

ECE7115 ~~Multimodal VLM~~ LLM

14. Reasoning

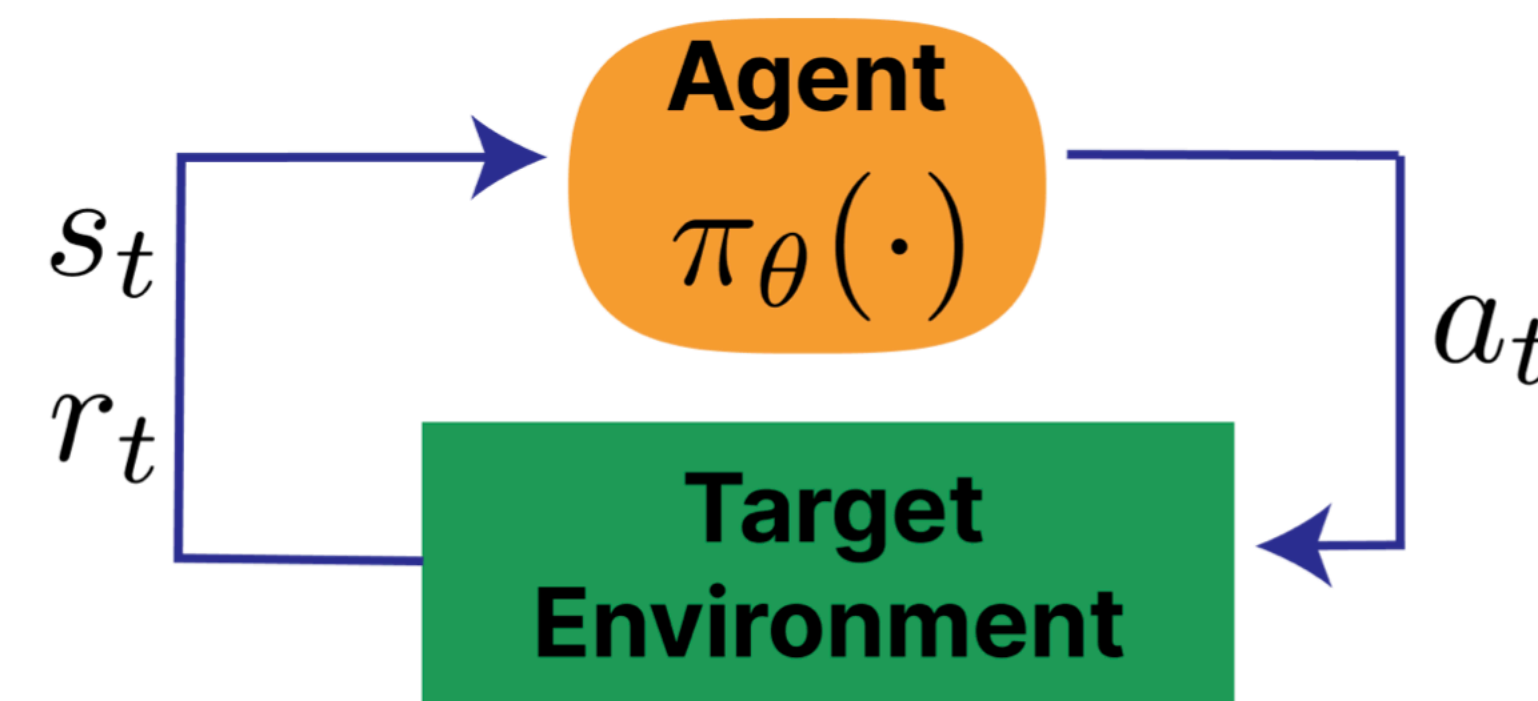
Spring 2026

Namhyuk Ahn, Inha University



Last Week: RL

- RL is trial-and-error learning, balancing exploration and exploitation across long-term rewards
- **State**: the current situation the agent is in
- **Action**: what the agent does next
- **Reward**: the signal for how good that action was
- **Policy**: the strategy for choosing actions
- **Trajectory**: a sequence of states, actions, and rewards
 - $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$
- **Trajectory probability**: $P(\tau | \pi) = p(s_0) \prod_{t=0}^T \pi(a_t | s_t) p(s_{t+1} | s_t, a_t)$



Last Week: Step 1/3: SFT

- Start from pre-trained LMs
- Collect demonstrations of desired assistant behavior
- Train with standard supervised learning on prompt-response pairs

$$L_{\text{SFT}}(\theta) = - \sum_{t=1}^T \log \pi_{\theta}(y_t^* | x, y_{<t}^*).$$

SYSTEM

You are a helpful, harmless assistant. A system message like this can be used to steer the model to specific personas or behaviors.

USER

Write me a short poem about an optimistic goldfish.

ASSISTANT

**Bright little goldfish
Finds a sunrise in each wave
Small bowl, endless hope**

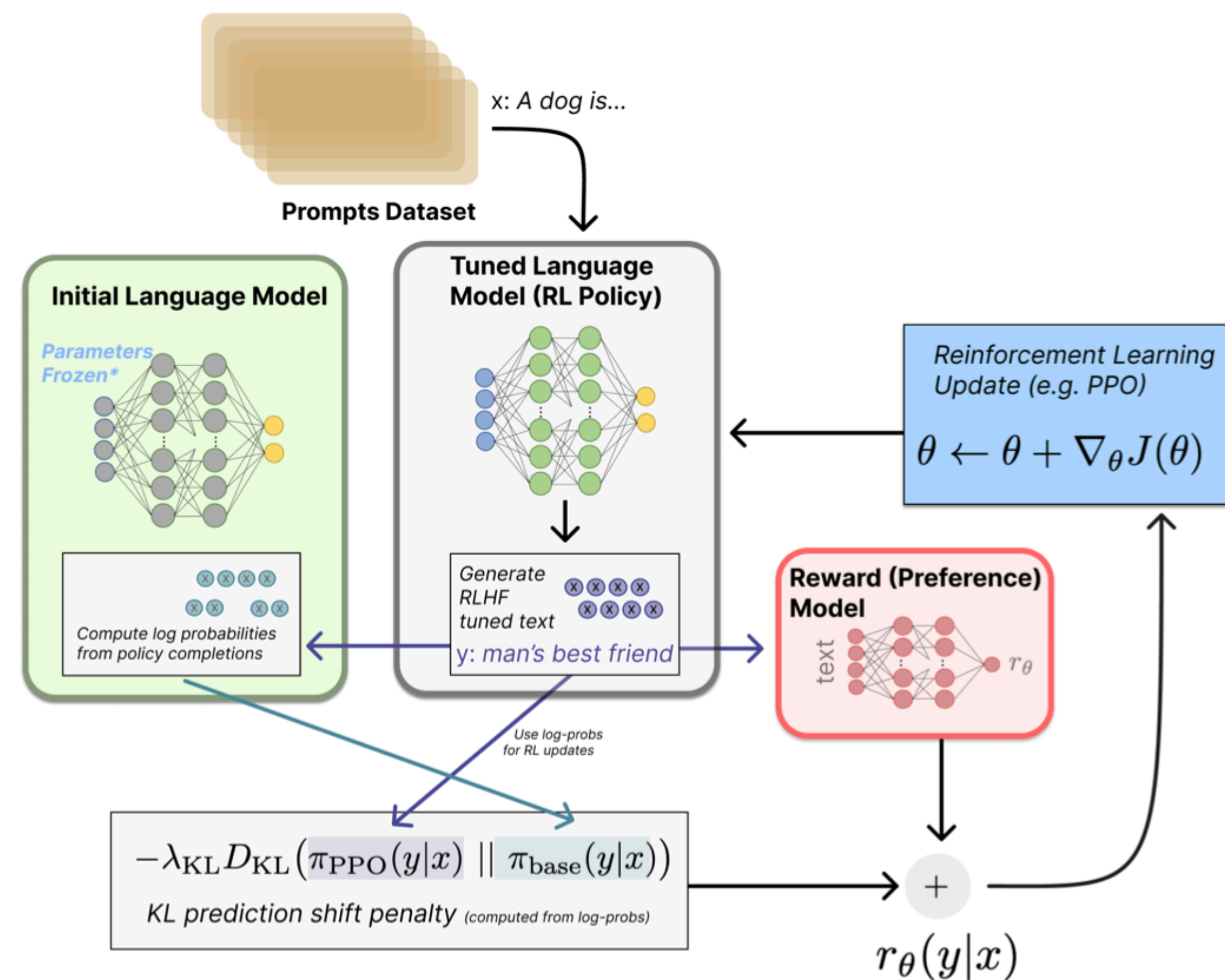
Last Week: Step 2/3: Reward Modeling

- Core idea: The reward used in RLHF is the **model predicts** the probability that a given piece of text would be the "winning" or "chosen" completion in a pair/batch
 - Why? there is no practical way to collect human preference on the fly during training... → approximate with a reward model!
- Collect comparisons for the same prompt (x, y_w, y_l)
 - RLHF gets its name from collecting human feedback between completions, but today much of it is AI feedback
- Train a reward model r_ϕ to score **preferred completions higher**

Last Week: Step 3/3: RL Against RM

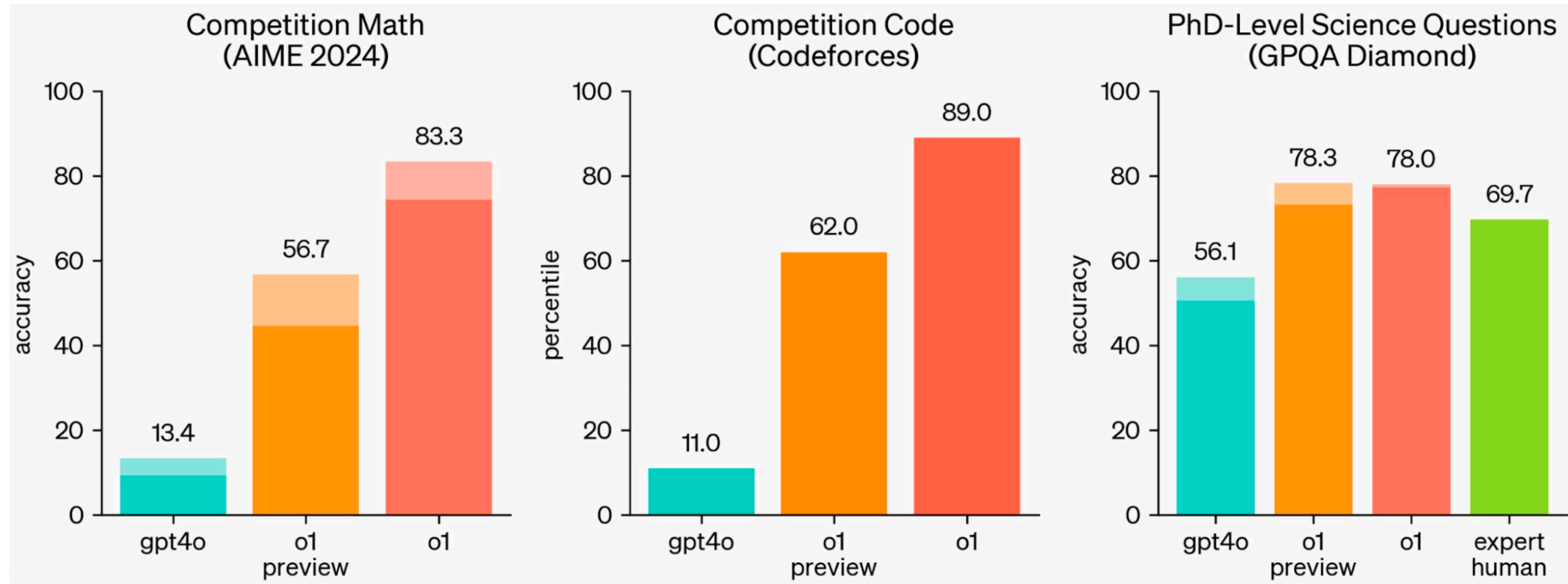
- Where everything comes together
- Sample a batch of prompts $x \sim \mathcal{D}$
- Generate completions $y_i \sim \pi_\theta(\cdot | x_i)$
- Score with the reward model $r_\phi(x_i, y_i)$
- Apply PPO or other RL algorithms
 - Add a KL penalty so the policy stays close to the SFT/reference model

$$J(\pi) = \mathbb{E}[r_\phi(x, y)] - \beta D_{\text{KL}}(\pi \parallel \pi_{\text{ref}})$$



LLMs in 2025 ~

- The advancement of reasoning models



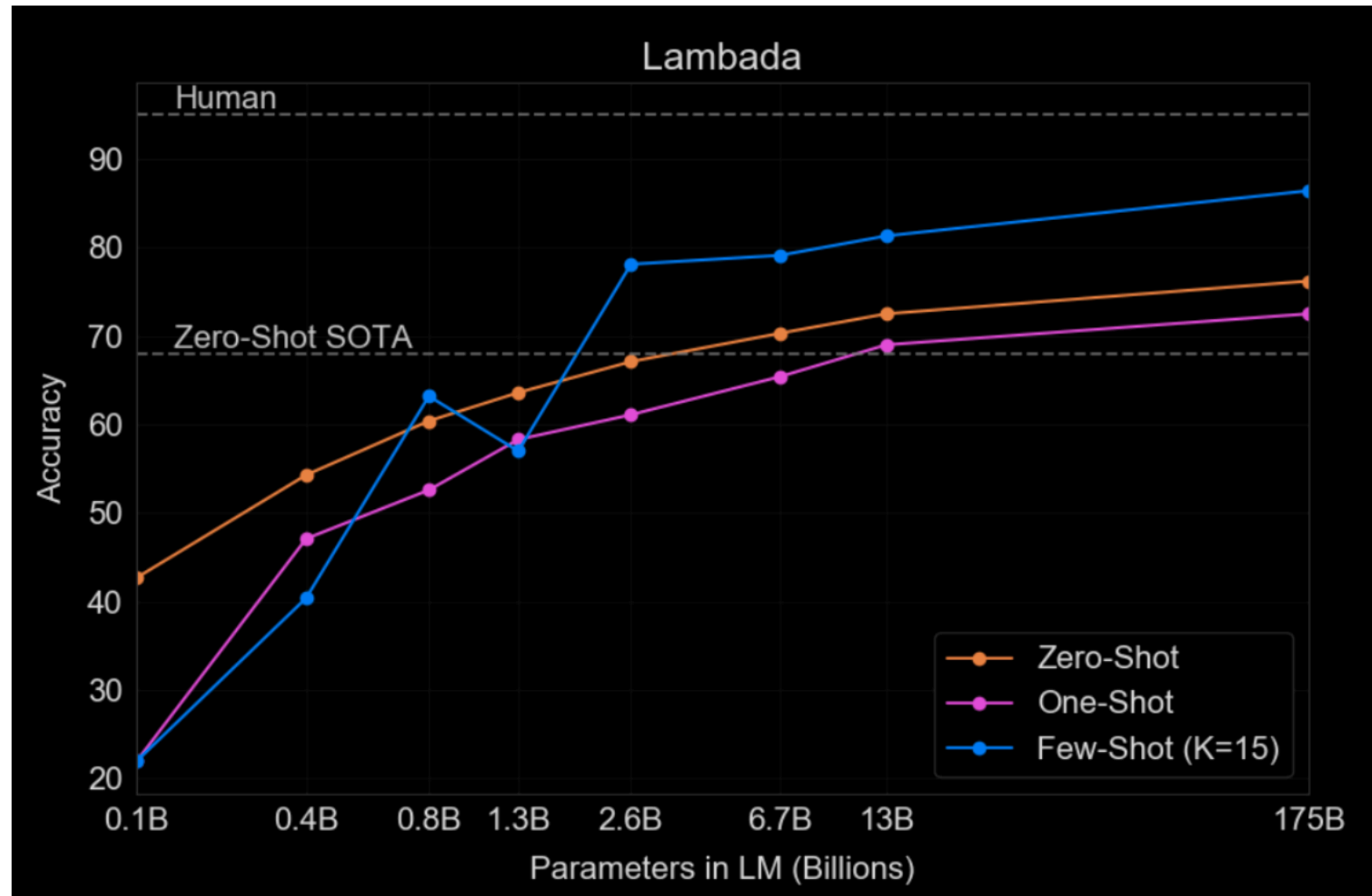
OpenAI o1 started to achieve impressive performance across various challenging reasoning tasks in math, coding, STEM, etc.

LLM History

- 2017: the Transformer is born
- 2018: GPT-1, ELMo, and BERT released
- 2019: GPT-2 and scaling laws
- 2020: GPT-3 surprising capabilities
- 2022: ChatGPT
- 2023: GPT-4 and frontier-scale
- 2024: o1 and reasoning models
- 2025: DeepSeek-R1, o3, Claude Code
- 2026: Agentic systems

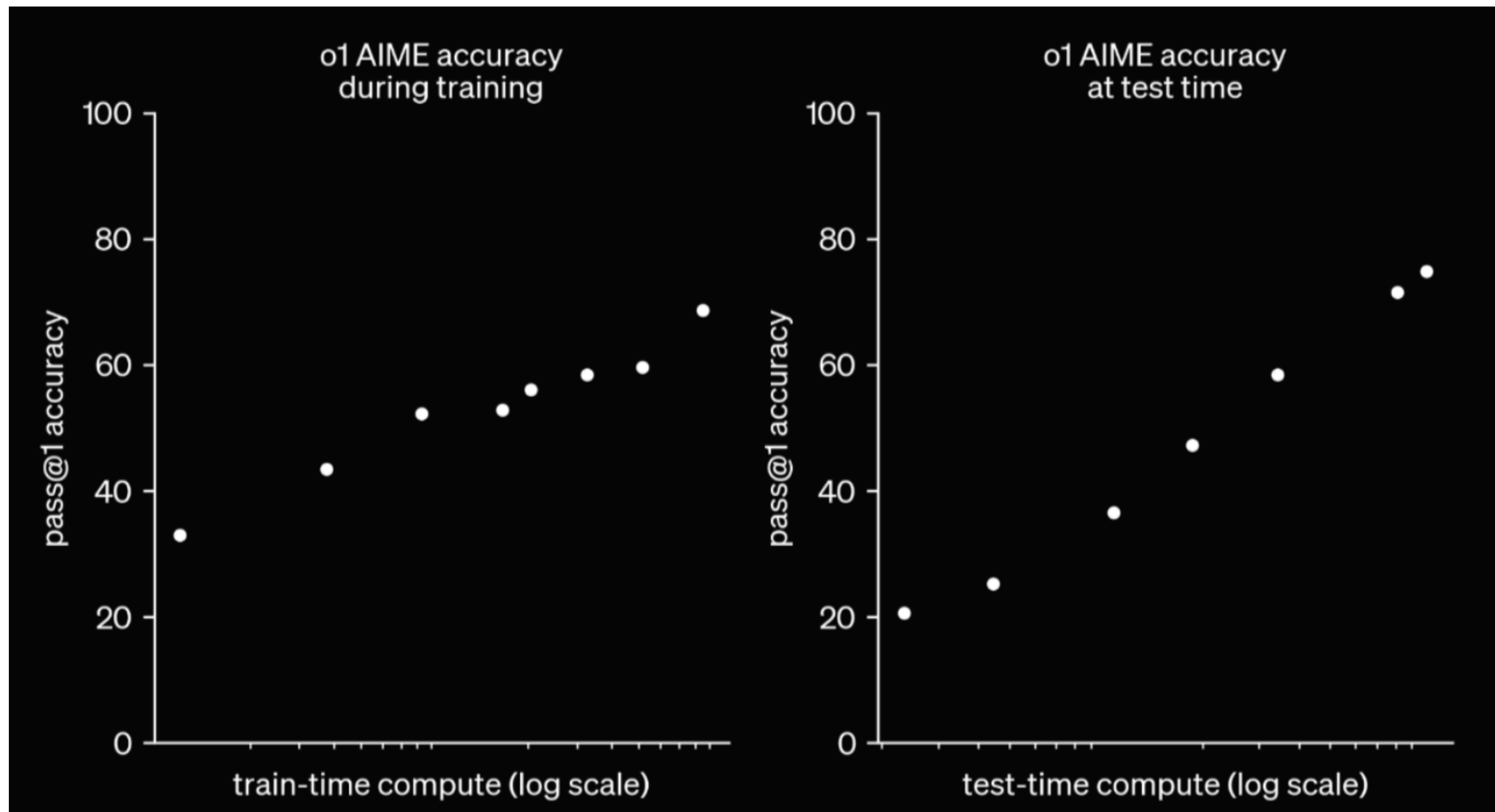
Train-Time Scaling

- We know the "train-time" scaling



Test-Time Scaling

- Increasing test-time (inference) also improves the performance



Note: AIME Task

For any finite set X , let $|X|$ denote the number of elements in X . Define

$$S_n = \sum |A \cap B|,$$

where the sum is taken over all ordered pairs (A, B) such that A and B are subsets of $\{1, 2, 3, \dots, n\}$ with $|A| = |B|$. For example, $S_2 = 4$ because the sum is taken over the pairs of subsets

$$(A, B) \in \{(\emptyset, \emptyset), (\{1\}, \{1\}), (\{1\}, \{2\}), (\{2\}, \{1\}), (\{2\}, \{2\}), (\{1, 2\}, \{1, 2\})\}$$

giving $S_2 = 0 + 1 + 0 + 0 + 1 + 2 = 4$. Let $\frac{S_{2022}}{S_{2021}} = \frac{p}{q}$, where p and q are relatively prime positive integers. Find the remainder when $p + q$ is divided by 1000.

o1: Hidden Thought

User query

You are visiting San Francisco for the day and want to meet as many friends as possible. Solve the problem by considering various different schedules and picking the best one to optimize your goals.

Travel distances (in minutes):
Bayview to Alamo Square: 16.
Bayview to Union Square: 17.
Bayview to Golden Gate Park: 22.
Alamo Square to Bayview: 16.
Alamo Square to Union Square: 14.
Alamo Square to Golden Gate Park: 9.
Union Square to Bayview: 15.
Union Square to Alamo Square: 15.
Union Square to Golden Gate Park: 22.
Golden Gate Park to Bayview: 23.
Golden Gate Park to Alamo Square: 10.
Golden Gate Park to Union Square: 22.

CONSTRAINTS: You arrive at Bayview at 9:00AM. Ronald will be at Alamo Square from 8:30AM to 7:45PM. You'd like to meet Ronald for a minimum of 90 minutes. Richard will be at Union Square from 2:30PM to 9:45PM. You'd like to meet Richard for a minimum of 30 minutes. Kenneth will be at Golden Gate Park from 10:00AM to 3:15PM. You'd like to meet Kenneth for a minimum of 60 minutes.

 Thought about optimizing meeting schedules for 1m 14s

Below is one clear way to fit in all three visits without wasting time, together with the travel times and the required minimum meeting windows. The key insights are:

Model-generated response

Details ×

Crafting meeting logistics

I'm collaborating on a schedule to meet Ronald, Richard, and Kenneth, ensuring each meeting fits within their specified time slots, factoring in travel time and optimal routing.

Mapping travel logistics

Evaluating the best route to Golden Gate Park first, considering travel time and waiting for Kenneth. Balancing early arrival against travel efficiency.

Crafting the itinerary

Piecing together a travel plan, starting with Kenneth, then Ronald, and ending with Richard. Assessing the optimal sequence and meeting durations.

Mapping travel routes

OK, let me see. Traveling from Golden Gate Park to Alamo Square, I'm thinking about extending Ronald's meeting to 1.5 hours then moving to Union Square to meet Richard.

Evaluating options

I'm weighing the benefits of a longer Ronald meeting or a later one to avoid waiting for Richard. The goal is to strike a good balance.

Balancing schedules

I'm organizing a plan to visit Alamo Square first, then head to Golden Gate Park to meet Kenneth. This sequence maximizes time management and minimizes travel between meetings.

Streamlining the timeline

I'm thinking through the optimal approach: meeting Ronald for an uninterrupted 90 minutes to avoid waiting for Richard and making efficient use of the available time before 2:30.

Tweaking the schedule

Meeting Ronald for 90 minutes as scheduled, then quickly traveling to Union Square. This reduces waiting time significantly.

Mapping the timeline

Summarized hidden thought

Lecture Overview

- Reasoning without Training
 - Inference-time scaling
 - Increasing token budget
 - Increasing "width" of the trajectory
 - Increasing "depth" of the trajectory
- Training Reasoning
 - RLVR
 - GRPO family

Reasoning without Training

Core Idea: Generate a Long Sequence [Wei+ 2022]

- Trigger the LLM to generate a long chain-of-thought (CoT)

Chain-of-Thought Prompting

Model Input

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✓ **Thought**

Approaches to trigger CoT generation

- Few-shot CoT prompting
- Instruction prompting
- Instruction tuning
- Reinforcement learning

Models

- Reasoning and inference-time scaling with
 - Increase the **token budget** to generate a single solution
 - Increase the **width** to explore the solution space
 - Increase the **depth** to reach the final solution

Increasing the Token-Budget

Recap: Standard Prompting

- Before advancement in post-training, standard prompting performance is poor on reasoning benchmarks
- Why?
 - Standard few-shot exemplars only provide information on the **final solution format**, not the rationale to derive the solution

Standard Prompting

Model Input

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?

Exemplar

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27.



"Thinking" in LLM

- **Standard model** answers directly:

What is the sum of all prime numbers less than 20?

The primes less than 20 are 2, 3, 5, 7, 11, 13, 17, 19. Their sum is 77.

- **Reasoning model** generates `<think>` tokens before answering:

`<think>` *I need to find all primes less than 20. Starting from 2: 2 is prime. 3 is prime. 4 = 2x2, not prime. 5 is prime... Let me add them up: 2+3=5, 5+5=10, 10+7=17, 17+11=28, 28+13=41, 41+17=58, 58+19=77.*

`</think>`

The answer is `77`.

Verification: `extracted_answer == 77` → *Reward = 1*

- + additional inference-time scaling we will learn

Chain-of-Thought (CoT) [Wei+ 2022]

Standard Prompting

Model Input

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?

Exemplar

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

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? **Thought**

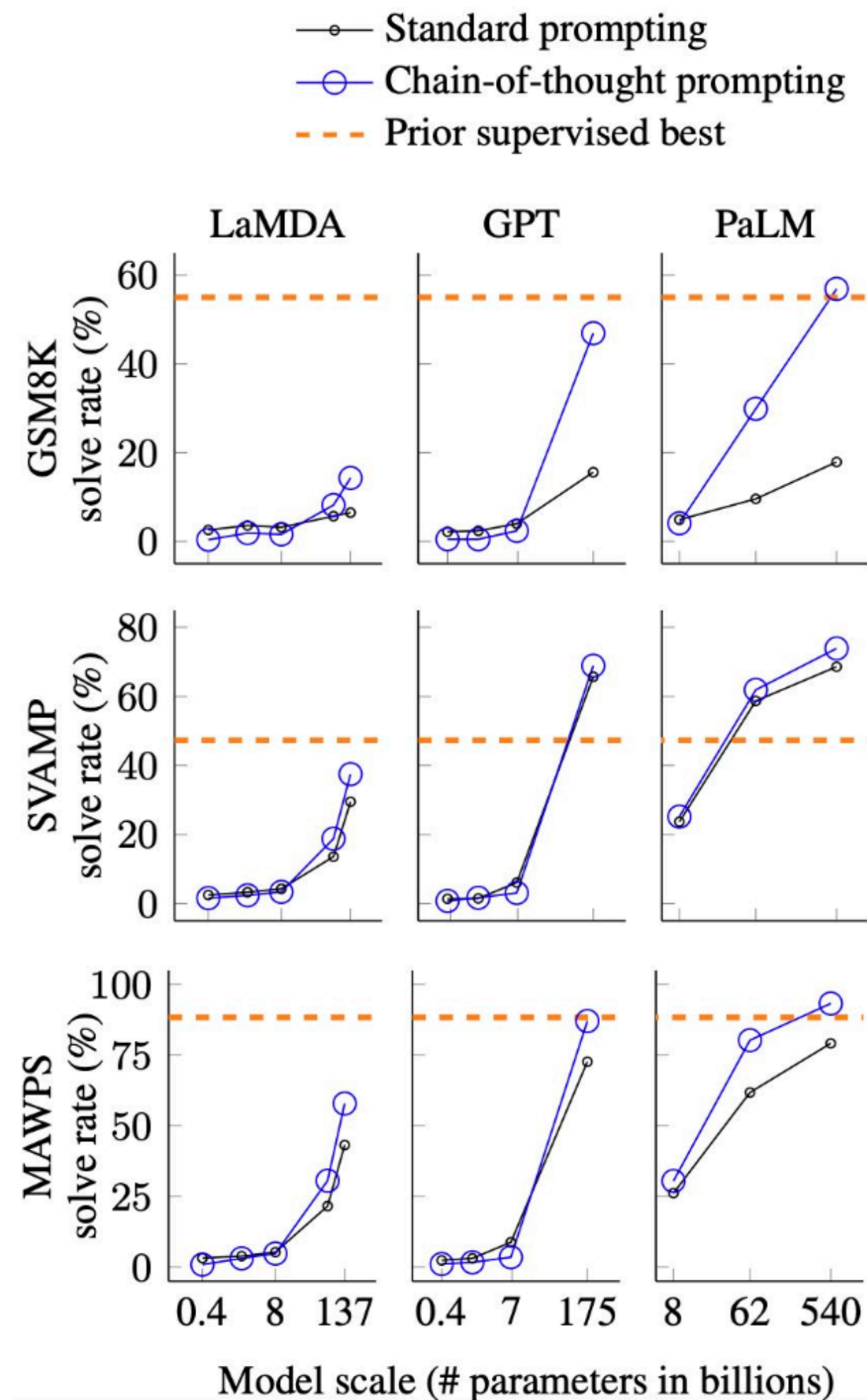
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

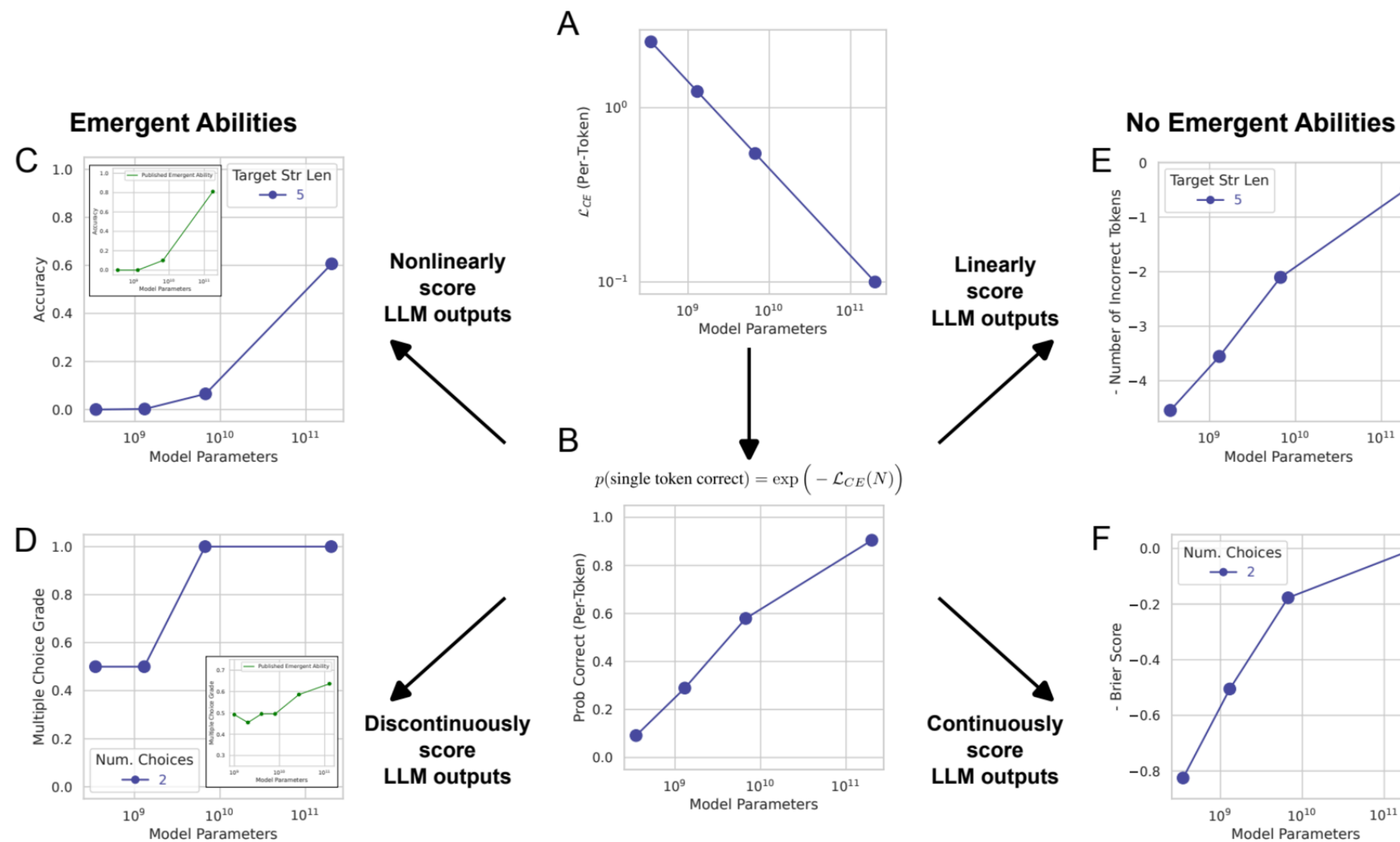
Chain-of-Thought (CoT) [Wei+ 2022]



- CoT performance improves more significantly with the increase of the model size
- Better models benefit more from CoT
 - A drastic improvement on reasoning performance when the model reaches a certain scale (**emergent abilities**)
- Note: these experiments used base LLMs
 - Recent post-trained LLMs might have different scaling curves, but the main conclusions still hold

Note: Emergent Ability

- **Emergent abilities:** only appear when models are very large
- Note: emergent abilities are an artifact of how we measure accuracy



Zero-Shot CoT [Kojima+ 2022]

- "Let's think step by step" triggers CoT generation without 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. **X**

(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 **X**

(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. **✓**

Zero-Shot CoT [Kojima+ 2022]

	Arithmetic					
	SingleEq	AddSub	MultiArith	GSM8K	AQUA	SVAMP
zero-shot	74.6/78.7	72.2/77.0	17.7/22.7	10.4/12.5	22.4/22.4	58.8/58.7
zero-shot-cot	78.0/78.7	69.6/74.7	78.7/79.3	40.7/40.5	33.5/31.9	62.1/63.7
	Common Sense		Other Reasoning Tasks		Symbolic Reasoning	
	Common SenseQA	Strategy QA	Date Understand	Shuffled Objects	Last Letter (4 words)	Coin Flip (4 times)
zero-shot	68.8/72.6	12.7/54.3	49.3/33.6	31.3/29.7	0.2/-	12.8/53.8
zero-shot-cot	64.6/64.0	54.8/52.3	67.5/61.8	52.4/52.9	57.6/-	91.4/87.8

Zero-Shot CoT [Kojima+ 2022]

- Current (but old...) LLMs are sensitive to prompt design

No.	Category	Template	Accuracy
1	instructive	Let's think step by step.	78.7
2		First,	77.3
3		Let's think about this logically.	74.5
4		Let's solve this problem by splitting it into steps.	72.2
5		Let's be realistic and think step by step.	70.8
6		Let's think like a detective step by step.	70.3
7		Let's think	57.5
8		Before we dive into the answer,	55.7
9		The answer is after the proof.	45.7
10	misleading	Don't think. Just feel.	18.8
11		Let's think step by step but reach an incorrect answer.	18.7
12		Let's count the number of "a" in the question.	16.7
13		By using the fact that the earth is round,	9.3
14	irrelevant	By the way, I found a good restaurant nearby.	17.5
15		AbraKadabra!	15.5
16		It's a beautiful day.	13.1
-		(Zero-shot)	17.7

Zero-shot CoT on MultiArith with text-davinci-002

Zero-Shot CoT [Kojima+ 2022]

- But zero-shot CoT is still worse than the few-shot CoT

	MultiArith	GSM8K
Zero-Shot	17.7	10.4
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
Zero-Shot-CoT	78.7	40.7
Few-Shot-CoT (2 samples)	84.8	41.3
Few-Shot-CoT (4 samples : First) (*1)	89.2	-
Few-Shot-CoT (4 samples : Second) (*1)	90.5	-
Few-Shot-CoT (8 samples)	93.0	48.7
Zero-Plus-Few-Shot-CoT (8 samples) (*2)	92.8	51.5

- How to improve CoT performance without manually labeled examples?

Analogical Prompting [Chen+ 2024]

- Prompt LLM to recall relevant examples, before solving the problem

Model Input

Q: What is the area of the square with the four vertices at (-2, 2), (2, -2), (-2, -6), and (-6, -2)?

Instruction:

Recall relevant exemplars:

Solve the initial problem:

Model Output

Relevant exemplars:

Q: What is the area of the square with a side length of 5?

A: The area of a square is found by squaring the length of its side. So, the area of this square is $5^2 = 25$

Solve the initial problem:

To find the area of the square, we need to find the side length. The length is ... $\sqrt{(2 - (-2))^2 + (-2 - 2)^2} = \sqrt{32}$. So, the area of the square is $(\sqrt{32})^2 = 32$.

- 👍 Examples are self-generated by LLMs, no manual labeling
- 👍 Examples are tailored to individual problems

Analogical Prompting [Chen+ 2024]

Codeforces problem

Our Prompt: Self-generate Exemplars + Knowledge

Your goal is to write Python3 code to solve competitive programming problems. Given a problem, explain the core concepts in it and provide other relevant problems. Then solve the original problem.

Problem:
You are given a sequence a_1, a_2, \dots, a_n . Each element of a is 1 or 2. Find out if an integer k exists so that the following conditions are met. $1 \leq k \leq n-1$, and $a_1 \cdot a_2 \dots a_k = a_{k+1} \dots a_n$. If there exist multiple k that satisfy the given condition, print the smallest.

Instruction:

Algorithms:

Identify the core concepts or algorithms used to solve the problem.

Tutorial:

Write a tutorial about these algorithms.

Example Problems:

Provide three examples of relevant competitive programming problems that involve these algorithms. For each problem, describe the problem, explain the solution in detail, and then write the correct Python3 code.

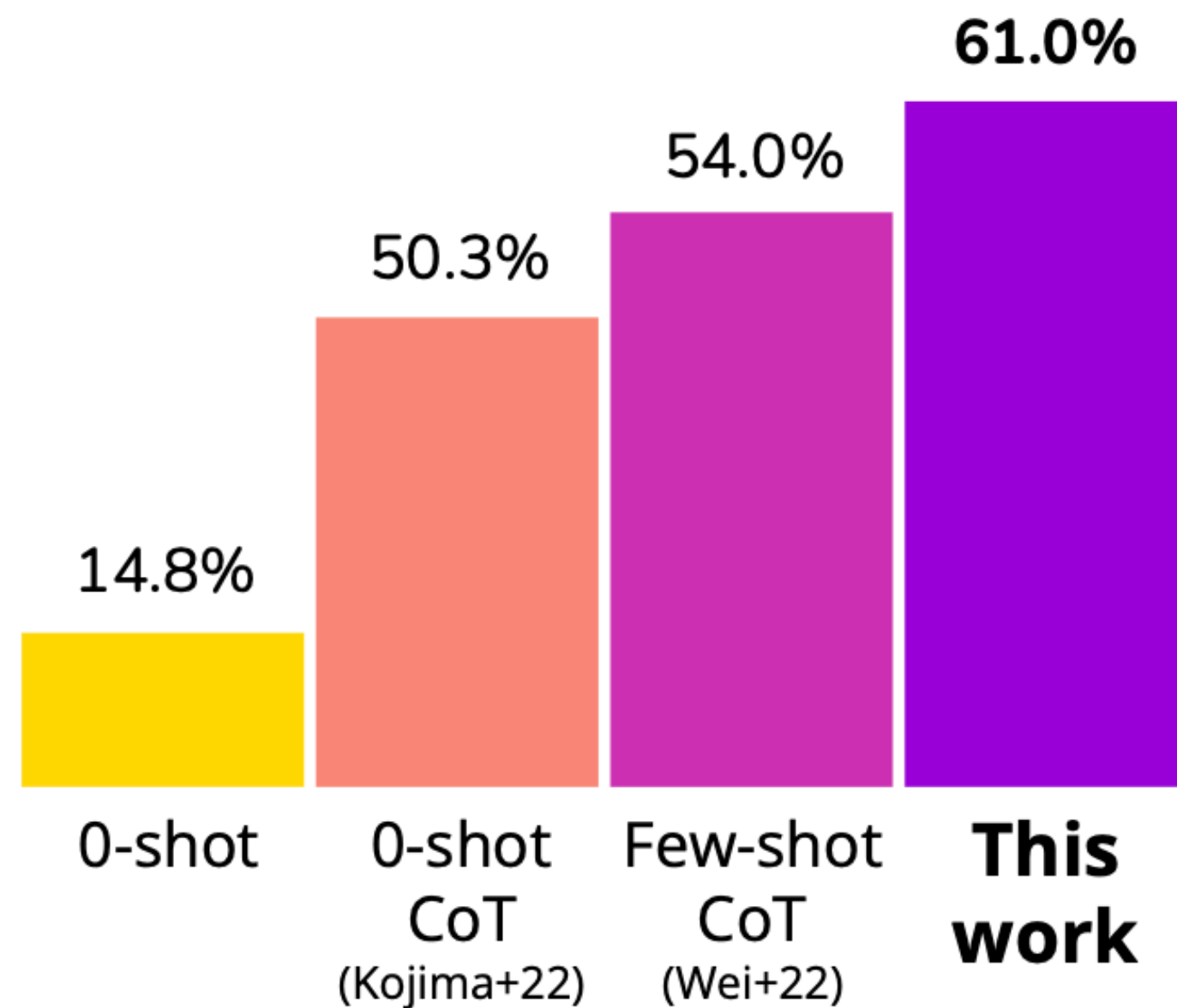
Python3 code to solve the original problem:

- Explanation of the solution:
- Python3 code to solve the problem:

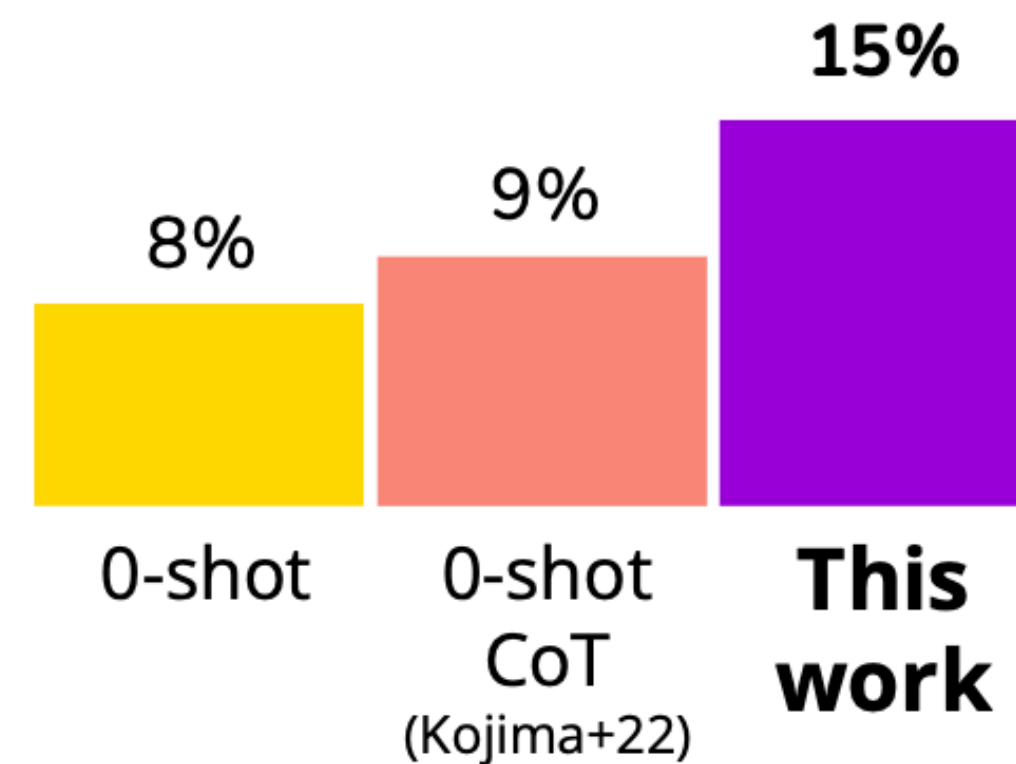
Analogical Prompting [Chen+ 2024]

- It outperforms zero-shot CoT and manual few-shot CoT

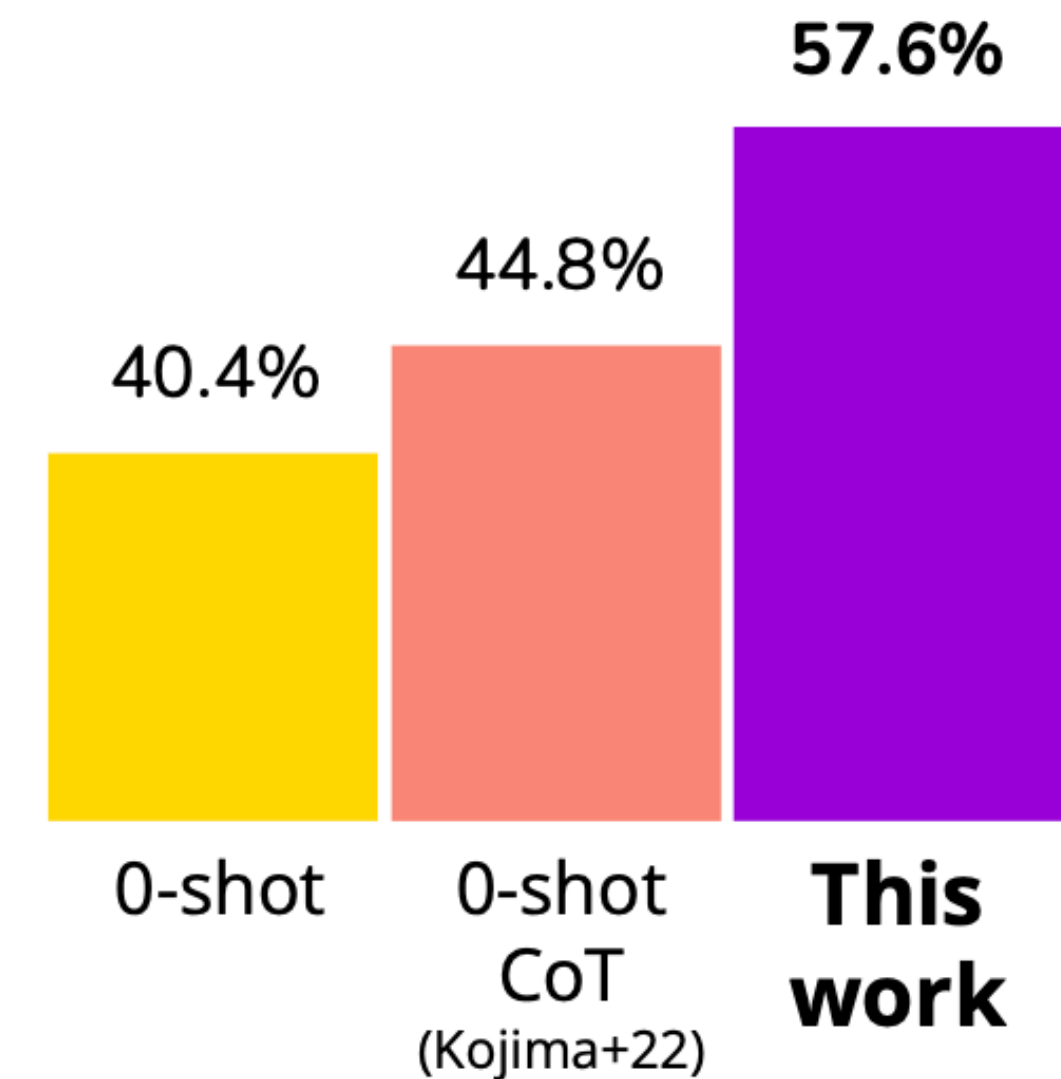
Math problems
(GSM8K, text-davinci-003)



Code generation
(Codeforces, GPT3.5 turbo)



Temporal reasoning
(BIG-Bench, GPT3.5 turbo)



Analogical Prompting [Chen+ 2024]

- Stronger LLMs are better analogical reasoners
- Actually, most of the reasoning methods (both training/non-training) heavily are affected by the LLM size
 - Good pre-trained LLMs are all we need!

Prompting Method	← scale down		scale up →	
	text-curie-001	text-davinci-001	text-davinci-002	text-davinci-003
0-shot	2%	6%	13%	14%
0-shot CoT	2%	6%	22%	50%
5-shot (fixed) CoT	2%	10%	43%	54%
5-shot retrieved CoT	3%	11%	47%	57%
Ours: Self-generated Exemplars	2%	9%	48%	61%

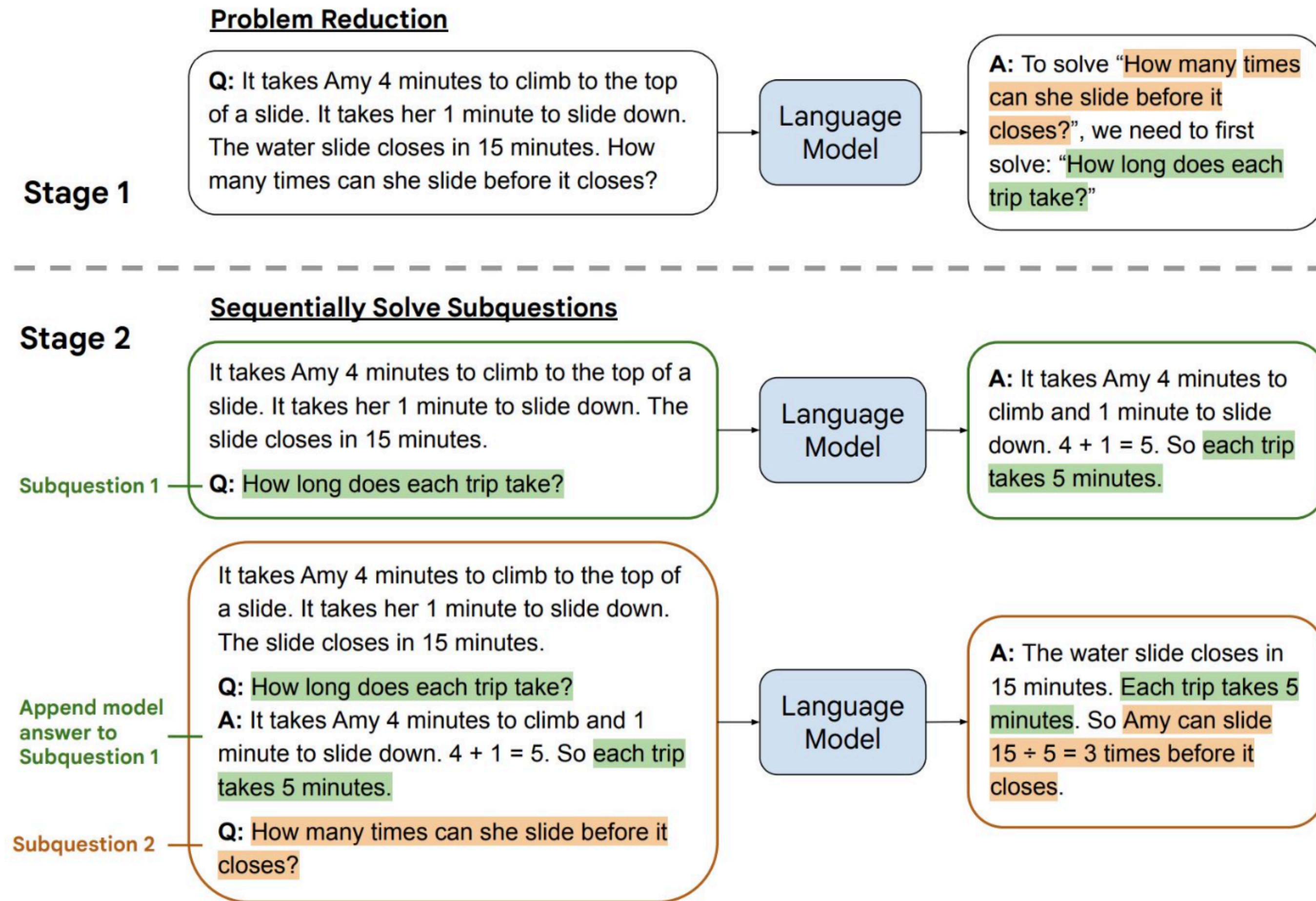
GSM8K for math reasoning

What Does CoT Brings into Reasoning?

- CoT prompting: **variable computation of the thought process** adapting to tasks of different difficulty levels
 - More complex questions → more reasoning steps in the CoT
- Reasoning strategies enabled by CoT
 - Decomposition
 - Planning
 - ...
- We can explicitly instruct the LLM with the desired reasoning strategies for problem solving

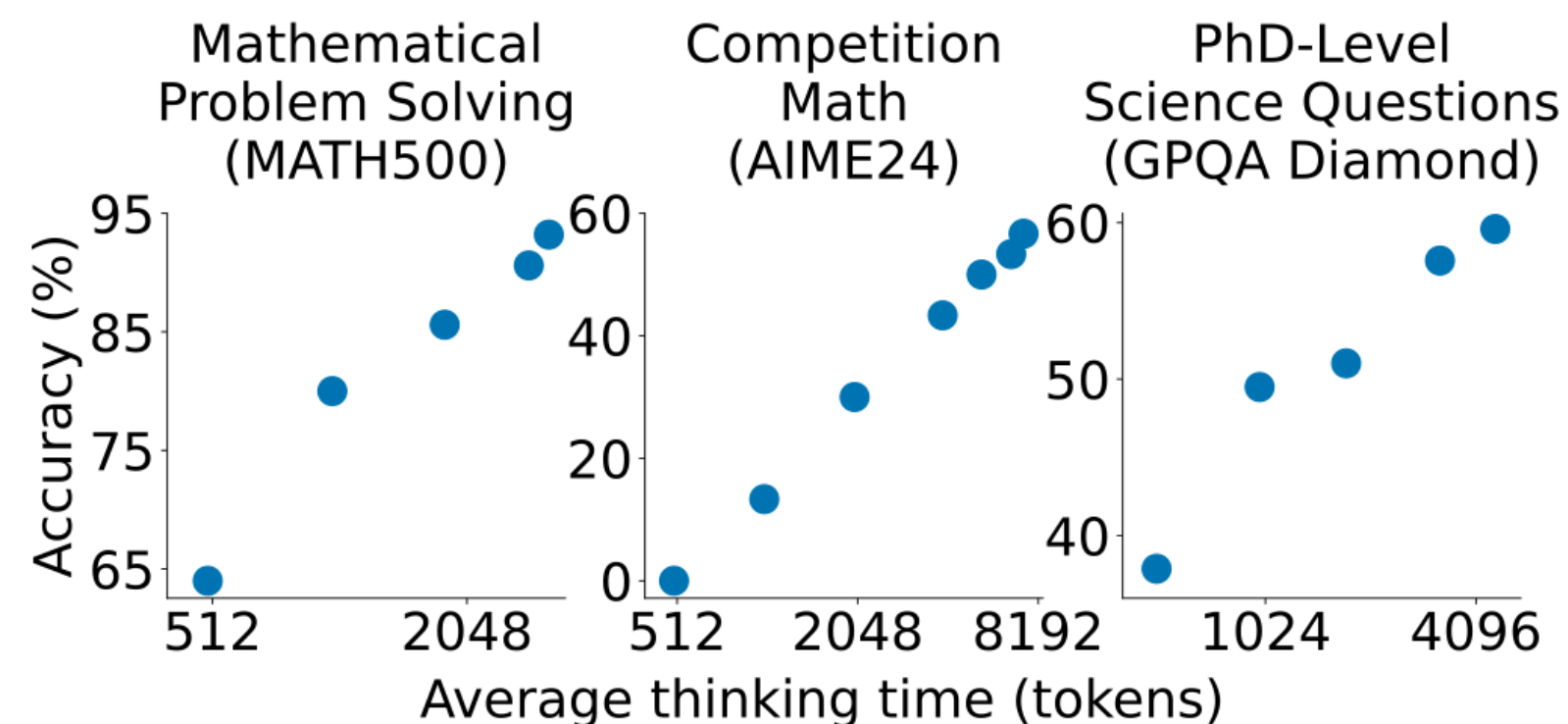
Least-to-Most Prompting [Zhou+ 2023]

- Easy-to-hard generation via decomposition



s1: Simple Test-Time Scaling [Muennighoff+ 2025]

- **Budget forcing**: controls test-time compute by forcefully terminating the model's thinking process (by "Final Answer:") or lengthening it by appending ("Wait") to the model's generation when it tries to end
- This leads the model to double-check its answer



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 **3** **Response**

Figure 3. Budget forcing with s1-32B. The model tries to stop after "...is 2.", but we suppress the end-of-thinking token delimiter instead appending "Wait" leading s1-32B to self-correct its answer.

Summary So Far

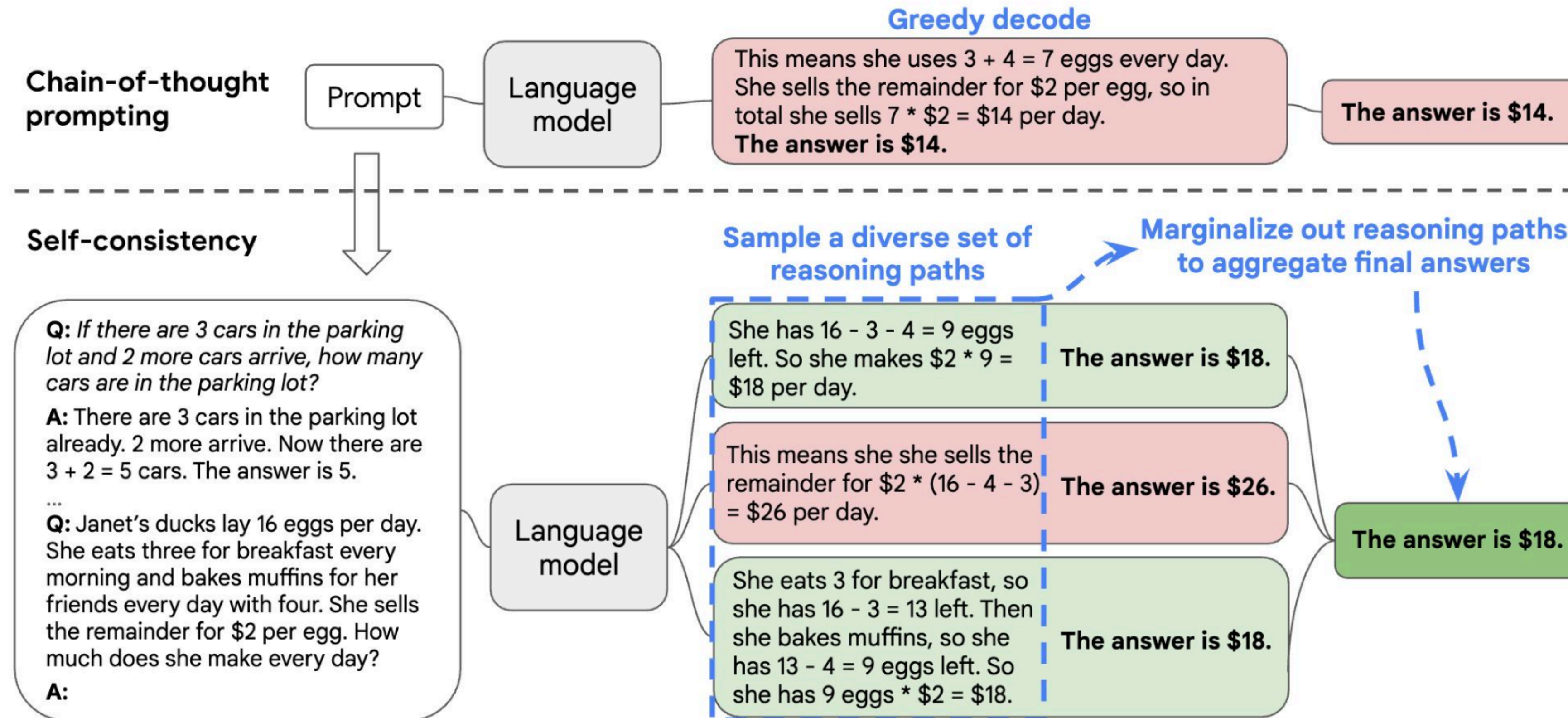
- Chain-of-thought generation: **variable computation of the thought process** adapting to tasks of different difficulty levels
- How to improve the CoT performance at inference time
 - Few-shot prompting with labeling of thoughts
 - Instruction prompting to trigger CoT generation
- Note: **the best practice for interacting with LLMs evolves over time**
 - But the principles of how to discover good prompting strategies for reasoning hold true
 - Encourage longer CoT for complex tasks
 - Support reasoning strategies required for the task

Increasing the Trajectory Width

What Was Missing?

- We should not limit LLM to **generate only one solution per problem!**
- Exploring multiple branches allows the LLM to recover from mistakes in a single generation
 - Generate multiple candidate solutions per problem
 - Generate multiple potential next reasoning steps given the current (partial) thought
- Challenge: how to select the best response from multiple candidates?
 - In most cases, we do not have an oracle scorer at inference time

CoT with Self-Consistency [Wang+ 2023]

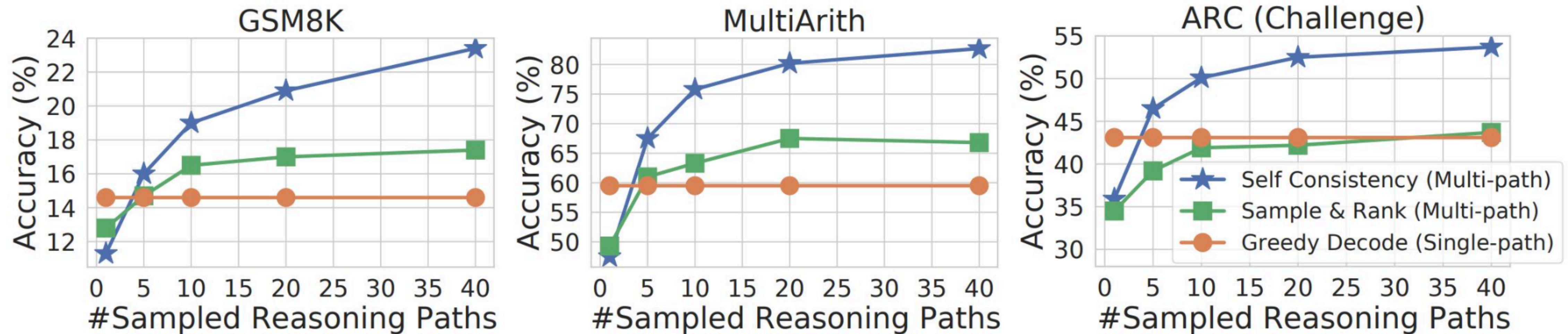


- Note: the selection is only based on the final answer, the reasoning paths do not need to be the same across different sampled responses

CoT with Self-Consistency [Wang+ 2023]

	Method	AddSub	MultiArith	ASDiv	AQuA	SVAMP	GSM8K
	Previous SoTA	94.9^a	60.5 ^a	75.3 ^b	37.9 ^c	57.4 ^d	35 ^e / 55 ^g
UL2-20B	CoT-prompting	18.2	10.7	16.9	23.6	12.6	4.1
	Self-consistency	24.8 (+6.6)	15.0 (+4.3)	21.5 (+4.6)	26.9 (+3.3)	19.4 (+6.8)	7.3 (+3.2)
LaMDA-137B	CoT-prompting	52.9	51.8	49.0	17.7	38.9	17.1
	Self-consistency	63.5 (+10.6)	75.7 (+23.9)	58.2 (+9.2)	26.8 (+9.1)	53.3 (+14.4)	27.7 (+10.6)
PaLM-540B	CoT-prompting	91.9	94.7	74.0	35.8	79.0	56.5
	Self-consistency	93.7 (+1.8)	99.3 (+4.6)	81.9 (+7.9)	48.3 (+12.5)	86.6 (+7.6)	74.4 (+17.9)
GPT-3 Code-davinci-001	CoT-prompting	57.2	59.5	52.7	18.9	39.8	14.6
	Self-consistency	67.8 (+10.6)	82.7 (+23.2)	61.9 (+9.2)	25.6 (+6.7)	54.5 (+14.7)	23.4 (+8.8)
GPT-3 Code-davinci-002	CoT-prompting	89.4	96.2	80.1	39.8	75.8	60.1
	Self-consistency	91.6 (+2.2)	100.0 (+3.8)	87.8 (+7.6)	52.0 (+12.2)	86.8 (+11.0)	78.0 (+17.9)

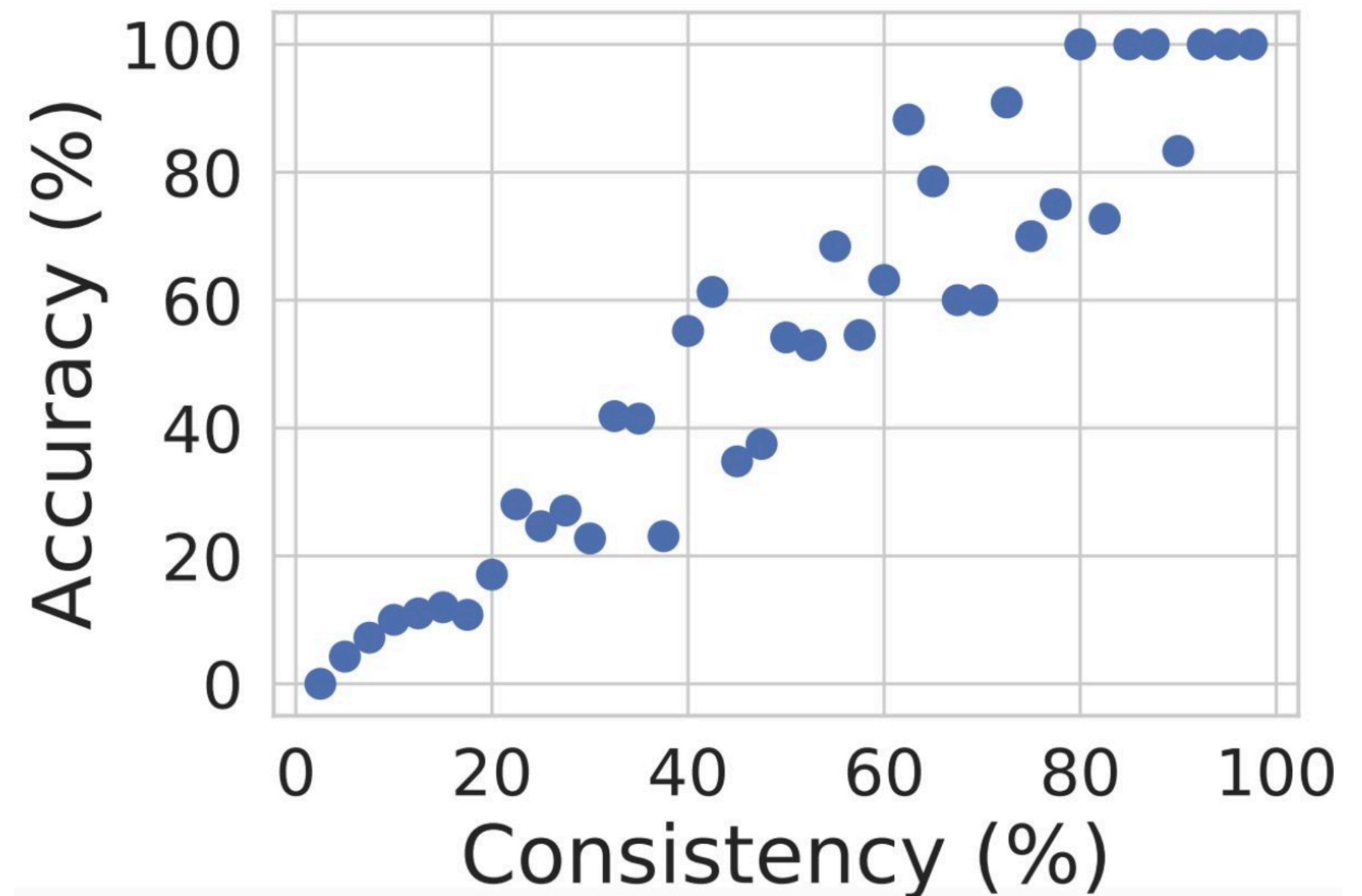
CoT with Self-Consistency [Wang+ 2023]



- Sample-and-Rank: select the response with the highest log probability
- CoT-SC performance scales much better than prob.-based ranking
 - Unless the model is trained to be a good verifier

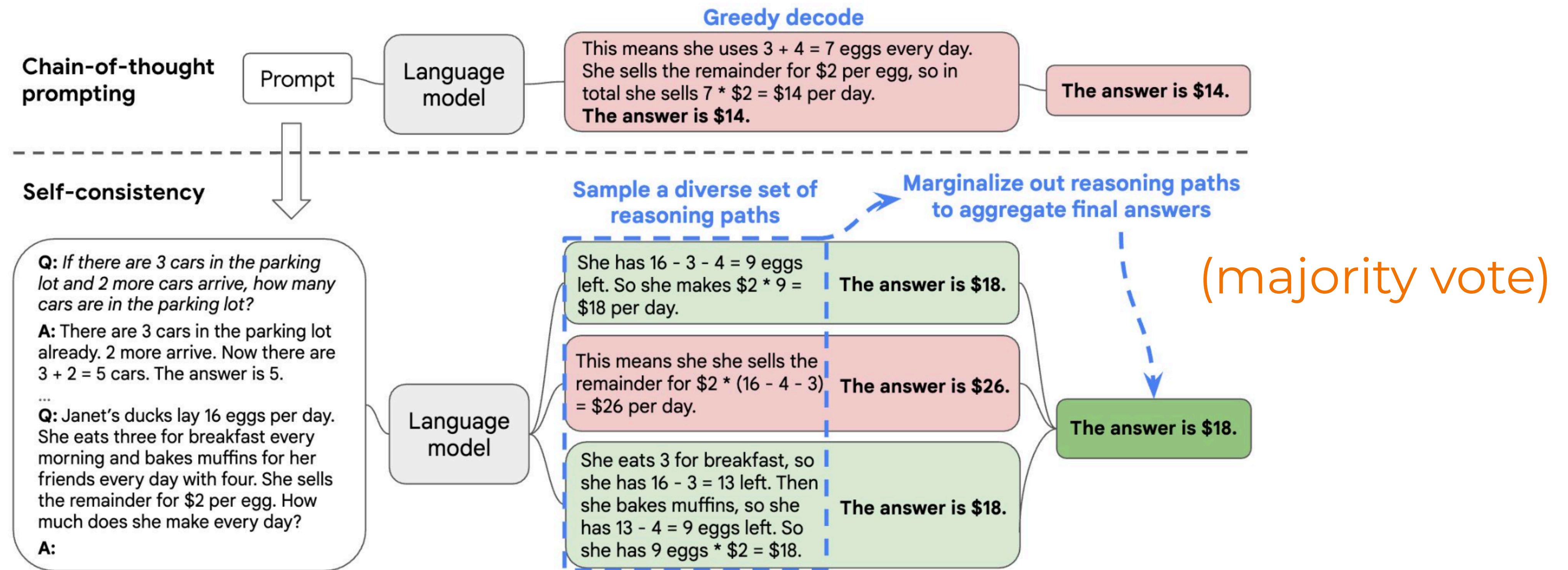
CoT with Self-Consistency [Wang+ 2023]

- Consistency is highly correlated with accuracy
- If more sampled responses lead to the same final answer:
- The LLM is more certain about its predicted conclusion
- The aggregated solution is more likely to be correct



CoT with Self-Consistency [Wang+ 2023]

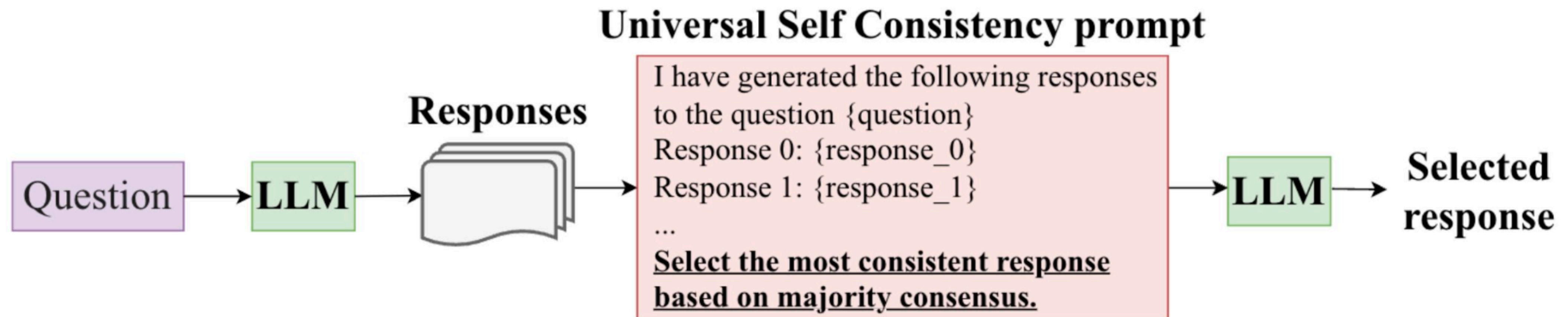
- Limitation: requires an answer extraction process



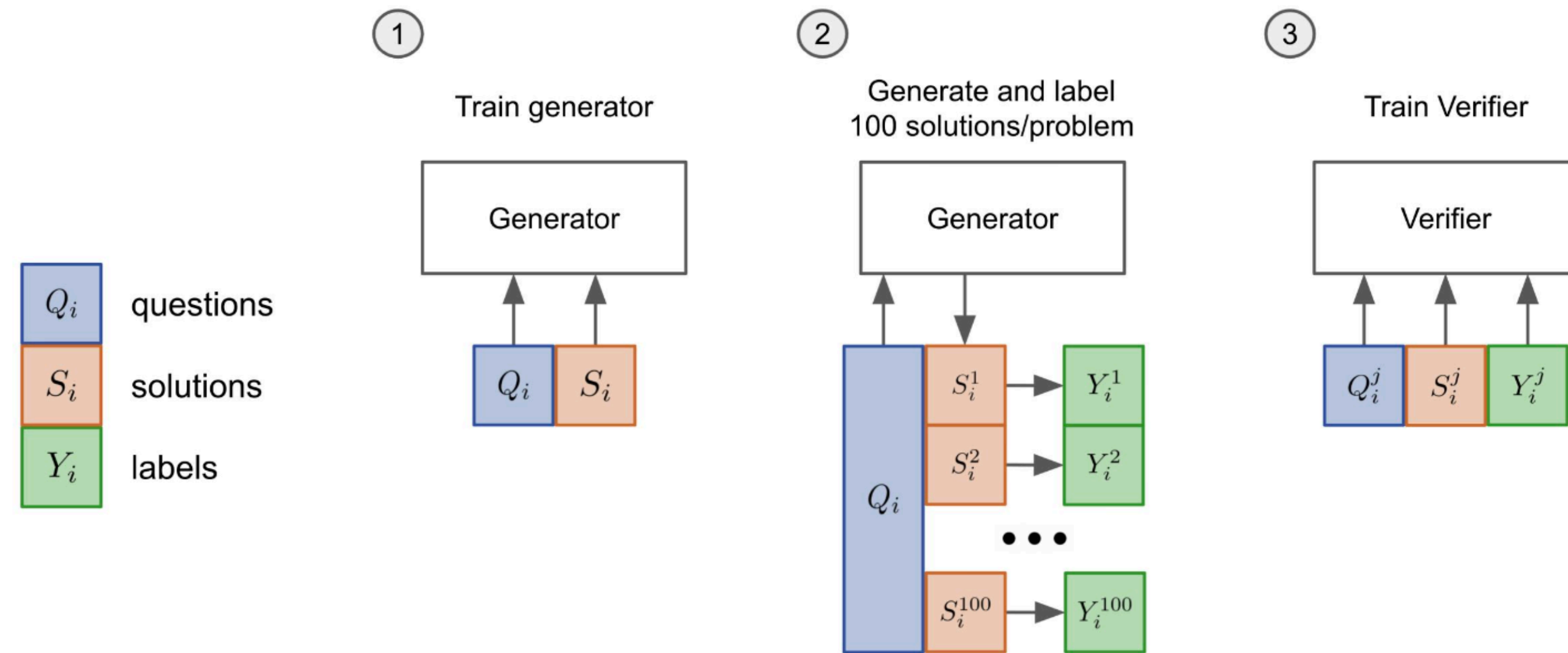
- Can we enable consistency-based decoding for free-form generation?

Universal Self-Consistency [Chen+ 2023]

- Ask LLMs to perform consistency-based selection!



Training Verifier [Cobbe+ 2023, Lightman+ 2023]



- Train an external verifier to predict whether the answer is correct
- Two types of LLM-based verifiers/reward models
 - Outcome-supervised reward model (ORM): verify at the solution level
 - Process-supervised reward model (PRM): verify at the step level

Training Verifier [Cobbe+ 2023, Lightman+ 2023]

- For a given task (e.g. Math), annotate intermediate solving process

MATH Dataset (Ours)

Problem: Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?

Solution: There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.

Problem: If $\sum_{n=0}^{\infty} \cos^{2n} \theta = 5$, what is $\cos 2\theta$?

Solution: This geometric series is $1 + \cos^2 \theta + \cos^4 \theta + \dots = \frac{1}{1 - \cos^2 \theta} = 5$. Hence, $\cos^2 \theta = \frac{4}{5}$. Then $\cos 2\theta = 2 \cos^2 \theta - 1 = \boxed{\frac{3}{5}}$.

Problem: The equation $x^2 + 2x = i$ has two complex solutions. Determine the product of their real parts.

Solution: Complete the square by adding 1 to each side.

Then $(x + 1)^2 = 1 + i = e^{\frac{i\pi}{4}} \sqrt{2}$, so $x + 1 = \pm e^{\frac{i\pi}{8}} \sqrt[4]{2}$.

The desired product is then

$$\begin{aligned} & (-1 + \cos(\frac{\pi}{8}) \sqrt[4]{2}) (-1 - \cos(\frac{\pi}{8}) \sqrt[4]{2}) = \\ & 1 - \cos^2(\frac{\pi}{8}) \sqrt{2} = 1 - \frac{(1 + \cos(\frac{\pi}{4}))}{2} \sqrt{2} = \boxed{\frac{1 - \sqrt{2}}{2}} \end{aligned}$$

Problem

Generator

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $2/5$, what is the numerator of the fraction? (Answer: $\boxed{14}$)

Let's call the numerator x.

So the denominator is $3x - 7$.

We know that $x/(3x - 7) = 2/5$.

So $5x = 2(3x - 7)$.

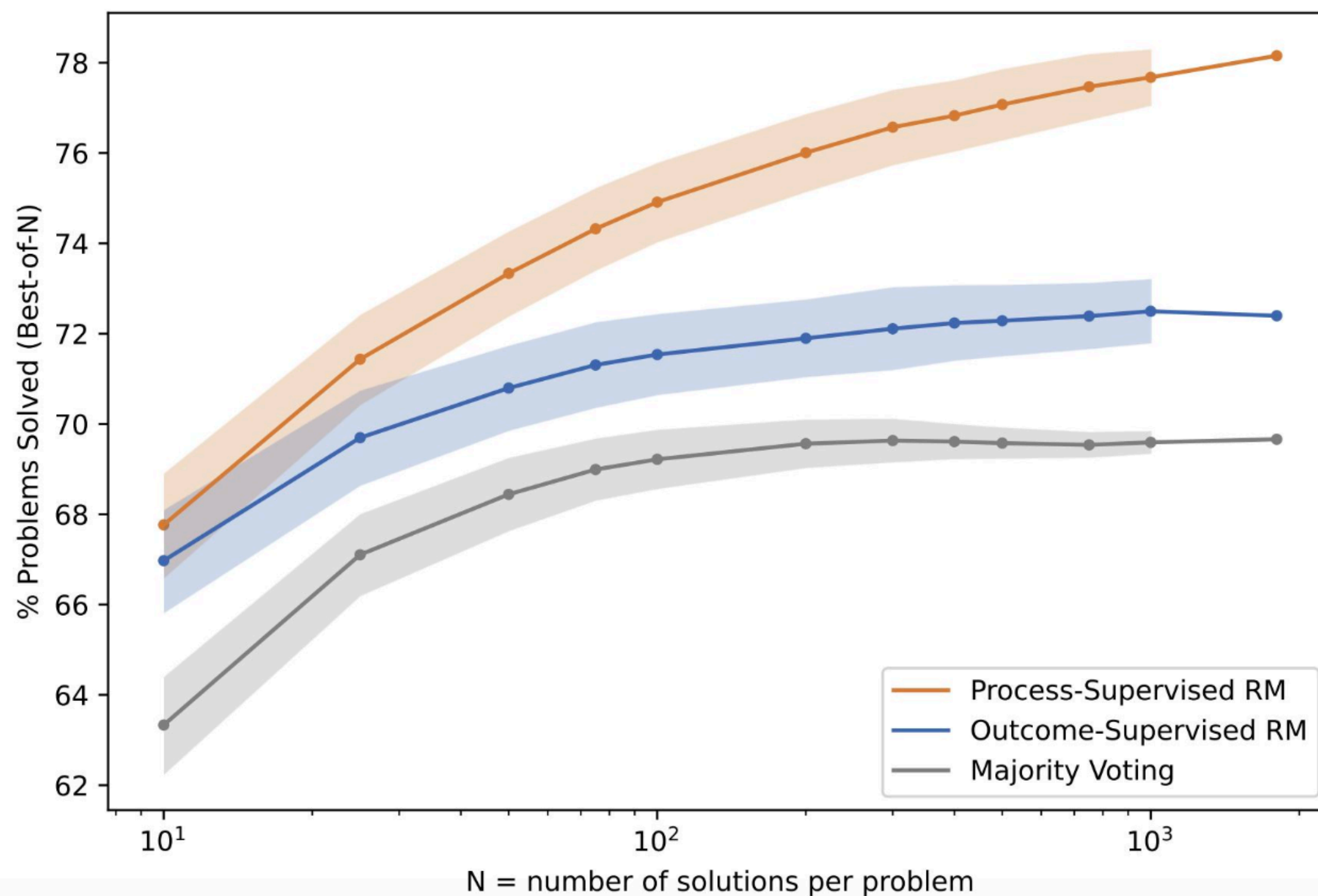
$5x = 6x - 14$.

So $x = 7$.

Training Verifier [Cobbe+ 2023, Lightman+ 2023]

- Strong LLM-based verifiers outperform consistency-based selection

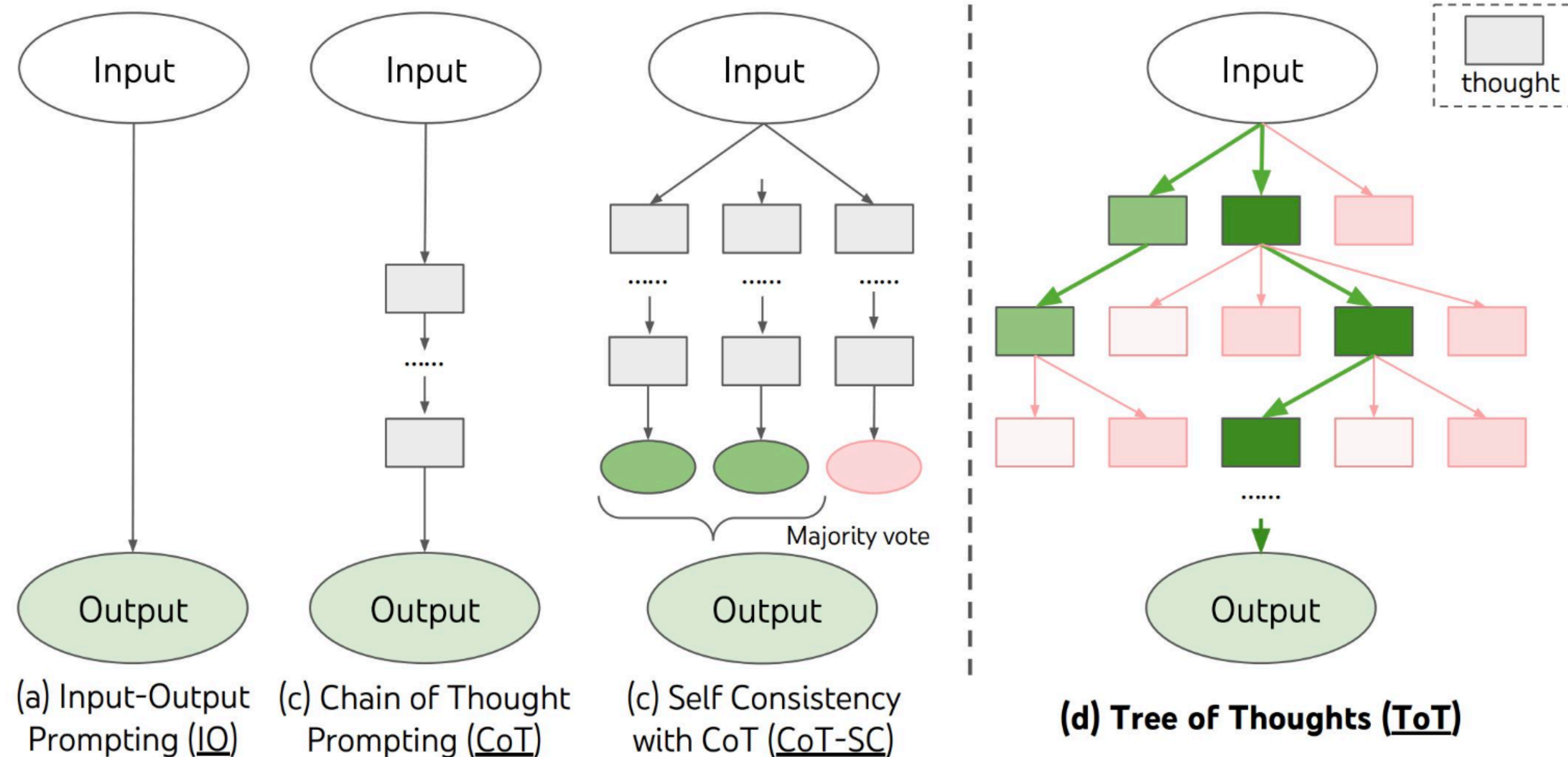
	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	78.2	69.6



- Process-supervised reward model (PRM) scales better with more samples
- Note: the performance is highly dependent on the verifier quality
- The same verifier might not generalize across tasks

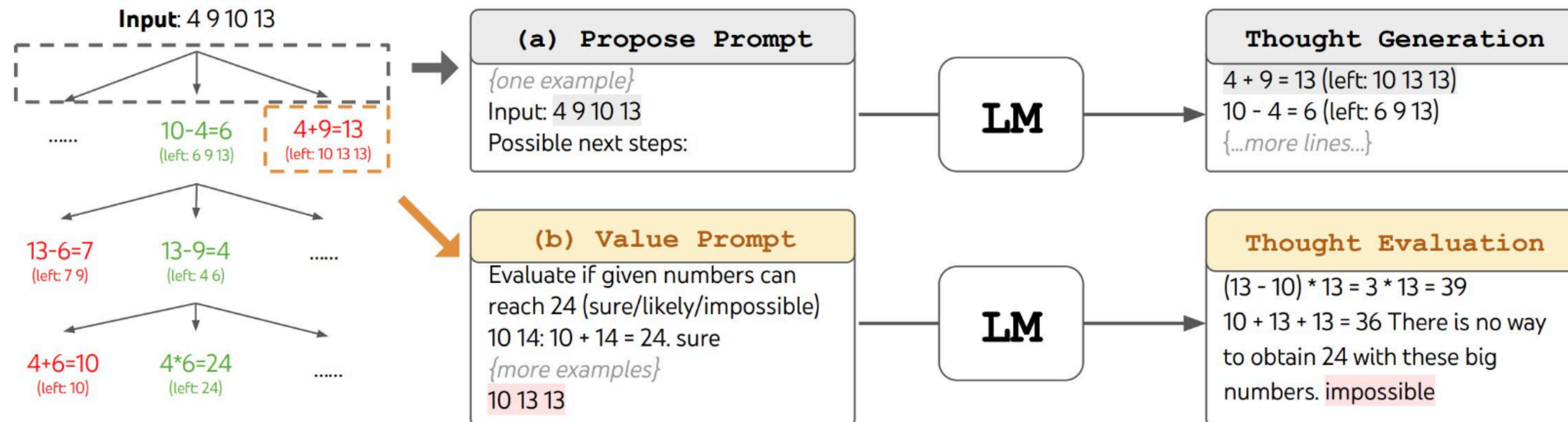
Tree-of-Thought [Yao+ 2023]

- CoT selects responses only after the full responses are generated
 - This does not fully utilize a step-wise scorer!
- LLM + tree search: prioritize the exploration of better partial solutions



Tree-of-Thought [Yao+ 2023]

- Example: Game of 24 (make 24 with given numbers)

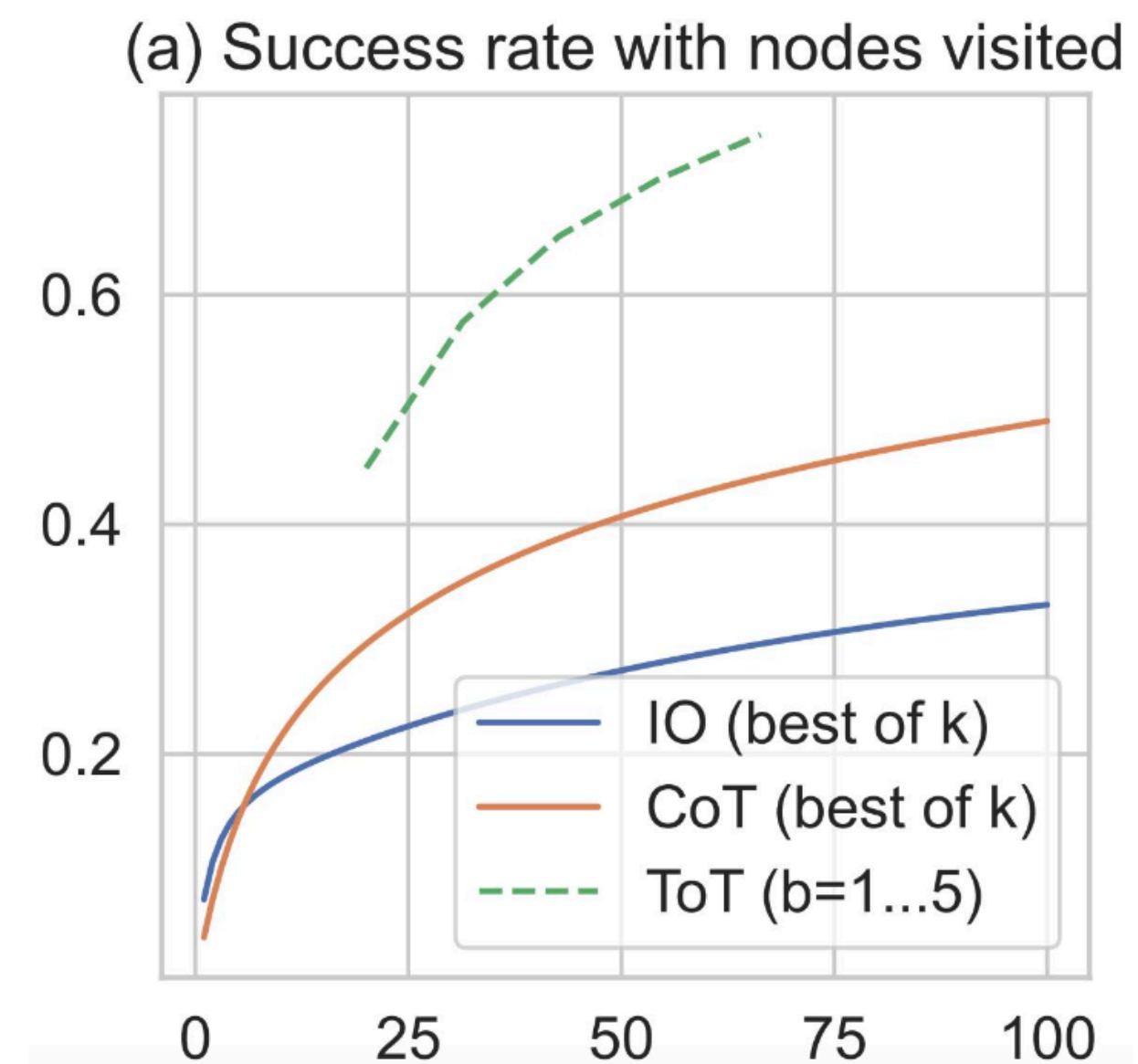


- At each step:
 - Generation: prompt the LLM to propose next thinking steps
 - Evaluation: prompt the LLM to evaluate how good the current state is

Tree-of-Thought [Yao+ 2023]

- ToT with breadth-first search scales better than standard prompting and CoT w.r.t. token budget
- Going further: we can integrate more advanced search algorithms
 - e.g. Monte-Carlo tree search (MCTS)
 - Need a good LLM + prompt design for self-evaluation

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC ($k=100$)	9.0%
ToT (ours) ($b=1$)	45%
ToT (ours) ($b=5$)	74%



Summary So Far

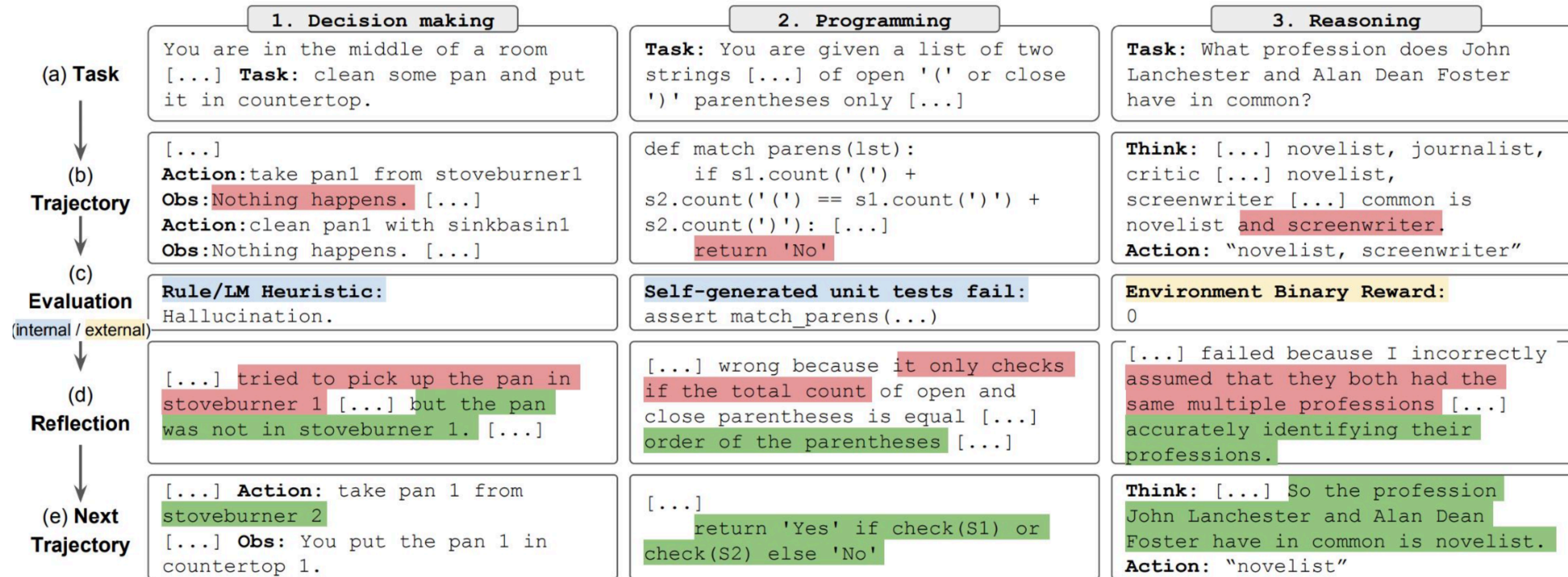
- We can further scale the inference-time compute by **sampling multiple branches in the solution space**
- Consistency-based selection: a simple, effective and general principle
 - CoT-SC: marginalize out (majority vote) reasoning paths and select based on the final answer
- We can attach (trained) external verifiers
- We can search in the partial solution space (tree-of-thought)

Increasing the Trajectory Depth

Even the Best LLMs Make Mistakes...

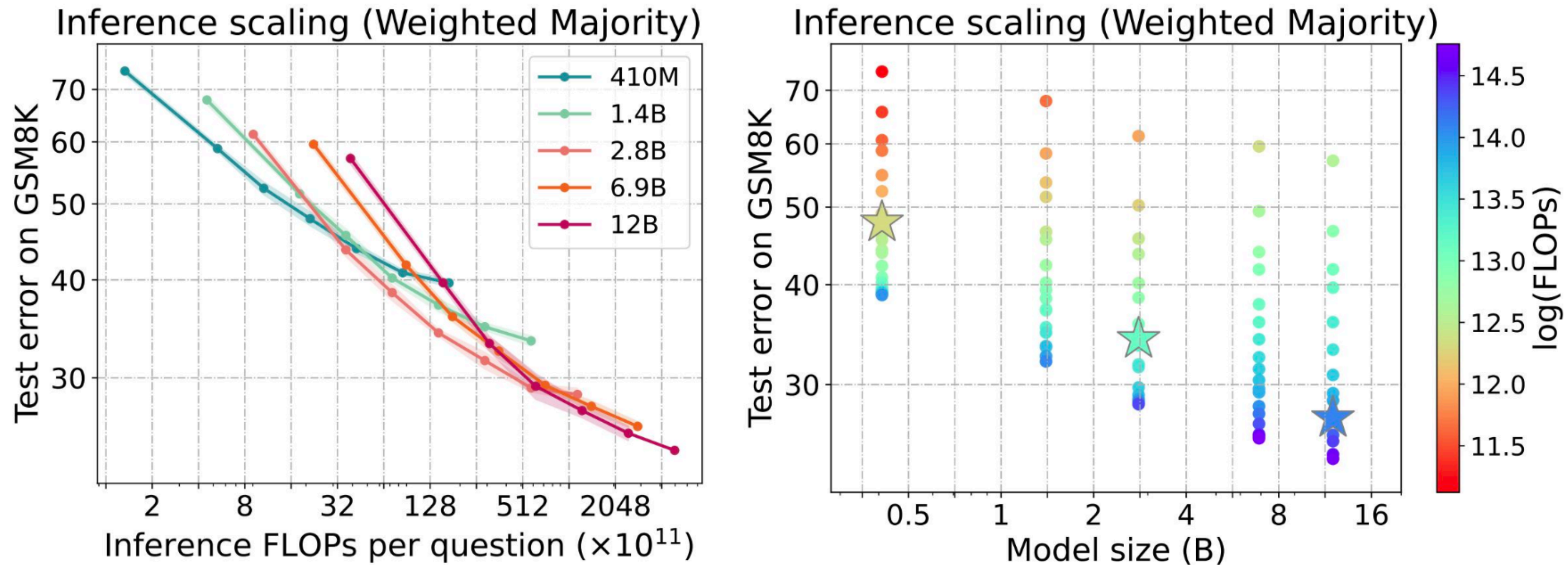
- Humans also make (sometimes trivial) mistakes on first thought!
- Sampling multiple solutions can reduce mistakes from a single prediction, but it is still suboptimal
 - **No feedback loop to correct the mistakes** after a complete solution is generated
- Inference-time self-improvement
 - LLM iteratively improves its own response for the given task

Reflexion and Self-Refine [Shinn+ 2023, Madaan+ 2023]



- LLM generates feedback on its output
- Use external evaluation when available
- Self-refines by referring to internal feedback and external evaluation

Rethinking the Scaling Law [Wu+ 2024]

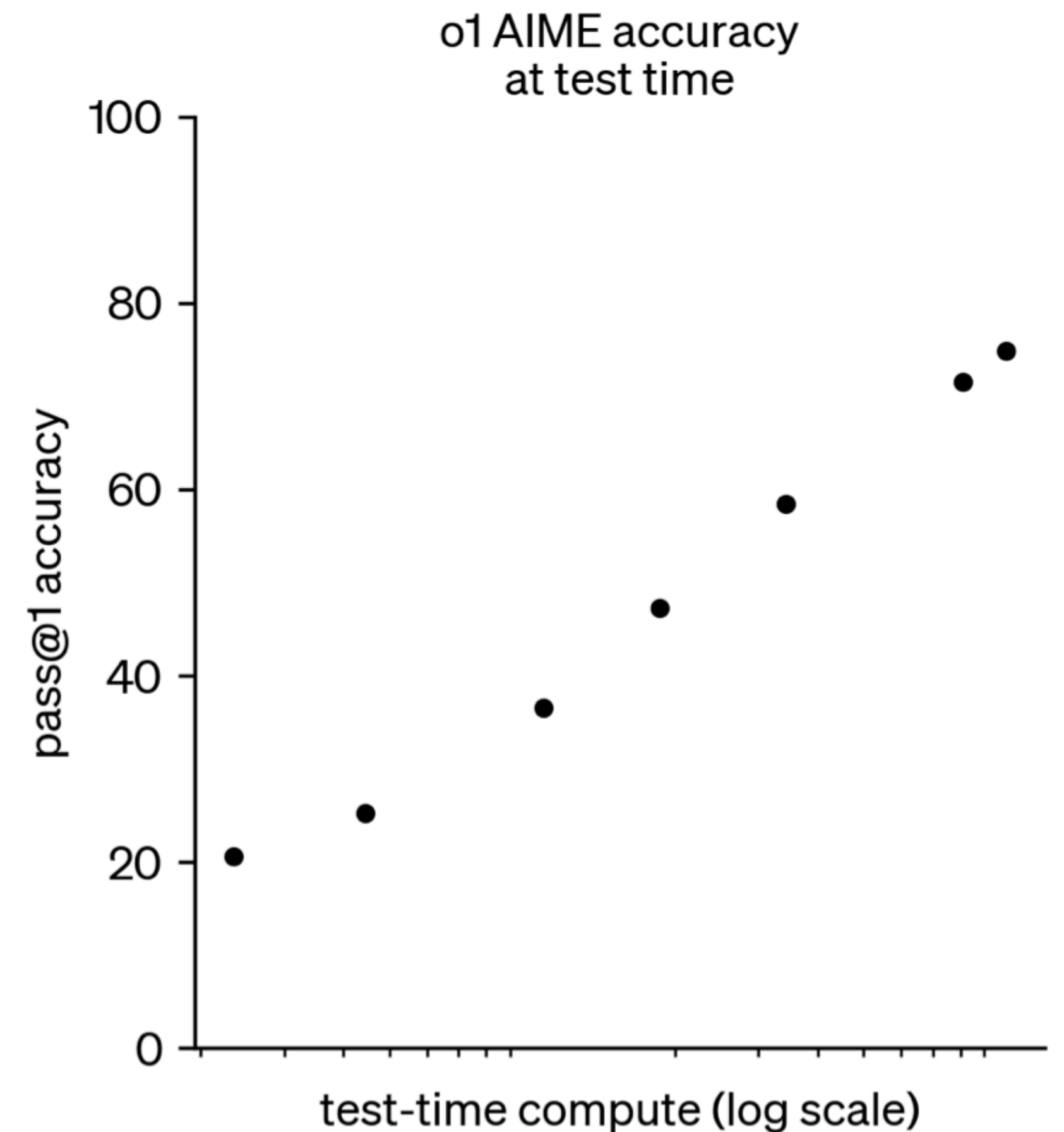


- With the same FLOPs budget at inference, we can sample more solutions (or more reasoning tokens) from a lighter model
- The optimal model can differ depending on the inference budget

Training Reasoning

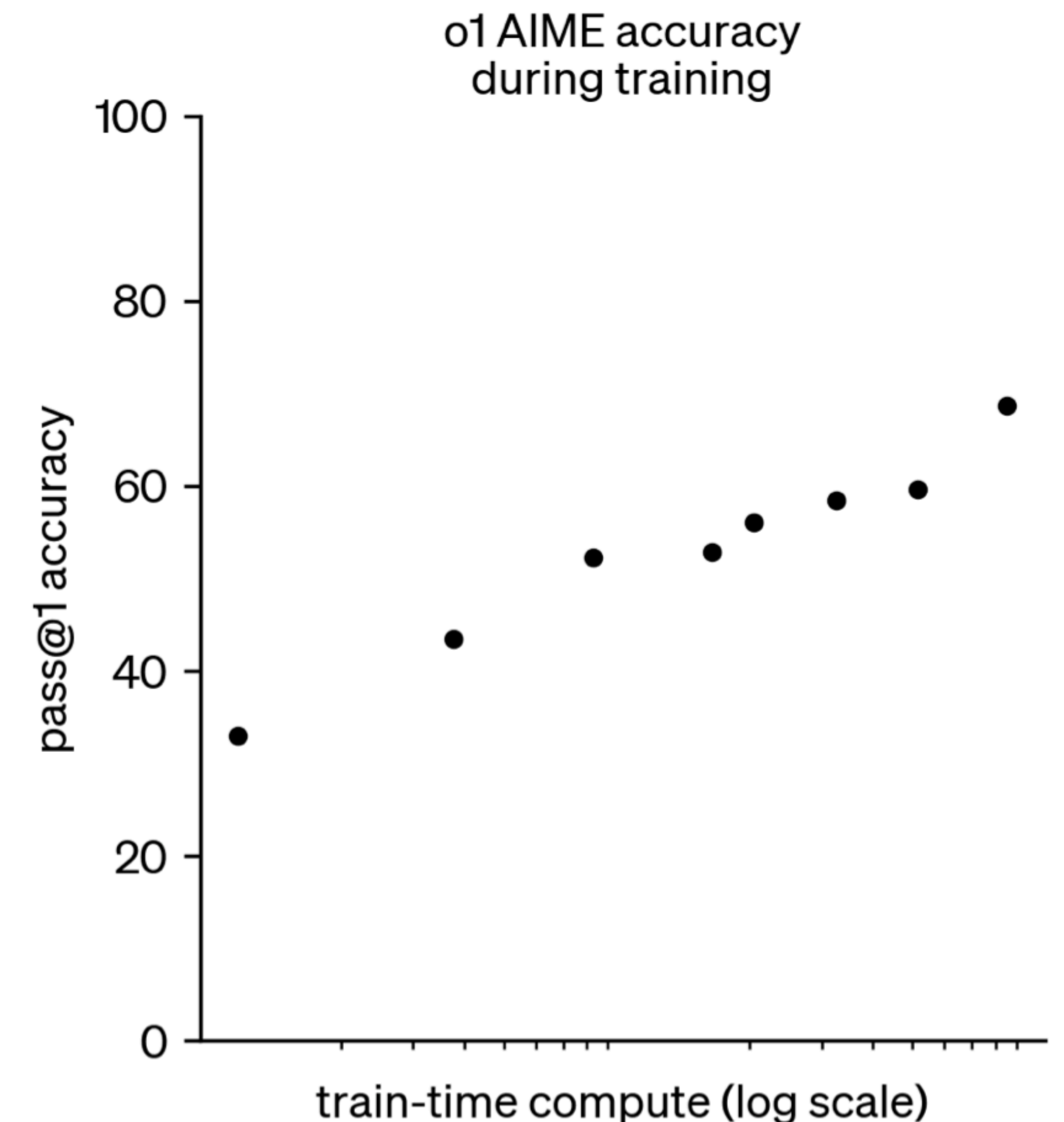
o1: Test-Time Scaling [OpenAI 2024]

- A log-linear relationship between **inference compute** (# tokens generated) and **downstream performance**
 - A fundamental property of models, unlocked in its popular form with RL
 - Can be done in many ways: one long CoT sequence, multiple agents in parallel, or mixes of the two, etc
 - Improving inference-time scaling changes the slope and offset of the curve



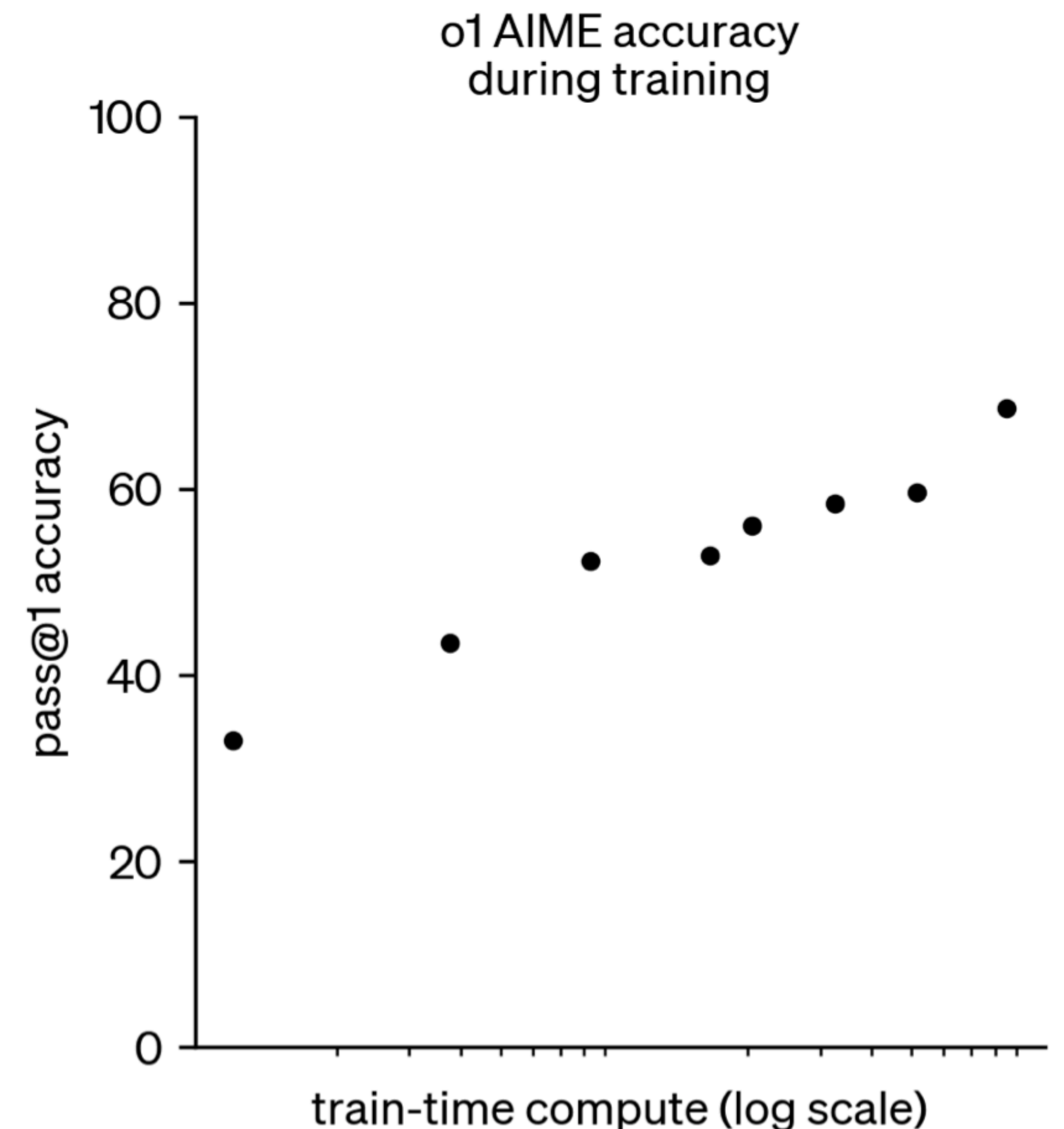
o1: Training-Time Scaling (with RL!) [OpenAI 2024]

- **Scaling RL compute** also has a log-linear return on performance!
- Core question: is scaling RL training just eliciting more from the base model or actually teaching new abilities?
- The results in a two-sided scaling landscape for training language models, both **pre-training** and **post-training**
- The third scaling axis is at **inference** (no weight updates there)



The Scaling RL Era

- RL is now the load-bearing step in training the most capable models
- Reasoning models (o1, DS-R1, etc.) are trained with these on **verifiable rewards**
 - Train-time compute scaling via RL
 - Test-time compute scaling via extended reasoning
 - RL went from "cherry on top" to a core capability driver



The S

- RL is not the most accurate
- Reasoner trained on human data
 - Train
 - Test-reasc
- RL with capabilities

■ "Pure" Reinforcement Learning (cherry)

- ▶ The machine predicts a scalar reward given once in a while.

- ▶ **A few bits for some samples**

■ Supervised Learning (icing)

- ▶ The machine predicts a category or a few numbers for each input

- ▶ Predicting human-supplied data

- ▶ **10→10,000 bits per sample**

■ Unsupervised/Predictive Learning (cake)

- ▶ The machine predicts any part of its input for any observed part.

- ▶ Predicts future frames in videos

- ▶ **Millions of bits per sample**

■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)



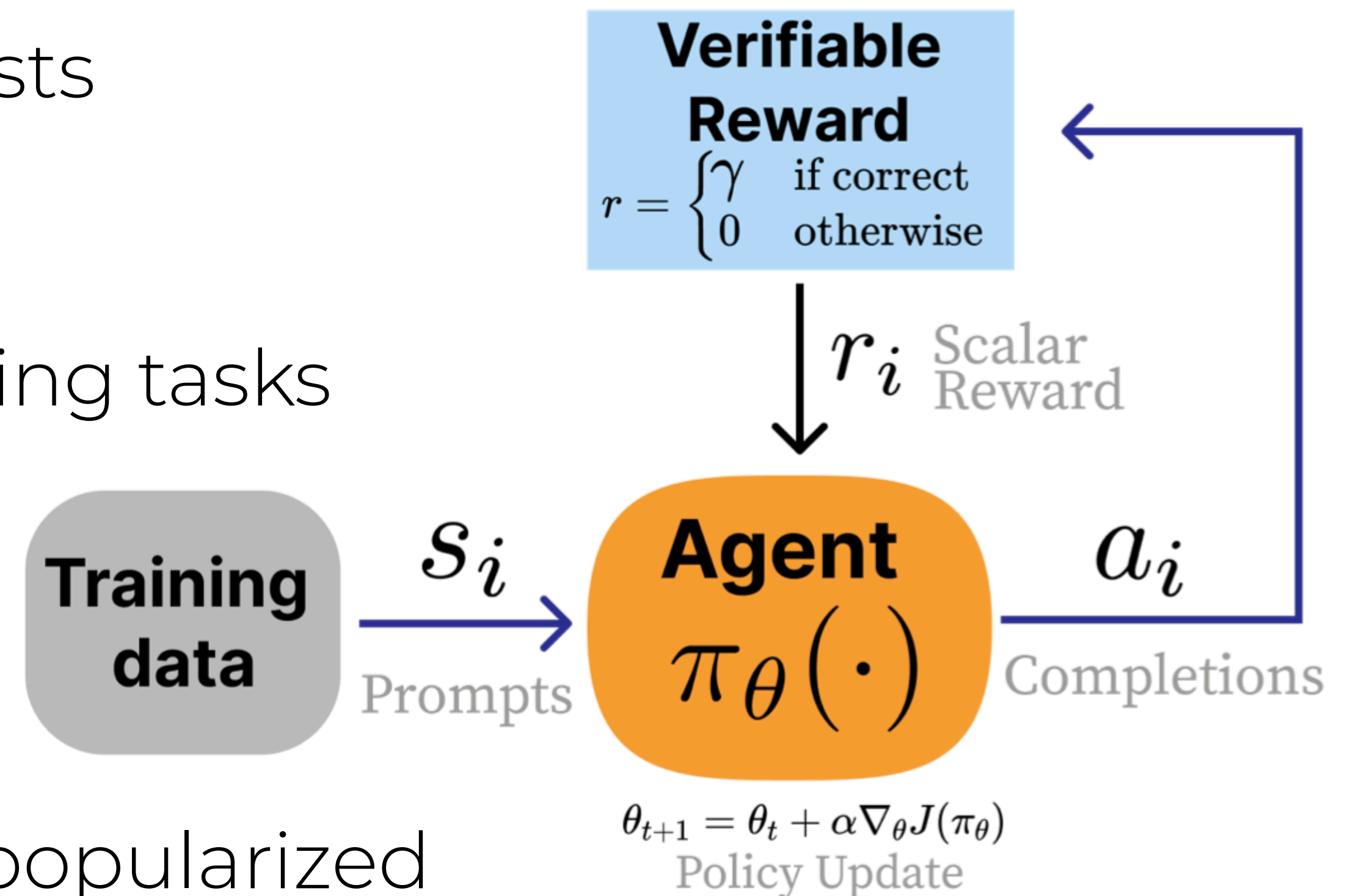
accuracy
training



Yann LeCun, NeurIPS16

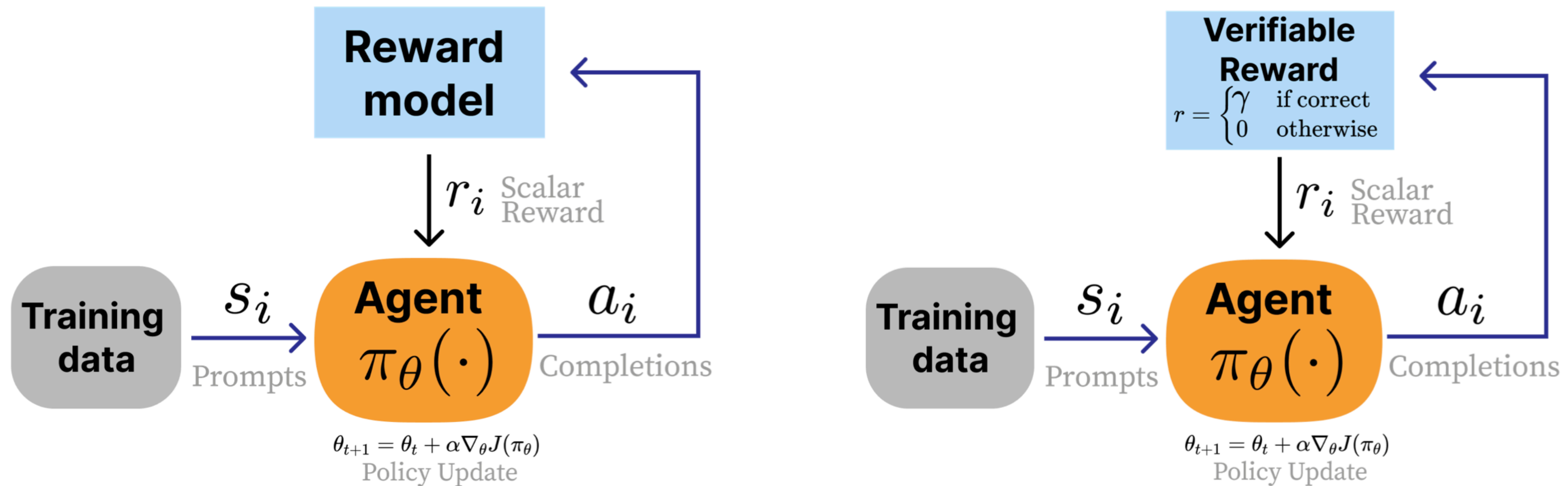
RL with Verifiable Rewards [Lambert+ 2024, Guo+ 2025]

- Apply RL to LLMs when the **answer can be checked directly**
- No need to train a reward model!
 - e.g. Math: final answer / Code: unit tests
- **No learned RM**, so no proxy objective
- Enables scaling RL compute on reasoning tasks
- Unlocked inference-time scaling
 - Spending more token compute increases reasoning performance
- Note: RLVR was named by Tulu 3 and popularized by DeepSeek-R1



RLVR: Same RL, Verifiable Rewards

- RL methods power both RLHF and RLVR
 - **RLHF**: reward model scores subjective quality (human preference)
 - **RLVR**: verification function checks correctness (math, code)
- Same policy gradient algorithms, different reward source



Verifiable Reward

Domain	Verifier Example	Reward Example
Math	Final answer extraction	1 if correct, 0 otherwise
Coding	Unit tests / hidden tests	Fraction of unit tests passed
Instruction following	Constraint checker	Whether format/constraints are satisfied
Theorem/proof	Proof checker	1 if the proof is valid
Tool use	Tool environment	1 when the target state is reached
Agent task	Browser, shell, DB sandbox	Task success, with cost penalty

Classical RL vs. RLHF vs. RLVR

	Classical RL	RLHF	RLVR
Reward	Environment	Learned (proxy)	Verifiable (exact)
State transitions	Yes	No	No
Reward granularity	Per-step	Per-response	Per-response
Primary challenge	Explore-Exploit Trade-off	Over-optimization	Task generalization
Example	CartPole	Chat style tuning	Math reasoning

Why Does RL Work Now?

- **Stability is much more tractable:** Still a first-class research problem (entropy collapse, long-horizon credit), but tooling and recipes are mature enough for widespread adoption
- **Base models are good enough:** Multiple sources suggest RL reasoning training only became viable with models from ~2024 onwards
 - A capability floor was needed
- **Verifiable domains provide clean signal:** Math and code give unambiguous rewards, avoiding the reward hacking issues of RLHF

New Algorithm: GRPO [Shao+ 2024]

- **Group Relative Policy Optimization (GRPO)** was introduced in DeepSeekMath for math reasoning
 - Start with PPO (many parts are similar)
 - Remove the value function / advantage computation
 - Calculate the advantage as "z-score within a group"

Group Relative Policy Optimization In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question q , GRPO samples a group of outputs $\{o_1, o_2, \dots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model π_{θ} by maximizing the following objective:

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

$$\mathbb{D}_{KL}(\pi_{\theta} || \pi_{ref}) = \frac{\pi_{ref}(o_i|q)}{\pi_{\theta}(o_i|q)} - \log \frac{\pi_{ref}(o_i|q)}{\pi_{\theta}(o_i|q)} - 1, \quad (2)$$

where ϵ and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \dots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})}. \quad (3)$$

Recap: REINFORCE

- Gradient ascent on reward can be written as minimizing

$$L_{\text{REINFORCE}}(\theta) = -R(y) \log \pi_{\theta}(y | x)$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{y \sim \pi_{\theta}(\cdot | x)} [R(y) \nabla_{\theta} \log \pi_{\theta}(y | x)].$$

- The core idea is that we sample over trials and estimate the gradient
- For a sampled answer:
 - If the return is high, it will push up the probabilities of the response
 - If the return is low, it will push down the probabilities of the response

- Token-level REINFORCE: $L_{\text{REINFORCE}}(\theta) = -R(x, y) \sum_{t=1}^T \log \pi_{\theta}(y_t | x, y_{<t})$.

Recap: PPO

- Core idea 1: **constrained updates**
 - Large gradient steps can destroy the policy (instability, over-optimization, etc.)
- **Trust regions** limit how far the policy can move in a single update
 - TRPO [Schulman+ 2015] solved this with a hard trust-region constraint, but required expensive second-order optimization (and very hard to understand and implement)

Recap: PPO

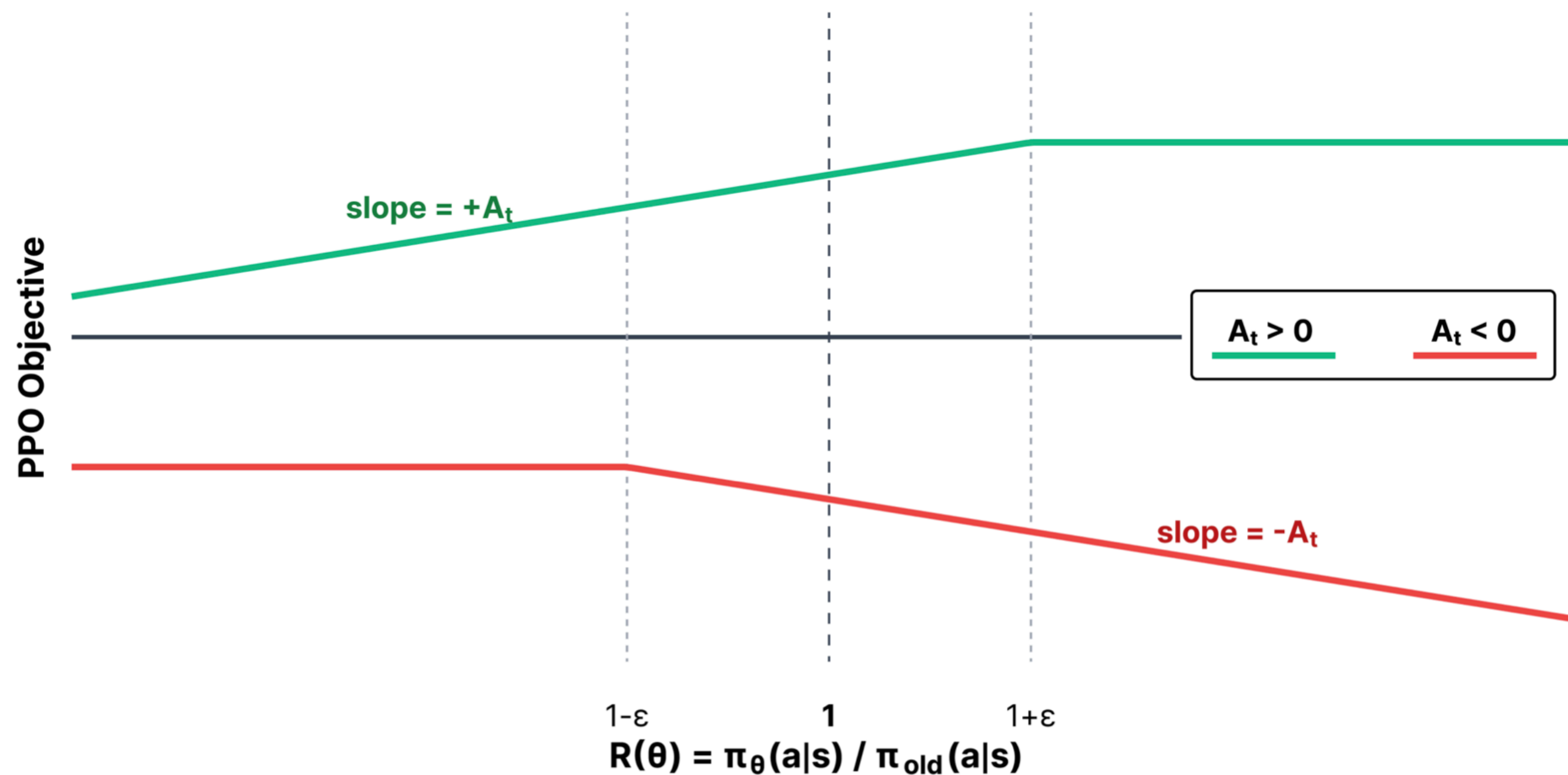
- Core idea 2: **importance sampling**
 - We want to take multiple gradient steps on a batch, but the data came from an old policy
- Define the **importance sampling ratio** $\rho_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\text{old}}(a_t | s_t)}$
 - If $\rho_t = 1$: current and old policy agree on this action
 - If $\rho_t > 1$: current policy assigns higher probability than old
 - If $\rho_t < 1$: current policy assigns lower probability than old
- This re-weights old-policy samples to estimate new-policy gradients

Recap: PPO

- PPO clips the ratio ρ_t to prevent large updates
 - A practical surrogate inspired by trust-region ideas

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t [\min(\rho_t(\theta)A_t, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t)]$$

use $\epsilon = 0.1-0.2$
in practice



Recap: DPO

$$L_{\text{DPO}}(\theta) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_{\text{pref}}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

- DPO increases the probability of chosen responses **relative to the reference model**, and decreases that of rejected responses

- DPO loss gradient is

$$z_{\theta} = \beta \left[\log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right]$$

$$\nabla_{\theta} L_{\text{DPO}} = -\beta \underbrace{\sigma(-z_{\theta})}_{\text{higher weight when (implicit) reward estimate is wrong}} \left[\underbrace{\nabla_{\theta} \log \pi_{\theta}(y_w | x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_{\theta} \log \pi_{\theta}(y_l | x)}_{\text{decrease likelihood of } y_l} \right]$$

higher weight when (implicit) reward estimate is wrong
 If policy already prefers y_w over y_l , less gradient

increase likelihood of y_w

decrease likelihood of y_l

- DPO works like adaptively weighted preference matching

Recap: PPO vs. DPO

Aspect	PPO-RLHF	DPO
Feedback	RM score	preference pair
Reward model	explicit	implicit
Sampling during update	yes	no, usually offline
Main regularizer	KL to reference	reference log-ratio
Optimization	RL loop	supervised-style loss

- PPO requires four models: policy, reference, value function, RM
- DPO requires two models: policy, reference

Are PPO/DPO All We Needed?

- Downsides of **PPO**
 - In practice, complicated implementation
 - PPO needs a value model
 - Memory hungry, involves additional tuning for training
- Downsides of **DPO**
 - Data not inherently pairwise (or in the form of BT comparisons)
 - Offline (though could be made online by iterating)

REINFORCE Leave-One-Out (RLOO)

- **RLOO baseline**: (1) generates K responses per prompt (2) uses the other $K - 1$ rewards as the baseline

$$b(s, a_k) = \frac{1}{K - 1} \sum_{i=1, i \neq k}^K R(s, a_i)$$

- The advantage for response K : $\hat{A}(s, a_k) = R(s, a_k) - b(s, a_k)$
- This is a **per-prompt baseline** that naturally captures prompt difficulty
 - Hard prompts get low rewards across all completions, so the baseline is low

RLOO Example

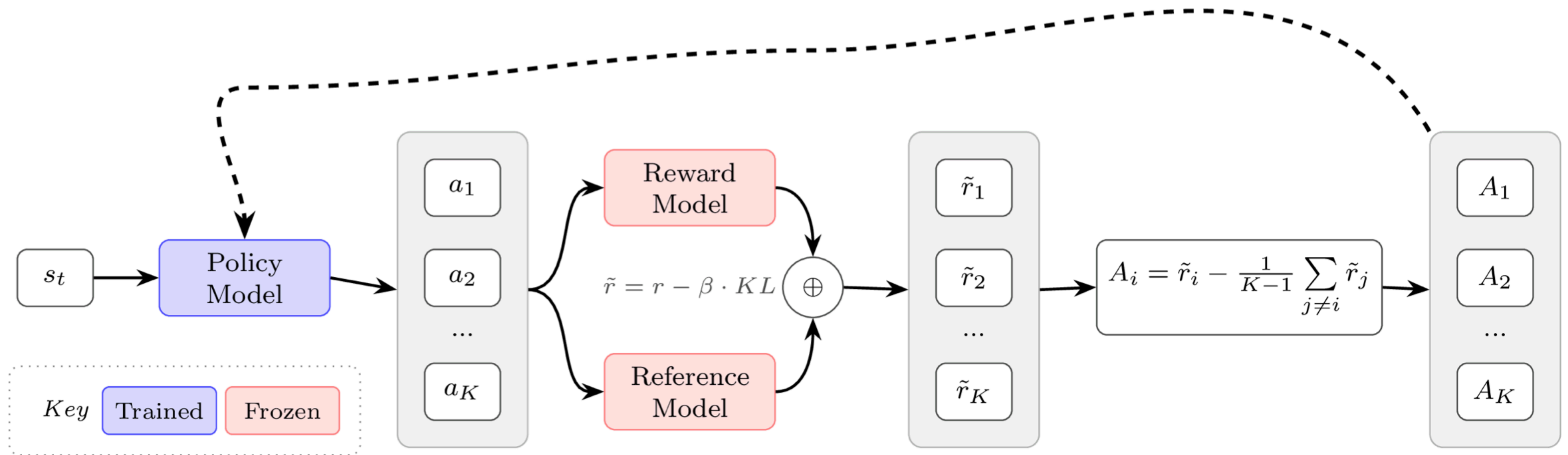
- $K = 4$ responses for one prompt, with rewards $[0.8, 0.3, 0.6, 0.5]$

Completion	Reward	Baseline (avg of others)	Advantage
1	0.8	$(0.3 + 0.6 + 0.5)/3 = 0.467$	+0.333
2	0.3	$(0.8 + 0.6 + 0.5)/3 = 0.633$	-0.333
3	0.6	$(0.8 + 0.3 + 0.5)/3 = 0.533$	+0.067
4	0.5	$(0.8 + 0.3 + 0.6)/3 = 0.567$	-0.067

- Completion 1 (best) gets reinforced
- Completion 2 (worst) gets suppressed
- Completions 3 and 4 get small updates

RLOO

- No value function needed and RLOO baseline is strong in practice
- RLOO is linked intuitively to GRPO



RLOO architecture for language models. Same structure as REINFORCE, but multiple completions per prompt provide the leave-one-out baseline.

GRPO: PPO for Reasoning [Shao+ 2024]

- **Group relative policy optimization (GRPO)**: (1) generate G responses per prompt, (2) use the group's reward statistics as the baseline
 - No learned critic (value function) needed (saves memory)
 - Simpler to implement than PPO
- Natural fit for RLVR where you have a clear verification signal and higher variance rewards (0 or 1 for correctness)

GRPO Advantage

- For a group of G responses with rewards R_1, \dots, R_G

$$\hat{A}_i = \frac{R_i - \text{mean}(R_1, \dots, R_G)}{\text{std}(R_1, \dots, R_G)}$$

- **Z-score normalization**: positive advantage for the above-average responses, negative for below
- Each token in completion i gets the same advantage (sequence-level, not per-token)

GRPO Example

$$\hat{A}_i = \frac{R_i - \text{mean}(R_1, \dots, R_G)}{\text{std}(R_1, \dots, R_G)}$$

- $G = 4$ responses for one prompt, with verifier rewards $R = [1, 0, 1, 0]$
 - Then, $\mu_G = 0.5$, $\sigma_G = 0.5$

- **GRPO** advantage is

$$A^{\text{GRPO}} = \left[\frac{1 - 0.5}{0.5}, \frac{0 - 0.5}{0.5}, \frac{1 - 0.5}{0.5}, \frac{0 - 0.5}{0.5} \right] = [1, -1, 1, -1]$$

- **RLOO** advantage is

$$A^{\text{RLOO}} = \left[1 - \frac{0 + 1 + 0}{3}, 0 - \frac{1 + 1 + 0}{3}, 1 - \frac{1 + 0 + 0}{3}, 0 - \frac{1 + 0 + 1}{3} \right] = \left[\frac{2}{3}, -\frac{2}{3}, \frac{2}{3}, -\frac{2}{3} \right]$$

GRPO Example $\hat{A}_i = \frac{R_i - \text{mean}(R_1, \dots, R_G)}{\text{std}(R_1, \dots, R_G)}$

- What if verifier rewards are $R = [0, 0, 0, 0]$ or $R = [1, 1, 1, 1]$?
 - There are no "relatively better (worse)" responses
 - First prompt is too difficult and second prompt is too easy
 - → No meaningful group relative advantage signal
- Difficulty filtering, dynamic sampling and curriculum would be the key ideas to make useful relative signal

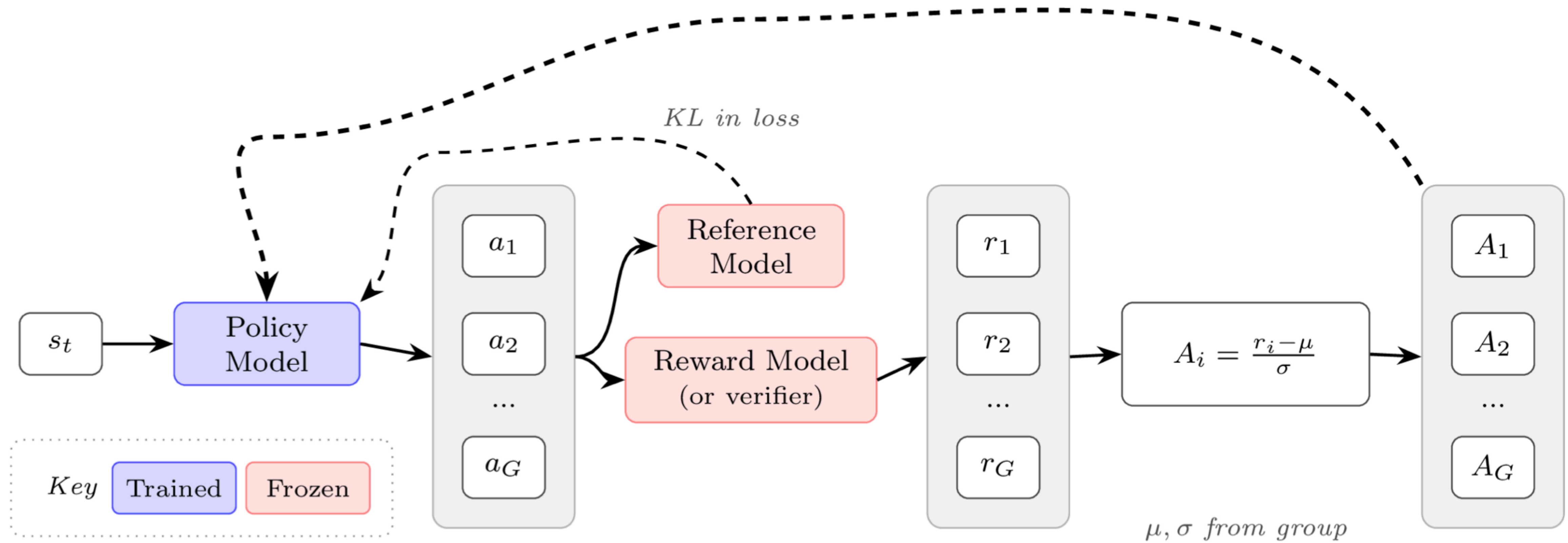
GRPO Objective

- Clipped ratio (like PPO) + GR advantages + KL penalty directly in loss

$$J_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min(\rho_{i,t} A_i, \text{clip}(\rho_{i,t}, 1 - \epsilon, 1 + \epsilon) A_i) \right] - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}})$$

- Per-token importance ratio $\rho_{i,t}(\theta) = \frac{\pi_{\theta}(y_{i,t} \mid x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} \mid x, y_{i,<t})}$, but advantage is shared across all tokens in the response (sequence-level advantage)

GRPO



GRPO architecture for language models. Multiple completions per prompt, group-normalized advantages, no value model needed.

PPO vs. GRPO

	PPO	GRPO
Value function	Learned V_ϕ	None
Advantage	Per-token via GAE	Sequence-level, group z-score
KL penalty	In reward (before advantages)	In loss (default, but optional)
Models in memory	4 (policy, value, ref, RM)	3 (policy, ref, RM) or 2 without KL
Complexity	Higher	Lower
Popular for	General RLHF	Reasoning / RLVR (DeepSeek R1)

- Intuitively, GRPO is PPO minus the value function, with a statistical baseline instead

RLOO vs. GRPO

	RLOO	GRPO
Baseline	Leave-one-out mean	Group mean (z-scored)
Update style	REINFORCE (no clipping)	PPO-style clipped ratio
KL penalty	Optional (in reward)	In loss (default, but optional)
Advantage	$R_k - \frac{1}{K-1} \sum_{j \neq k} R_j$	$\frac{R_i - \text{mean}}{\text{std}}$

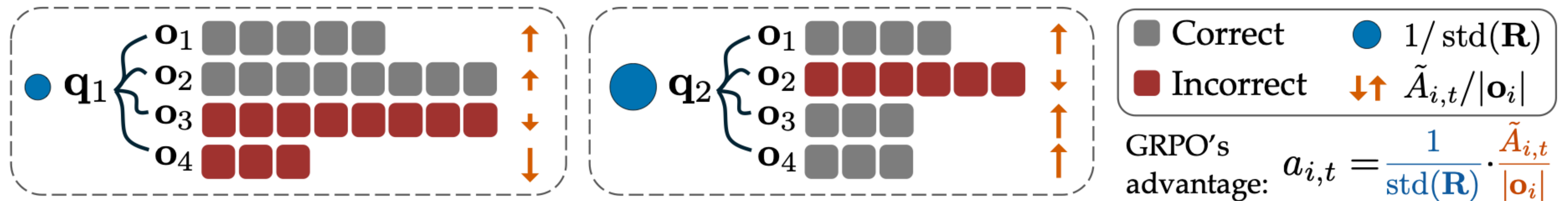
- The key difference is in the PPO-style clipping for GRPO
- Without std normalization, the GRPO-style advantage estimate becomes equivalent to RLOO up to a scaling constant

Dr. GRPO

[Liu+ 2025]

$$J_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min(\rho_{i,t} A_i, \text{clip}(\rho_{i,t}, 1 - \epsilon, 1 + \epsilon) A_i) \right]$$

- Optimization bias in GRPO
 - **Question-level difficulty bias:** GRPO divides rewards by (in-group) σ_G , hence questions with lower std are given higher weights during policy updates
 - **Response-level length bias:** GRPO divides \hat{A} by $|y_i|$ (sampled response length), hence longer responses are penalized less due to their large $|y_i|$, causing the policy to prefer lengthier responses



Dr. GRPO [Liu+ 2025]

- GRPO

- Advantage: $A_i^{\text{GRPO}} = \frac{R_i - \mu_G}{\sigma_G + \epsilon}$

- Length norm: $L_{\text{seq}} = \frac{1}{B} \sum_{i=1}^B \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \ell_{i,t}$

- Dr. GRPO

- Advantage: $A_i^{\text{Dr. GRPO}} = R_i - \mu_G$

- Length norm: $L_{\text{fixed}} = \frac{1}{B} \sum_{i=1}^B \frac{1}{L_{\text{max}}} \sum_{t=1}^{|y_i|} \ell_{i,t}$

- Dr. GRPO = GRPO without std norm and with fixed length norm

GRPO

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_{\theta}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_{\theta}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] \right\},$$

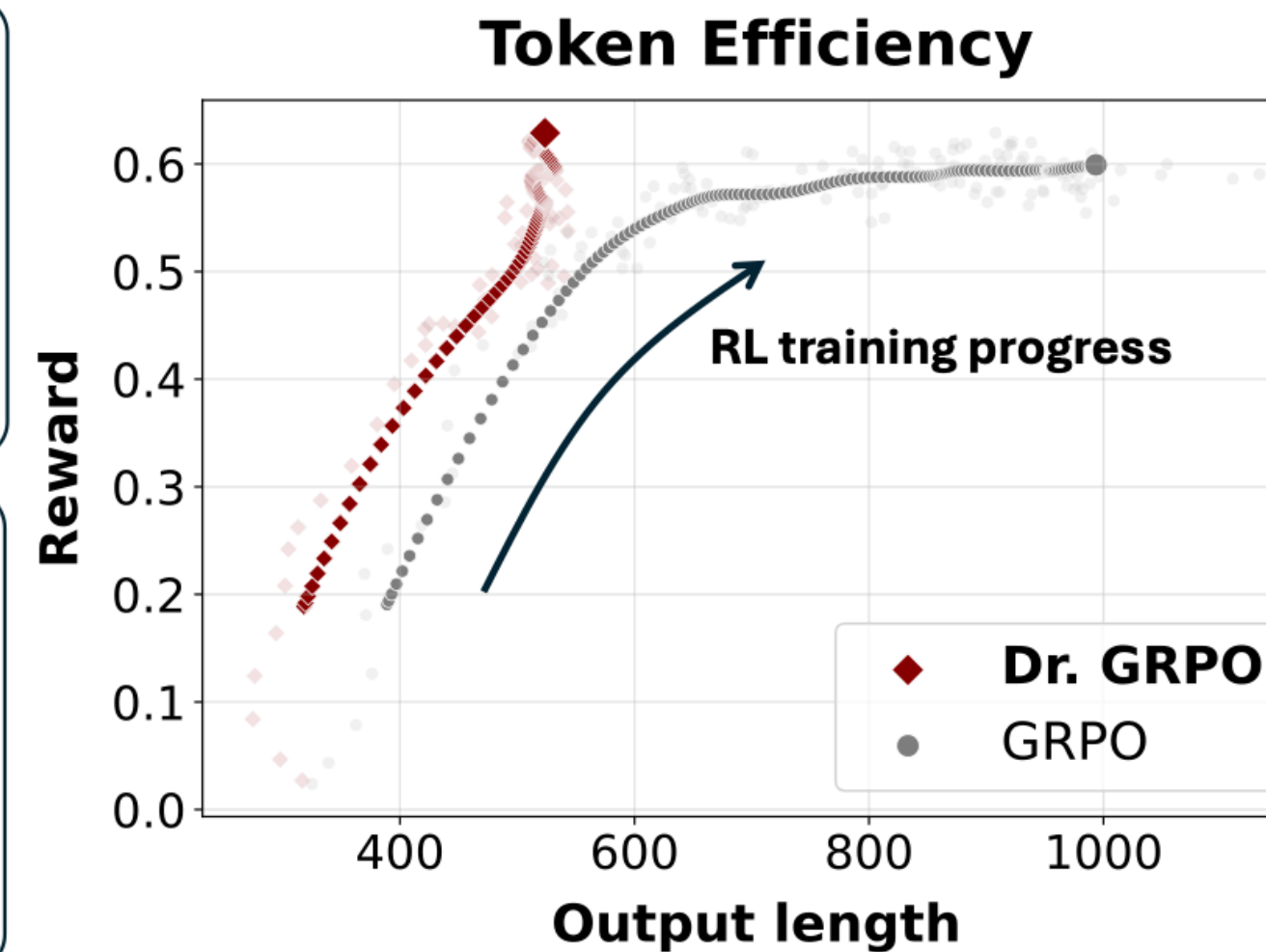
where $\hat{A}_{i,t} = \frac{R(\mathbf{q}, \mathbf{o}_i) - \text{mean}(\{R(\mathbf{q}, \mathbf{o}_1), \dots, R(\mathbf{q}, \mathbf{o}_G)\})}{\text{std}(\{R(\mathbf{q}, \mathbf{o}_1), \dots, R(\mathbf{q}, \mathbf{o}_G)\})}$.

Dr. GRPO

GRPO Done Right (without bias)

$$\frac{1}{G} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_{\theta}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_{\theta}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | \mathbf{q}, \mathbf{o}_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] \right\},$$

where $\hat{A}_{i,t} = R(\mathbf{q}, \mathbf{o}_i) - \text{mean}(\{R(\mathbf{q}, \mathbf{o}_1), \dots, R(\mathbf{q}, \mathbf{o}_G)\})$.



Group Sequence PO (GSPO) [Zheng+ 2025]

- Aggregating token-level importance ratio $\rho_{i,t}(\theta) = \frac{\pi_{\theta}(y_{i,t} | x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} | x, y_{i,<t})}$ across long sequence is unstable
 - A single token with a large ratio can dominate the update
- **GSPO** uses a length-normalized importance weight per response

$$\rho_i^{\text{seq}}(\theta) = \exp \left(\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \log \frac{\pi_{\theta}(y_{i,t} | x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} | x, y_{i,<t})} \right)$$

$$J_{\text{GSPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \min(\rho_i^{\text{seq}} A_i, \text{clip}(\rho_i^{\text{seq}}, 1 - \epsilon, 1 + \epsilon) A_i) \right]$$

$$J_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min(\rho_{i,t} A_i, \text{clip}(\rho_{i,t}, 1 - \epsilon, 1 + \epsilon) A_i) \right]$$

DAPO [Yu+ 2025]

- Decoupled Clip and Dynamic sAmpling Policy Optimization
- What is missing in GRPO?
 - Key technical details of SoTA models (o1, DS-R1, etc) are concealed
 - Naive GRPO does not perform well in practice
 - In particular, this does not work as expected in long-CoT settings

- → A practical recipe of GRPO for long-CoT RLVR model

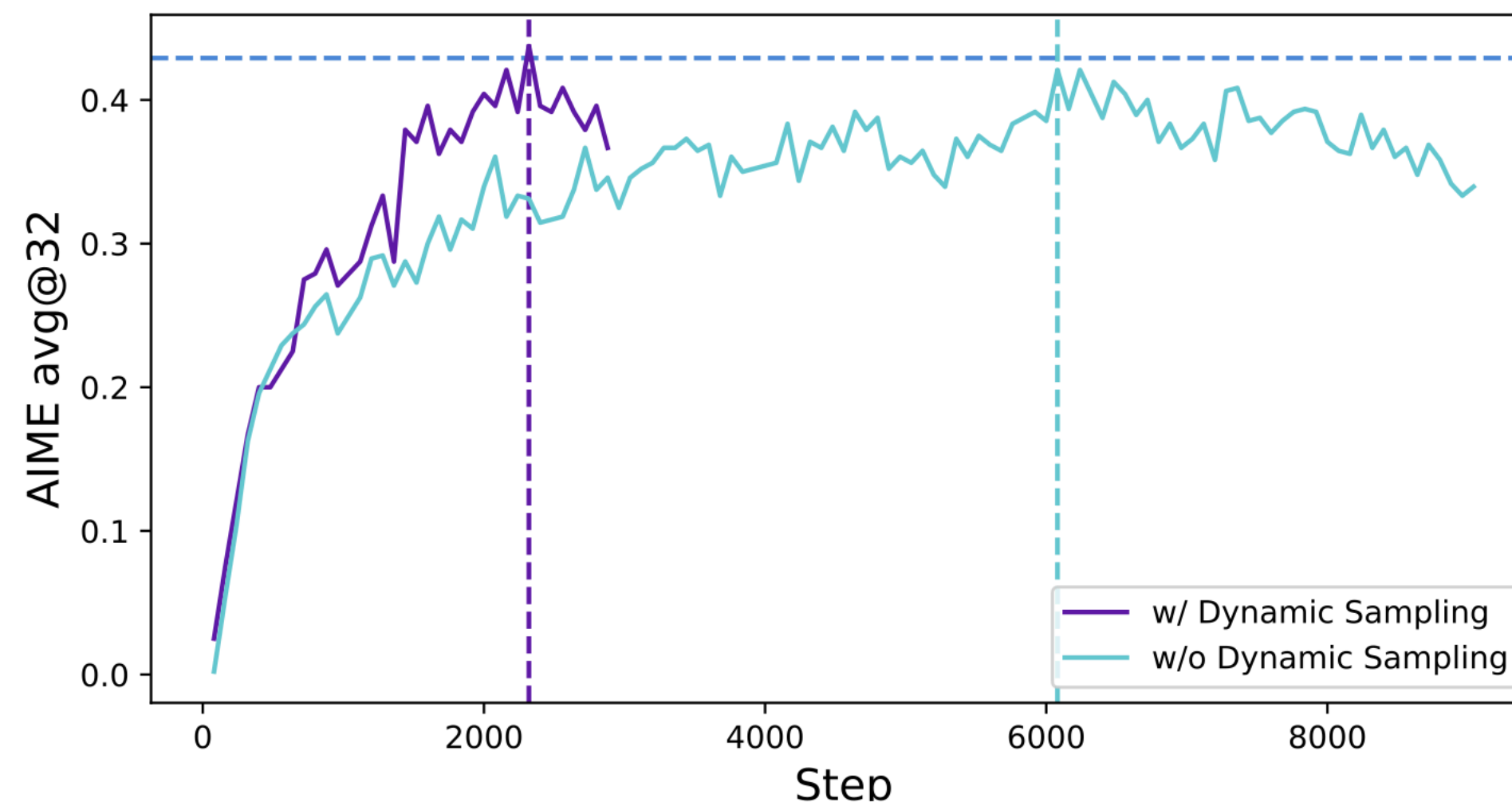
Model	AIME24 _{avg@32}
DeepSeek-R1-Zero-Qwen-32B	47
Naive GRPO	30
+ Overlong Filtering	36
+ Clip-Higher	38
+ Soft Overlong Punishment	41
+ Token-level Loss	42
+ Dynamic Sampling (DAPO)	50

DAPO [Yu+ 2025]

- GRPO baseline suffers from **entropy collapse**
 - RLVR model uses explicit rewards: correct = 1 / wrong = 0
 - The model puts too much probability on a few tokens or patterns
 - The output distribution becomes too sharp
- As a result, the model may:
 - Repeat the same reasoning template
 - Use fewer diverse tokens
 - Generate much shorter or longer outputs
 - Collapse into a fixed response style

DAPO: Dynamic Sampling [Yu+ 2025]

- GRPO works well when success/fail examples are mixed within a group
 - What if the verifier rewards are $R = [0, 0, 0, 0]$ or $R = [1, 1, 1, 1]$?
 - In this case, all the G samples have zero gradient
- **Dynamic sampling**: over-sample and filter out prompts with the accuracy equal to 1 or 0 $0 < \left| \{o_i \mid \text{is_equivalent}(a, o_i)\} \right| < G$.

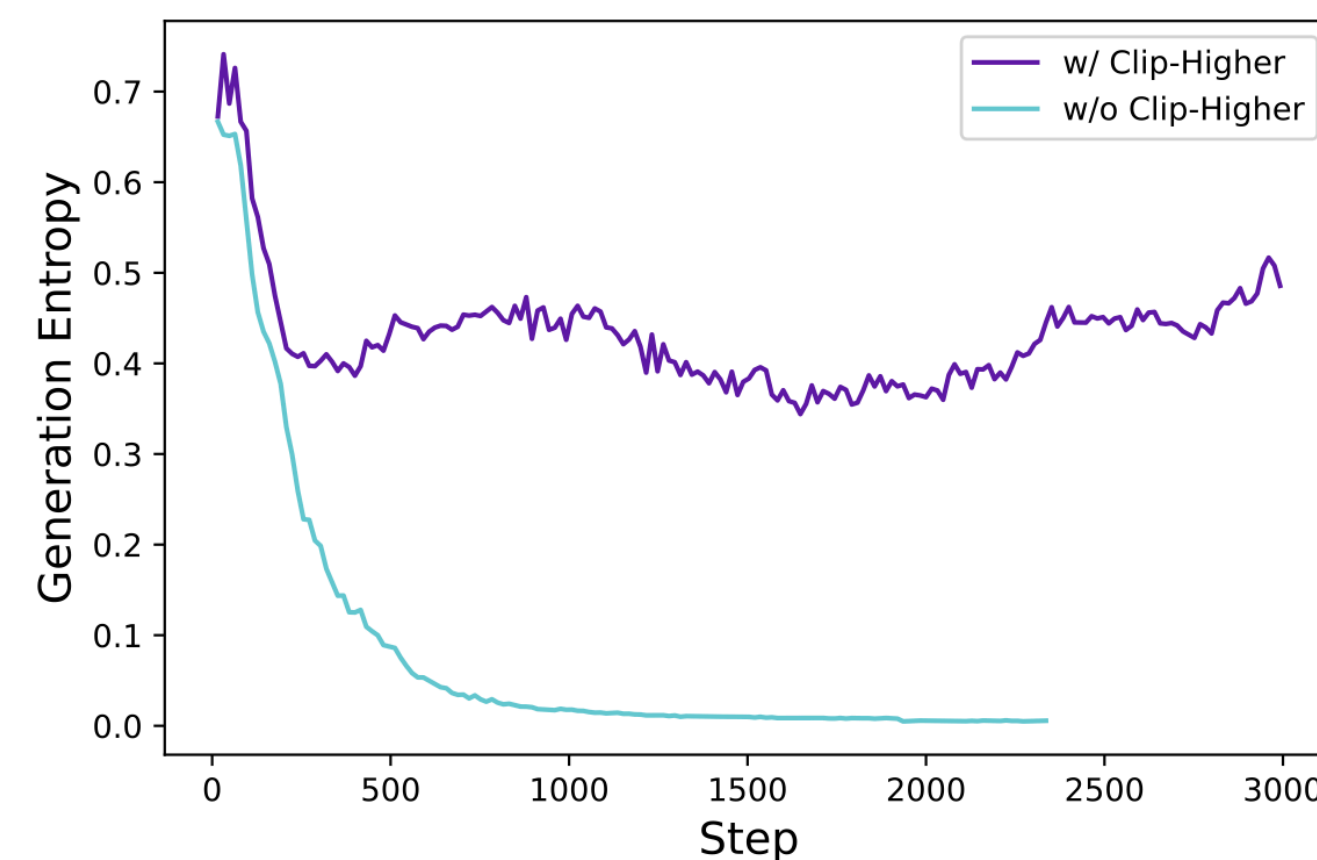
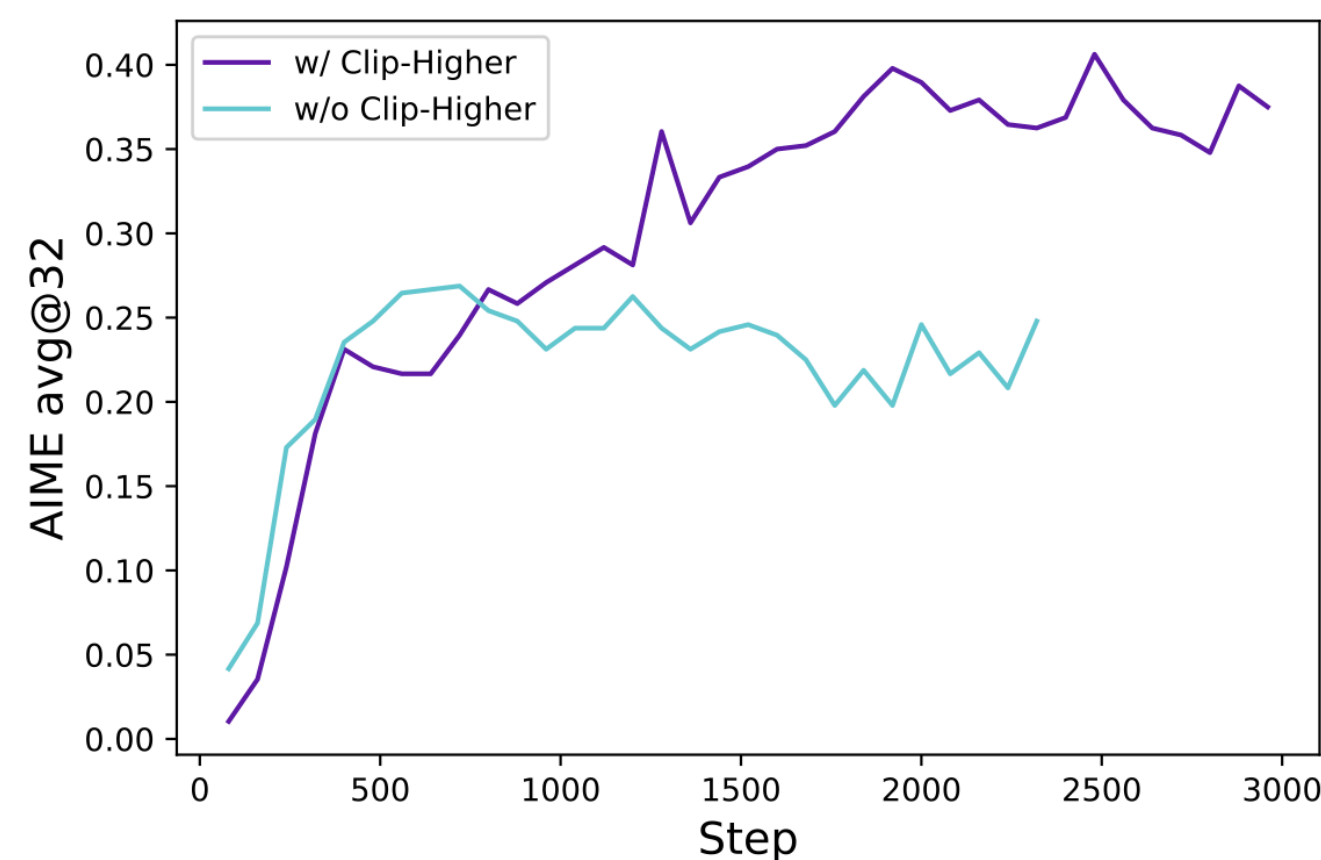


DAPO: Clip-Higher [Yu+ 2025]

- PPO/GRPO suppress large policy updates with clipping: $\text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon)$
 - But clipping can also limit exploration too much
- **Decoupled clipping**: allow a larger upper clip for good trajectories

$$\text{clip}(\rho, 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}), \quad \epsilon_{\text{high}} > \epsilon_{\text{low}}$$

- Intuition: Be conservative when pushing bad trajectories away, but more aggressive when reinforcing good reasoning trajectories
- 👍 Better entropy, exploration, and diversity in long-CoT reasoning



DAPO: Token-Level PG [Yu+ 2025]

- Issue of sample-level loss in GRPO $\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|}$
 - Long-CoT contains many tokens
 - → Each token receives a weaker learning signal
 - Long, high-quality reasoning may be under-trained
 - Leads to an unhealthy increase in entropy and response length
- **Token-level loss**: aggregate over all tokens across the group $\frac{1}{\sum_{i=1}^G |y_i|} \sum_{i=1}^G \sum_{t=1}^{|y_i|}$
 - This does not fully solve credit assignment
 - But it reduces the chance that the learning signal from long reasoning traces gets diluted.

DAPO: Soft Overlong Punishment [Yu+ 2025]

- In GRPO, if the output exceeds the max length, it is truncated and receives reward = -1 (**reward noise**)
 - e.g. The reasoning is almost correct, but the last 10 tokens are truncated, then the whole response receives negative reward
- **Soft overlong punishment**: Add a buffer zone near the max length
 - Avoid a sudden reward drop to -1
 - Treat length overflow as a gradual penalty, not a binary failure

$$R_{\text{length}}(y) = \begin{cases} 0, & |y| \leq L_{\text{max}} - L_{\text{cache}} \\ \frac{(L_{\text{max}} - L_{\text{cache}}) - |y|}{L_{\text{cache}}}, & L_{\text{max}} - L_{\text{cache}} < |y| \leq L_{\text{max}} \\ -1, & L_{\text{max}} < |y| \end{cases}$$

DAPO: Objective [Yu+ 2025]

$$J_{\text{DAPO}}(\theta) = \mathbb{E}_{(x,a) \sim \mathcal{D}, \{y_i\}_{i=1}^G \sim \pi_{\text{old}}(\cdot|x)} \left[\frac{1}{\sum_{i=1}^G |y_i|} \sum_{i=1}^G \sum_{t=1}^{|y_i|} \min(\rho_{i,t} A_i, \text{clip}(\rho_{i,t}, 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}) A_i) \right]$$

s.t. $0 < |\{i : y_i \text{ correct}\}| < G.$

- Normally, DAPO removes KL penalty
- GRPO:

$$J_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min(\rho_{i,t} A_i, \text{clip}(\rho_{i,t}, 1 - \epsilon, 1 + \epsilon) A_i) \right] - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}})$$

Summary: RLVR

	Classical RL	RLHF	RLVR
Reward	Environment	Learned (proxy)	Verifiable (exact)
State transitions	Yes	No	No
Reward granularity	Per-step	Per-response	Per-response
Primary challenge	Explore-Exploit Trade-off	Over-optimization	Task generalization
Example	CartPole	Chat style tuning	Math reasoning

Summary: GRPO Family

Method	What Changes?	Key Idea	Main Goal
PPO	Uses a value model and GAE	Actor-critic RL with clipping	Stable RLHF
RLOO	Removes the value model	Uses leave-one-out group average as the	Critic-free RL
GRPO	Uses group-normalized advantage	Subtracts group mean and divides by group std	Value-free RL for RLVR
Dr. GRPO	Removes biased normalization terms	Uses group mean only; removes std and length normalization	Less biased GRPO
GSPO	Changes the importance ratio	Uses sequence-level ratio instead of token-level ratio	Stable sequence-level RL
DAPO	Adds practical long-CoT training tricks	Dynamic sampling, clip-higher, token-level loss, soft overlong penalty	Exploration and scaling