()

一旦所有子线程终止并返回相应消息，主线程的执行将继续进行，其执行条件仅取决于生成子线程前的原有上下文及子线程返回的消息。



APR 算法示例



SoS+
APR
x
inputs
22, 26, 31, 53
(53 - 22)
(53 - 31)
x - 26
26 / x
31 - x
26 + x
x / 31
26 * x
31 + x
x - 22
26 - x
SoS+ Result:
No solution found
Parent thread
spawn(target=27, inputs={26, 31}, x = 31)
spawn(target=27, inputs={22, 26}, x = 22)
x =(53 - 22)
x = (53 - 31)
Child #1
join(null)
Child #2
x / 22
join(27 = 26 + (x / 22))
Child #2 found solution!
x = 53 - 31
26 + ((53 - 31) / 22) = 27
APR Result:
26 + ((53 - 31) / 22)

APR 算法示例

Parent thread

Moving to Node #0 Current State: 27:[22, 26, 31, 53], Operations: []
Exploring Operation: 53-22=31, Resulting Numbers: [26, 31, 31] Generated Node #0,0: 27:[26, 31, 31] Operation: 53-22=31
…
<Calling Sub Searches> <Start Sub Search 0 at level 1> Moving to Node #0,0 <Start Sub Search 1 at level 1> Moving to Node #0,1 <End Calling Sub Searches>

Child #1

Moving to Node #0,0 Current State: 27:[26, 31, 31], Operations: ['53-22=31']
…
Exploring Operation: 31-5=26, Resulting Numbers: [26] 26,27 **unequal: No Solution No solution found.**

Child #2

Moving to Node #0,1 Current State: 27:[26, 31, 31], Operations: ['53-22=31']
…
Exploring Operation: 26+1=27, Resulting Numbers: [27] 27,27 **equal: Goal Reached**

Parent thread continued

…<Sub Searches>
<No Solution in Sub Search 0 at level 1 at Node #0,0>
<Goal Reached in Sub Search 1 at level 1 at Node #0,1>…<End Sub Searches>
27,27 equal: Goal Reached Moving to Node #0…

训练方法

- **Supervised initialization of parallel reasoning**：
  - 采用监督学习从头 pretrain 一个 200M 参数的语言模型，使其可以生成使用 spawn() 和 join() 的推理路径
  - 利用符号求解器（symbolic solver）生成推理轨迹
    - SoS+，它通过改造原始 SoS 求解器来生成无需 spawn() 和 join() 的序列化混合搜索路径
    - APR，它能创建具有并行化特性的混合演示，在其执行过程中会选择特定节点进行并行探索
  - 为了进行控制计算量的实验对比，我们训练了以 token 数量或者子线程数量为条件的模型

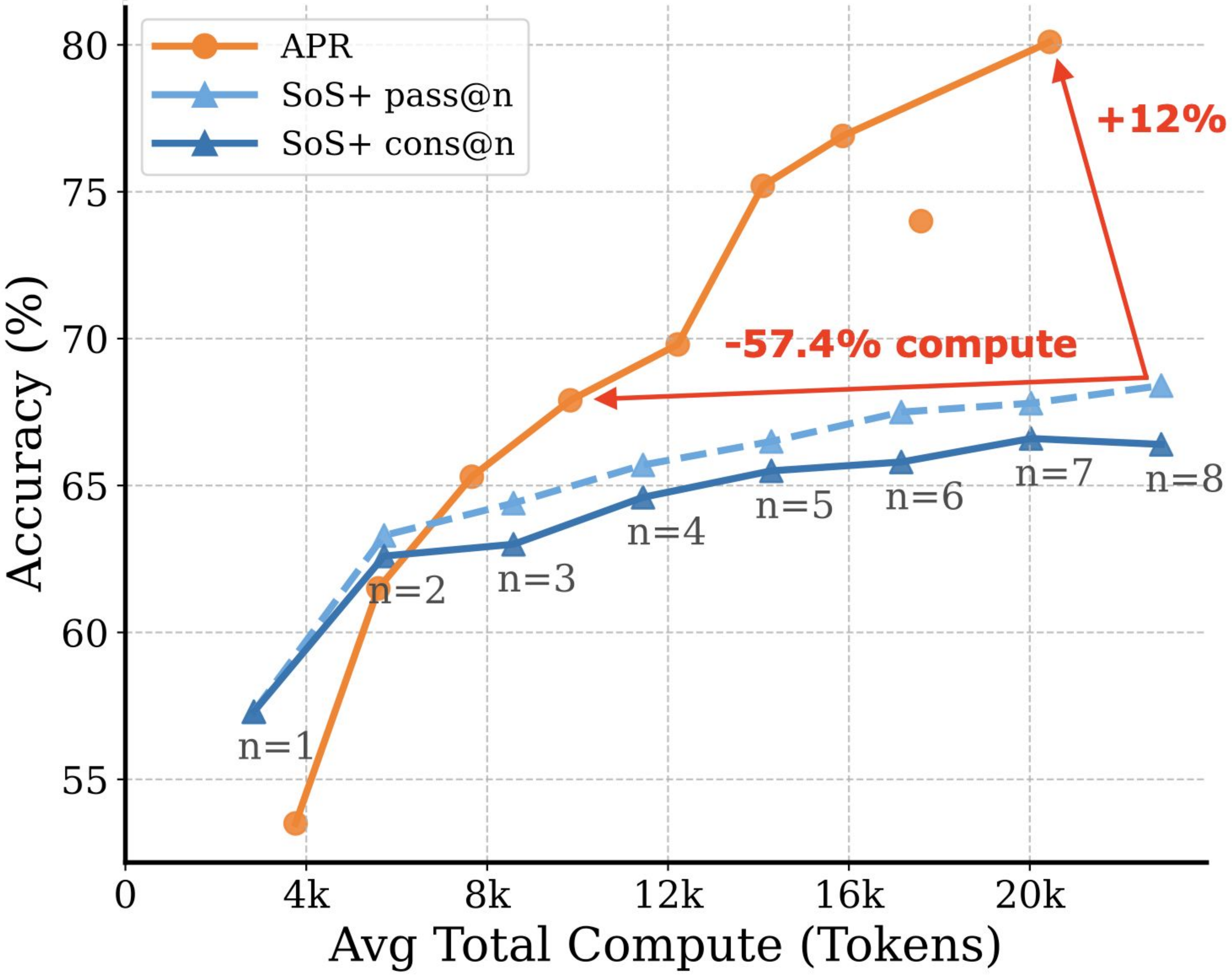
训练方法

- **Reinforcement learning for end-to-end policy optimization：**
  - 监督训练仅引导模型模仿所提供的演示，并未优化计算效率或推理效果
  - 用端到端的强化学习对语言模型进行微调，迭代地从当前策略中采样推理路径，根据其正确性分配 rewards，使用 GRPO 优化
  - 模型学会策略性地决定何时、如何及以何种广度调用子线程，通过平衡并行探索与上下文窗口限制之间的权衡来最大化性能

实验: Scaling with Higher Compute

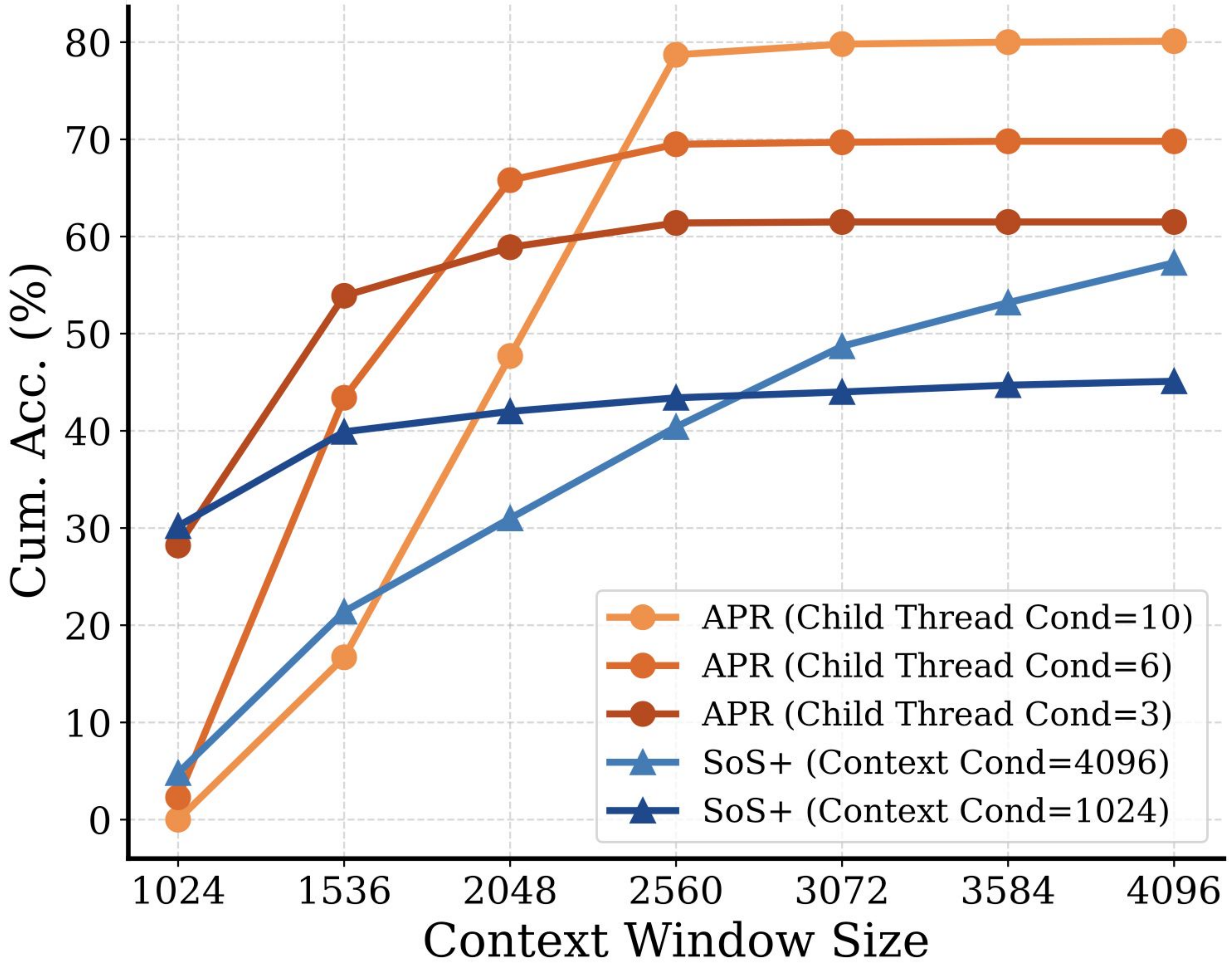
APR 通过给定子线程数量，SoS+ 通过采样数控制计算量

- **更优的扩展性:** 通过独立执行的子线程实现并行探索，APR 的性能能更有效地随计算量增加而扩展
  - APR 在低计算量情况下存在"并行性开销"——部分生成的 tokens 用于协调线程而非直接参与搜索
  - 随着计算量的增加，APR 显著优于 SoS+



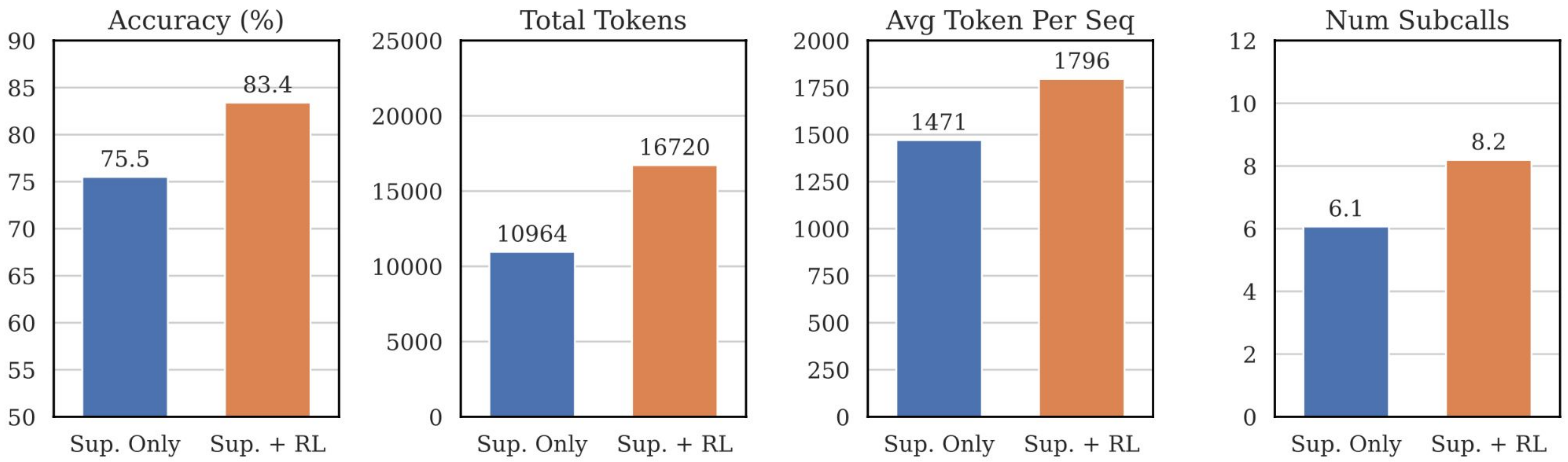
实验: Scaling with Context Window Size

- **相同上下文窗口内更高性能:**
  - 针对每种上下文窗口限制，我们采样一次并报告累计准确率:在每个窗口大小下，仅统计长度不超过该限制的结果
  - APR 将推理过程分配到并行线程中，无需将完整路径压缩至单个上下文窗口



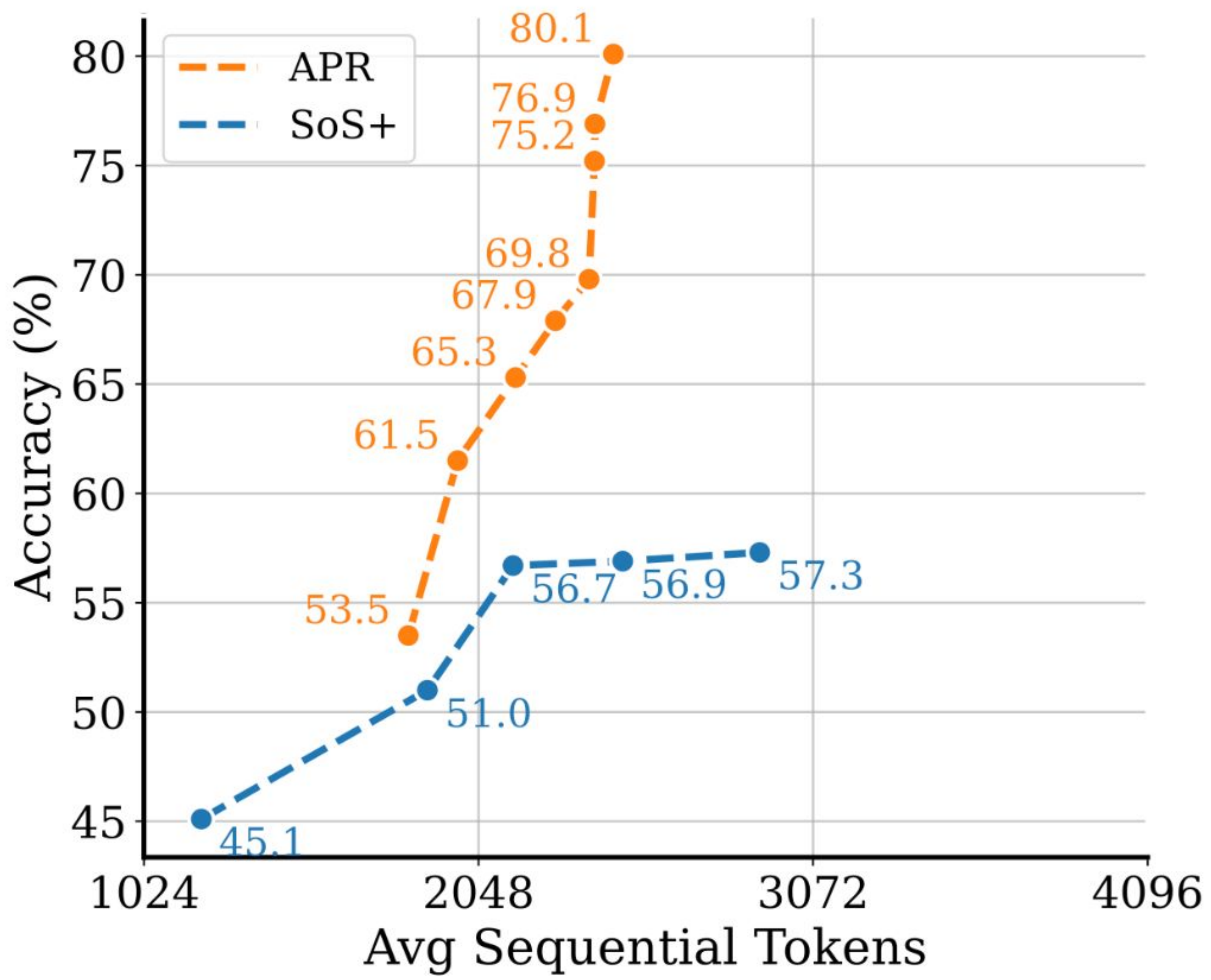
实验: End-to-end Optimization through RL

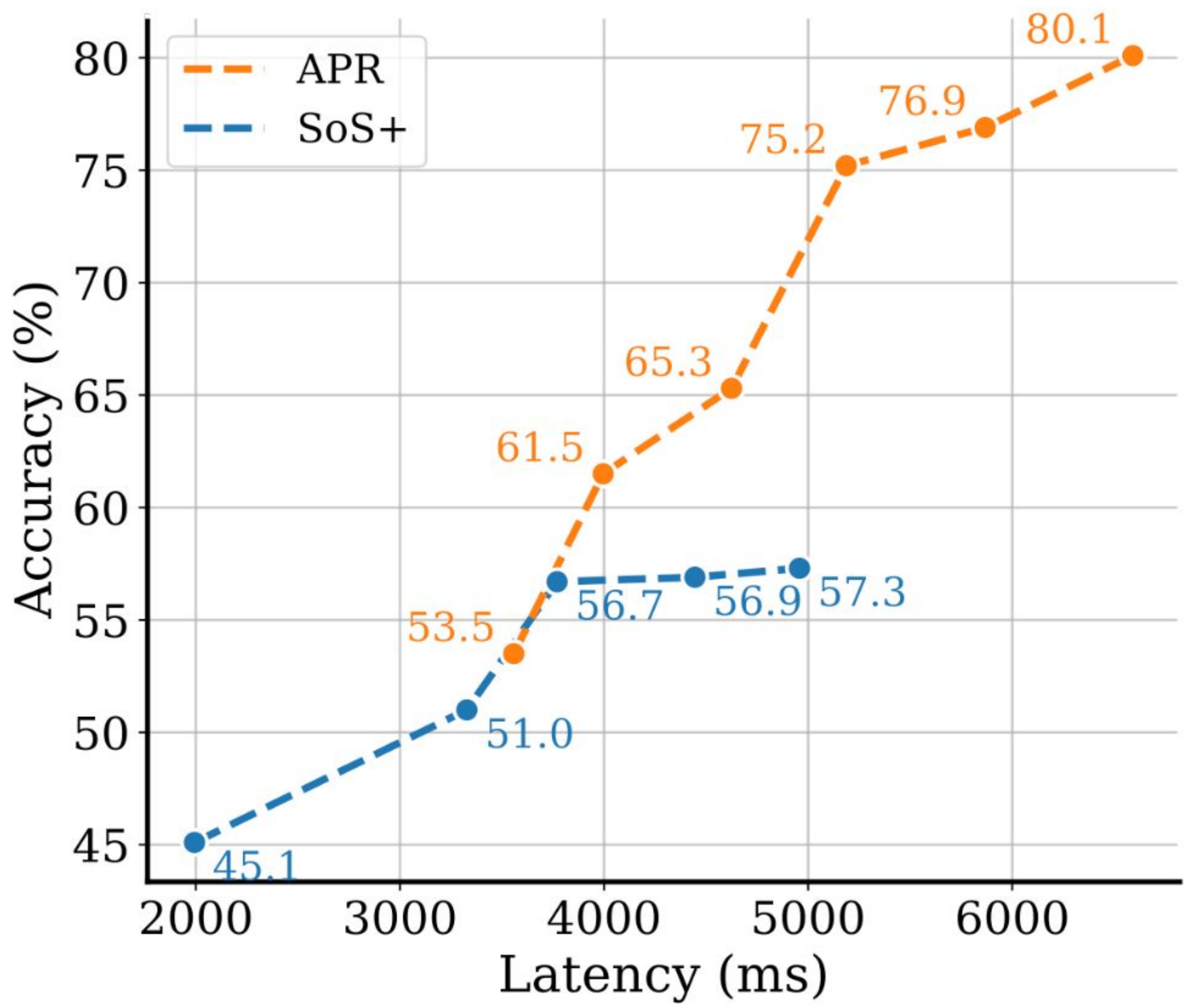
- **强化学习显著提升了模型性能**，最终准确率大幅提高（从 75.5%升至 83.4%）
- 强化学习使序列长度和子线程数量同步增加



实验: Efficiency of APR

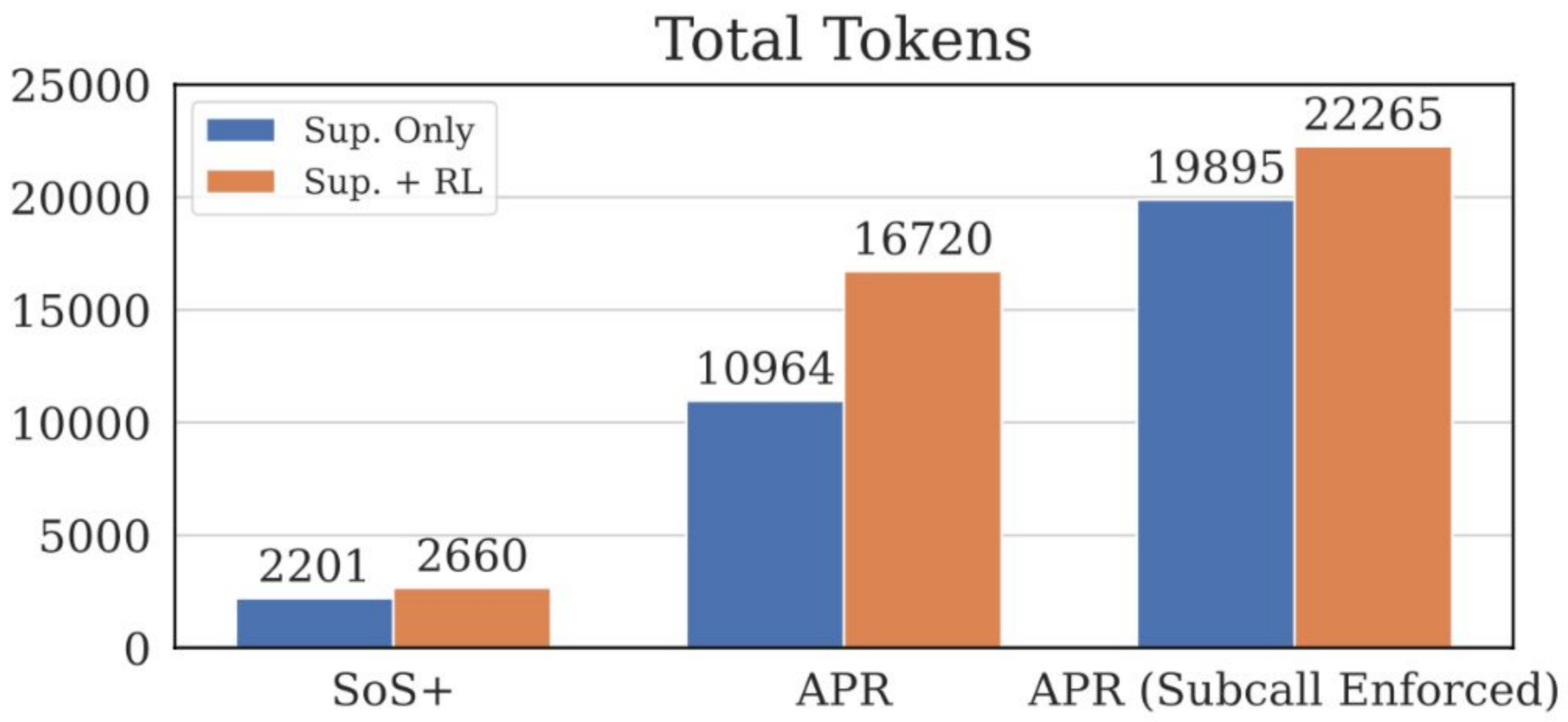
- **Avg Sequential Tokens:** 最长不可并行化部分的平均长度，作为 lower bound
- 基于 API 的服务场景 / 利用硬件并行化：模型部署在配备 8 x A6000 GPU 的服务器上，分配一块 GPU 处理主推理线程，其余 GPU 用于并行执行子线程。
- 在同等延迟约束下，APR 相比串行搜索方法**显著提高了成功率**

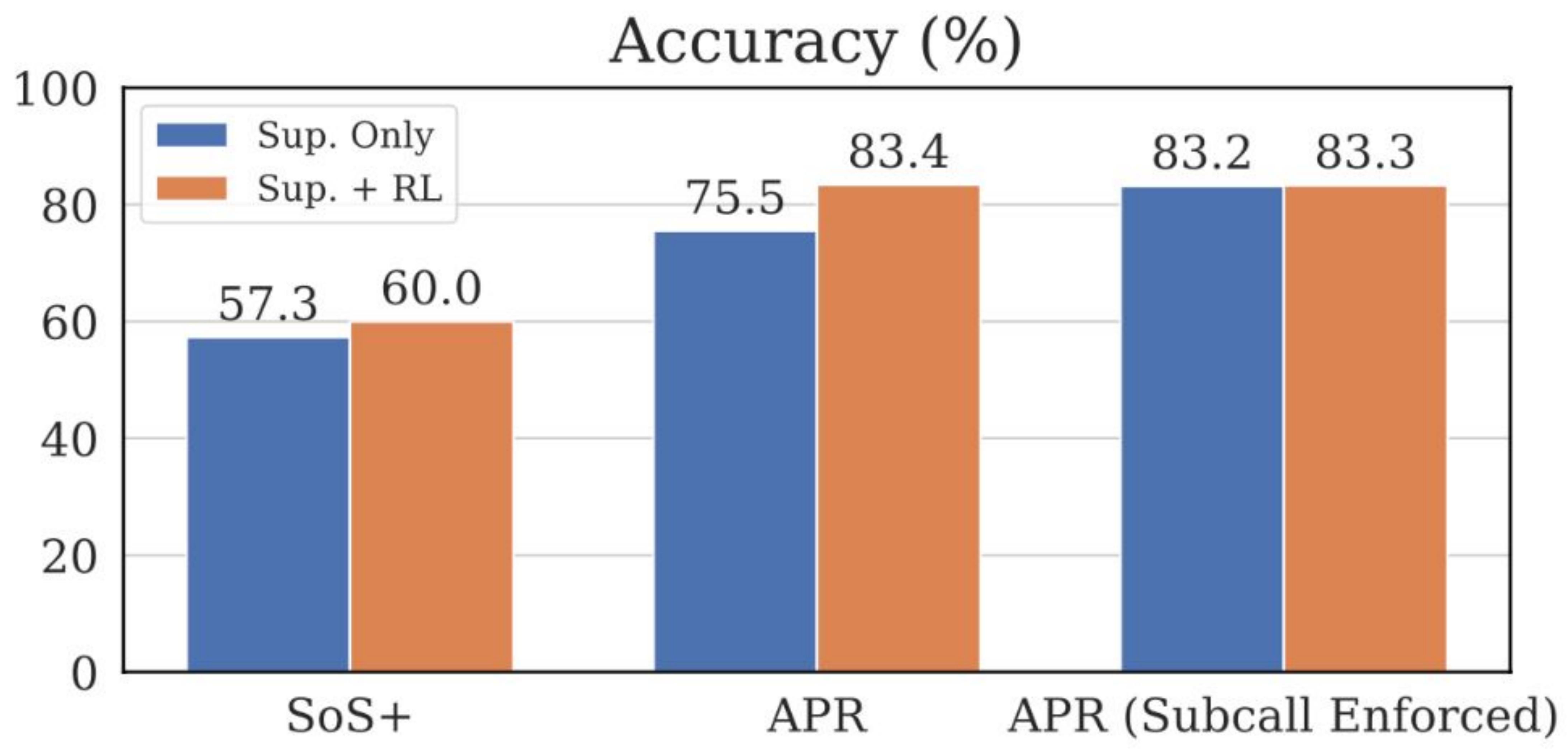




消融实验

- **RL on SoS+ vs APR：**与 SoS+（2.7%）相比，RL 为 APR 带来了更大的性能提升（7.9%）
- **理清 RL 的贡献：**通过限制子线程数量的消融实验表明，RL 带来的性能提升主要来源于扩展测试时计算资源，而非在相同资源条件下做出更优决策。





Future Work

- 向预训练 LLM 和通用任务扩展
- 降低对监督训练的依赖。当前设置需要通过模仿符号求解器进行启动，我们的目标是使用预训练 LLM 作为 checkpoint，探索像 DeepSeek R1-Zero 范式直接应用强化学习
- 优化协调与通信协议

谢谢 ！

- We are happy to take questions now!