

Reinforcement Learning for LLM Reasoning

Aviral Kumar

Carnegie Mellon University
School of Computer Science

Setup for Today: Solving Math Problems!

Problem 1

Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b .

Solution 1 (thorough)

We are tasked with finding the number of integer bases $b > 9$ such that $\frac{9b+7}{b+7} \in \mathbf{Z}$. Notice that

$$\frac{9b+7}{b+7} = \frac{9b+63-56}{b+7} = \frac{9(b+7)-56}{b+7} = 9 - \frac{56}{b+7}$$

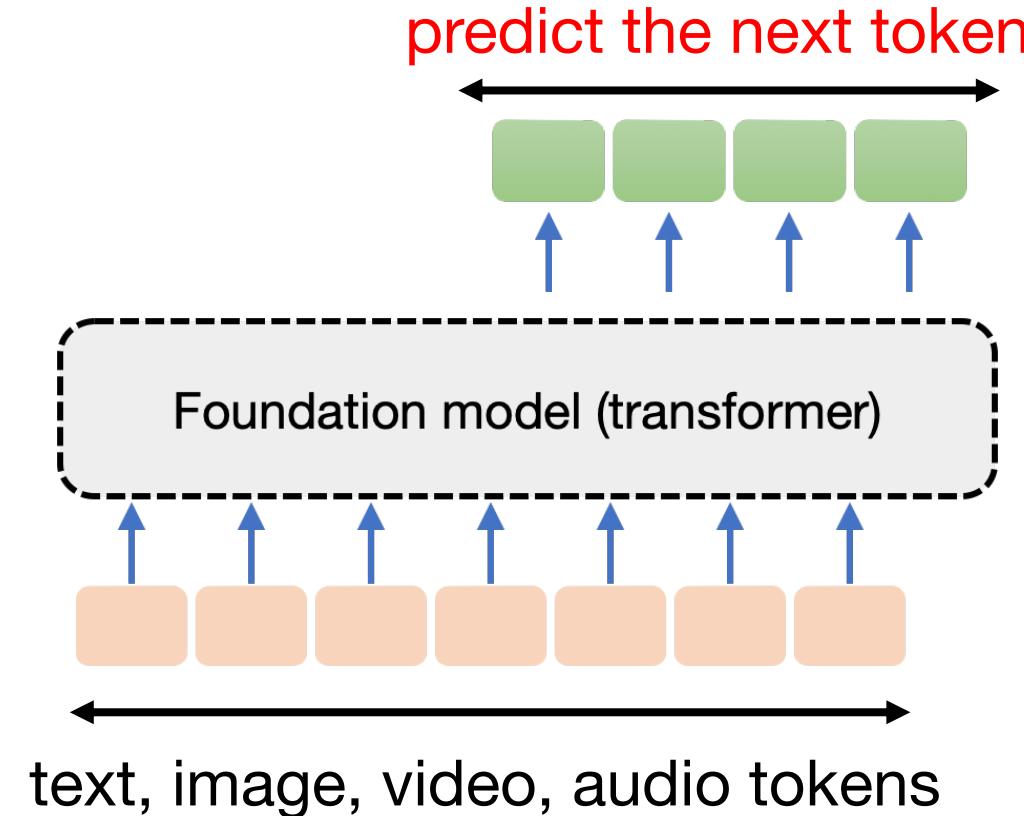
so we need only $\frac{56}{b+7} \in \mathbf{Z}$. Then $b+7$ is a factor of 56.

The factors of 56 are 1, 2, 4, 7, 8, 14, 28, 56. Of these, only 8, 14, 28, 56 produce a positive b , namely $b = 1, 7, 21, 49$ respectively. However, we are given that $b > 9$, so only $b = 21, 49$ are solutions. Thus the answer is $21 + 49 = \boxed{070}$. ~eevee9406

Solution 2 (quick)

We have, $b+7 \mid 9b+7$ meaning $b+7 \mid -56$ so taking divisors of 56 under bounds to find $b = 49, 21$ meaning our answer is $49 + 21 = \boxed{070}$.

The Conventional Way of Training



Why does next token prediction work?

ground-truth model

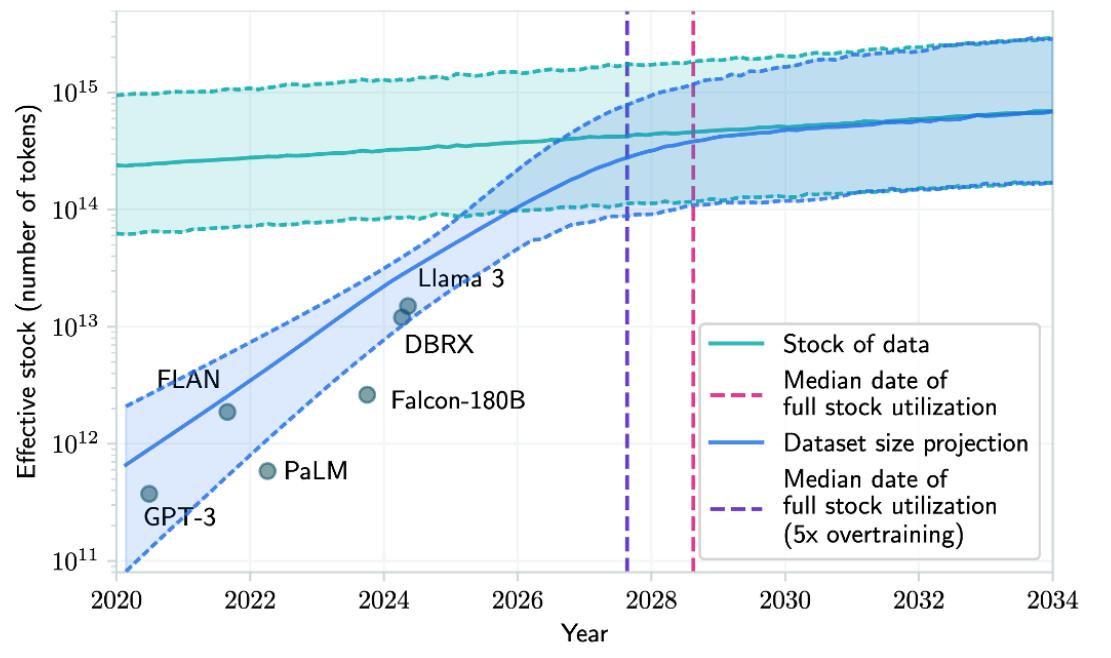
$$|\hat{p}_\theta(\mathbf{y}|\mathbf{x}) - p^*(\mathbf{y}|\mathbf{x})| \propto \frac{1}{|\mathcal{D}(\mathbf{y}|\mathbf{x})|^\alpha}$$

Learned model

Error reduces with more data similar to the target \mathbf{x}

But in Many Problems, Data is Limited/Biased...

Math reasoning

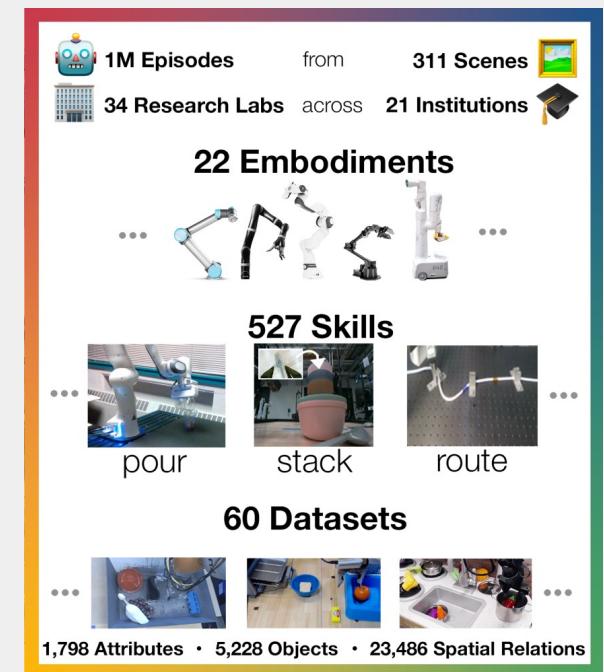


It is estimated we will run out of high-quality Internet text by 2028

Embodied AI

DROID Distributed Robot Interaction Dataset

- 76k Episodes
- 564 Scenes
- 52 Buildings
- 13 Institutions
- 86 Tasks / Verbs



Real-world is stochastic, this is little data compared to LLMs!

And Training on Current Data is Not Enough

Show that the inequality

$$\sum_{i=1}^n \sum_{j=1}^n \sqrt{|x_i - x_j|} \leq \sum_{i=1}^n \sum_{j=1}^n \sqrt{|x_i + x_j|}$$

holds for all real numbers x_1, x_2, \dots, x_n .

Analyzing individual terms is not good enough

The model “**asserts**” its way out, no realization of its mistake / backtracking

Analyzing $\sqrt{|x_i - x_j|}$ and $\sqrt{|x_i + x_j|}$

1. Absolute Value Cases:

- When $x_i = x_j$, $\sqrt{|x_i - x_j|} = \sqrt{0} = 0$ and $\sqrt{|x_i + x_j|} = \sqrt{|2x_i|}$.
- When $x_i \neq x_j$, we need to analyze how $|x_i - x_j|$ compares to $|x_i + x_j|$.

2. Comparing $|x_i - x_j|$ and $|x_i + x_j|$:

Consider two cases for (x_i, x_j) :

- **Case 1:** $x_i \geq 0$ and $x_j \geq 0$: Here both x_i and x_j are non-negative. We have:

$$|x_i - x_j| = x_i - x_j \quad \text{and} \quad |x_i + x_j| = x_i + x_j.$$

Clearly, $x_i + x_j \geq x_i - x_j$ if $x_i \geq x_j$ or $x_j \geq x_i$.

- **Case 2:** $x_i \leq 0$ and $x_j \leq 0$: Both are non-positive. We have:

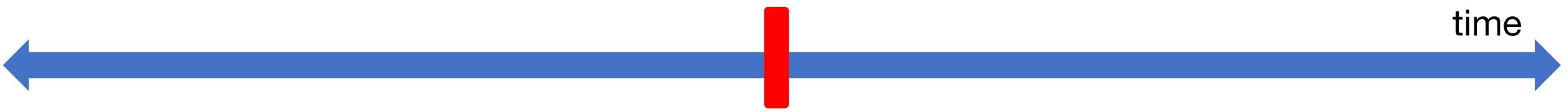
$$|x_i - x_j| = -(x_i - x_j) = x_j - x_i \quad \text{and} \quad |x_i + x_j| = -(x_i + x_j) = -(x_i + x_j).$$

Similarly, $|x_i + x_j|$ would be $|-x_i - x_j|$ and the same argument holds.

- **Case 3:** $x_i \geq 0$ and $x_j \leq 0$ or vice versa: This makes $(x_i - x_j)$ and $(x_i + x_j)$ different in signs or their modulus. The absolute values may change based on these signs but we still have the result as:

$$\sqrt{|x_i + x_j|} \geq \sqrt{|x_i - x_j|}$$

Outline: RL for Reasoning



DeepSeek-R1 /
“thinking” models

Classical RL techniques for LLM reasoning

1. Imitation Learning
2. Offline reinforcement learning
3. Online reinforcement learning

Takeaway: Training with RL can help improve efficiency of learning!

Modern extensions of these RL techniques

1. Online reinforcement learning
2. Some Extensions

Summary: Still the old recipes and RL ideas are helpful!

Problem Setup: LLM Math Reasoning Problems

Initial state

Problem: Suppose a and b are positive real numbers with $a > b$ and $ab = 8$. Find the minimum value of $\frac{a^2+b^2}{a-b}$.

Ground truth solution: We can write $\frac{a^2+b^2}{a-b} = \frac{a^2+b^2-2ab+16}{a-b} = \frac{(a-b)^2+16}{a-b} = a-b + \frac{16}{a-b}$. By AM-GM, $a-b + \frac{16}{a-b} \geq 2\sqrt{(a-b) \cdot \frac{16}{a-b}} = 8$. Equality occurs when $a-b = 4$ and $ab = 8$. We can solve these equations to find $a = 2\sqrt{3} + 2$ and $b = 2\sqrt{3} - 2$. Thus, the minimum value is 8.

Steps = actions

reward = 1 if
answer is correct

Intuition: Sparse-reward MDP with deterministic dynamics

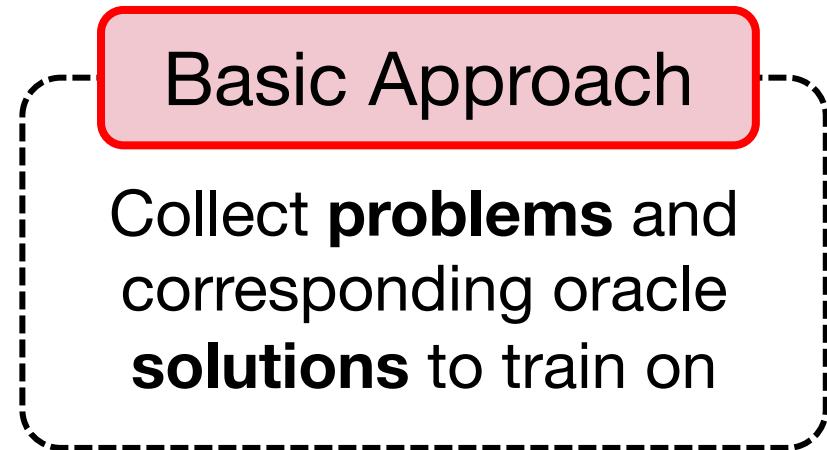
Part 1

Classical RL Methods for Reasoning

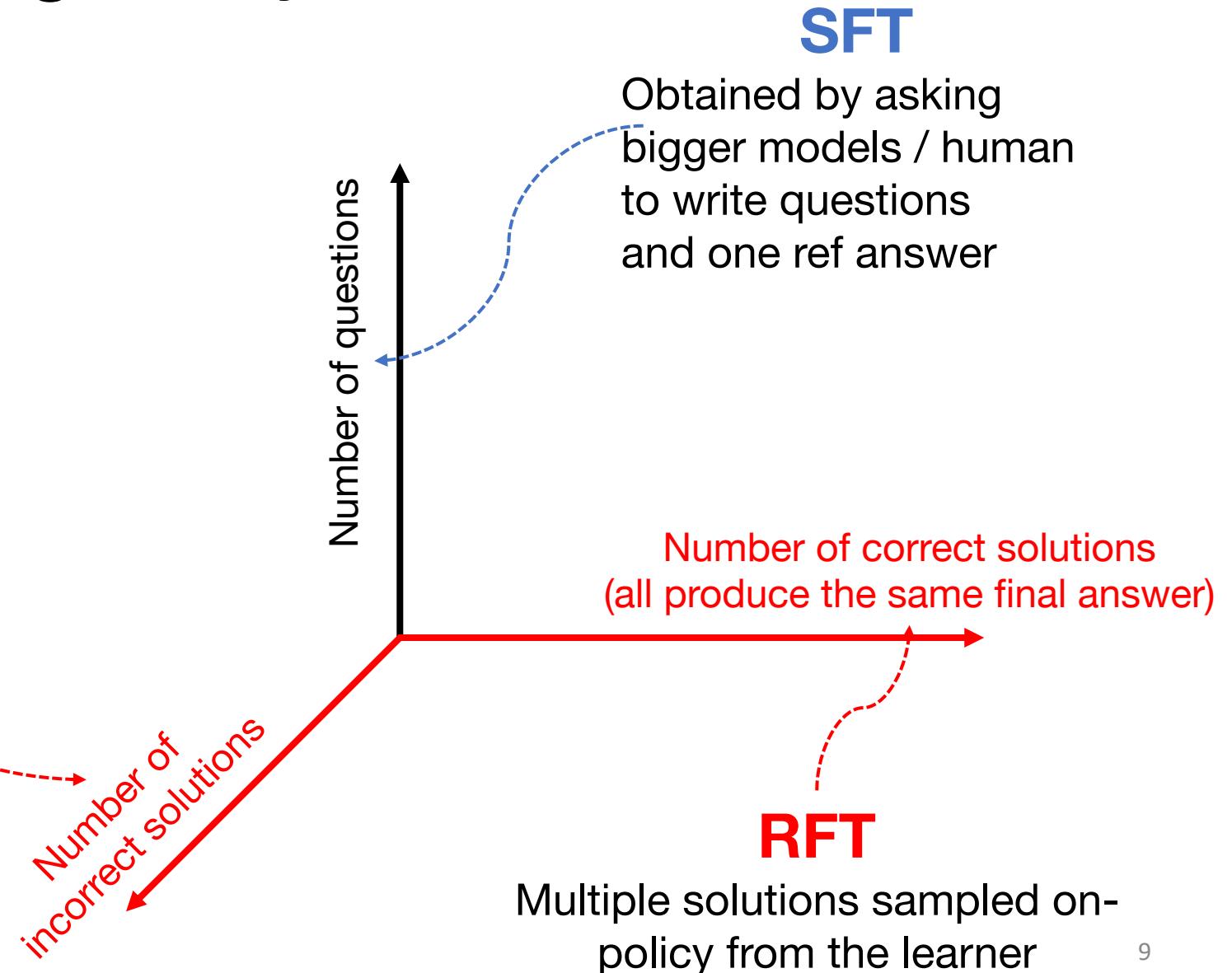
Main papers covered:

- RL on Incorrect Synthetic Data Scales the Efficiency of LLM Math Reasoning by Eight-Fold.
Setlur, Garg, Geng, Garg, Smith, Kumar. NeurIPS 2024
- Rewarding Progress: Scaling up Automated Process Supervision for LLM Reasoning
Setlur, Nagpal, Fisch, Geng, Eisenstein, R. Agarwal, A. Agarwal, Berant, Kumar. ICLR 2025

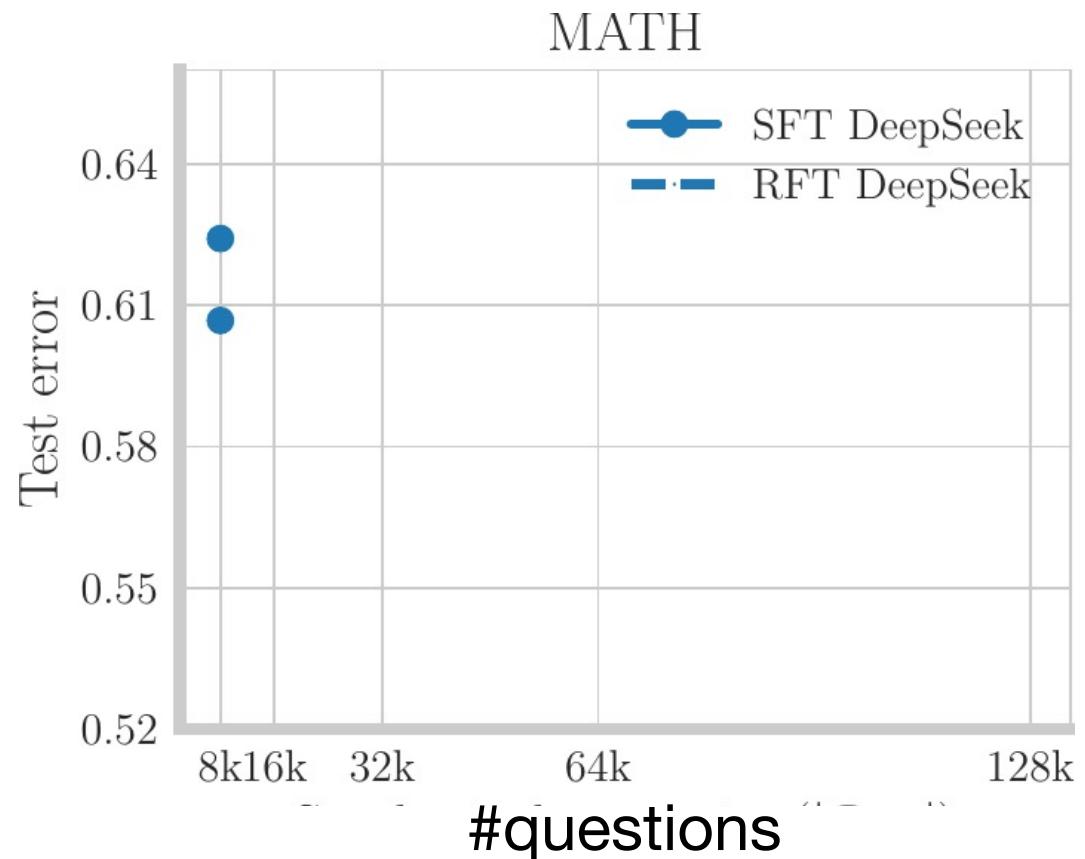
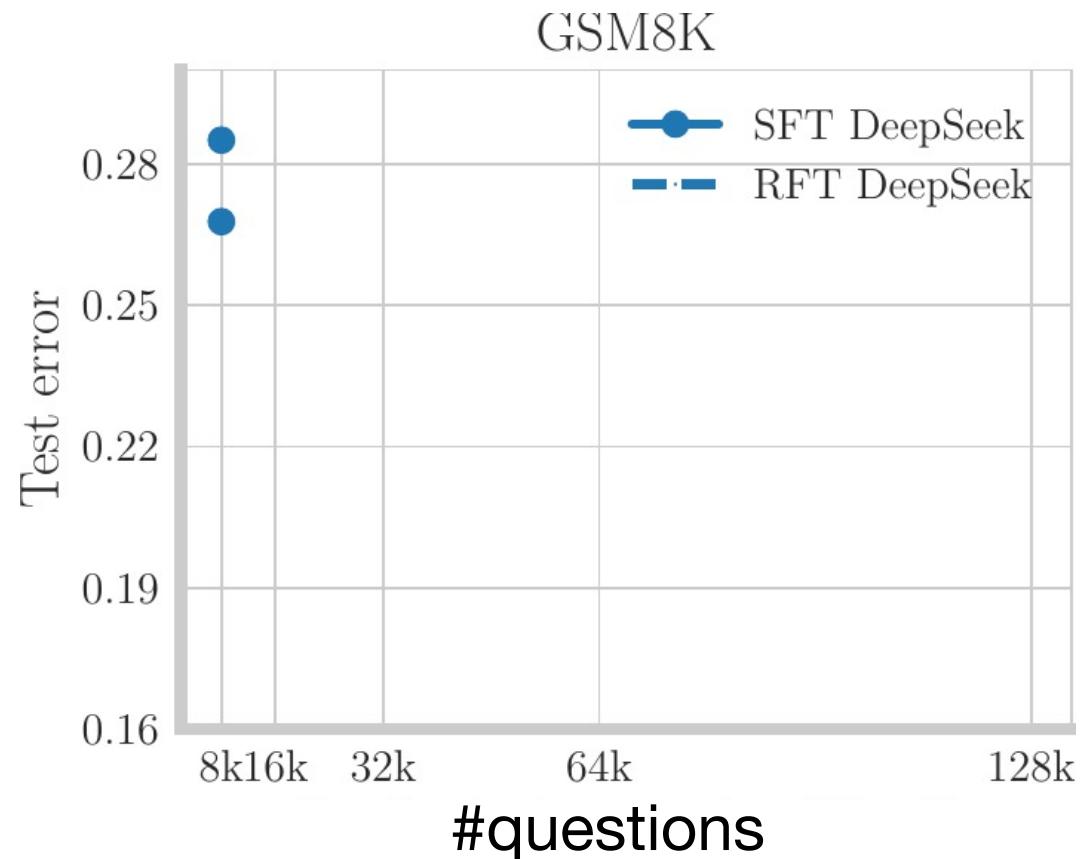
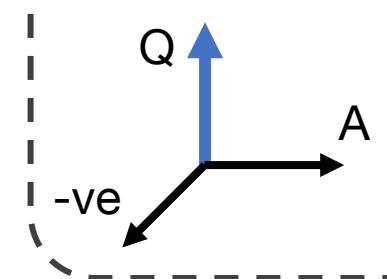
Setup: Data-Scaling Analysis



RL
Also using bad solutions produced by the learner on some questions



Warmup: Scale #Questions, Oracle Ans



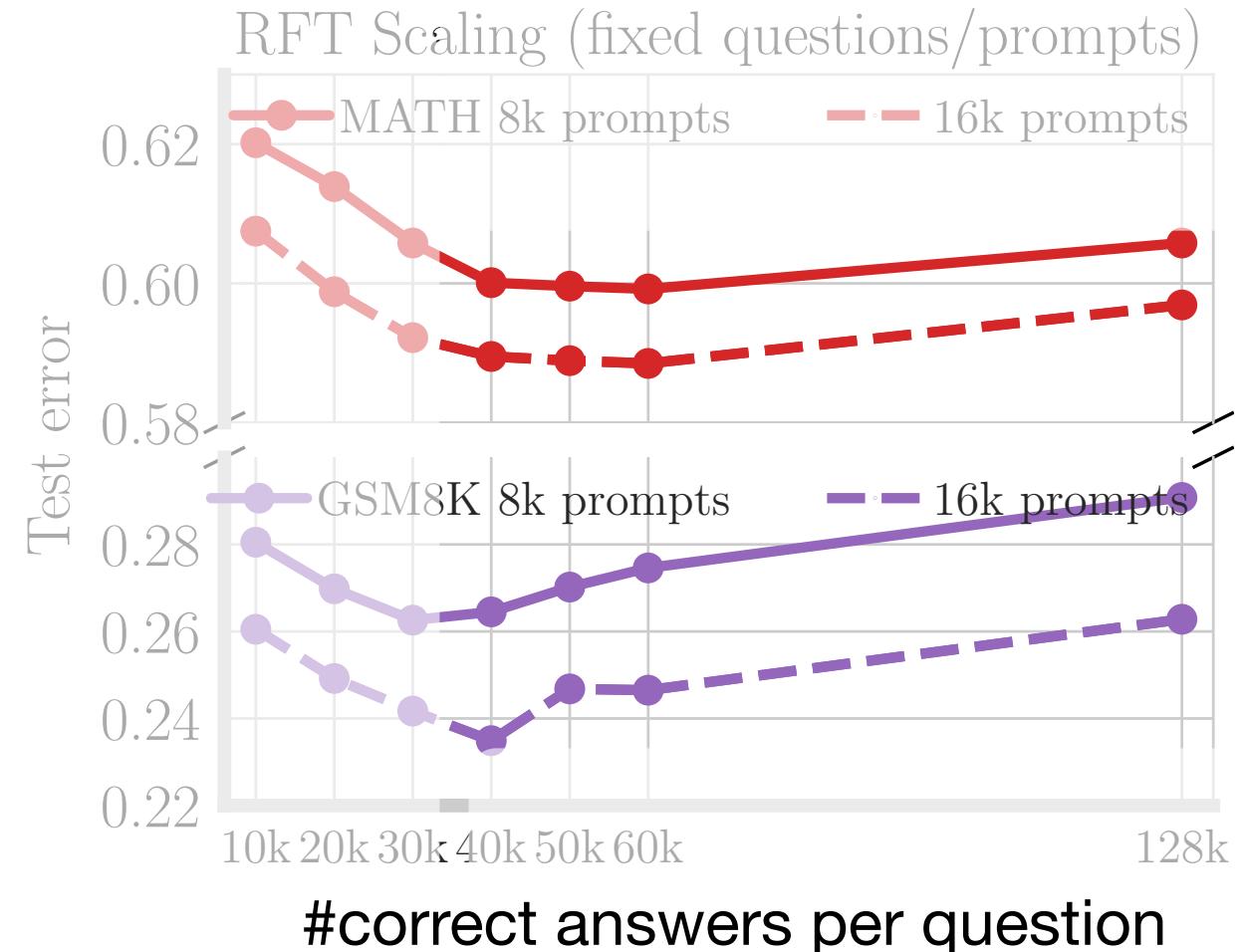
$$|\mathcal{D}|^{-0.15}$$

$$|\mathcal{D}|^{-0.05}$$

But.. We *Can't* Just SFT On-Policy Solutions

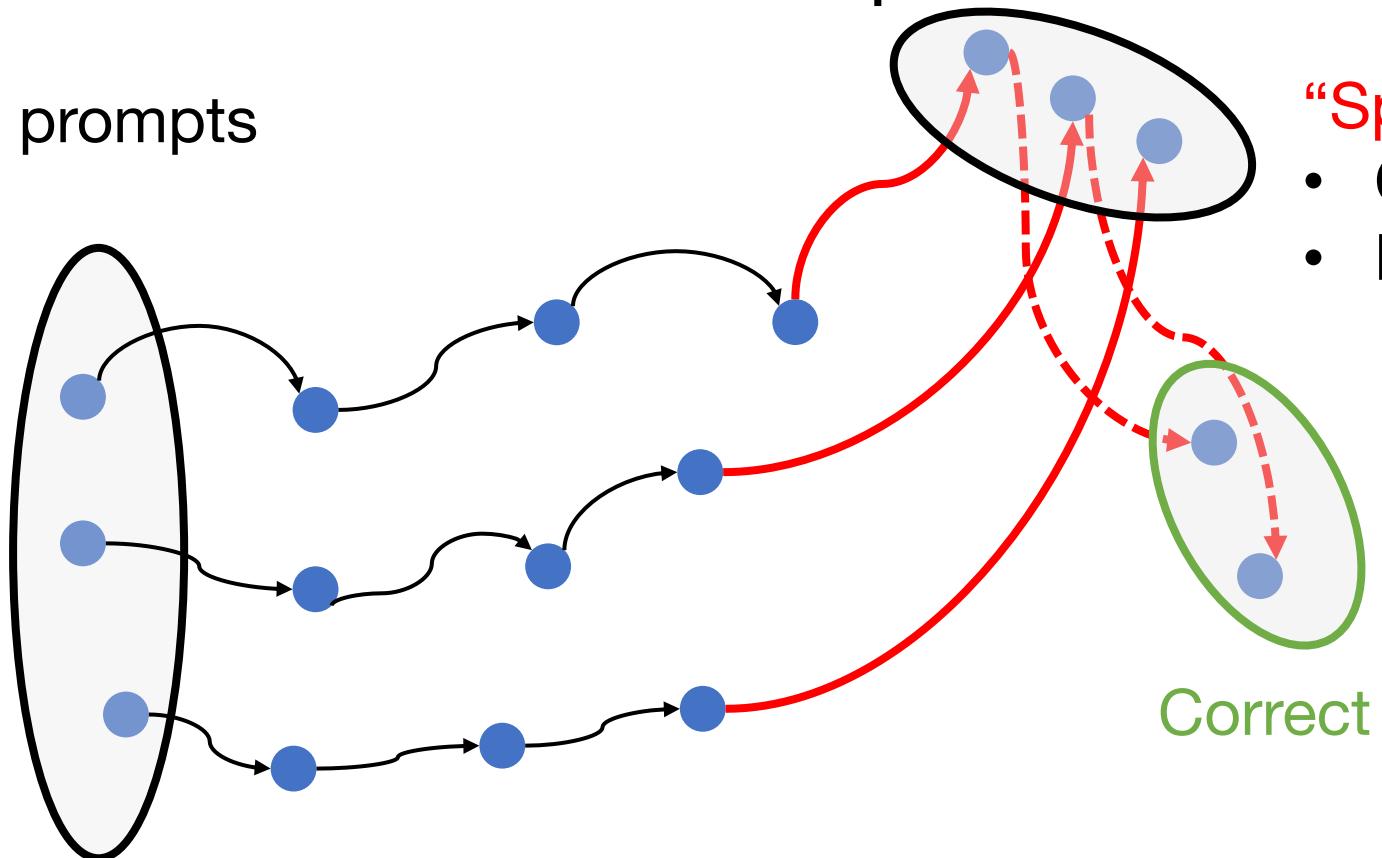
Observation: On-policy imitation eventually degrades if you train too much on it

Fitting self-generated data on limited initial states can hurt generalization on new initial states (prompts).



Mental Picture: Spurious Steps Derail the Model

prompts



“Spurious” steps

- Can somehow learn to recover on train
- But *not* on test, where it will derail

RFT data has spurious steps.

Question: Betty wants a new wallet costing \$100 and has only half the money she needs. Her parents give her \$15, and her grandparents twice as much as her parents. How much more money does she need?

Model generated solution: Betty's parents gave her 15, so her grandparents gave her $2 * 15 = \ll 15*2=30 \gg 30$. In total, Betty received $30 + 15 = \ll 30+15=45 \gg 45$. The total amount of money Betty needs is $100 * 2 = \ll 100*2=200 \gg 200$. Betty needs $100 / 2 = \ll 100/2=50 \gg 50$ in total. Betty still needs to save $50 - 45 = \ll 50-45=5 \gg 5$. Answer is 5.

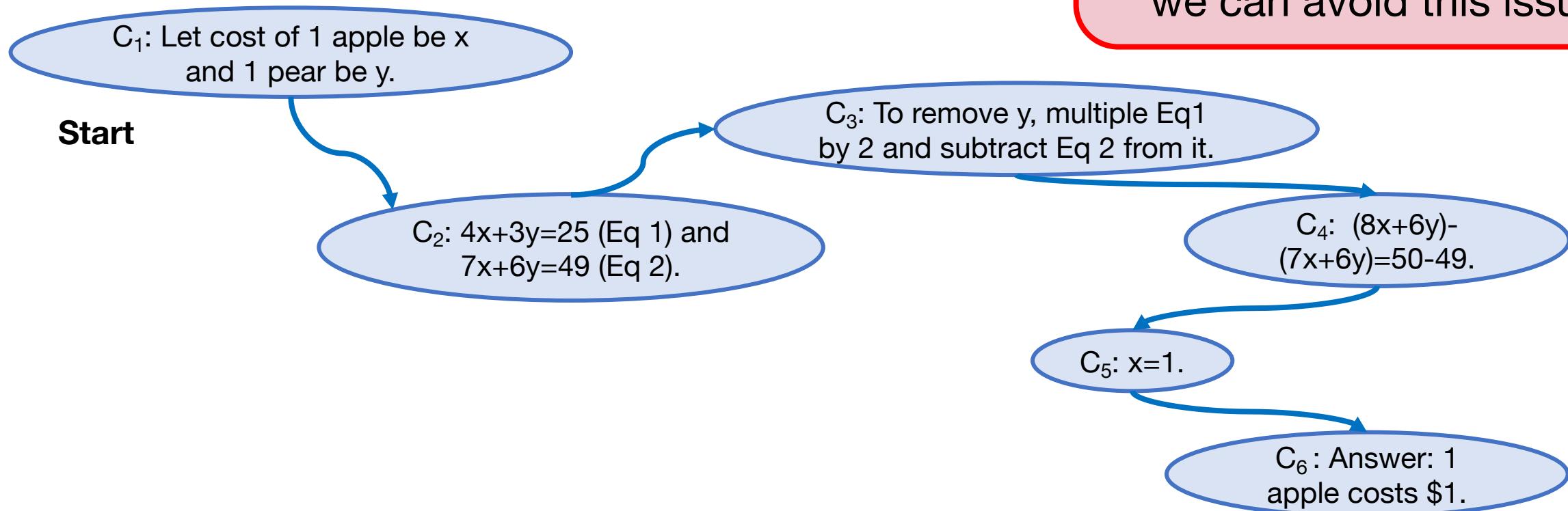
Akin to “causal confusion” in imitation when conditioned on history of past actions

How Can We Address This?

An Idea: Assigning “Credit” to Steps

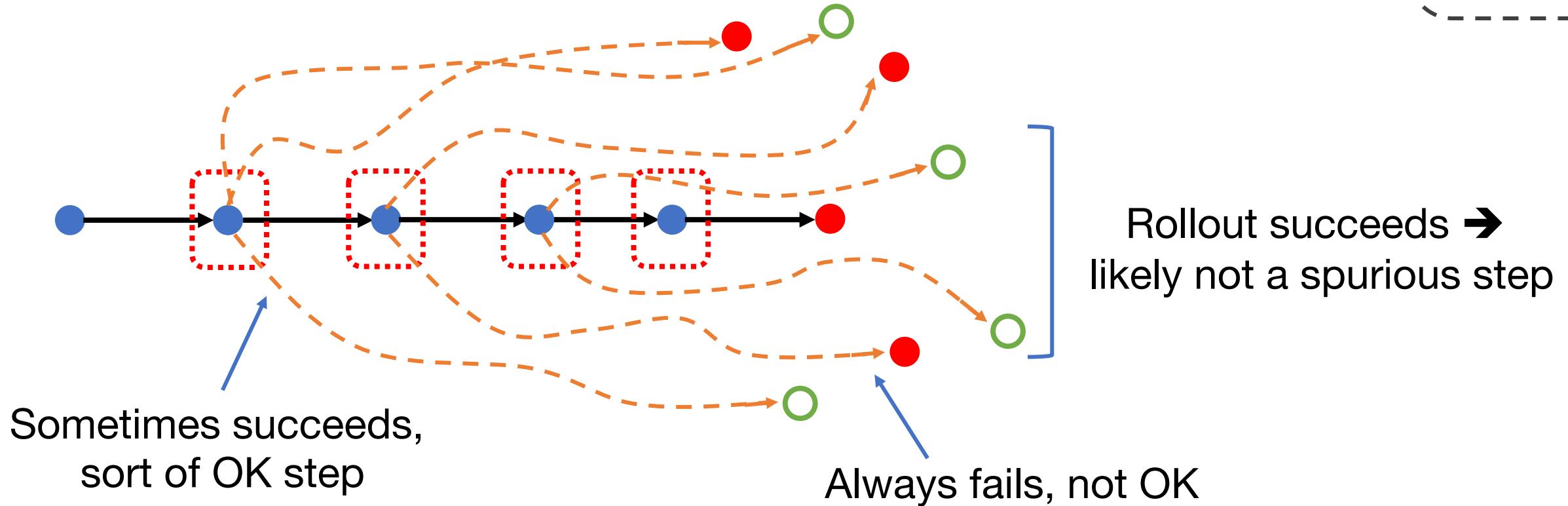
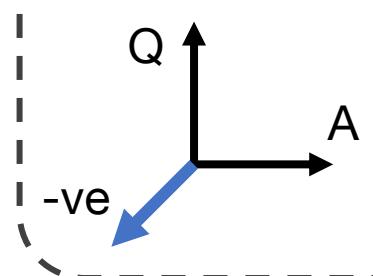
Question: 4 apples and 3 pears cost \$25, but 7 apples and 6 pears cost \$49. What is the cost of 1 apple?

Key insight: If we can identify spurious steps somewhat *precisely*, then we can avoid this issue



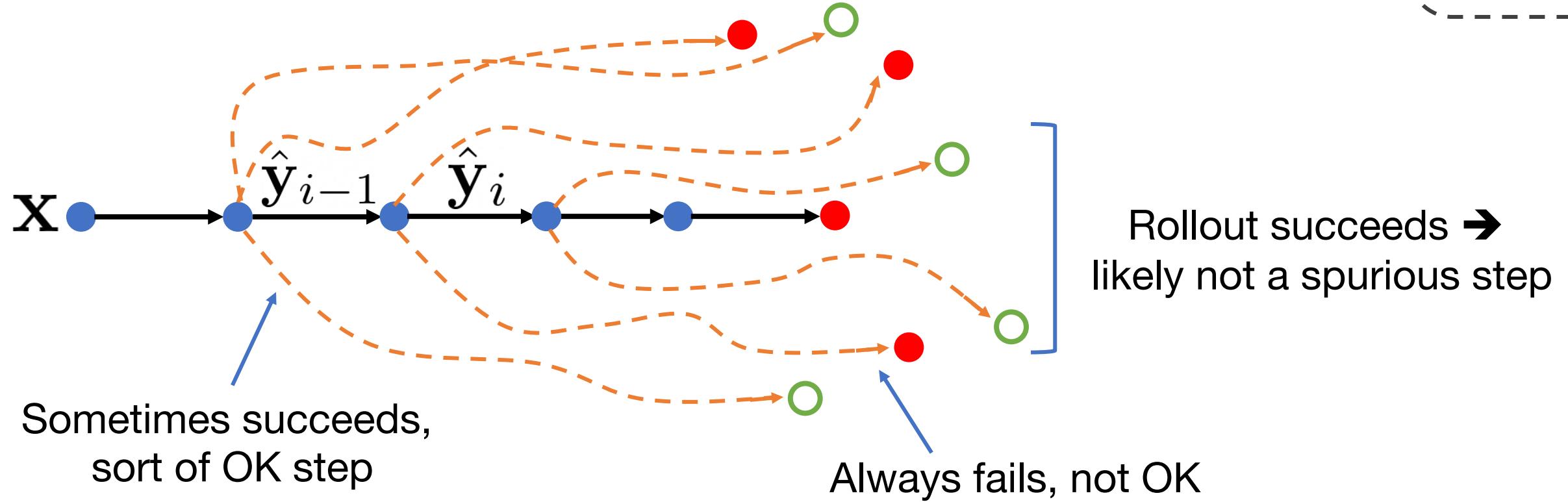
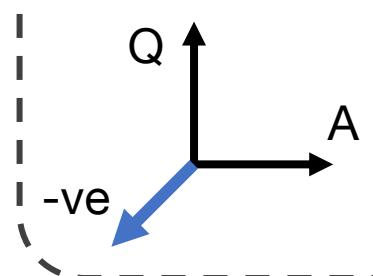
✓ Correct solution
14

Negative On-Policy Data Gives Advantages



Connection: This is equivalent to the value function of the *rollout policy*....

Negative On-Policy Data Gives Advantages

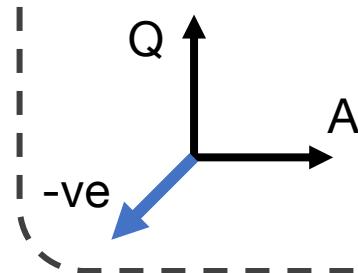


$$Q_{\tilde{\pi}}(\underbrace{x, \hat{y}_{1:i-1}}_{\text{state}}; \underbrace{\hat{y}_i}_{\text{action}}) =$$

$$\mathbb{E}_{\mathbf{y}_{i+1:L}^{\text{new}} \sim \tilde{\pi}(\cdot | x, \hat{y}_{1:i})} \left[r([\hat{y}_{1:i}, \mathbf{y}_{i+1:L}^{\text{new}}], \mathbf{y}) \right]$$

expected future reward under new actions (i.e., steps) sampled by policy $\tilde{\pi}$

Negative On-Policy Data Gives Advantages

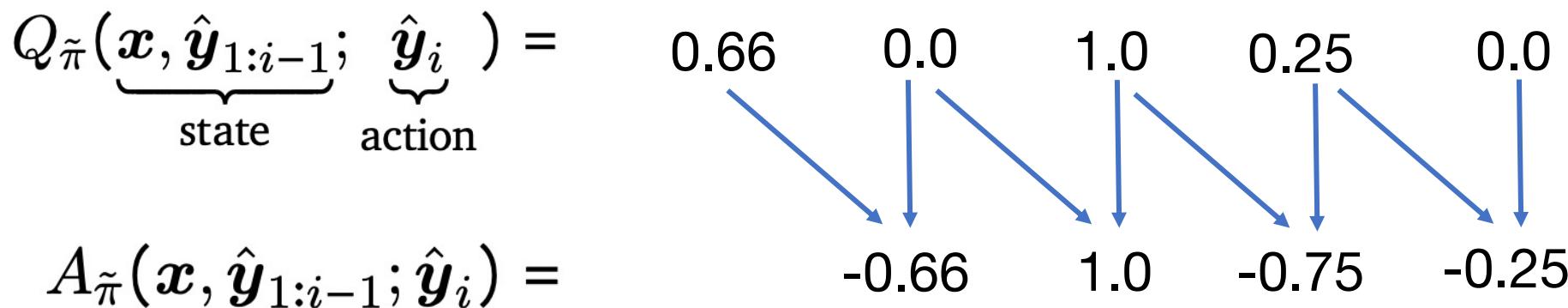


Q-value after step i Q-value after step $i-1$

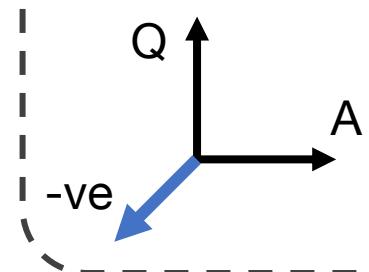
$$A_{\tilde{\pi}}(x, \hat{y}_{1:i-1}; \hat{y}_i) = Q_{\tilde{\pi}}(x, \hat{y}_{1:i-1}; \hat{y}_i) - Q_{\tilde{\pi}}(x, \hat{y}_{1:i-2}; \hat{y}_{i-1}).$$



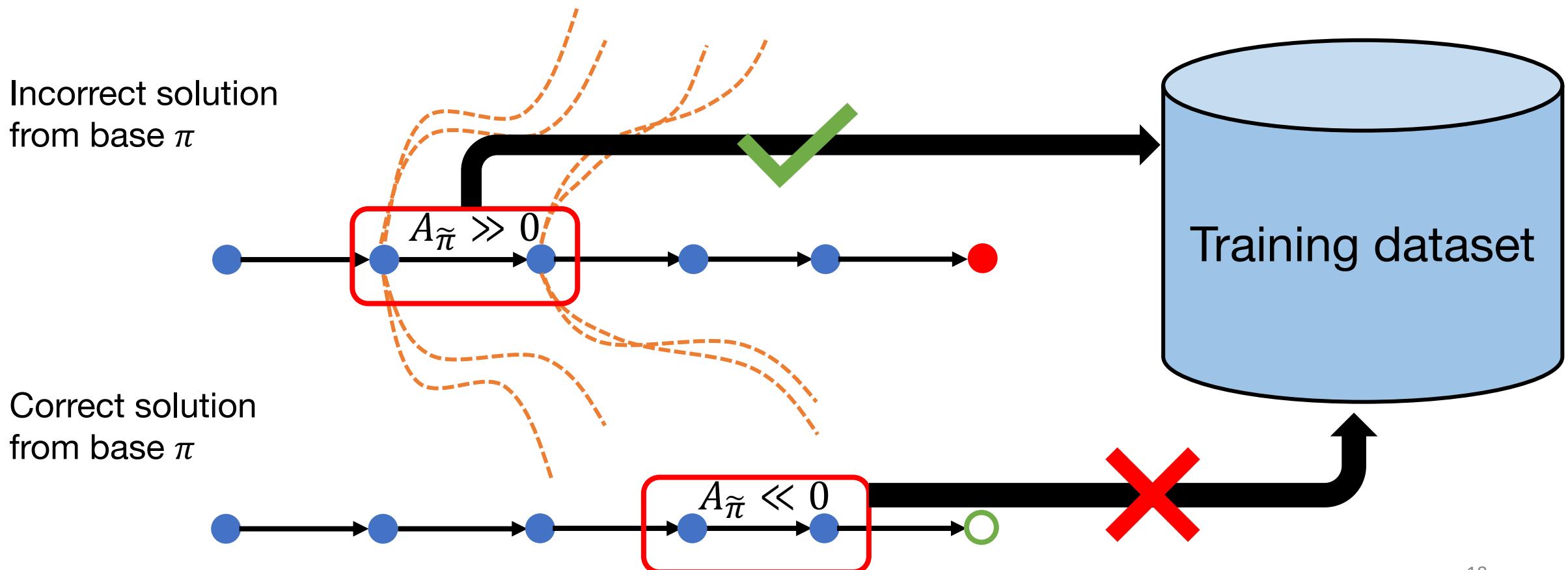
Advantage: relative change in the value function having committed to a step



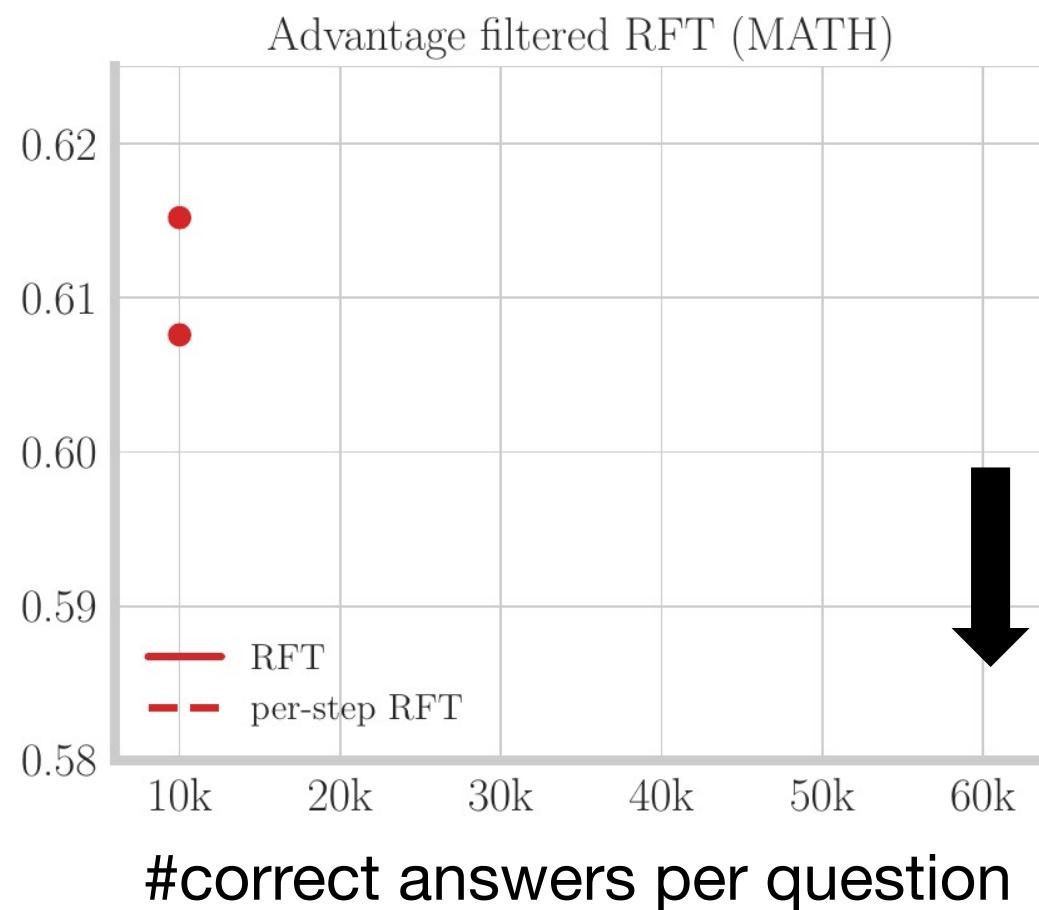
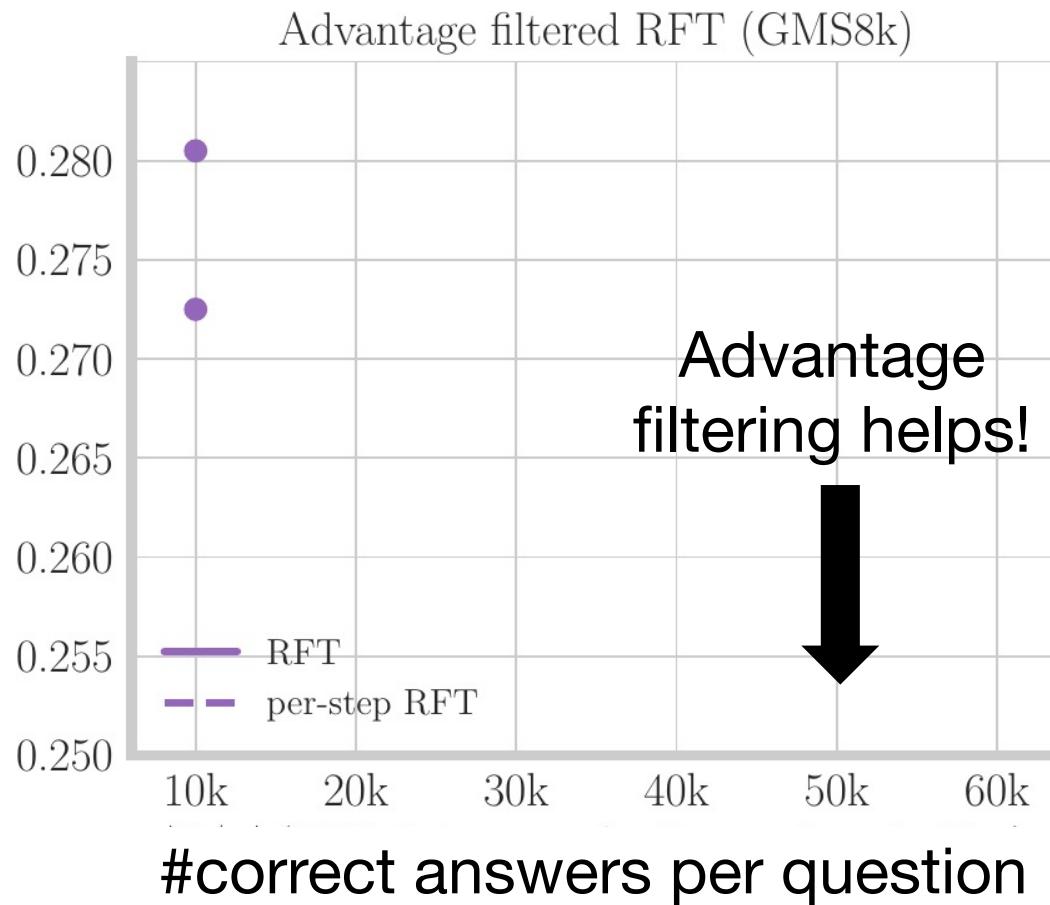
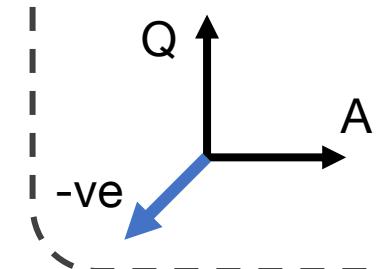
Using Advantages for Training



Option 1: Filter steps by advantages directly

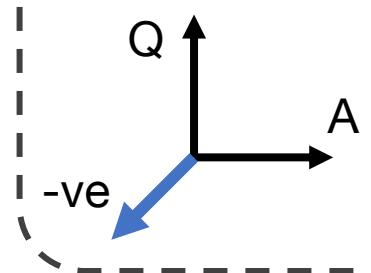


Results: Filtering Steps w/ Advantages



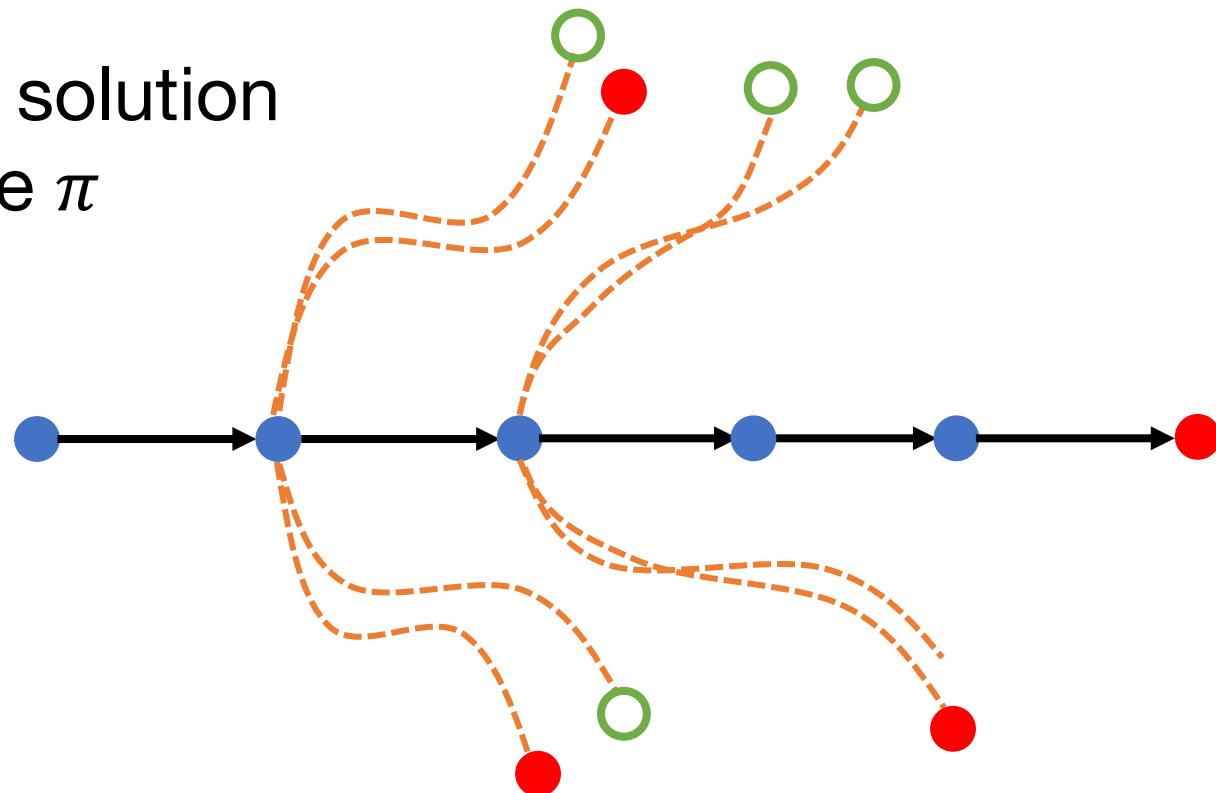
Finding: Advantage-filtered RFT does not fall prey to spurious steps

Using Advantages for Offline RL (DPO)

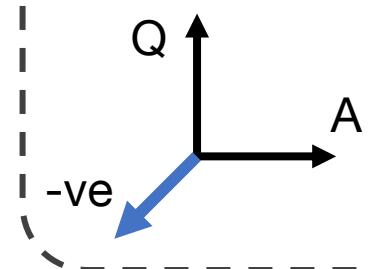


Option 2: Retain partial rollouts from $\tilde{\pi}$ for training

Incorrect solution
from base π

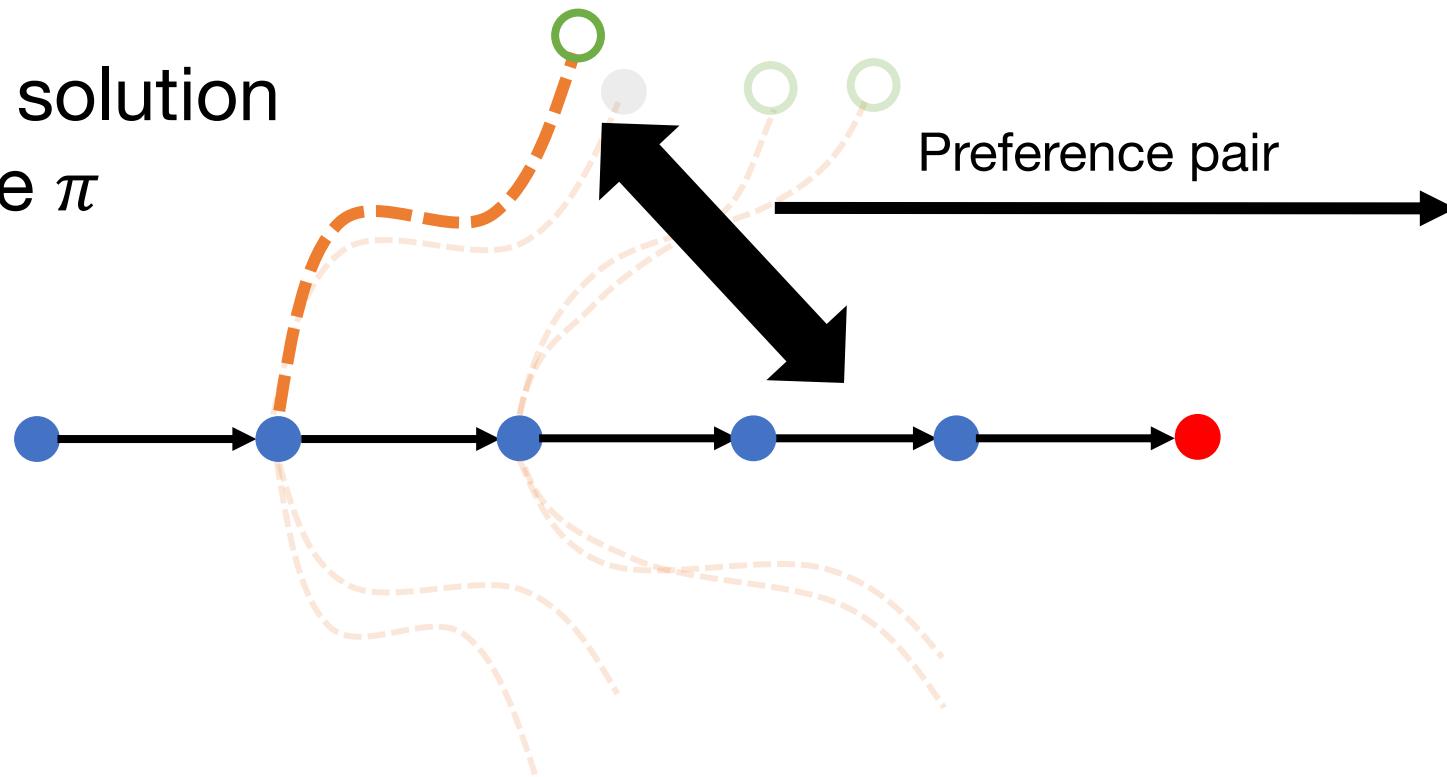


Using Advantages for Offline RL (DPO)

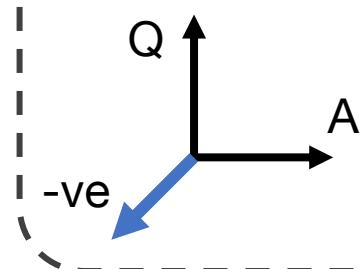


Option 2: Retain partial rollouts from $\tilde{\pi}$ for training

Incorrect solution
from base π

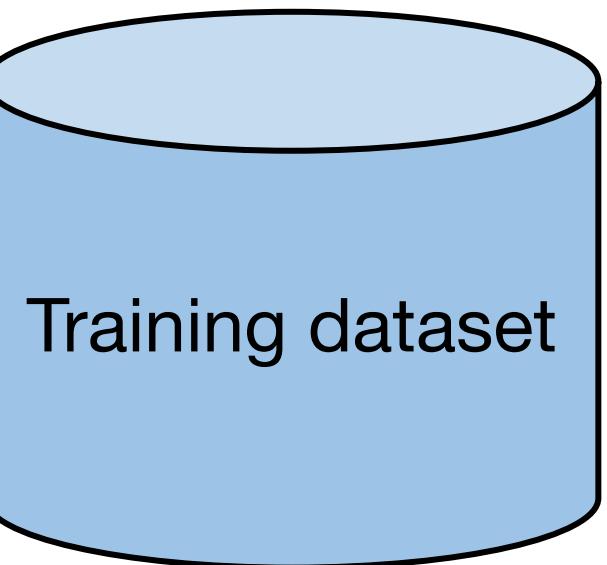
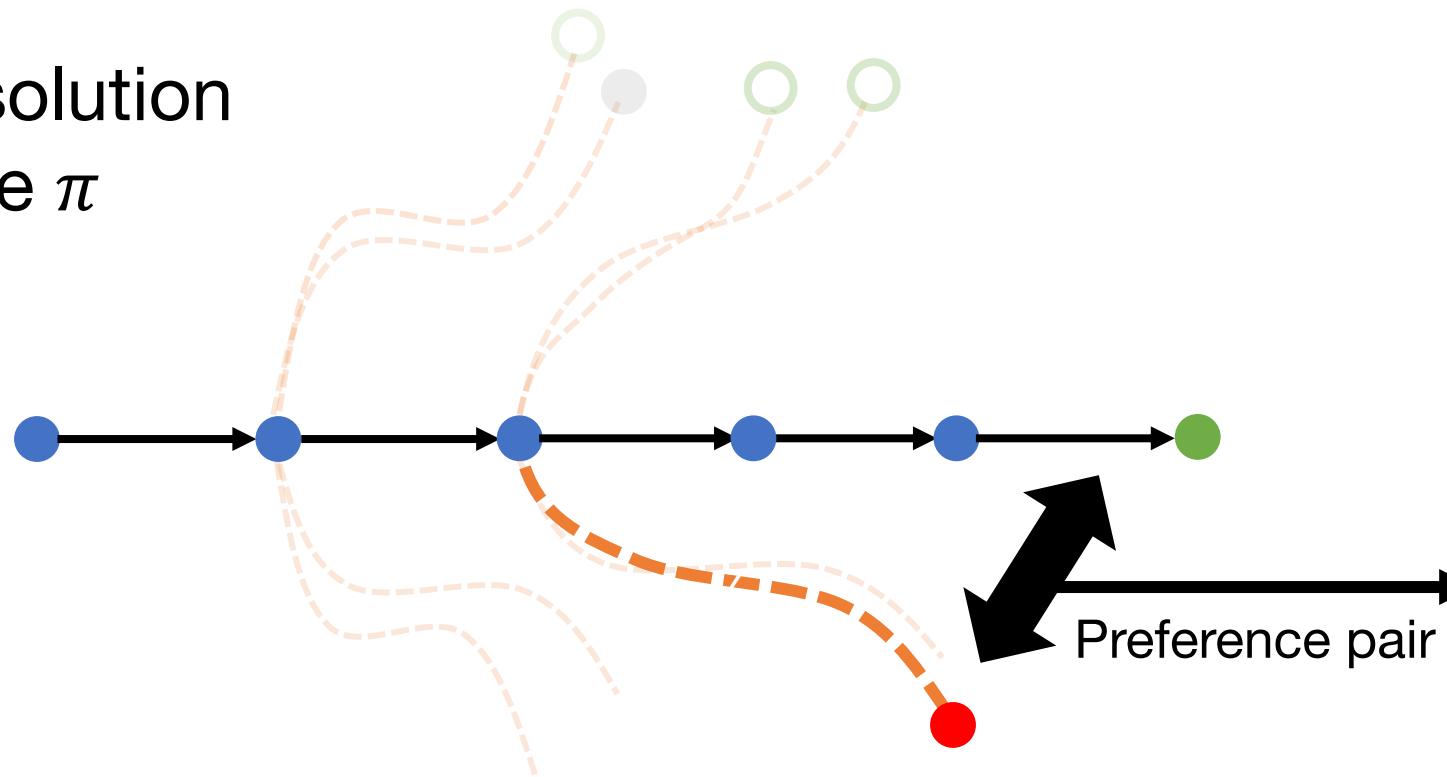


Using Advantages for Offline RL (DPO)

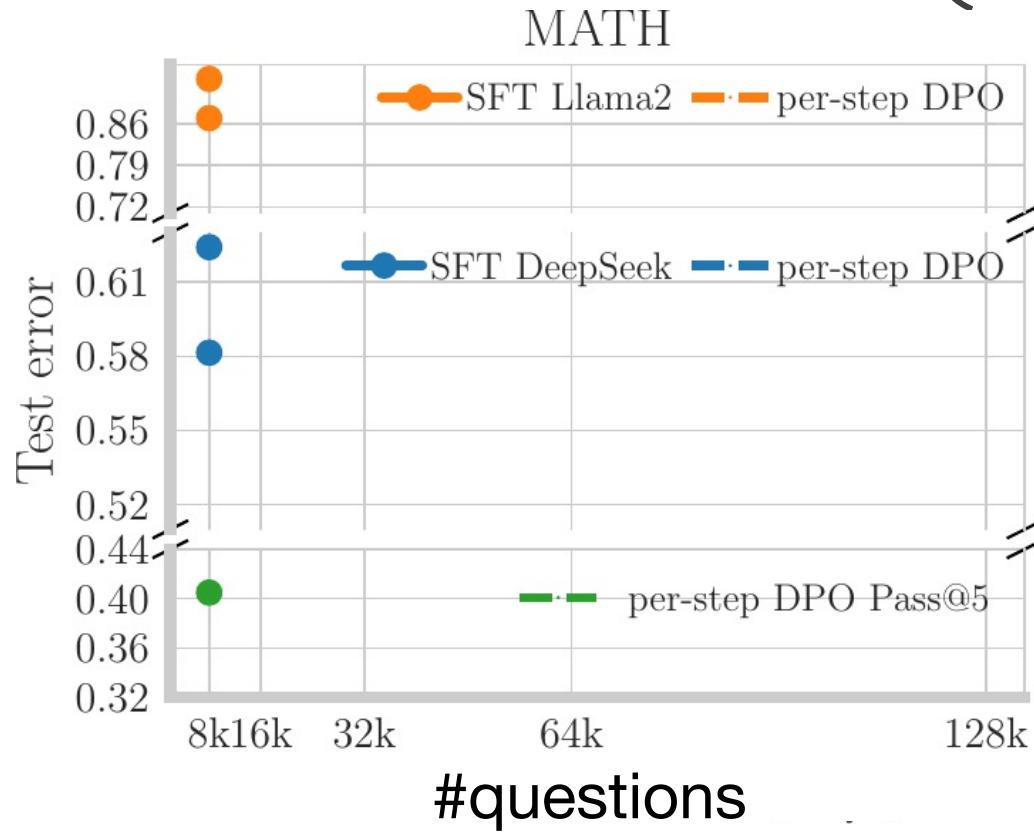
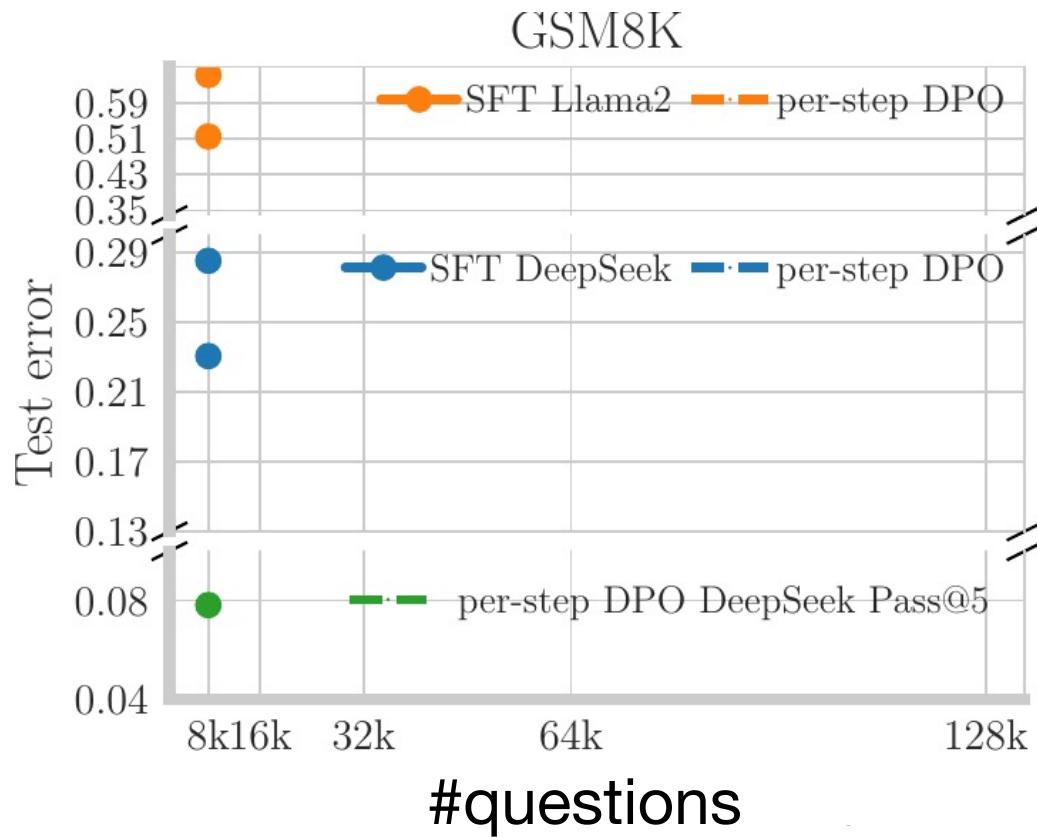
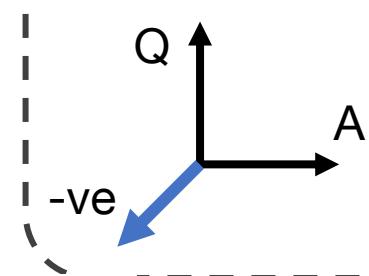


Option 2: Retain partial rollouts from $\tilde{\pi}$ for training

Correct solution
from base π



Results: Scaling for Offline RL



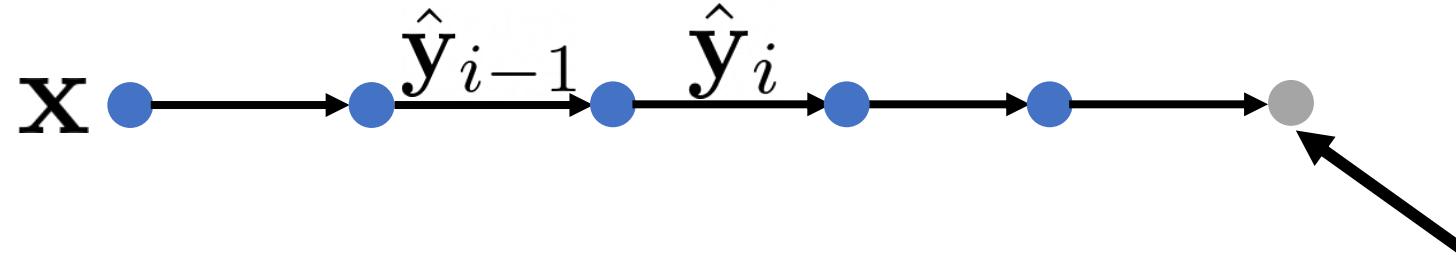
Finding: Using offline RL (via per-step RL) gives you 8x data efficiency in performance over imitation only.

Online RL: Basic Recipe with 0/1 rewards

A solution from
current π

Key idea: Train with a binary 0/1 reward using policy gradients

$$\mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{train}}, \mathbf{y} \sim \pi(\cdot | \mathbf{x})} \left[\sum_j \nabla_{\pi} \log \pi(\mathbf{y}_j | \mathbf{x}, \mathbf{y}_{0:j-1}) \cdot r(\mathbf{x}, \mathbf{y}) \right]$$



Can also use other policy gradient methods (PPO, GRPO, etc)!

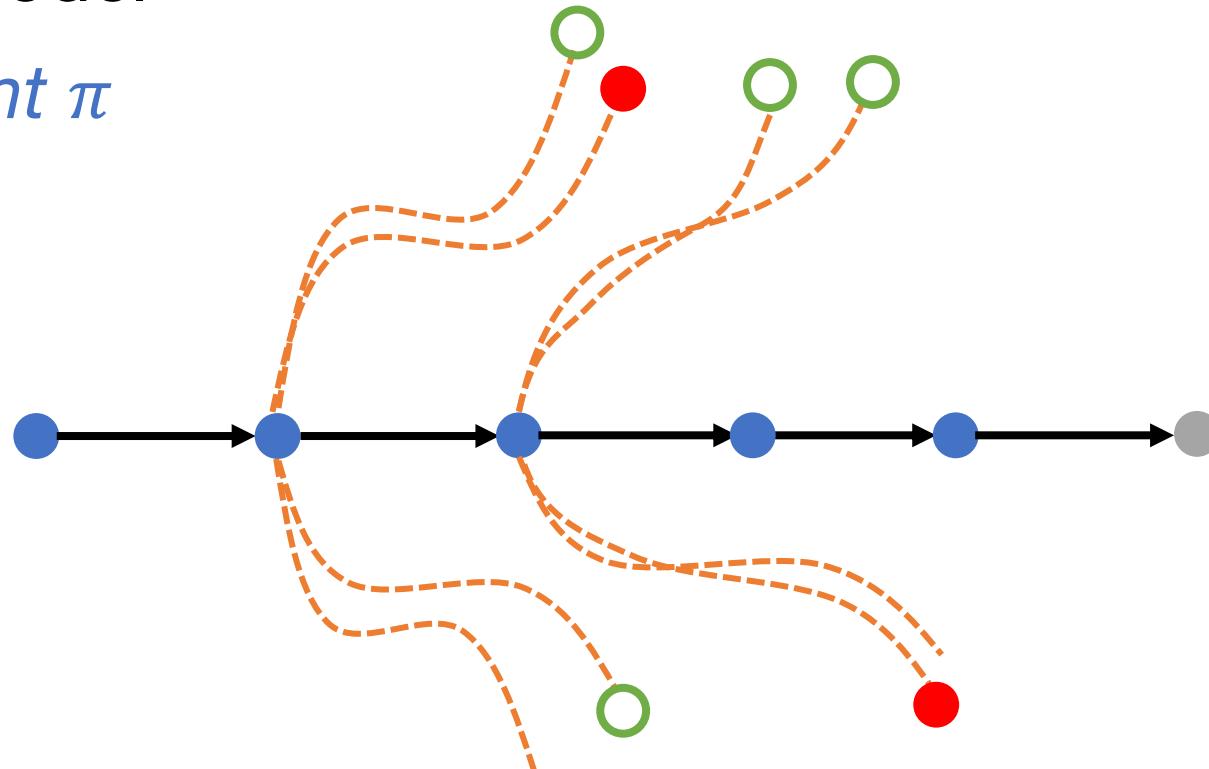
$$r(x, y) = \{0, 1\}$$

Per-Step Advantages in Online RL

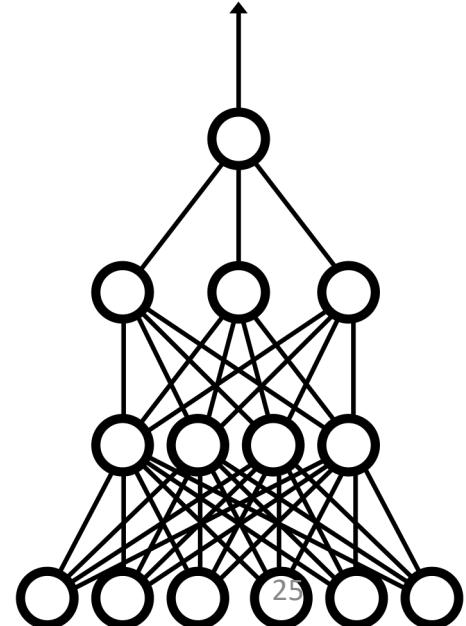
A solution from
~~base model~~

Question: Should we do rollouts from $\tilde{\pi}$ on the fly?

current π



Fit a parametric model to advantage predictions



Approach: Process Advantage Verifiers (PAVs)

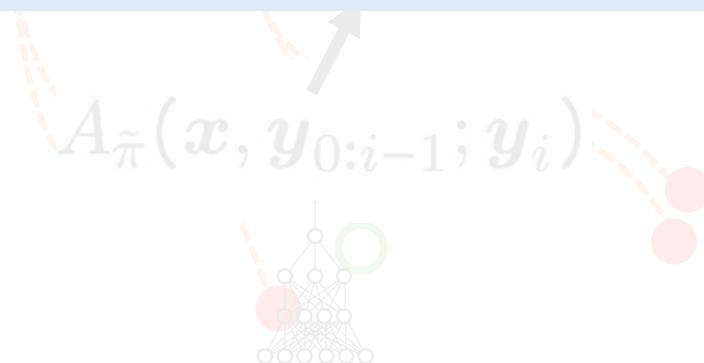
A solution from
current π

Key idea: Advantage as dense reward bonus in RL

[Informal] Optimal Rollout Policy, $\tilde{\pi}$

The optimal rollout policy $\tilde{\pi}$ is one which produces advantages that **most effectively distinguish** good and bad steps from *all* π .

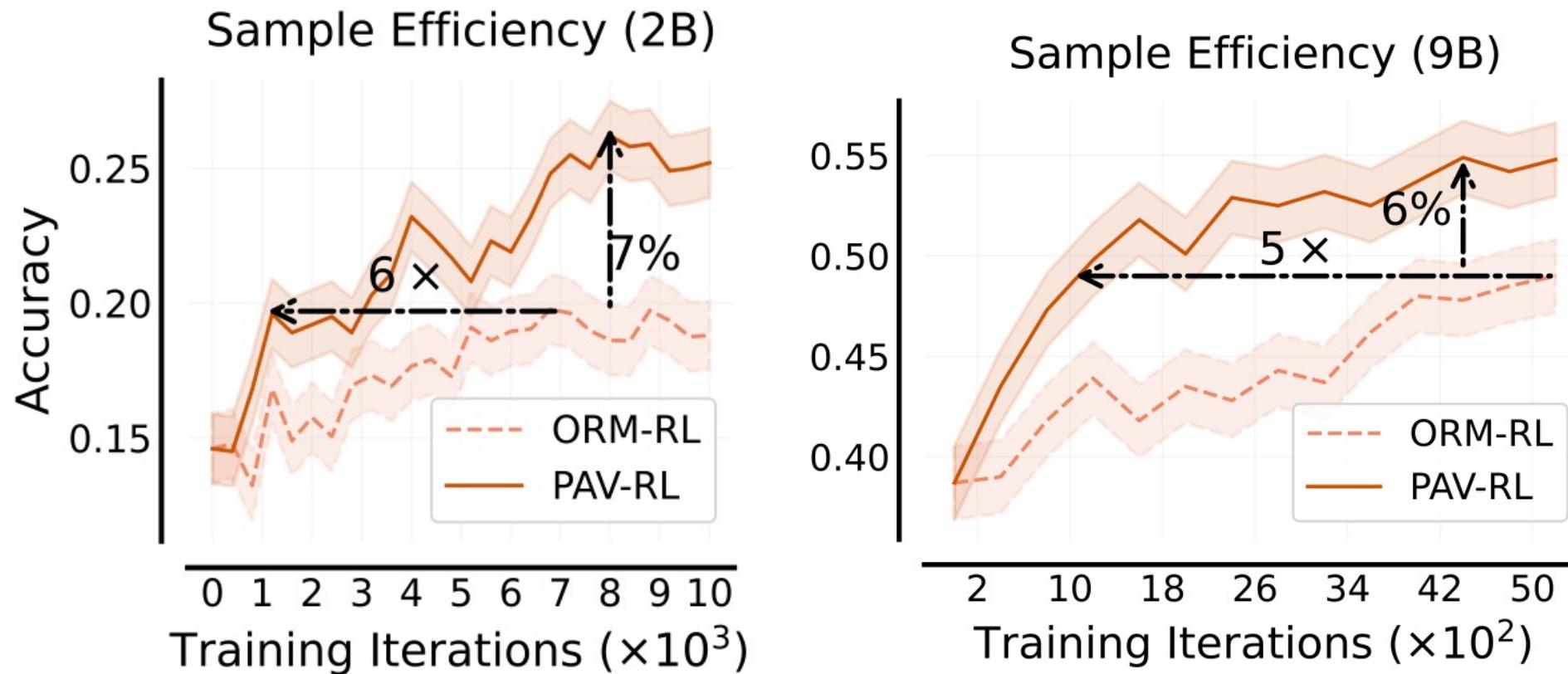
x



$$r(x, y) = \{0, 1\}$$

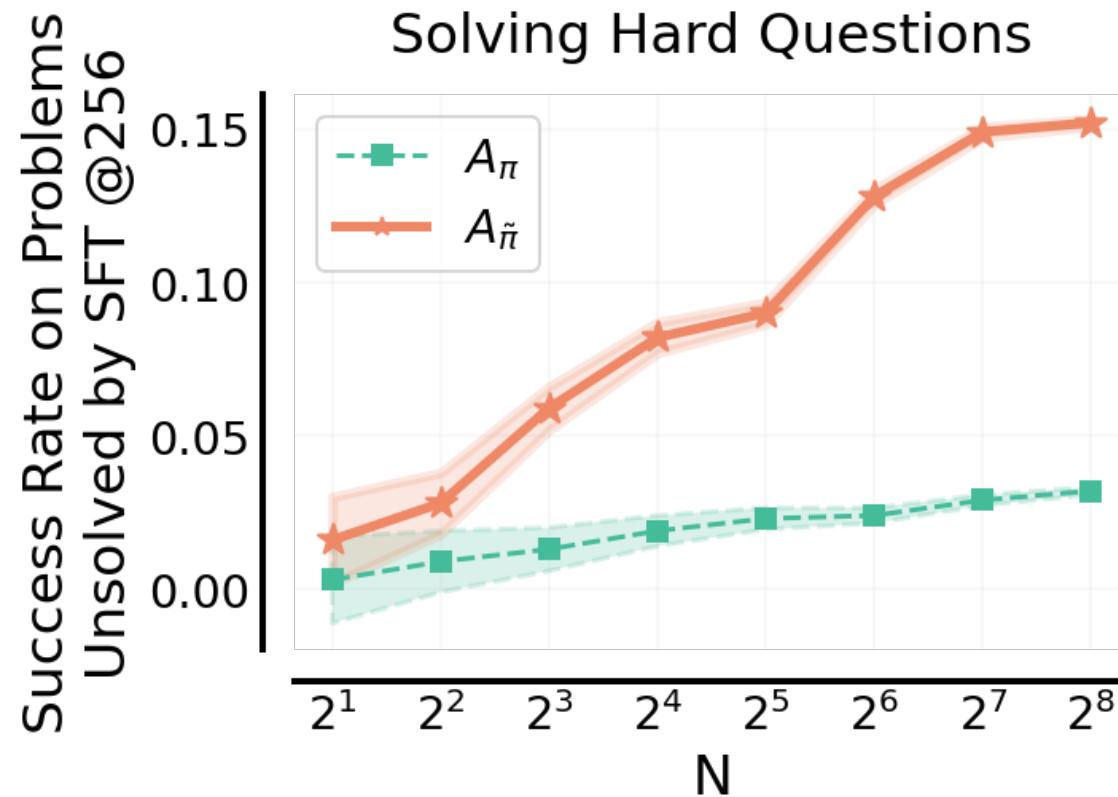
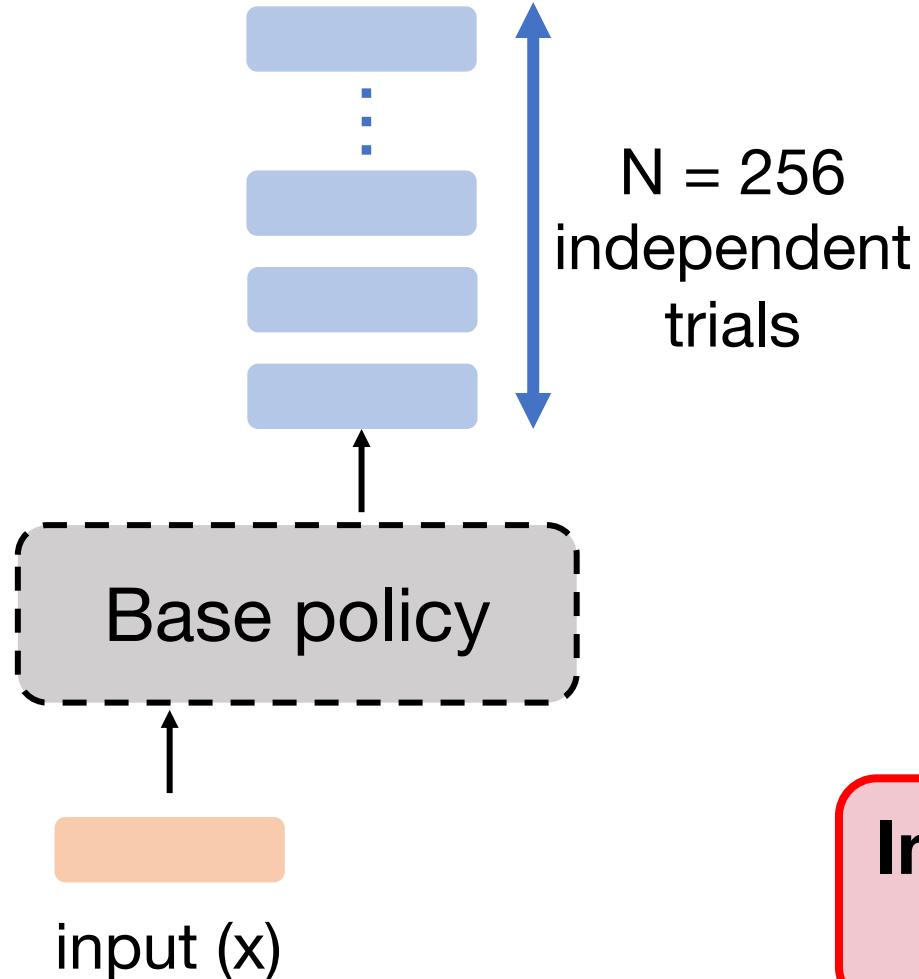
Challenge: These issues once some $\tilde{\pi}$ are fixed in RL

Results: Dense-Reward RL Performance



5-6x sample efficiency than outcome rewards only, 6-7% improvement

Results: Discovers Solutions to Hard Questions



Insight: Specifically, PAVs give exploration crucial for solving hard questions

Part 1

Takeaways

- **Challenge:** Imitating positive data scales (slowly), but it induces spurious steps, that are memorized.
- **Algorithm:** Careful, model-specific negative data can help with addressing this issue via offline / online RL
- **Result:** Offline RL gives **8x** sample efficiency vs imitation.
Online RL w/ dense advantage reward improves **5-6x**



Part 2

Training “Thinking” Models via RL

Main papers covered:

- DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
- Kimi K1.5: Scaling Reinforcement Learning with LLMs.
- **[Optional]** Optimizing Test-Time Compute requires Solving a Meta-RL Problem
Setlur, Qu, Yang, Zhang, Smith, Kumar. CMU MLD Blog, 2025

Example: A Trace From Thinking Models

Problem:

James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?

Thinking Process

<think>

Okay, let me try to figure out this problem. So, James is writing a 3-page letter to two different friends twice a week. We need to find out how many pages he writes in a year. Hmm, let's break this down step by step. First, let me understand the details. [...]

Wait, if he writes to 2 friends twice a week, does that mean he writes 2 letters each time, and he does this twice? [...] So per week, he writes 2 friends * 3 pages * 2 times. Let me check that. [...]

Alternatively, maybe it's 3-page letter to 2 different friends each time he writes, and he does that twice a week. So each time he writes, he's writing two letters (each 3 pages), so that's $2 * 3 = 6$ pages per writing session. Then, twice a week would be $6 * 2 = 12$ pages per week. Then, multiply that by the number of weeks in a year. But how many weeks are in a year? Typically, 52 weeks. So $12 \text{ pages/week} * 52 \text{ weeks} = 624 \text{ pages}$. Is that right?

Wait, let me confirm. So, per week: [...] That makes sense.

But maybe I need to double-check if the problem is interpreted differently. [...] That seems consistent.

Wait, but wait a second. Is there any chance that the problem might be considering a year as 52 weeks, but some people might account for leap years or something else? But generally, in problems like this, unless specified, we take 52 weeks per year. So that's standard.

So, conclusion: [...] So the answer is 624. So I think that's correct.

</think>

James writes 3-page letters to 2 friends twice a week. Here's the breakdown:

[...]

Answer: James writes 624 pages in a year.

Answer



What Changed?

(Disclaimer: this is my version!)

- **The RL training objectives remained the same:**
 - DeepSeek-R1 uses GRPO, another policy gradient method
 - Kimi-1.5 uses some version of APA (advantage-induced policy alignment)
 - Some minor changes in reward function
- **Crudely, the action space changed:**
 - These base models became better at performing “macro” actions, such as revisiting a given answer, verification, backtracking, planning, etc
 - Crudely, we can now think of operating in this new action space that presents several “meta” actions
- We **increased the token budget drastically** to chain multiple macro actions

Formulation: Training Thinking Models via RL

Let's start from the final goal

$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{test}}} \left[\mathbb{E}_{\mathbf{z} \sim \pi(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{z})] \right]$$

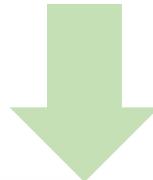
on test problems

response sampled
from model (longer
than typical solution)

Total compute
constraint per problem

Formulation: Training Thinking Models via RL

$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{test}}} \left[\mathbb{E}_{\mathbf{z} \sim \pi(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{z})] \right] \text{ s.t. } \forall \mathbf{x}, \mathbb{E}_{\pi(\cdot | \mathbf{x})} |\mathbf{z}| \leq C_0$$



$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{train}}} \left[\mathbb{E}_{\mathbf{z} \sim \pi(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{z})] \right] \text{ s.t. ...}$$

This compute budget is fixed!

Can optimize this via:

- RL (like DeepSeek-R1): outcome-reward RL
- SFT / RFT: collect data, filter by correctness, maximize likelihood

“Action” Space: Incorporating Meta Strategies

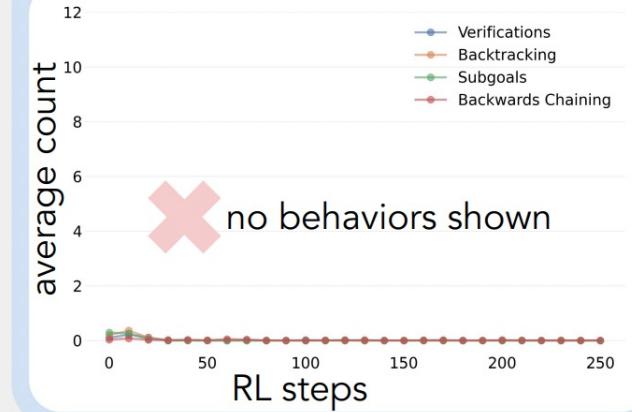
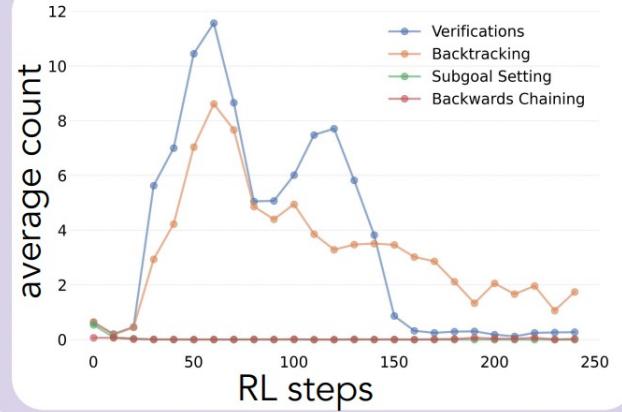
A contrast in behaviors explored by the two models

Verifications
“Let me check my answer ...”

Subgoal Setting
“Let’s try to get to a multiple of 10”

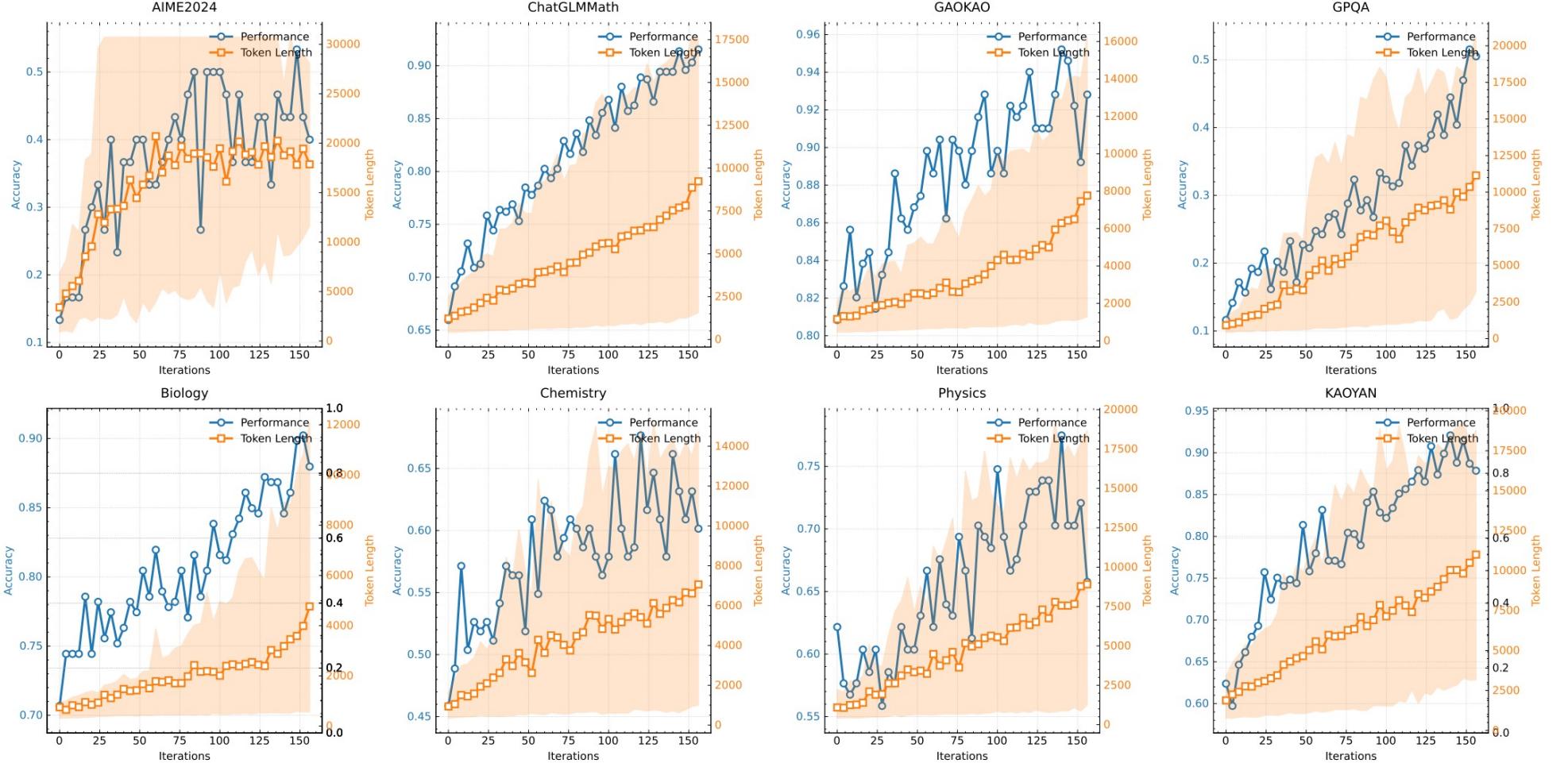
Backtracking
“Let’s try a different approach, what if we ...”

Backward Chaining
“Working backwards, 24 is 8 times 3”



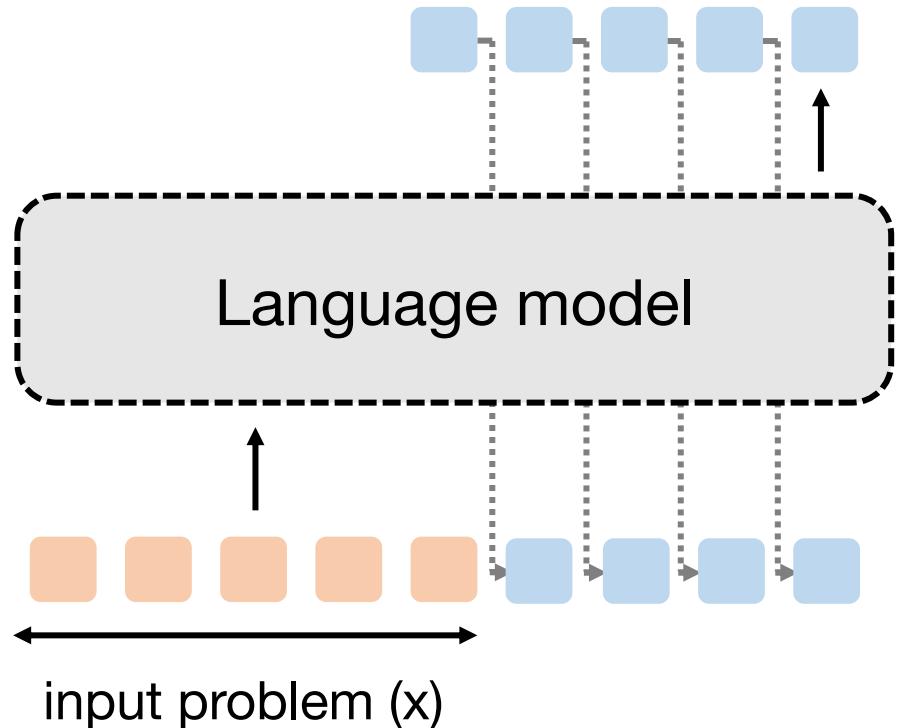
This model does not improve during RL

Longer Length: RL Training Amplifies Length

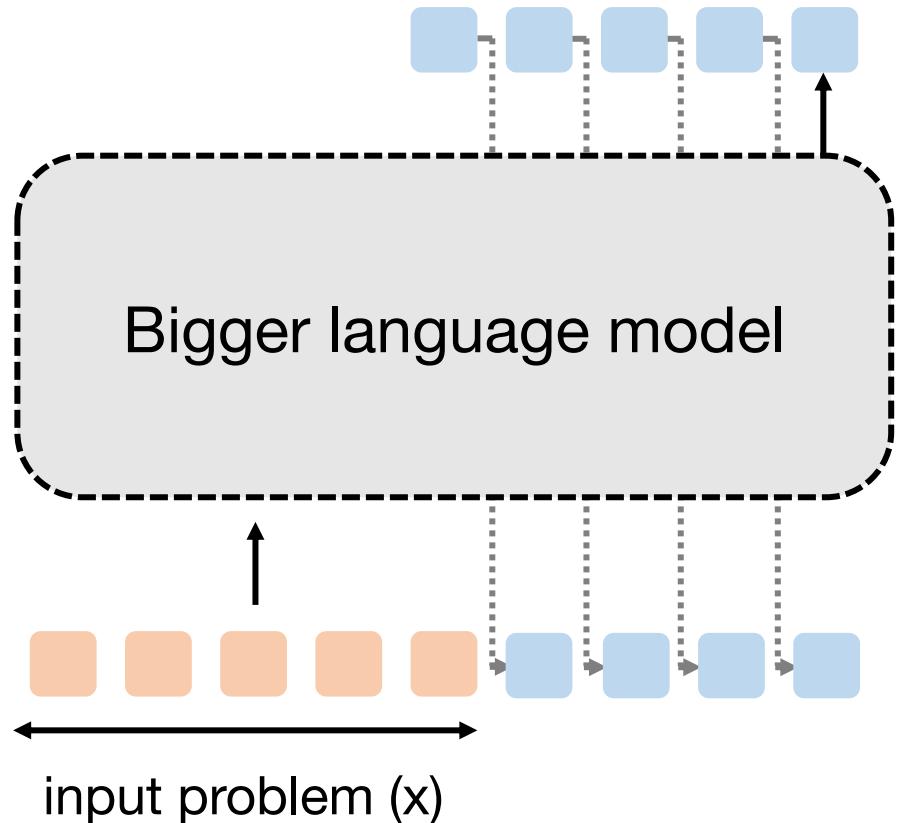


Why?

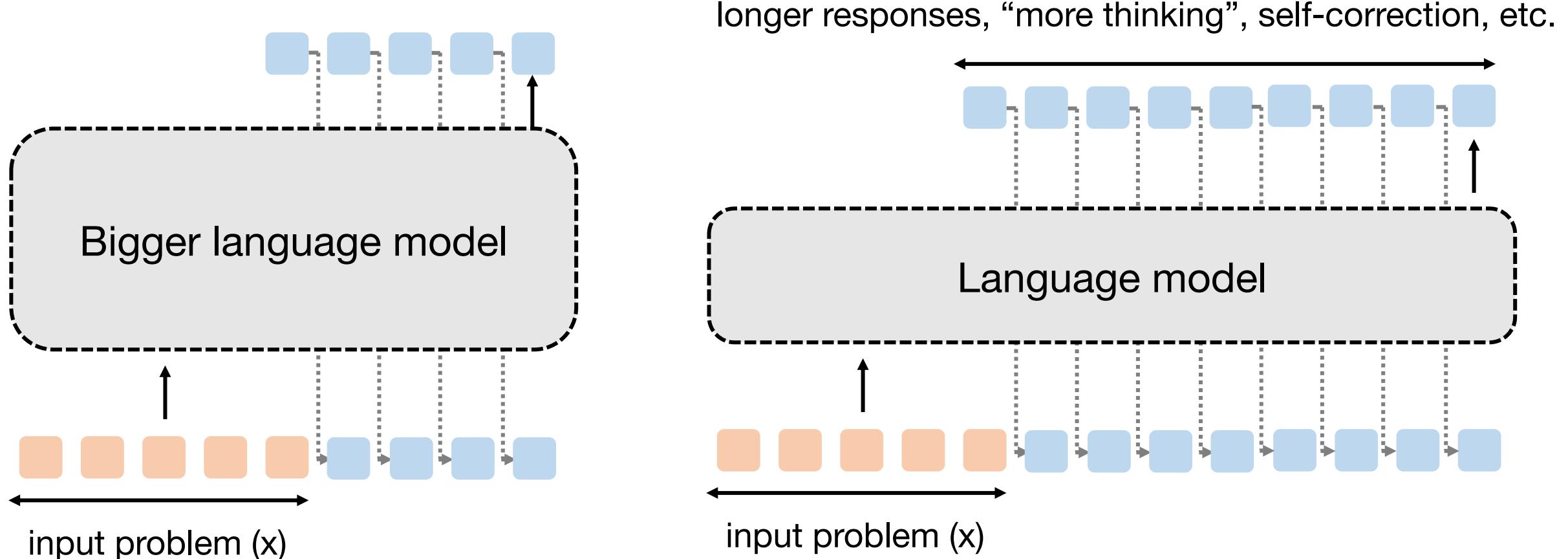
A New Paradigm: Test-Time Scaling



A New Paradigm: Test-Time Scaling

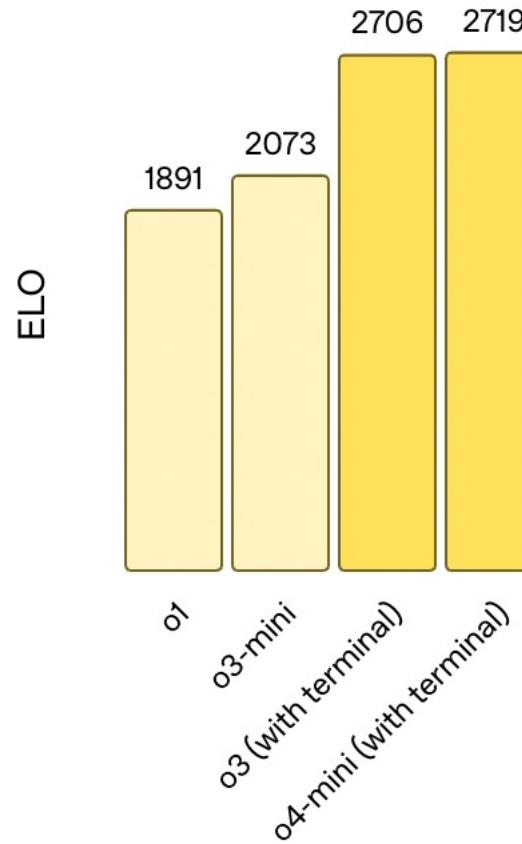


A New Paradigm: Test-Time Scaling

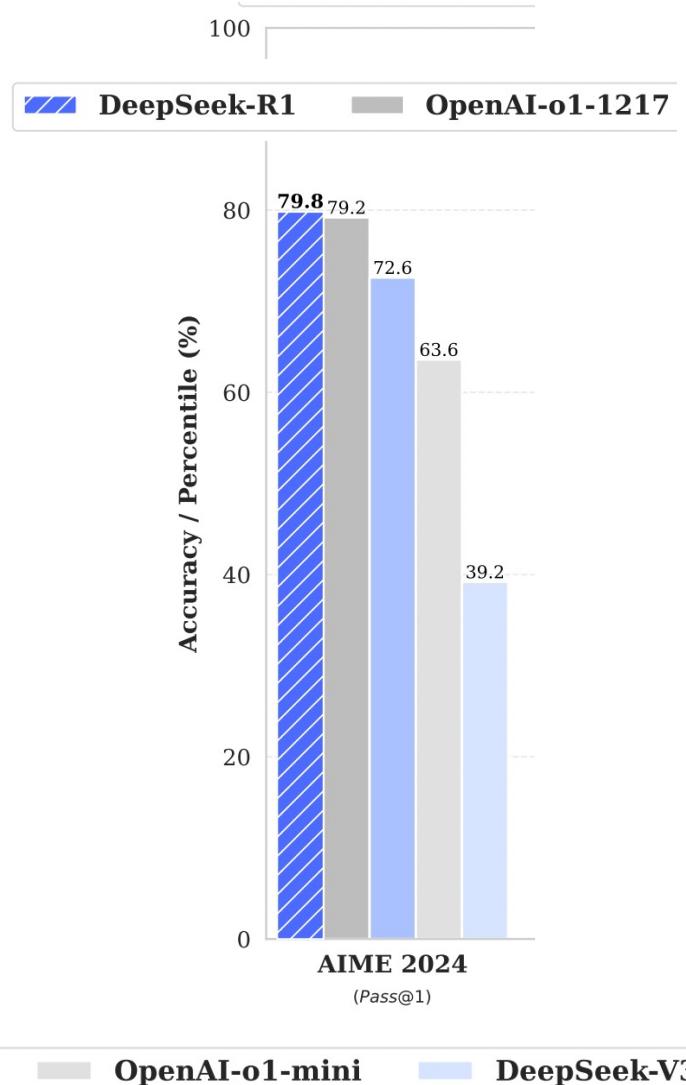
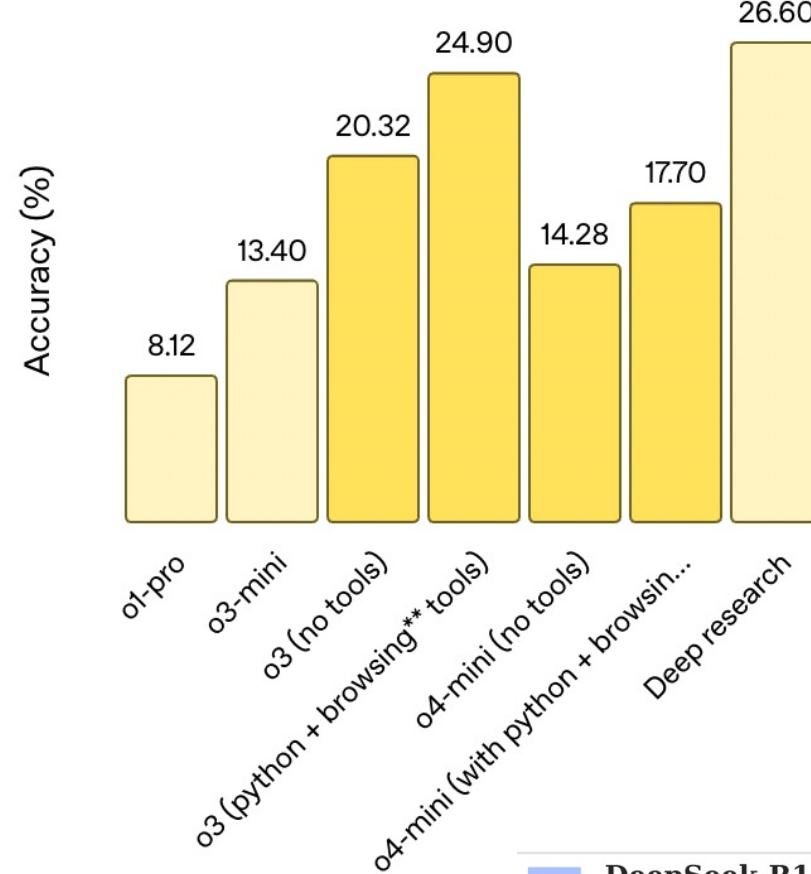


Results: Thinking Models Top the Leaderboards!

Codeforces
Competition Code



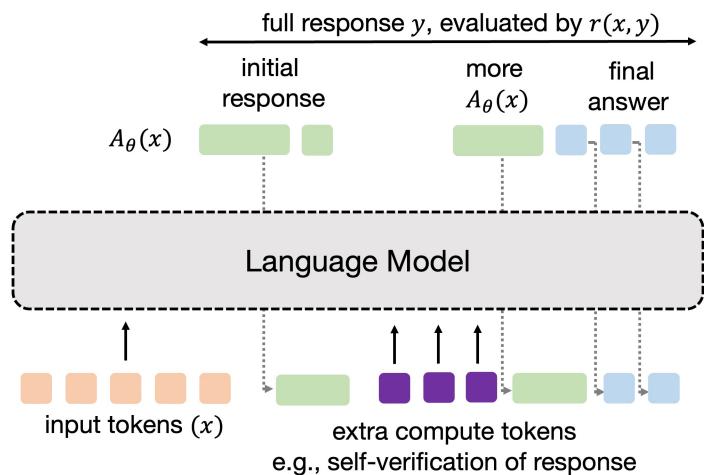
Humanity's Last Exam
Expert-Level Questions Across Subjects



But Lots of Interesting Questions Remain!

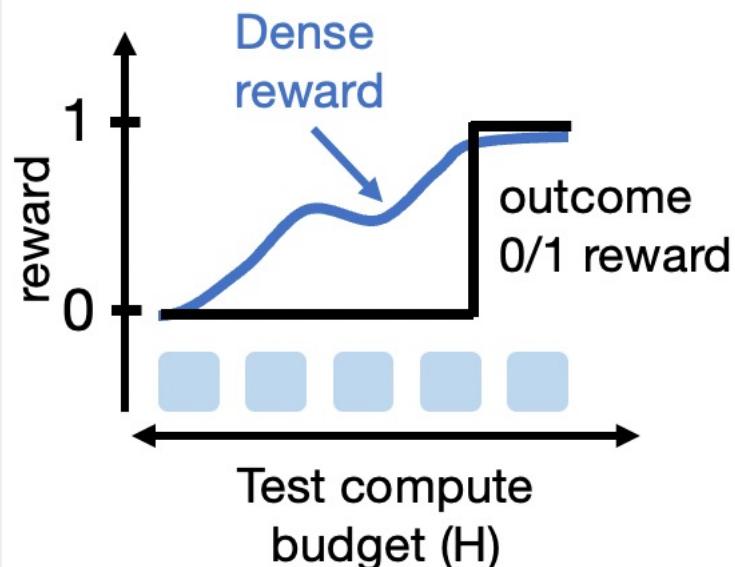
Desiderata and Formulation

Learning “how”: Train algorithm $A_\theta(x)$ to spend **extra test compute**, search over **responses** & discover the **final answer**.



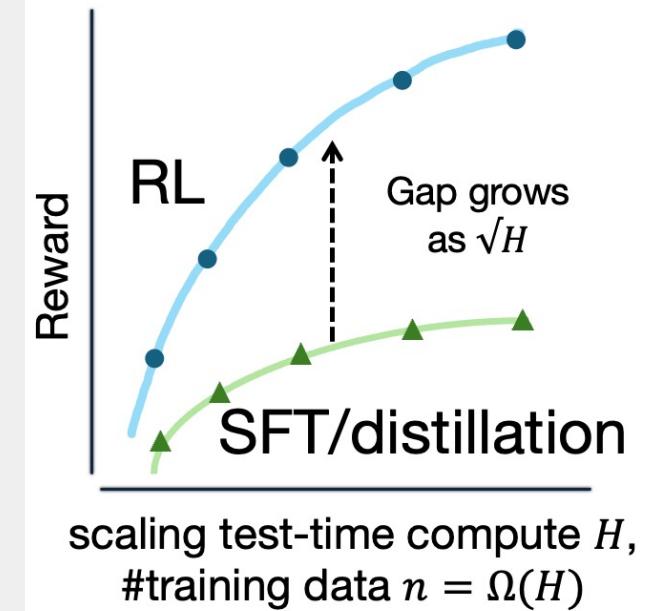
Summary: Pose it as an adaptation problem

Ingredient 1: Dense rewards



Summary: More than outcome reward needed

Ingredient 2: RL >> SFT



Summary: Use reward signals for training