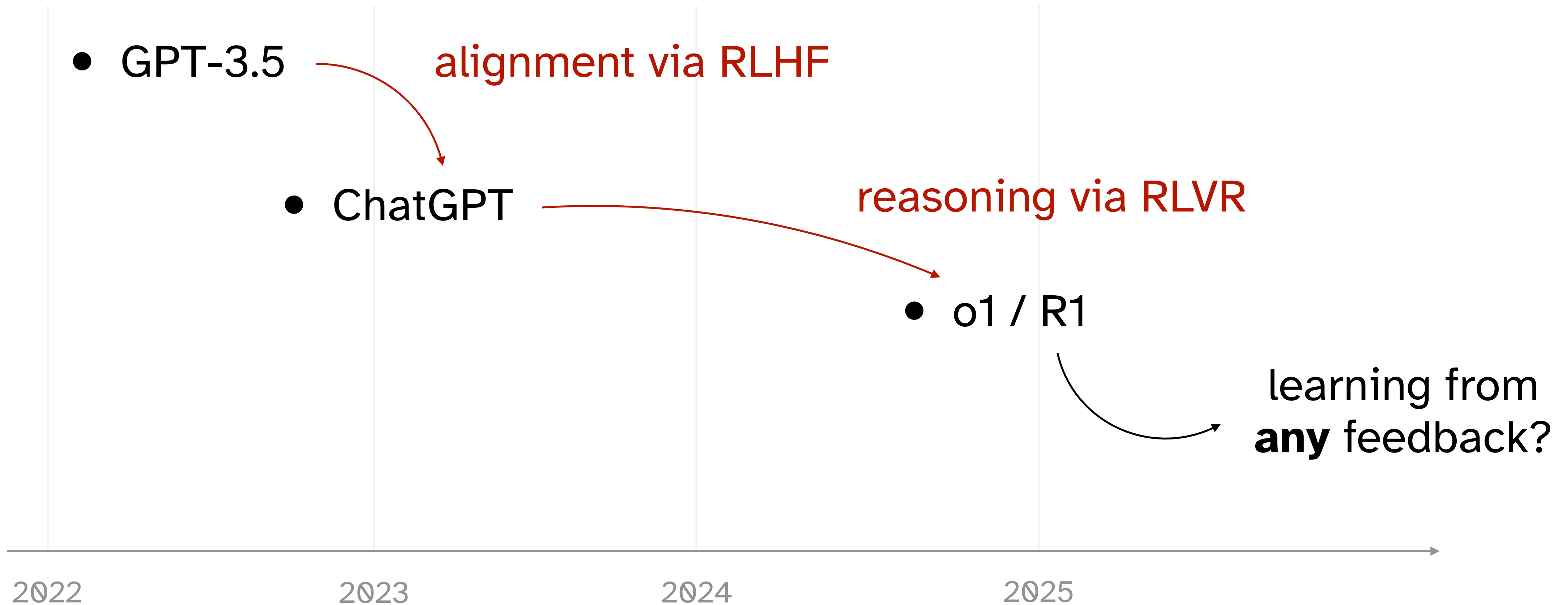


# **Rethinking Post-Training with Self-Distillation**

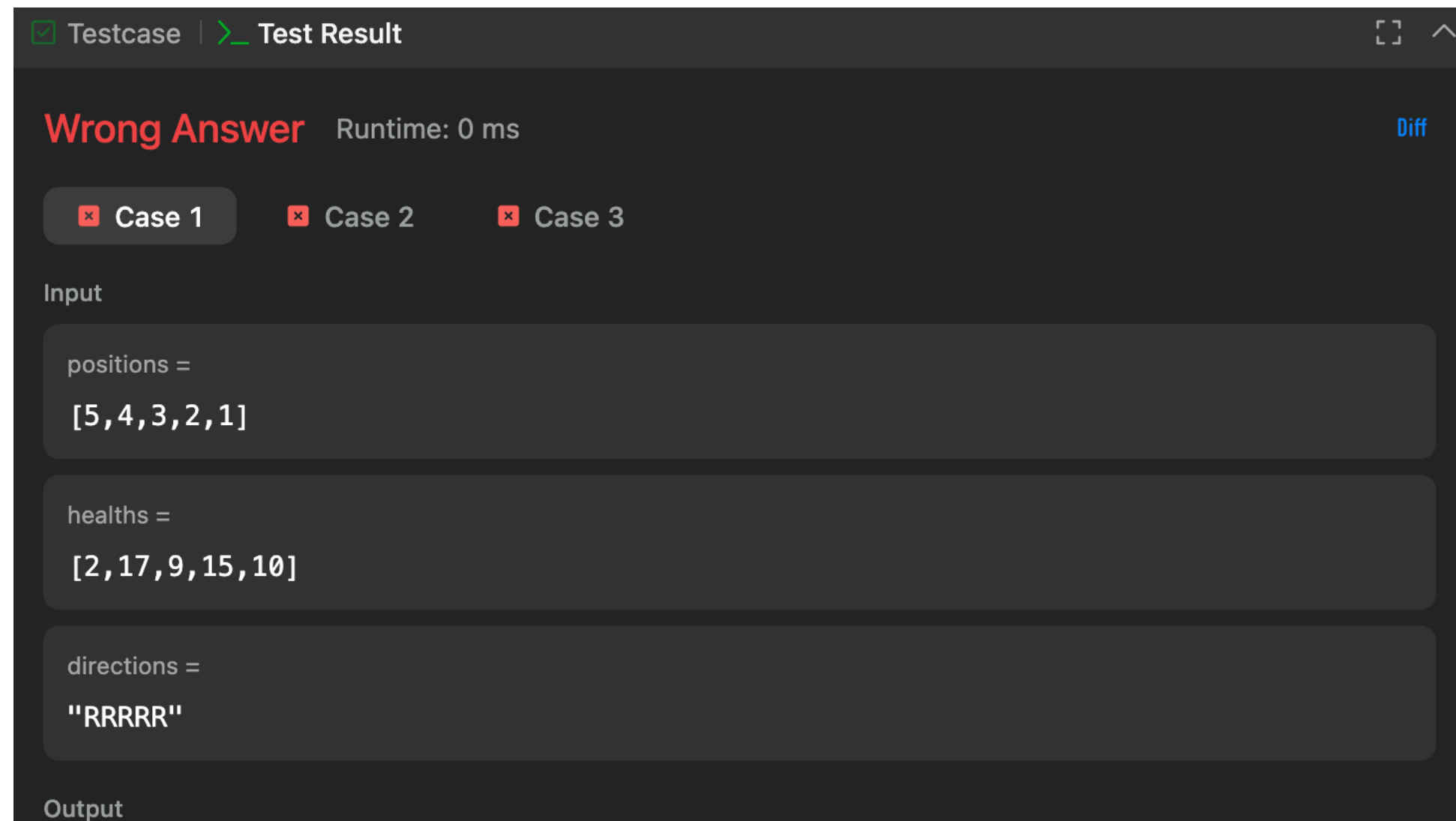
Jonas Hübötter / April 1, 2026

# Reinforcement learning made AI useful



# The feedback bottleneck in RL

How do you improve at coding?



Run unit tests,  
debug runtime errors, profiling, ...

0 / 1  
score



update  
 $\pi_\theta$

# The feedback bottleneck in RL

How do we learn about each other?



Are you a morning person?

Not even close. I just get weirdly cheerful before my brain fully shows up.



That explains a lot, actually.

better  
/  
worse



update  
 $\pi_\theta$

We ask & listen

Is there a way to learn from rich feedback?

# In-context learning is magical!

ChatGPT ⌵ 👤+

What can I help with?

Runtime Error  
ZeroDivisionError: division by zero  
Line 73 in separateSquares (Solution.py)  
Last Executed Input  
[[26,30,2],[11,23,1]]  
+ 🕒 Thinking ⌵

ChatGPT ⌵ 👤+

What can I help with?

Don't be so sycophantic!

+ 🕒 Thinking ⌵ 🎤 ⬆️

# In-context learning is magical!

- Learns from any kind of data
- Learns from little data
- Generalizes very well to specific contexts
- **BUT** learning is transient (disappears when context is removed)
- **AND** context windows are inherently bounded

Long-term learning requires some form of compression...

...and there is one other way:  $\pi_{\theta}(\cdot | x, c)$

# In-context learning is magical!

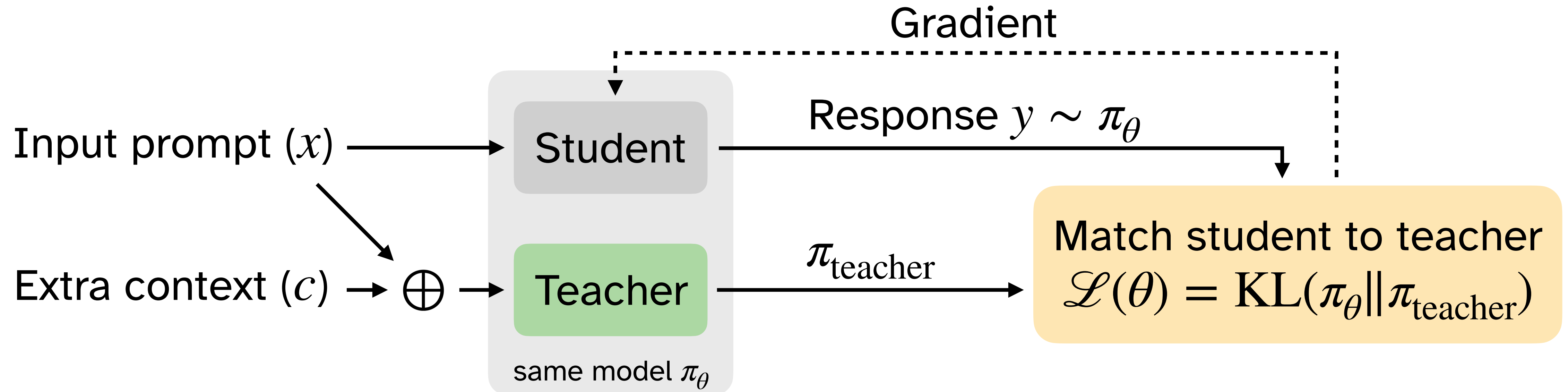
- Learns from any kind of data
- Learns from little data
- Generalizes very well to specific contexts
- **BUT** learning is transient (disappears when context is removed)
- **AND** context windows are inherently bounded

Long-term learning requires some form of compression...

...and there is one other way:  $\pi_{\theta}(\cdot | x, c)$  **in-weight learning!**

# The core idea of self-distillation

“make in-context learning permanent”



This is on-policy distillation (but using yourself with extra context as target):

- **Student:** the initial model
- **(Self-)Teacher:** the same model, but with feedback / additional information

# We let the model decide how to update its policy

more likely in hindsight

User: Answer with YES or NO: Is water wet?

Assistant: YES, water is wet.

User: I said YES or NO only.

less likely in hindsight

$x$  = prompt / conversation history

$y$  = model completion

$c$  = next user message

$$\log \frac{\pi_{\theta}(y_i | x, c, y_{<i})}{\pi_{\theta}(y_i | x, y_{<i})} \begin{array}{l} \longrightarrow \text{improved policy} \\ \longrightarrow \text{original policy} \end{array}$$

# Two equivalent perspectives come from this idea

**Classical RL View:** Policy Gradient with token-level advantages:

$$\text{Maximize } A_i(x, y, o) := \log \frac{\pi_\theta(y_i | x, c, y_{<i})}{\pi_\theta(y_i | x, y_{<i})}$$

**Self-Distillation:** Make  $\pi_\theta(\cdot | x)$  match the improved  $\pi_\theta(\cdot | x, c)$ .

$$\text{Minimize } \mathcal{L}_{\text{SDPO}}(\theta) := \frac{1}{|y|} \sum_{i=1}^{|y|} \text{KL}(\pi_\theta(\cdot | x, y_{<i}) \| \pi_\theta(\cdot | x, c, y_{<i}))$$

Equivalent gradient updates in expectation!

# Self-Distillation across post-training

1. Learning from rich feedback (RLVR setting)
2. Learning from demonstrations (SFT setting)
3. Alignment (RLHF setting)

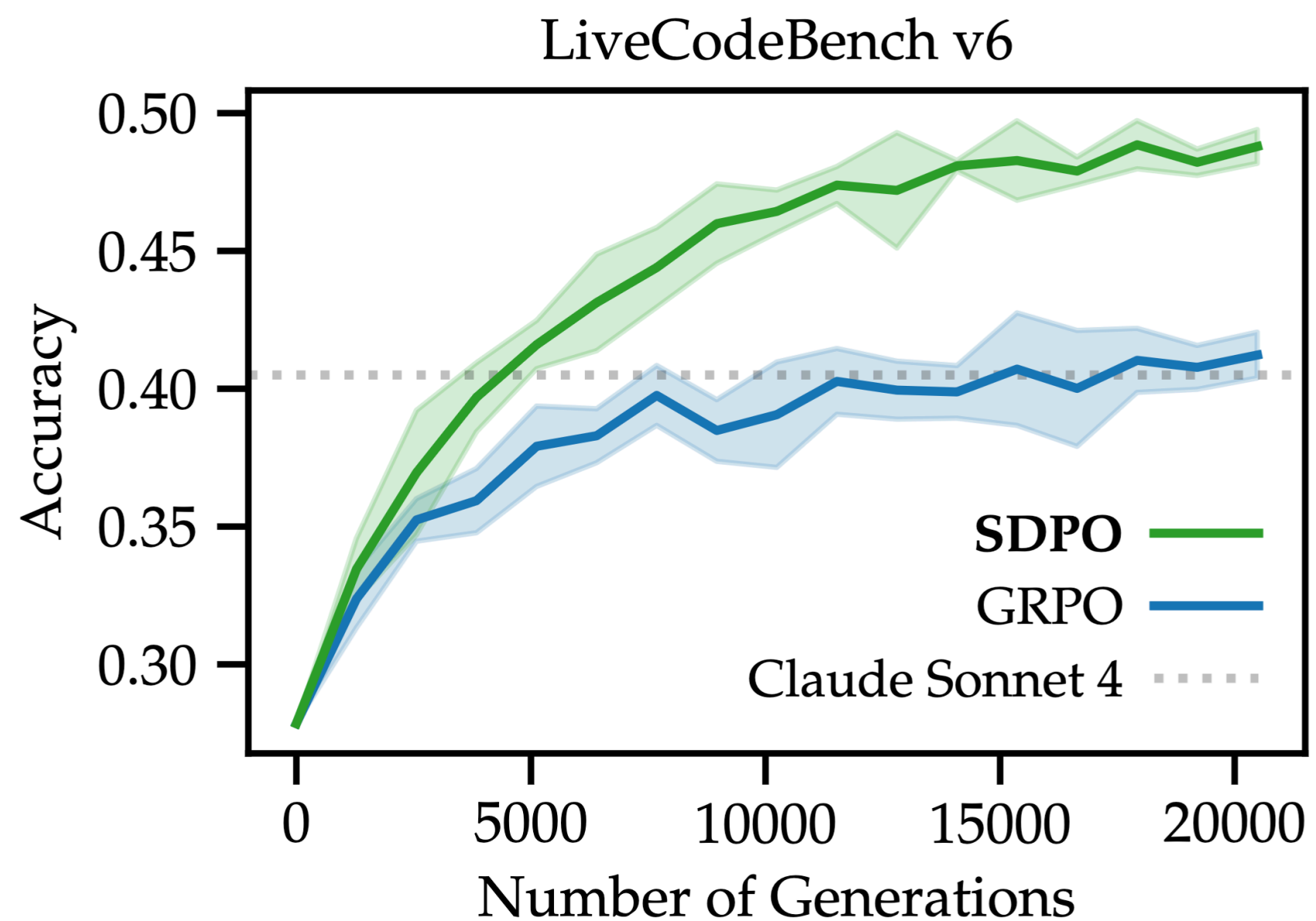
# Self-Distillation across post-training

- 1. Learning from rich feedback (RLVR setting)**
2. Learning from demonstrations (SFT setting)
3. Alignment (RLHF setting)

# Learning from rich feedback in LiveCodeBench

## Example of feedback

```
Runtime Error  
ZeroDivisionError: division by zero  
Line 73 in separateSquares (Solution.py)  
  
Last Executed Input  
[[26,30,2],[11,23,1]]
```

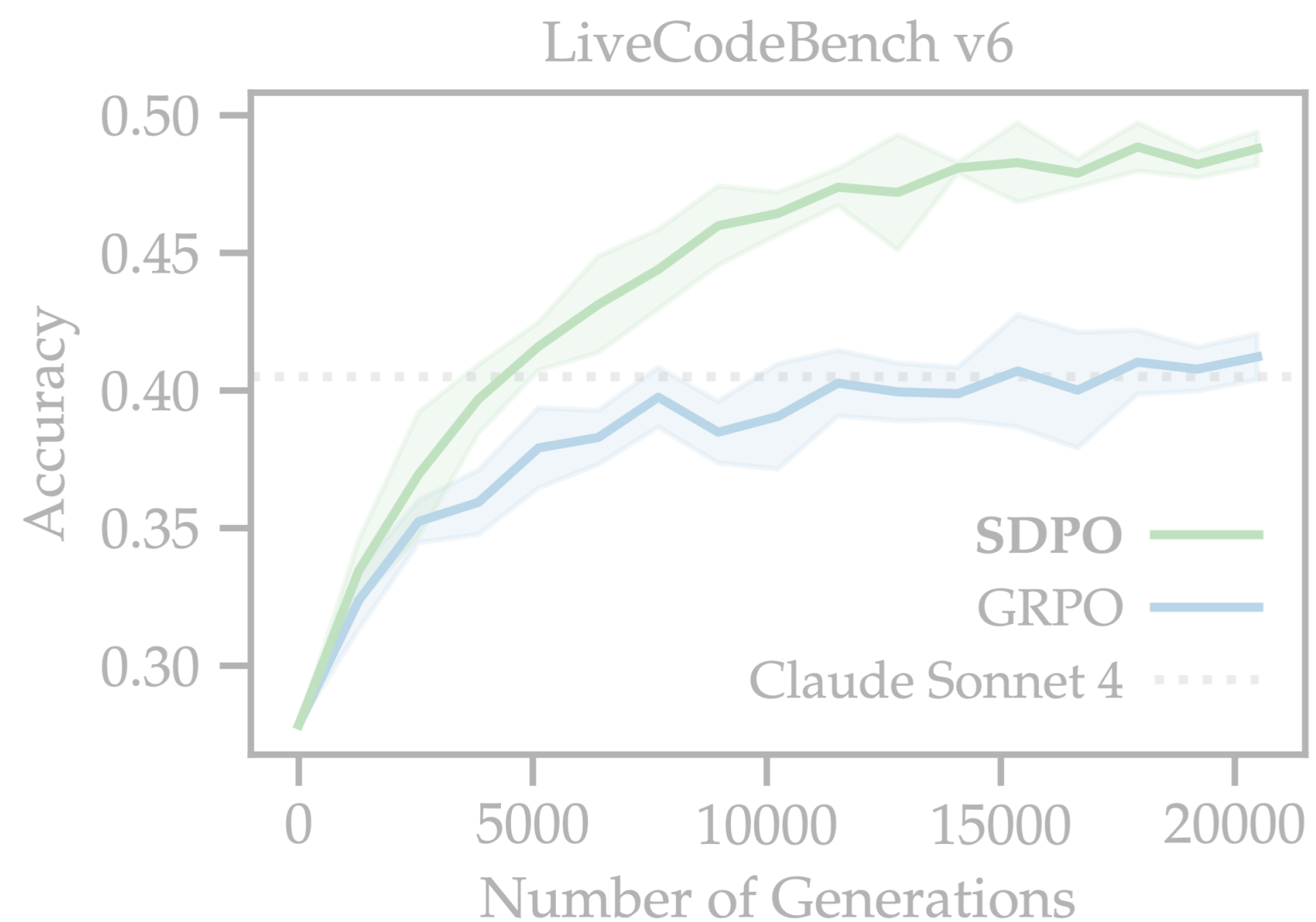


# Learning from rich feedback in LiveCodeBench

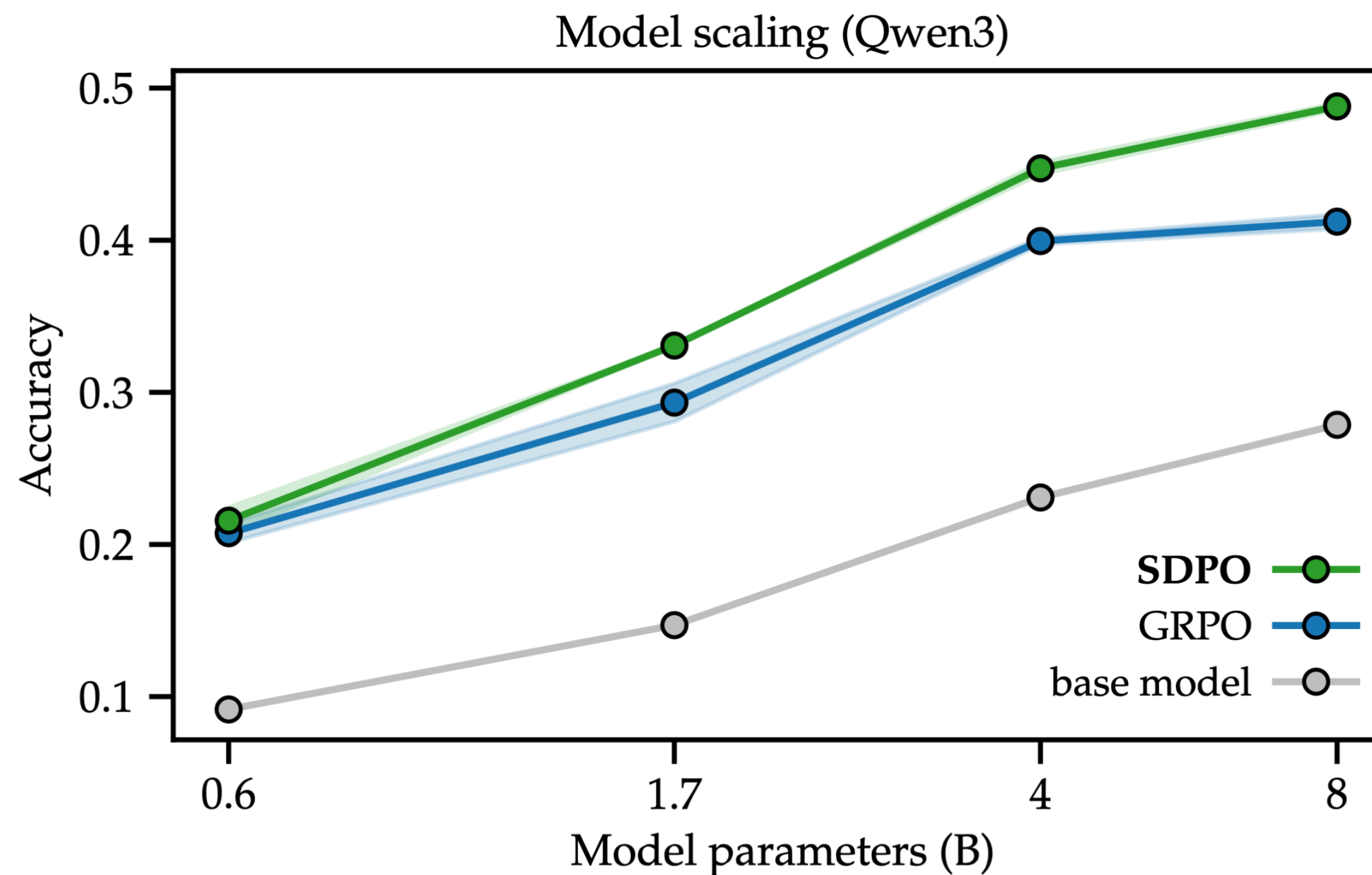
Example of feedback

```
Runtime Error  
ZeroDivisionError: division by zero  
Line 73 in separateSquares (Solution.py)
```

```
Last Executed Input  
[[26,30,2],[11,23,1]]
```



Better performance with stronger in-context learners!



# Existing methods are bottlenecked

Question:

Write a python function that returns all numbers from 1 to n. Answer briefly.

2) Answer  $y \sim \pi_{\theta}(\cdot | x)$ :

```
```python
def numbers_up_to_n(n):
    return list(range(1, n + 1))
```
```

Feedback:

Don't include n.

## GRPO

Generated tokens →

... (range ( 1 , n + 1 ))\n

1. Receiving a **scalar signal** per rollout
2. Using this signal for **rollout-level credit assignment**

# Existing methods are bottlenecked

Question:

Write a python function that returns all numbers from 1 to n. Answer briefly.

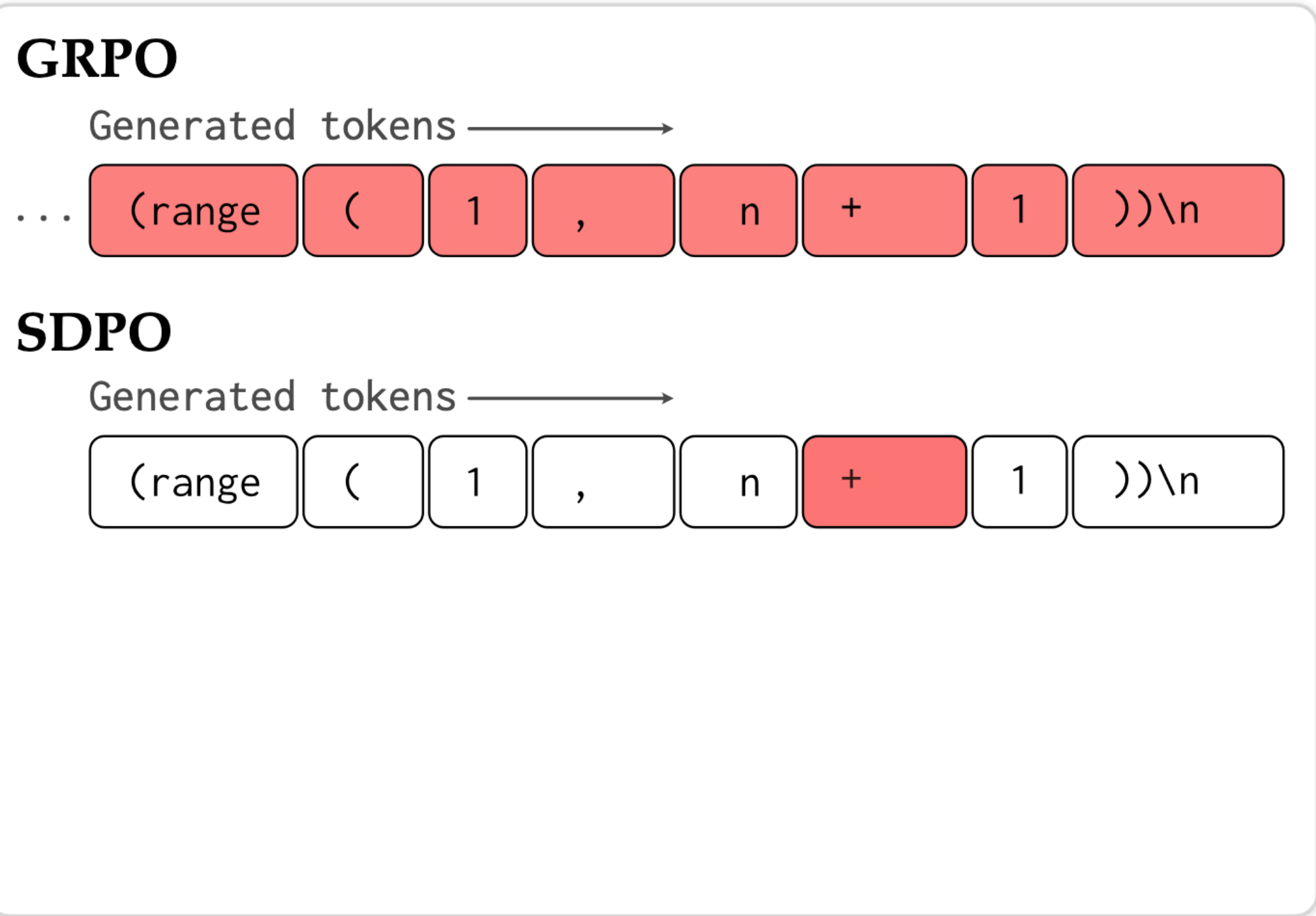
2) Answer  $y \sim \pi_{\theta}(\cdot | x)$ :

```
python
def numbers_up_to_n(n):
    return list(range(1, n + 1))
```

Feedback:

Don't include n.

1. Receiving a **scalar signal** per rollout
2. Using this signal for **rollout-level credit assignment**



# Existing methods are bottlenecked

Question:

Write a python function that returns all numbers from 1 to n. Answer briefly.

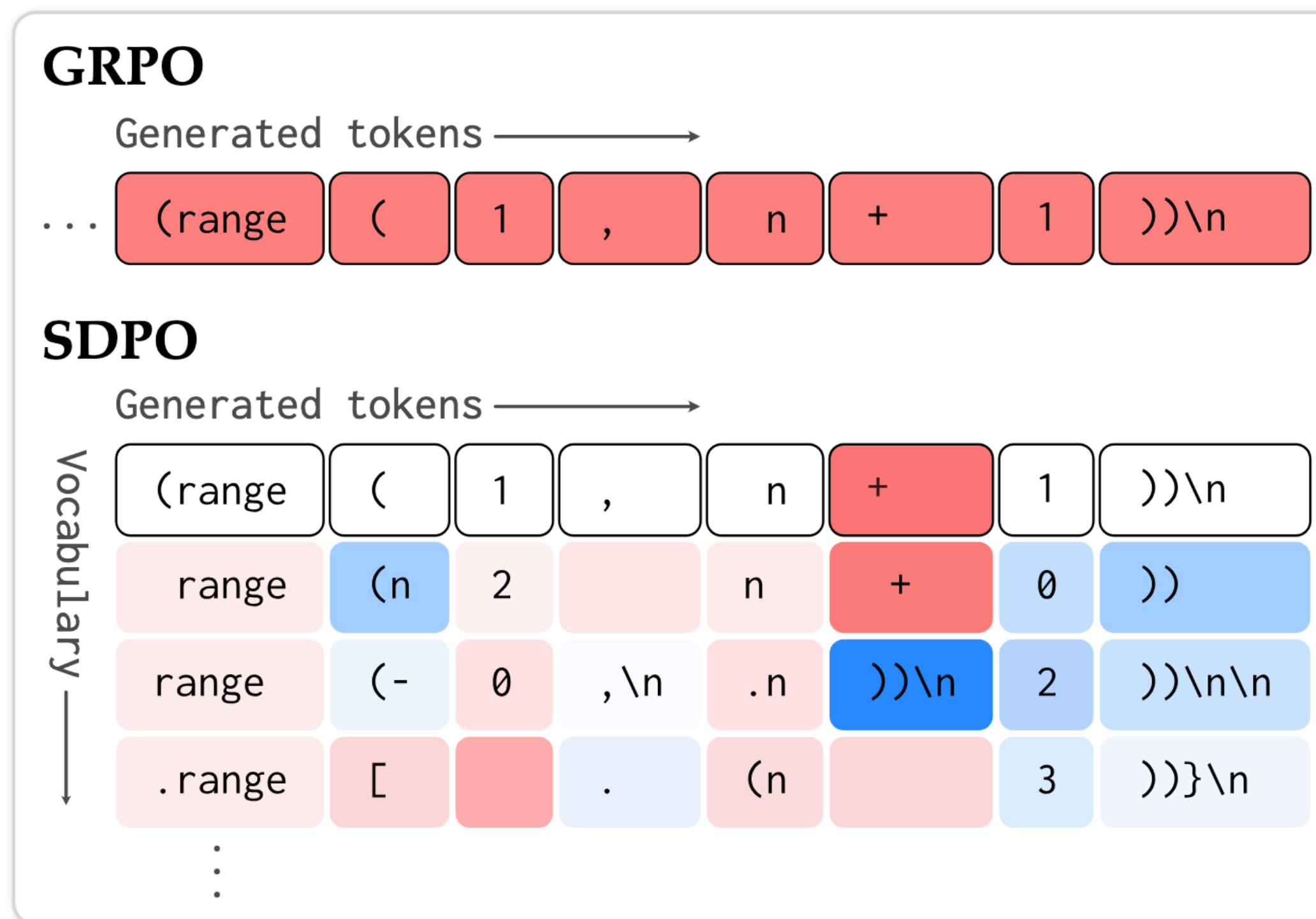
2) Answer  $y \sim \pi_{\theta}(\cdot | x)$ :

```
python
def numbers_up_to_n(n):
    return list(range(1, n + 1))
```

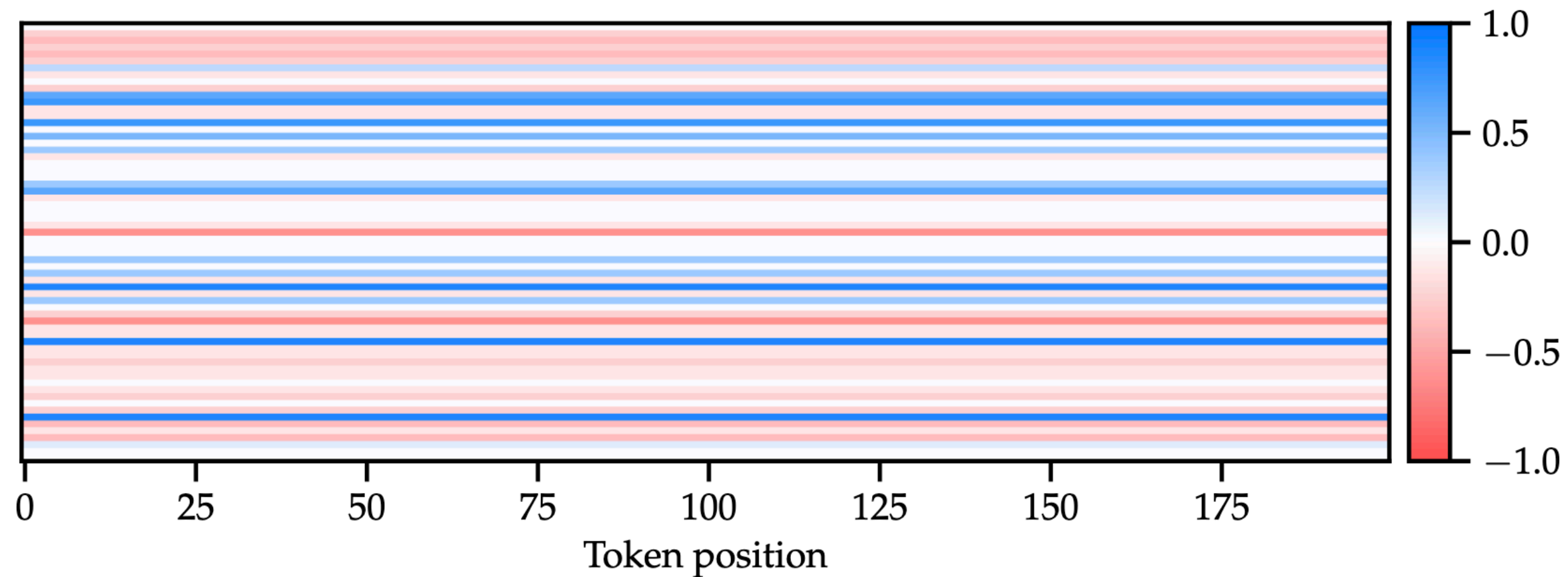
Feedback:

Don't include n.

1. Receiving a **scalar signal** per rollout
2. Using this signal for **rollout-level credit assignment**

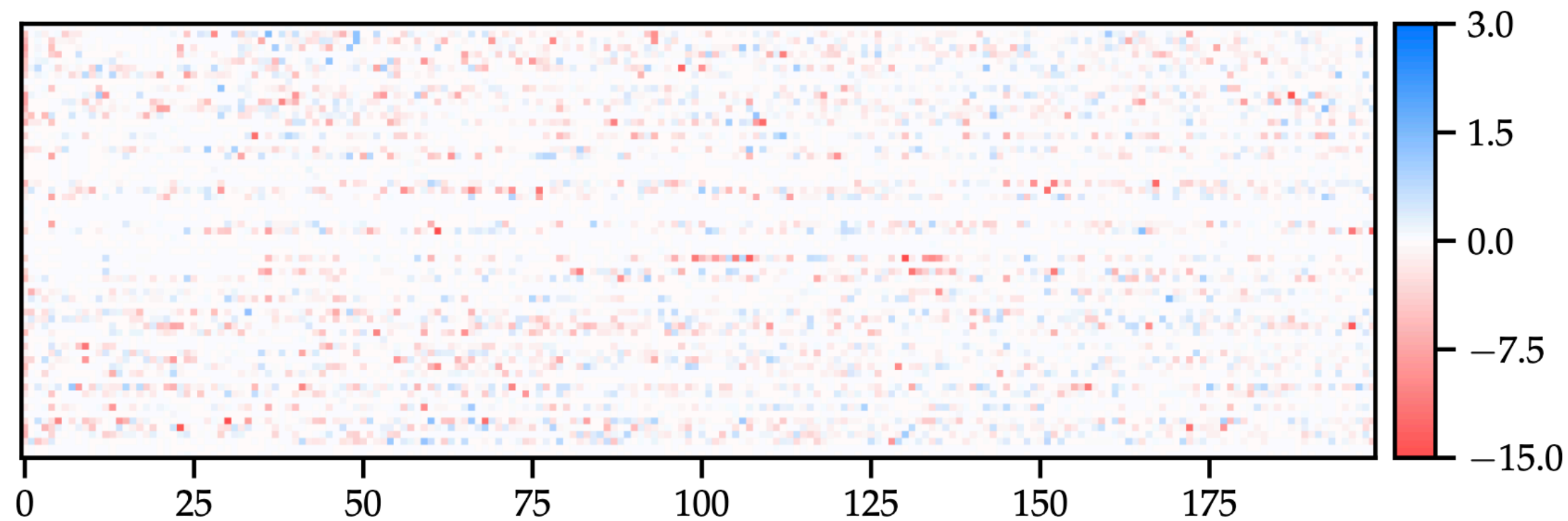


# Dense credit assignment of self-distillation



**GRPO:** Was a response better/worse than average?

→ make all tokens more/less likely



**SDPO:** Was this token good/bad in hindsight given additional context?

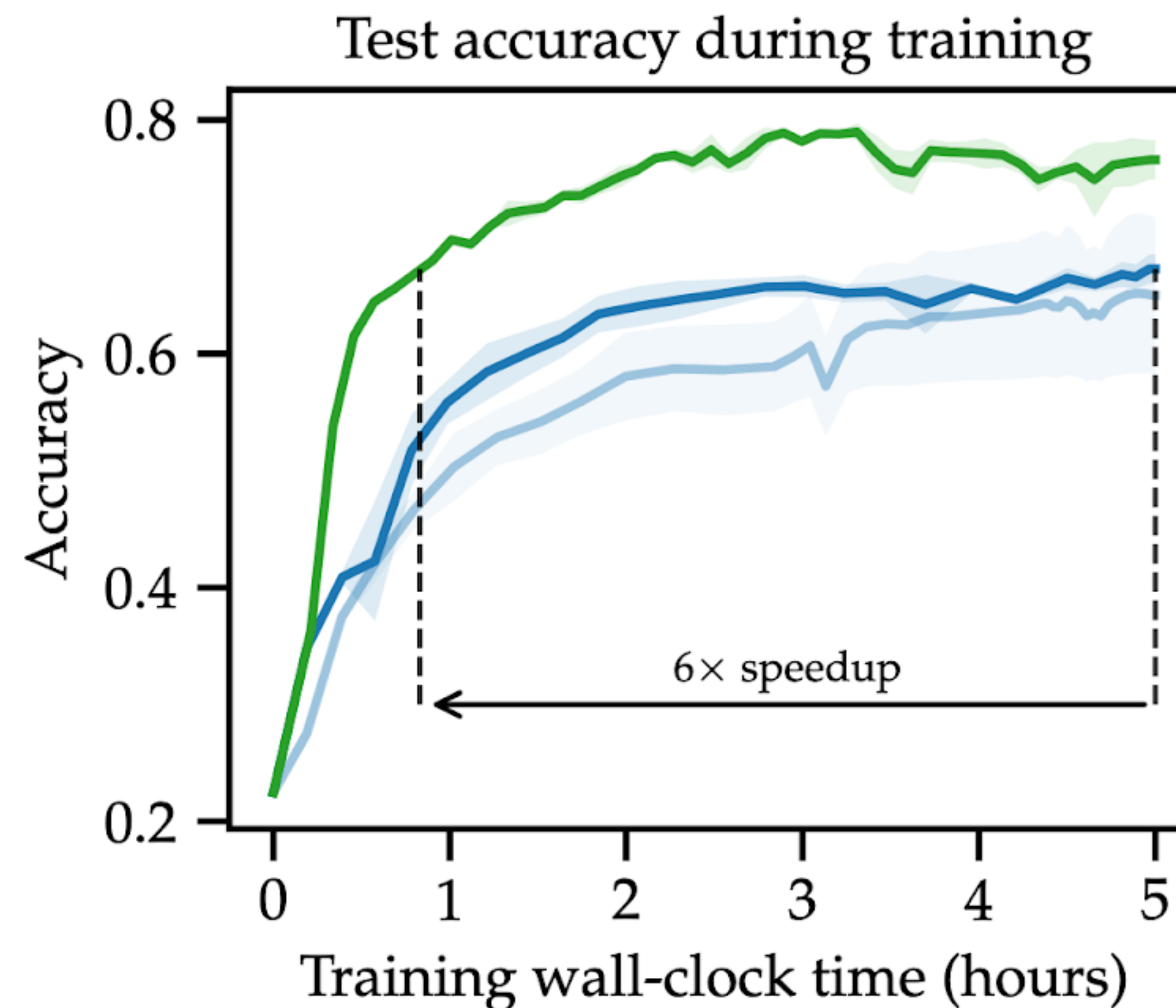
→ comments on *individual* tokens

# Credit assignment helps, even without rich feedback

Self-distillation in the RLVR setting:

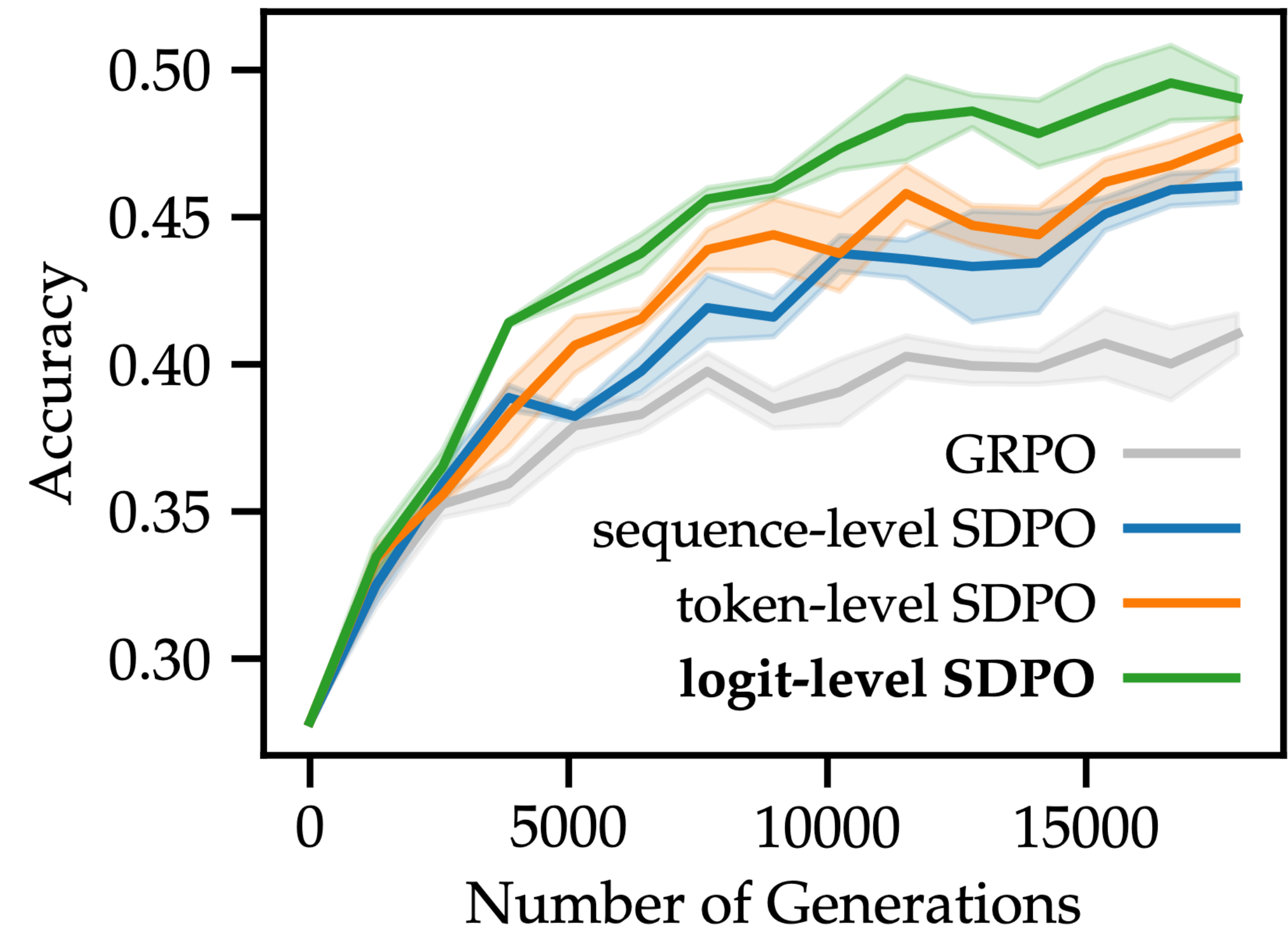
- Multiple rollouts per question
- Use correct attempts as teacher context for incorrect attempts

— SDPO (on-policy)    — GRPO (on-policy)    — GRPO



chemistry reasoning, avg over 3 seeds

**Rich feedback helps,  
even without credit  
assignment**



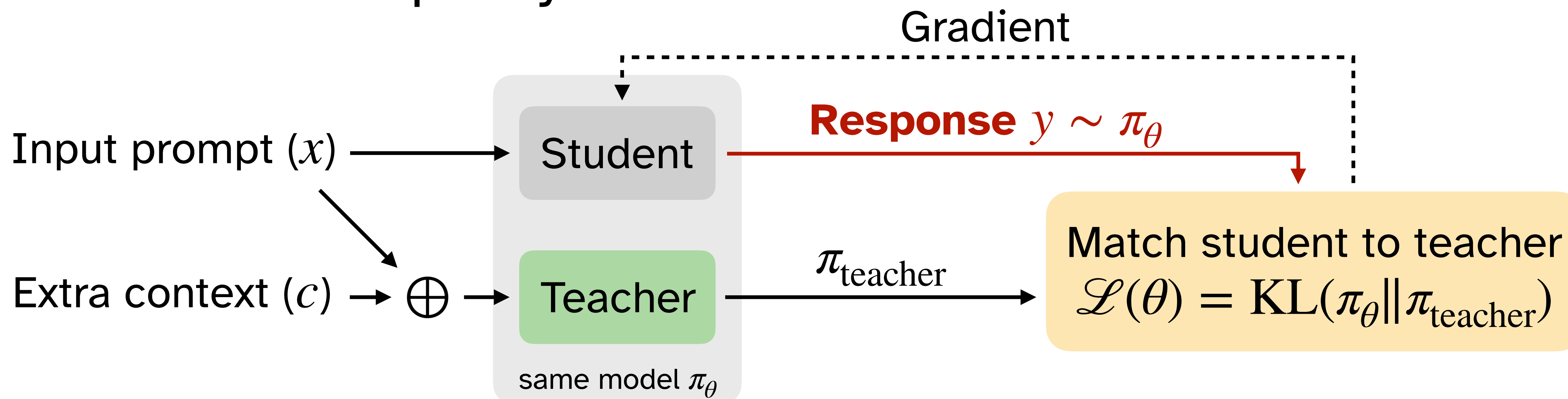
# Self-Distillation across post-training

1. Learning from rich feedback (RLVR setting)
- 2. Learning from demonstrations (SFT setting)**
3. Alignment (RLHF setting)

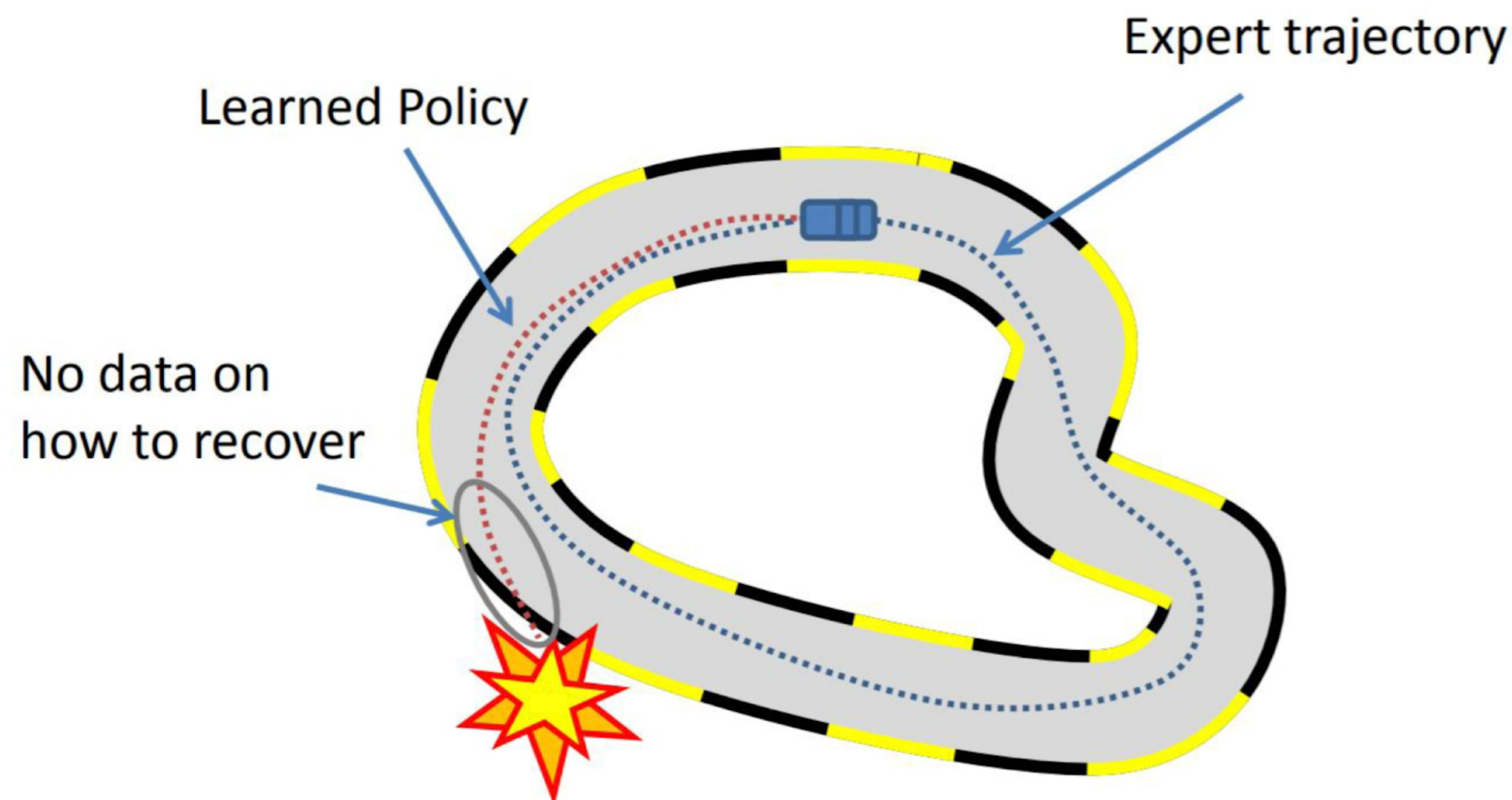
# There is one thing I have been hiding...

The fact that self-distillation is on-policy is important!

What does “on-policy” mean?



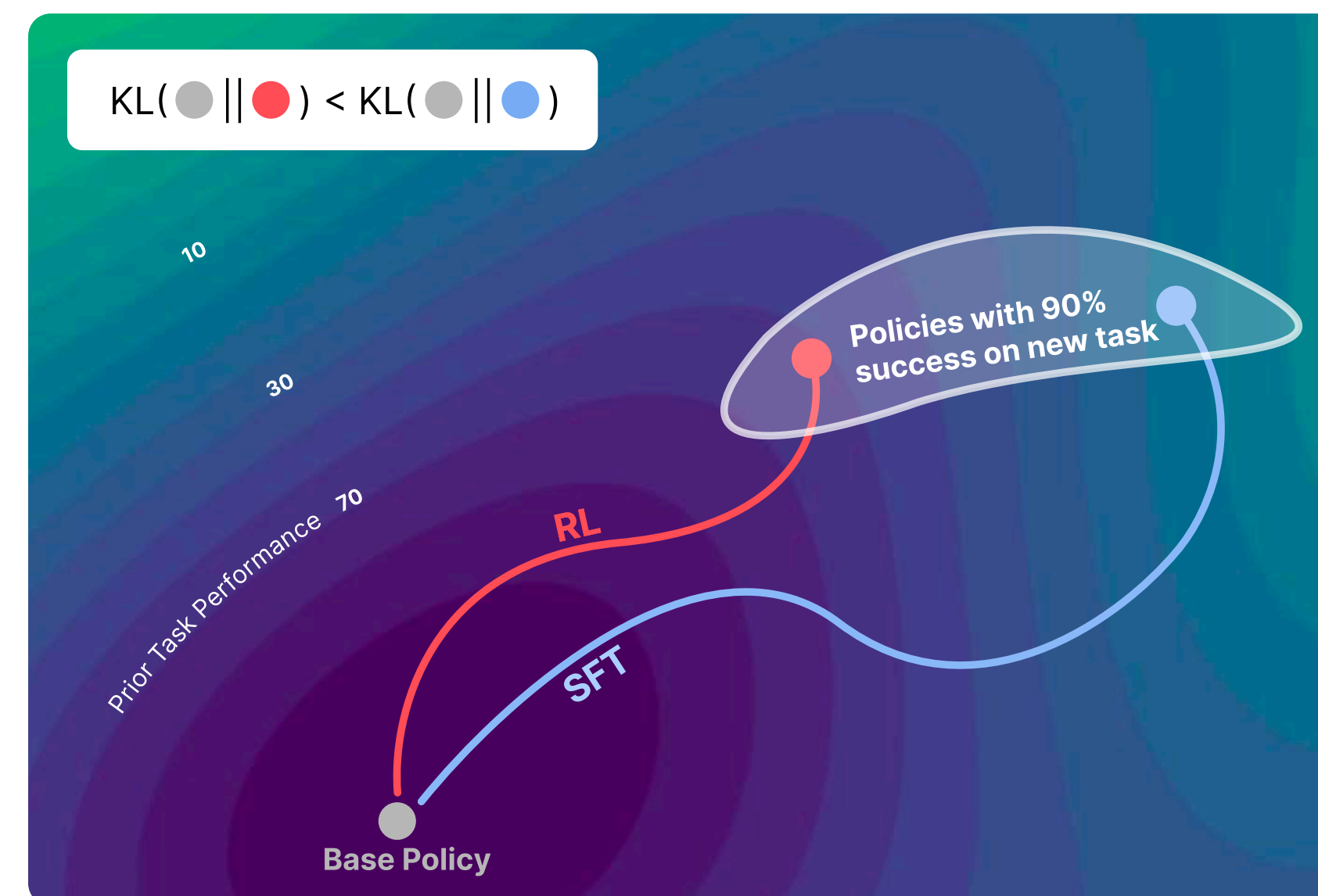
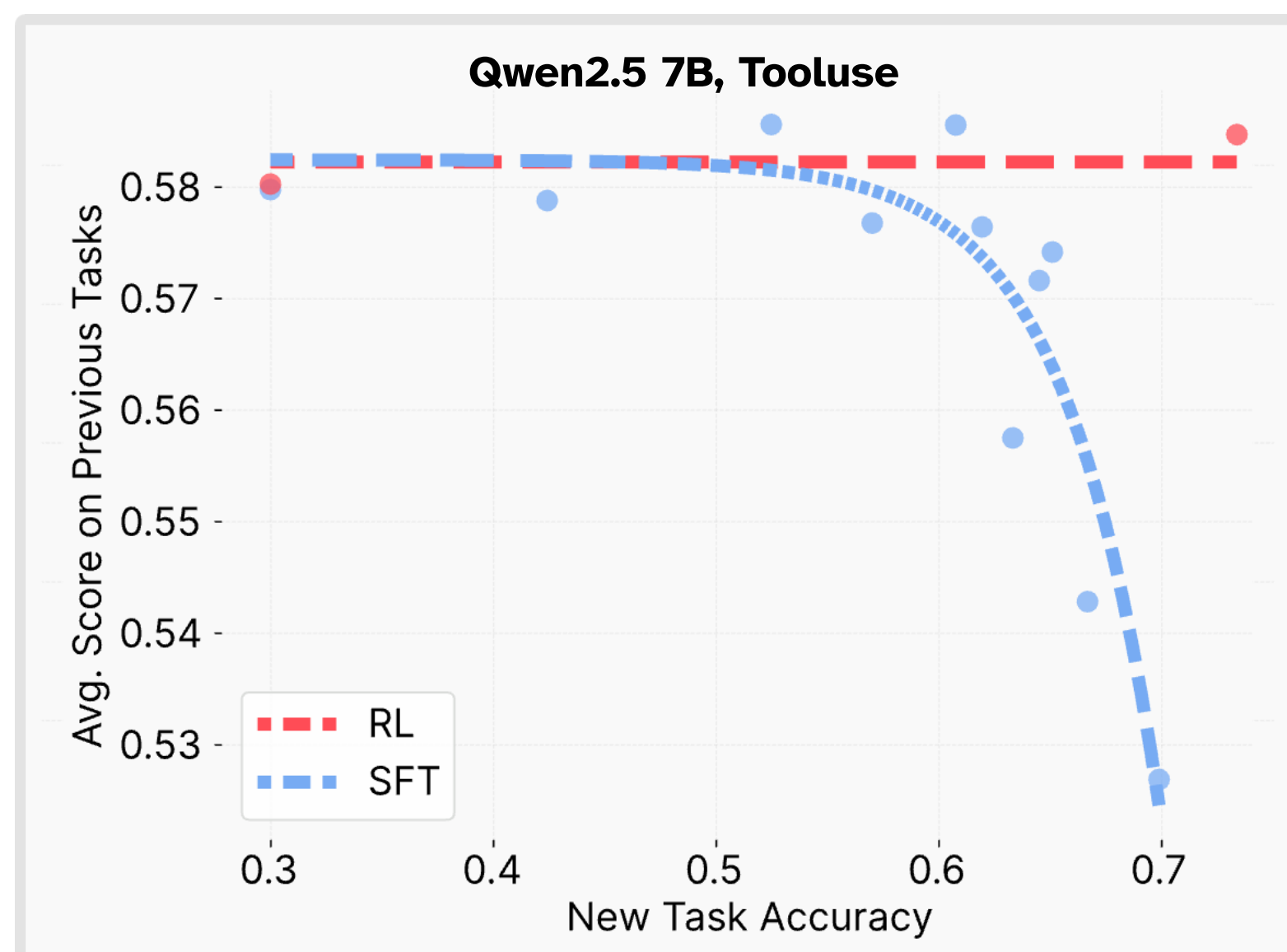
# Why on-policy leads to better performance



Copying demonstrations of an expert leads to **entering OOD states**.

A robust policy is able to deal with the states it visits itself!

# Why on-policy avoids forgetting



Among the many high-reward solutions for a new task, on-policy methods are implicitly biased toward solutions closer to the original policy in KL divergence.

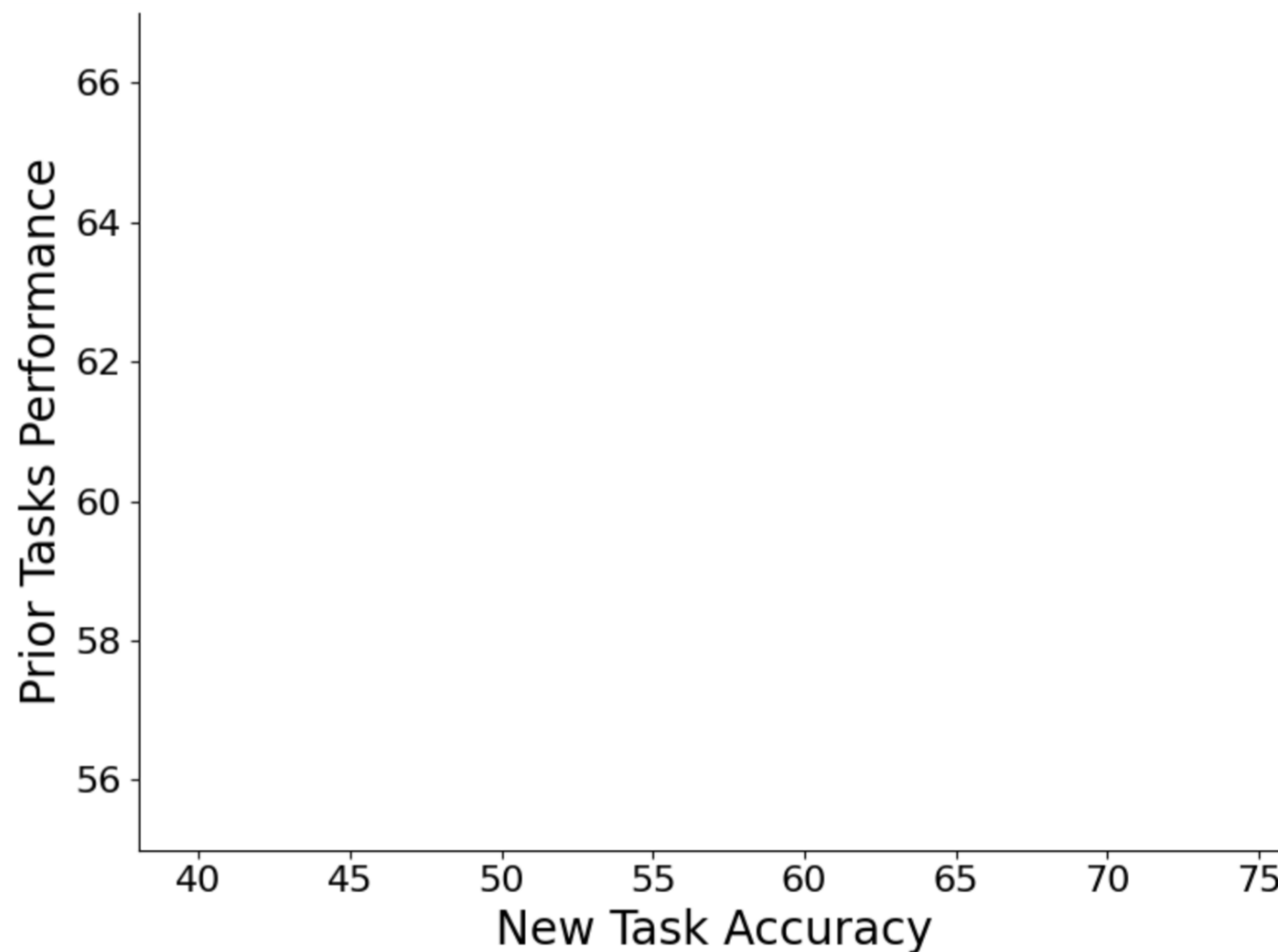
# Better performance, no forgetting

New tasks:

- science reasoning
- tool use
- medical reasoning

Holdout tasks:

- Hellaswag, TruthfulQA
- MMLU
- Humaneval
- IFeval



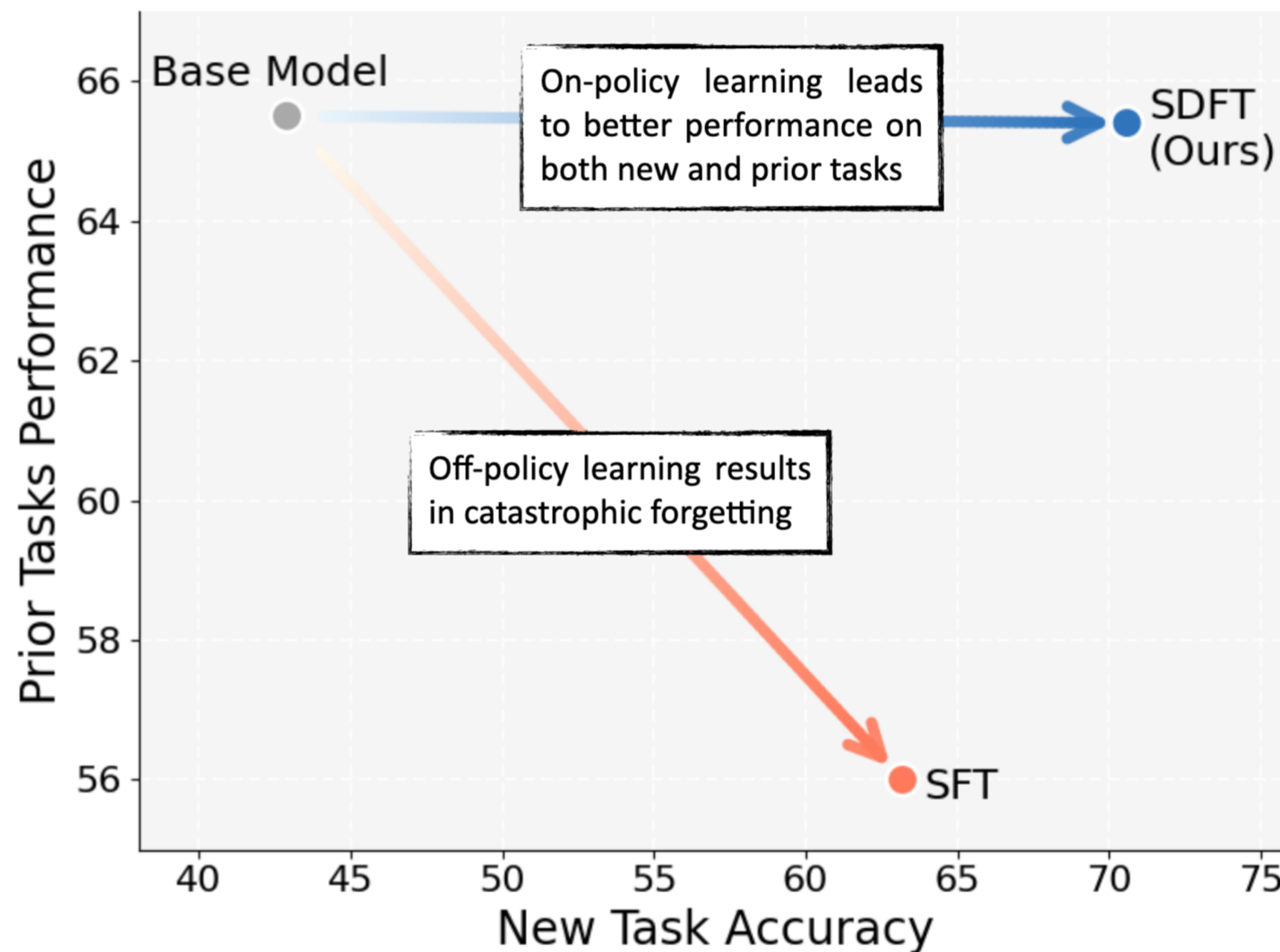
# Better performance, no forgetting

New tasks:

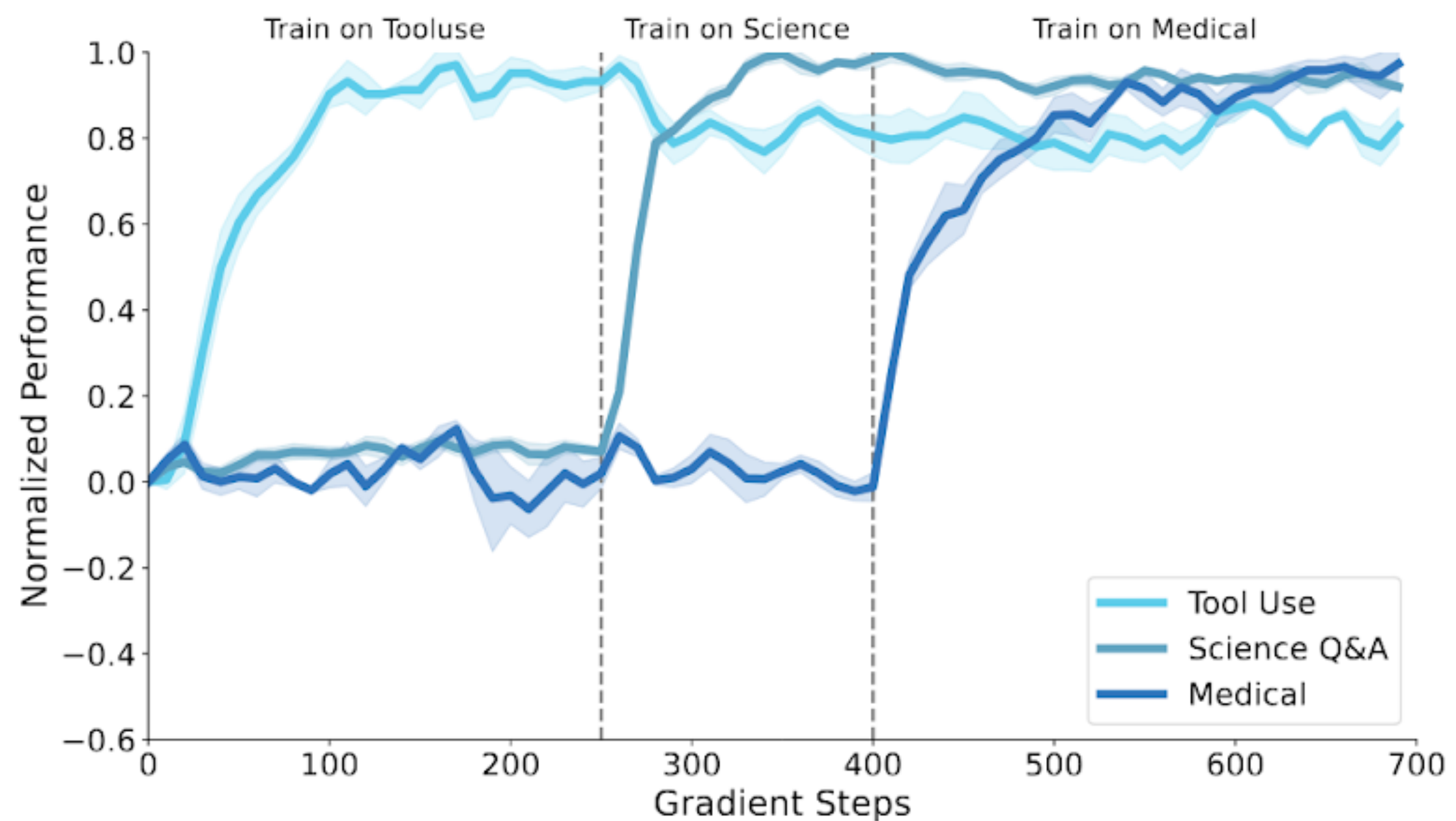
- science reasoning
- tool use
- medical reasoning

Holdout tasks:

- Hellaswag, TruthfulQA
- MMLU
- Humaneval
- IFeval

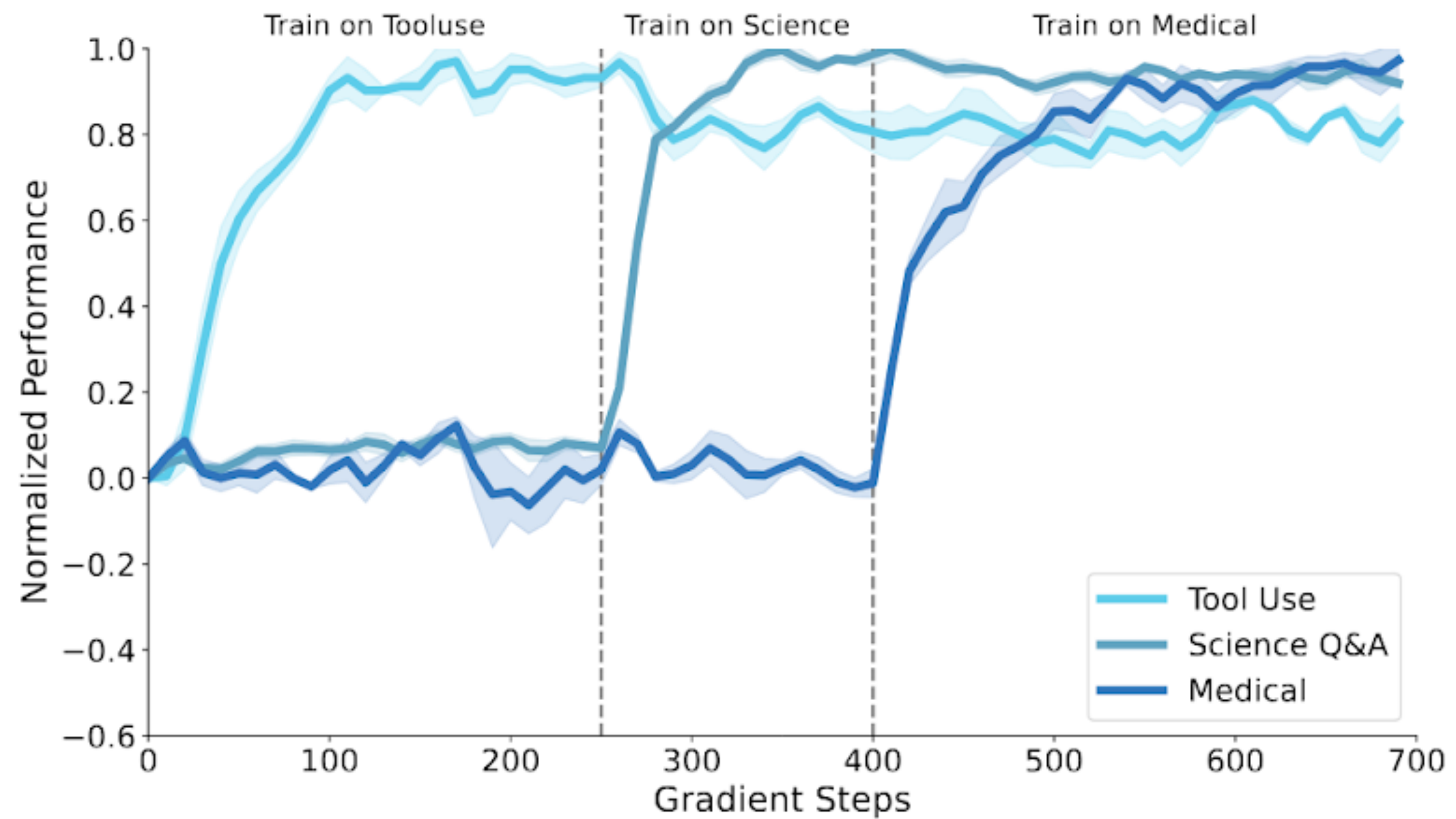


# Sequentially learning new skills

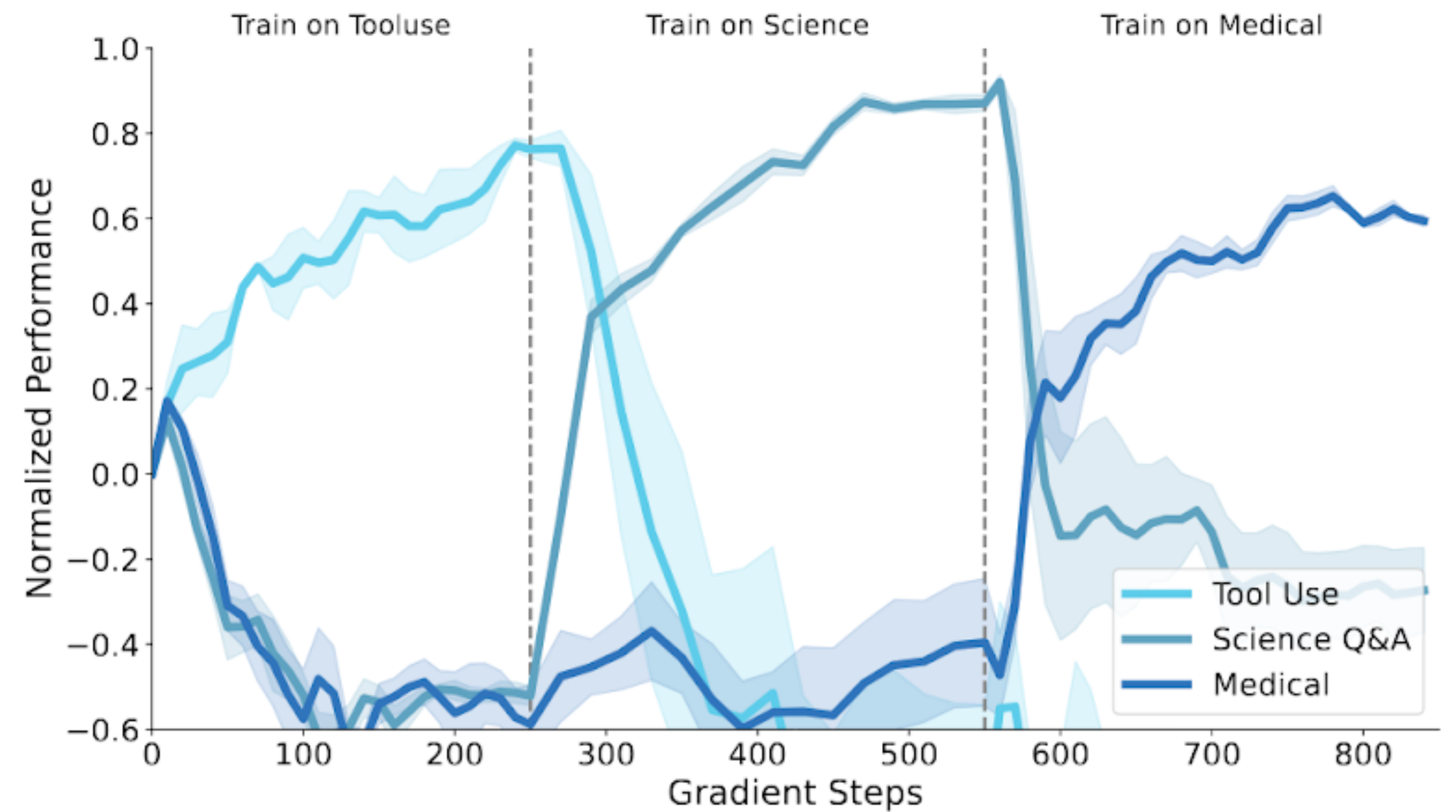


On-policy self-distillation

# Sequentially learning new skills



On-policy self-distillation



Supervised fine-tuning (SFT)

# Learning knowledge

- This is practically impossible with on-policy RL!
- Baselines:
  - Always correct article in context
  - CPT: Continued pre-training on dataset
  - SFT / “self-study”: Synthetically generate questions, then train on answers from model with data in context
- Ours: On-policy self-distillation with synthetically generated questions

---

|            | Accuracy<br>(strict) | Accuracy<br>(lenient) | OOD<br>Accuracy |
|------------|----------------------|-----------------------|-----------------|
| Base       | 0                    | 0                     | 0               |
| Oracle RAG | 91                   | 100                   | 100             |

---

# Learning knowledge

- This is practically impossible with on-policy RL!
- Baselines:
  - Always correct article in context
  - CPT: Continued pre-training on dataset
  - SFT / “self-study”: Synthetically generate questions, then train on answers from model with data in context
- Ours: On-policy self-distillation with synthetically generated questions

|             | Accuracy<br>(strict) | Accuracy<br>(lenient) | OOD<br>Accuracy |
|-------------|----------------------|-----------------------|-----------------|
| Base        | 0                    | 0                     | 0               |
| Oracle RAG  | 91                   | 100                   | 100             |
| CPT         | 9                    | 37                    | 7               |
| SFT         | 80                   | 95                    | 80              |
| SDFT (Ours) | <b>89</b>            | <b>100</b>            | <b>98</b>       |


On-policy self-distillation can integrate new knowledge


# Self-Distillation across post-training

1. Learning from rich feedback (RLVR setting)
2. Learning from demonstrations (SFT setting)
- 3. Alignment (RLHF setting)**


# Directly learning from real user conversations

Most compute today is spent on inference  
**Can we use this vast data for training?**

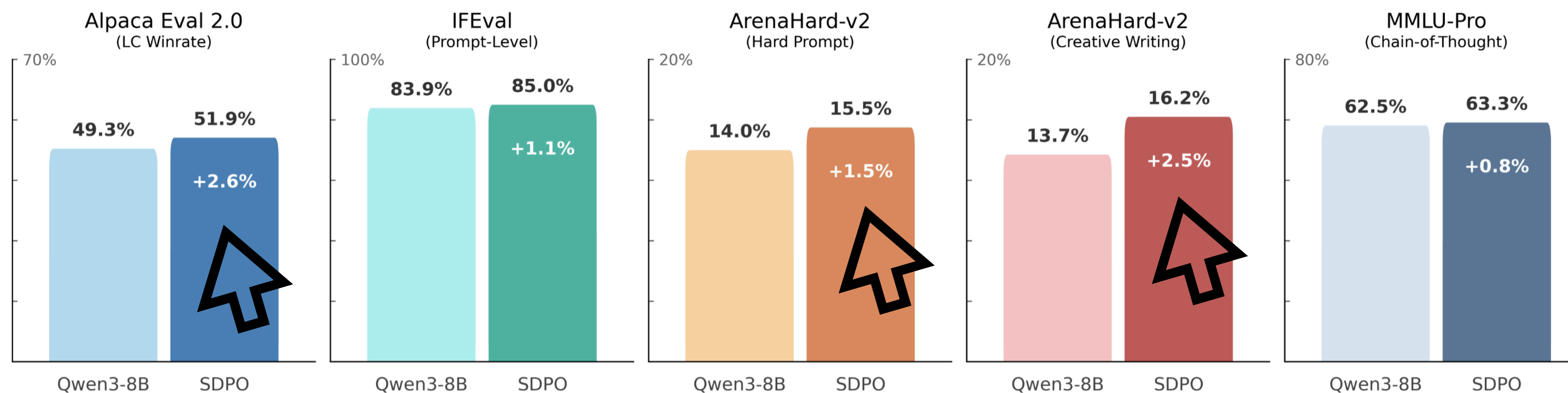
Should I standardize features before training a linear model? 

 Good question! This is something that many get wrong when they first start working with machine learning models. [...]

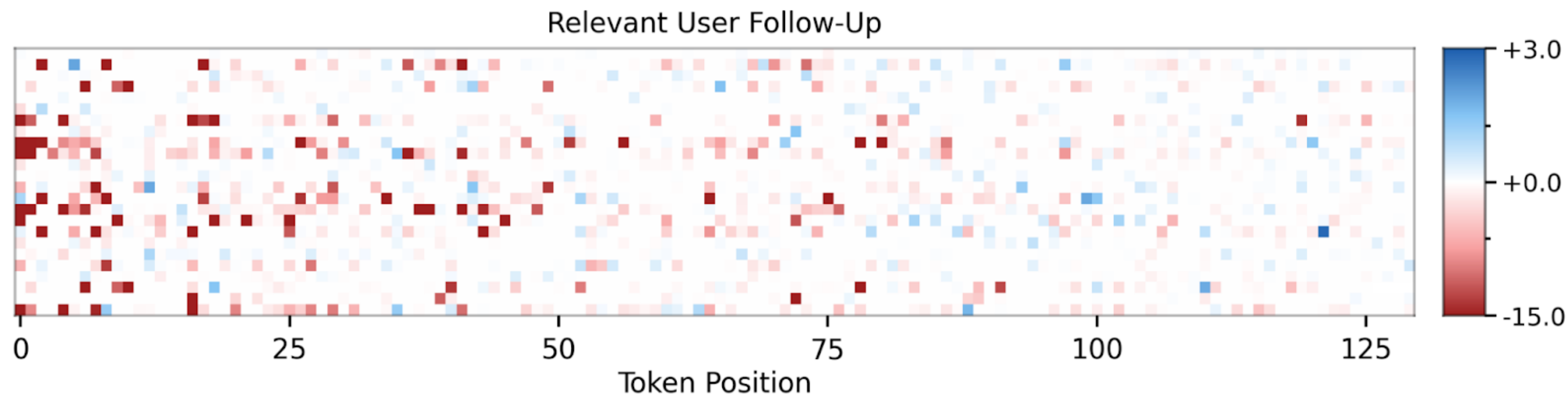
Yes. You should standardize features for most linear models because [...]

Get to the point... Just answer directly without filler. 

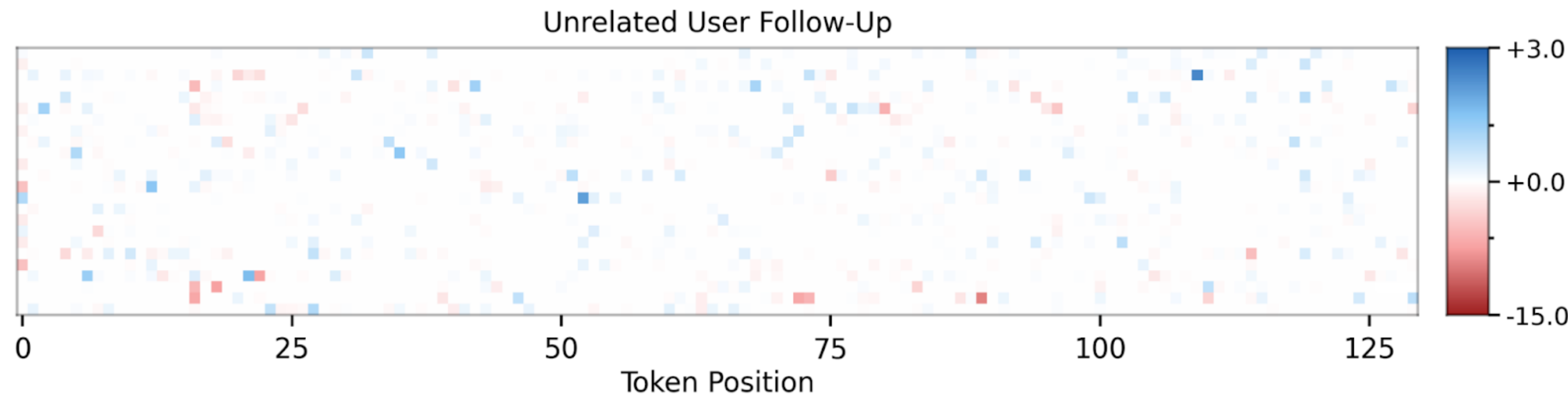
- We take 14k real-world user conversations from WildChat
- Train on triplets (prompt, model response, user response) with self-distillation



# The model identifies when to update & when not to



```
User: Write a short, friendly email to a teammate summarizing project status. Say progress is ...  
Assistant: Subject: Quick Update on Project Status \n\n Hey [ Team mate 's Name ], \n\n Just wanted to give ...  
User: Rewrite in a formal, professional tone.
```



```
User: Write a short, friendly email to a teammate summarizing project status. Say progress is ...  
Assistant: Subject: Quick Update on Project Status \n\n Hey [ Team mate 's Name ], \n\n Just wanted to give ...  
User: What is 27*4?
```

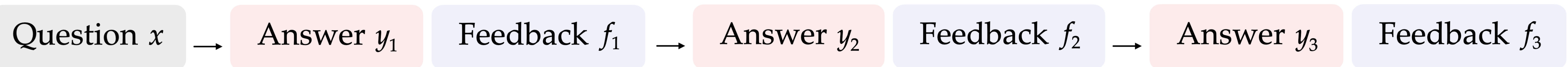
# What we want in a learning paradigm

- Performance ✓
- Avoiding forgetting ✓
- Learning from arbitrary, raw data ✓
- Learning efficiently from few samples ?

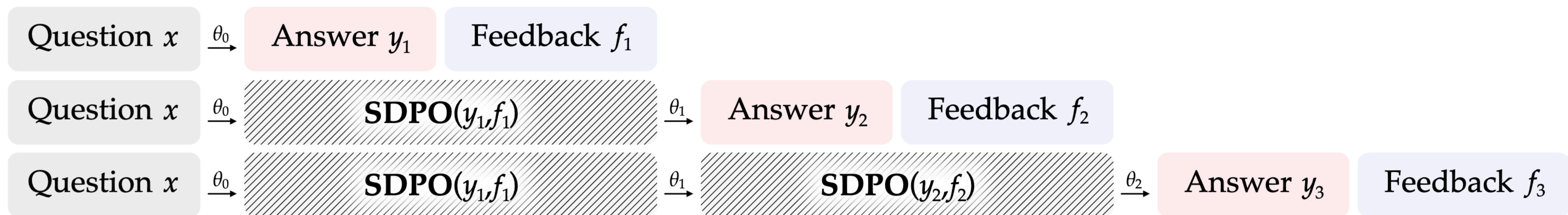
# **Towards a true continual learner: Learning when “deployed in the wild”**

# Towards a true continual learner

- Example: solving a single hard task in multiple iterations

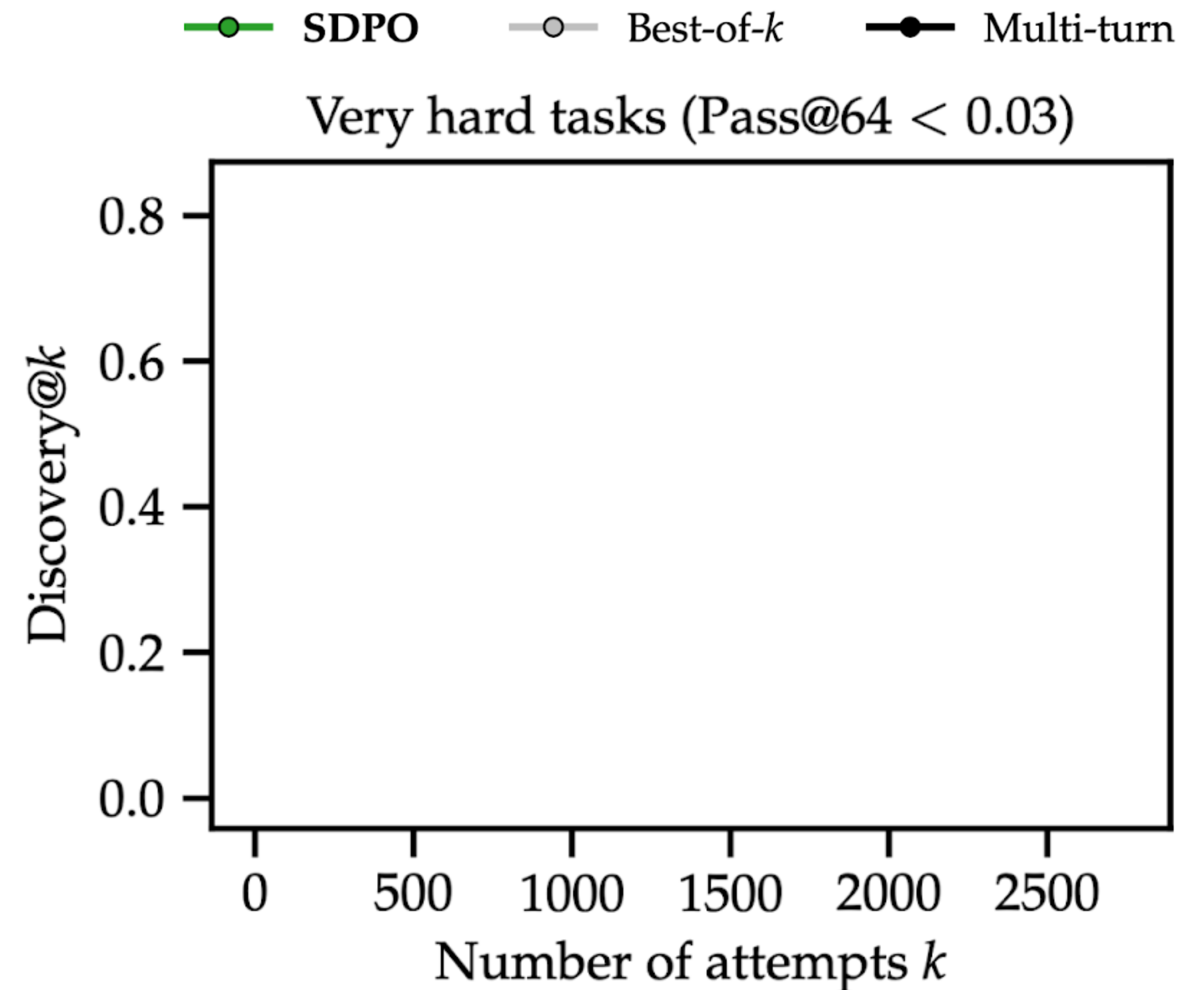


- Remember: Two ways of changing the model:  $\pi_{\theta}(\cdot | x, c)$ !



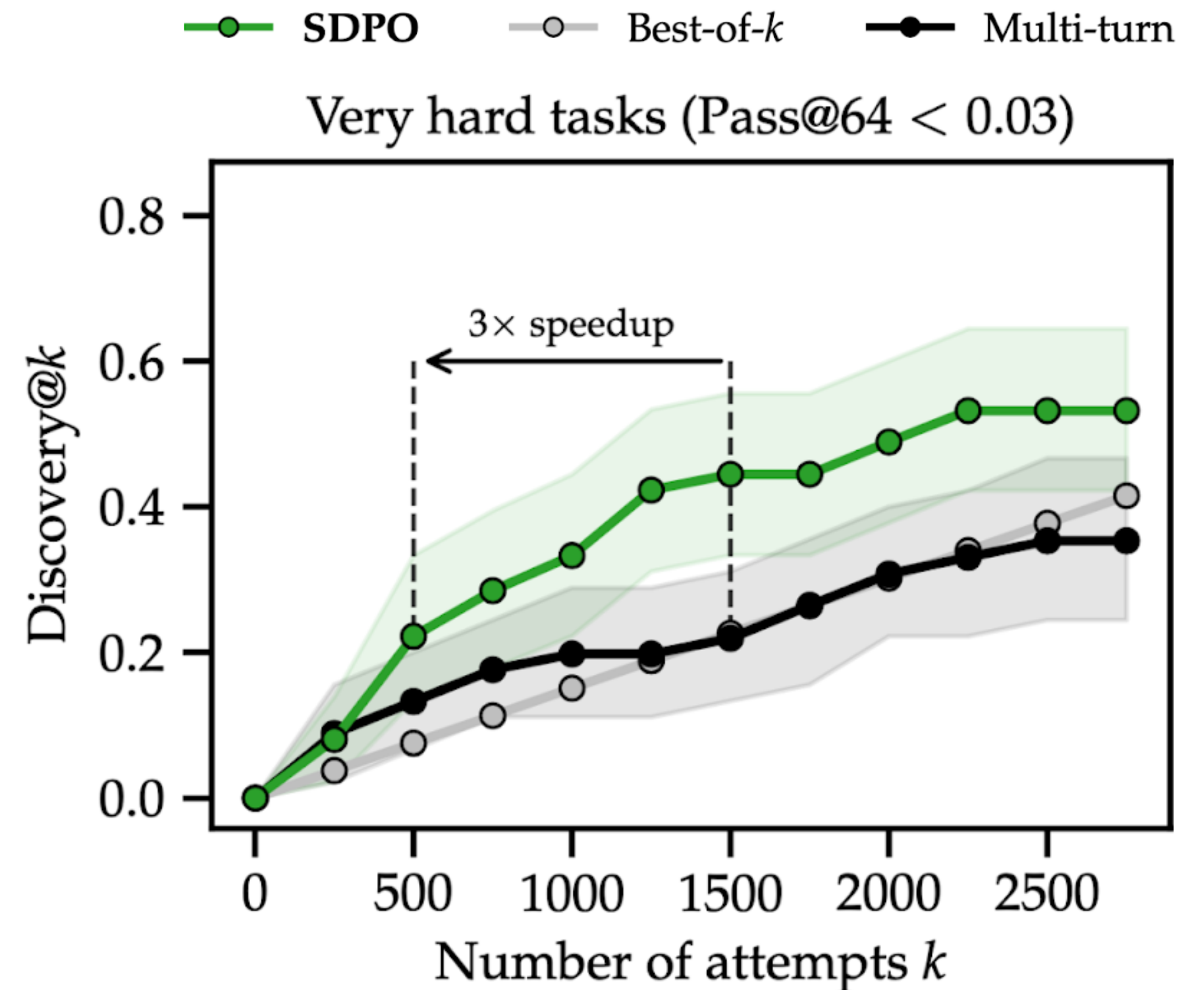
# Early results for discovery in hard problems

- **Goal:** Discovering solutions to hard coding tasks from LiveCodeBench
- **Discovery @  $k$**  = probability of solving task within  $k$  environment interactions
- Simplest baseline: Time until first success (**best-of- $k$** )



# Early results for discovery in hard problems

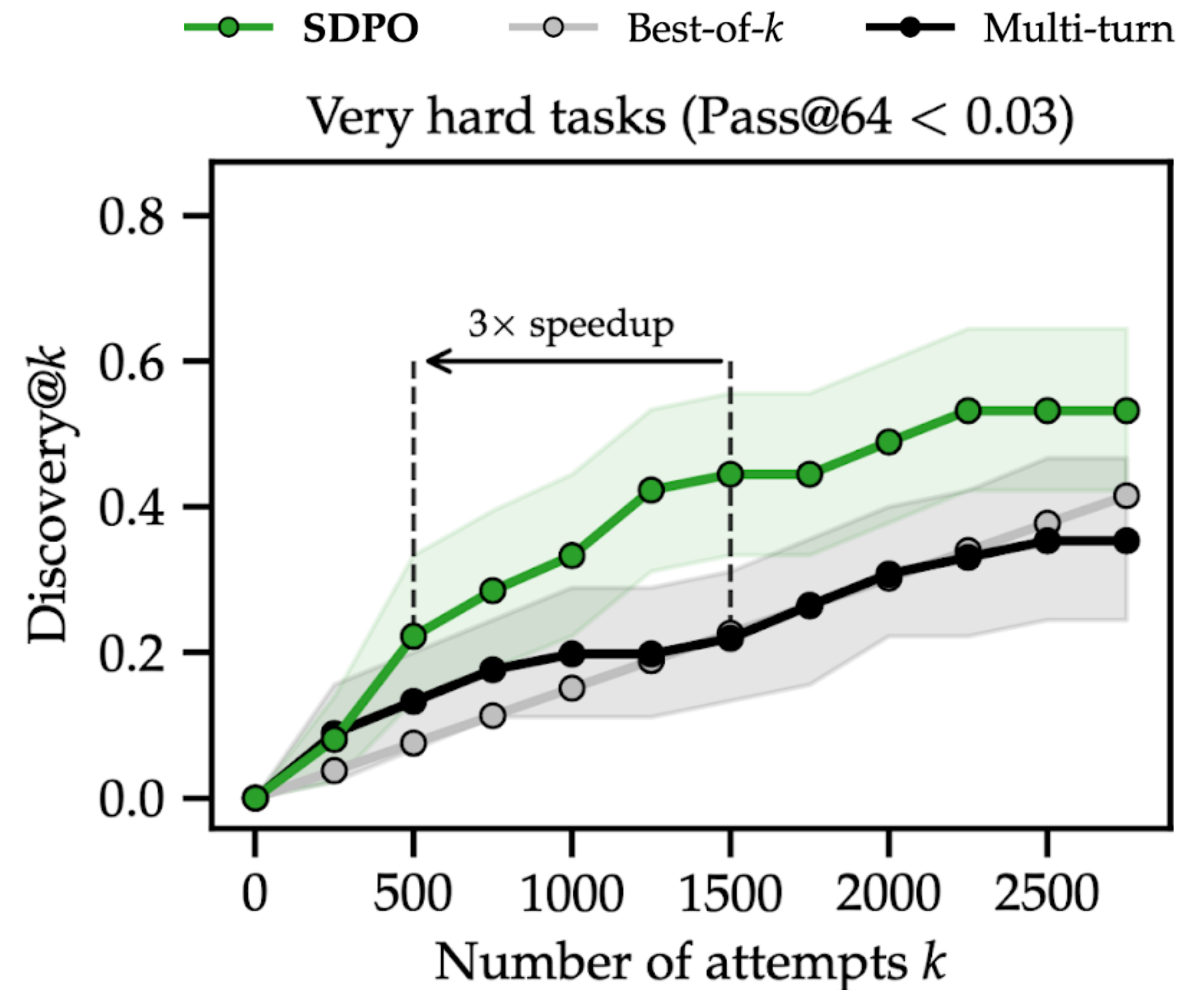
- **Goal:** Discovering solutions to hard coding tasks from LiveCodeBench
- **Discovery @  $k$**  = probability of solving task within  $k$  environment interactions
- Simplest baseline: Time until first success (**best-of- $k$** )



# Early results for discovery in hard problems

- **Goal:** Discovering solutions to hard coding tasks from LiveCodeBench
- Discovery @  $k$  = probability of solving task within  $k$  environment interactions
- Simplest baseline: Time until first success (**best-of- $k$** )

On-policy self-distillation can learn to solve hard tasks even before the task has ever been solved!



# Our research on self-distillation is already being used

Claude Code, but updating model weights



```
[> python -m continualcode

continualcode | Qwen/Qwen3-4B-Instruct-2507
/Users/sdan/Developer/continualcode
ready
> █
```



Student who uses OpenClaw to do homework,  
does not want to be found using AI.

**Before:**

The handbag cost \$220. Here's how:

1. Shoes cost \$80
2. 3 times the shoe cost =  $3 \times \$80 = \$240$
3. Subtract \$20 =  $\$240 - \$20 = **\$220**$

**\*\*Final Answer\*\*:** \$220

**AI-like !**

**After several conversations:**

Jillian's handbag cost \$220. Here's the step-by-step solution: 1. 3 times the shoe cost:  $3 * \$80 = \$240$ . 2. Subtract \$20:  $\$240 - \$20 = \$220$ . ✓

Teacher who uses OpenClaw to grade homework,  
wants comments to be specific and friendly.

**Before:**

Correct answer: 30. Your solution is correct.  
Total units: 120. Occupied units: 90.  
Unoccupied units: 30. **Cold !**

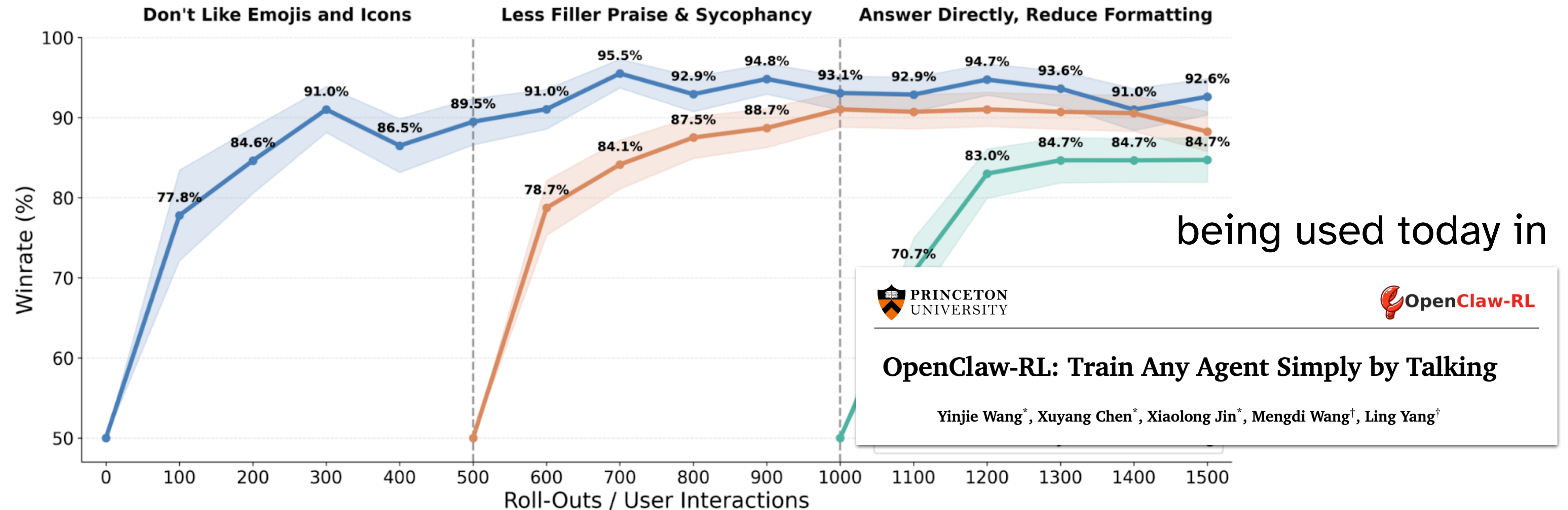
**After several conversations:**

Great job breaking down the problem! You correctly calculated 120 total units and found 90 occupied units. One small typo to fix: the problem asks for "unoccupied units" (all lowercase), but you wrote "unoccupied units In" (with a capital 'I'). Just remember to keep the words lowercase unless they're proper nouns! You've got the right answer (30) — well done! ✨ ✓

→ Your model learns as you interact with it

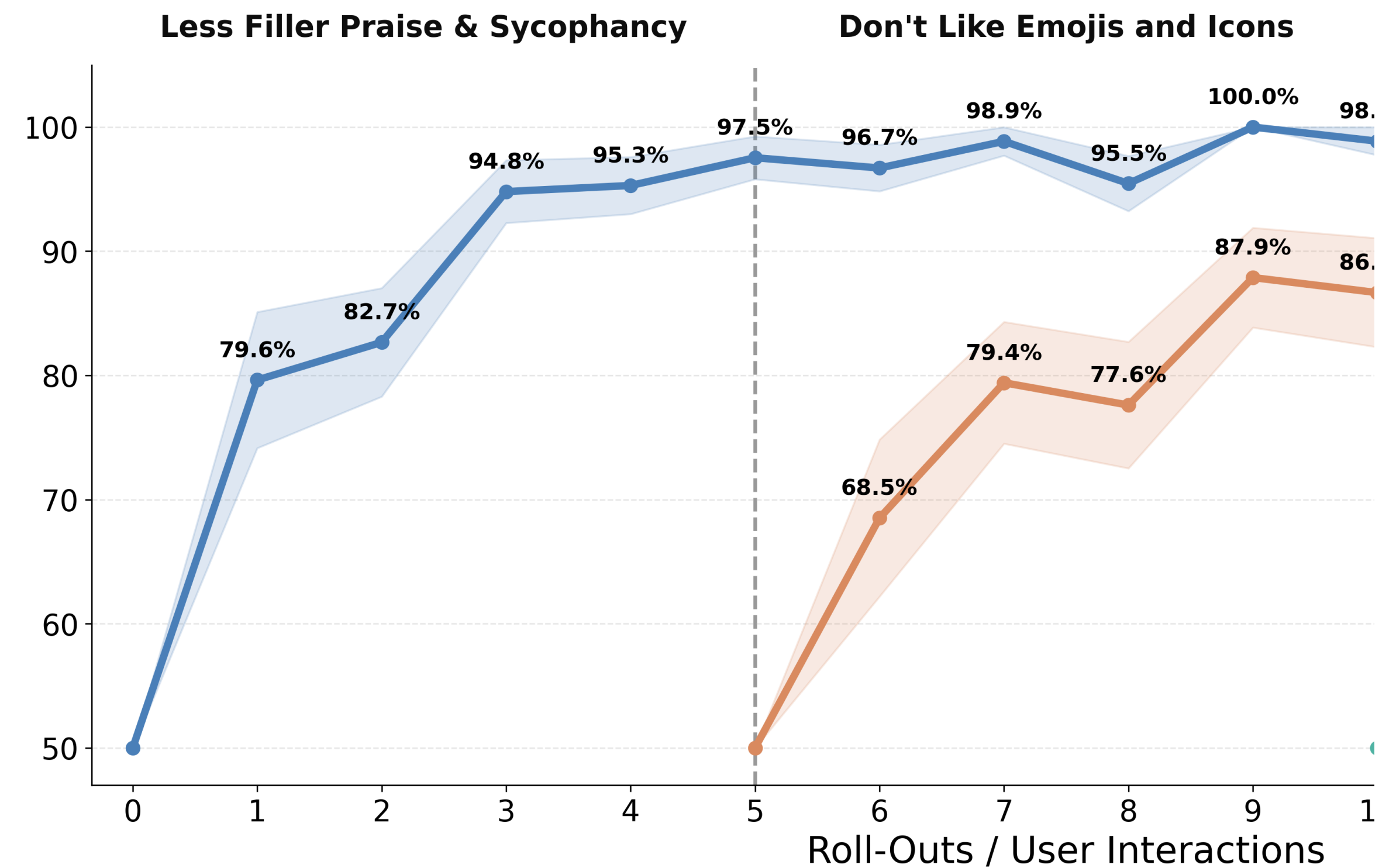
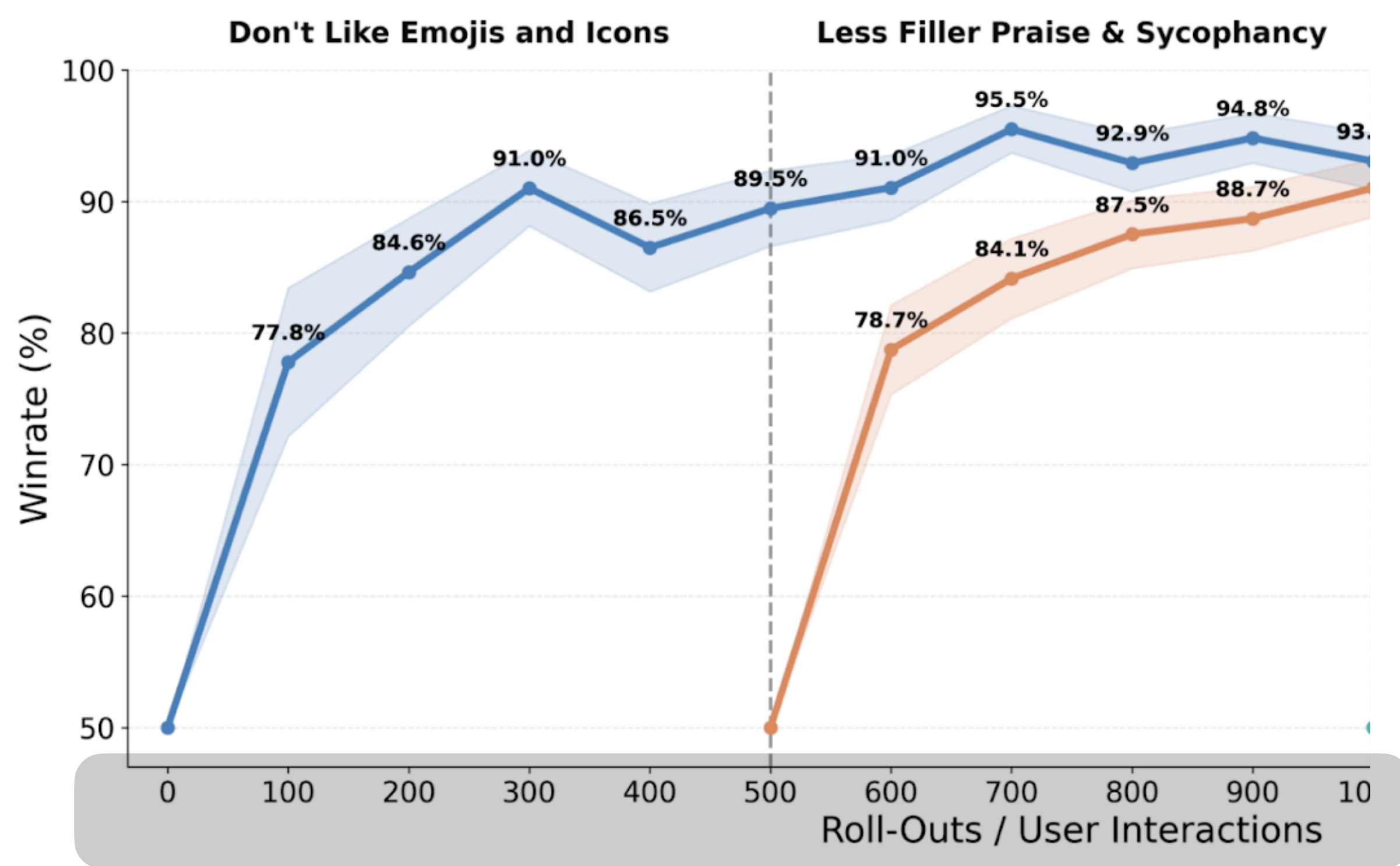
# Early results when “deploying in the wild”

**Goal:** Personalization towards a user with continually changing preferences



# Next frontier: Sample-efficiency

Can parametric learning be as sample-efficient as in-context learning?



## Reinforcement Learning via Self-Distillation

Jonas Hübotter<sup>1</sup> Frederike Lübeck<sup>\*,1,2</sup> Lejs Behric<sup>\*,1</sup> Anton Baumann<sup>\*,1</sup>  
Marco Bagatella<sup>1,2</sup> Daniel Marta<sup>1</sup> Ido Hakimi<sup>1</sup> Idan Shenfeld<sup>3</sup>  
Thomas Kleine Buening<sup>1</sup> Carlos Guestrin<sup>4</sup> Andreas Krause<sup>1</sup>  
<sup>1</sup>ETH Zurich <sup>2</sup>Max Planck Institute for Intelligent Systems <sup>3</sup>MIT <sup>4</sup>Stanford

Special thanks to



## SELF-DISTILLATION ENABLES CONTINUAL LEARNING

Idan Shenfeld<sup>1,2\*</sup> Mehul Damani<sup>1</sup> Jonas Hübotter<sup>3</sup> Pulkit Agrawal<sup>1,2</sup>  
<sup>1</sup>MIT <sup>2</sup>Improbable AI Lab <sup>3</sup>ETH Zurich

## Aligning Language Models from User Interactions

Thomas Kleine Buening<sup>1</sup> Jonas Hübotter<sup>1</sup> Barna Pásztor<sup>1</sup>  
Idan Shenfeld<sup>2</sup> Giorgia Ramponi<sup>3</sup> Andreas Krause<sup>1</sup>  
<sup>1</sup>ETH Zurich <sup>2</sup>MIT <sup>3</sup>University of Zurich

# Lots of other research interest too

## Reinforcement Learning via Self-Distillation

Jonas Hübottter<sup>1</sup> Frederike Lübeck<sup>\*,1,2</sup> Lejs Behric<sup>\*,1</sup> Anton Baumann<sup>\*,1</sup>  
Marco Bagatella<sup>1,2</sup> Daniel Marta<sup>1</sup> Ido Hakimi<sup>1</sup> Idan Shenfeld<sup>3</sup>  
Thomas Kleine Buening<sup>1</sup> Carlos Guestrin<sup>4</sup> Andreas Krause<sup>1</sup>  
<sup>1</sup>ETH Zurich <sup>2</sup>Max Planck Institute for Intelligent Systems <sup>3</sup>MIT <sup>4</sup>Stanford

<https://github.com/lasgroup/SDPO>

## Aligning Language Models from User Interactions

Thomas Kleine Buening<sup>1</sup> Jonas Hübottter<sup>1</sup> Barna Pásztor<sup>1</sup>  
Idan Shenfeld<sup>2</sup> Giorgia Ramponi<sup>3</sup> Andreas Krause<sup>1</sup>  
<sup>1</sup>ETH Zurich <sup>2</sup>MIT <sup>3</sup>University of Zurich

## SELF-DISTILLATION ENABLES CONTINUAL LEARNING

Idan Shenfeld<sup>1,2\*</sup> Mehul Damani<sup>1</sup> Jonas Hübottter<sup>3</sup> Pulkit Agrawal<sup>1,2</sup>  
<sup>1</sup>MIT <sup>2</sup>Improbable AI Lab <sup>3</sup>ETH Zurich

## Not only from us...

### On-Policy Context Distillation for Language Models

Tianzhu Ye\* Li Dong\*  
Xun Wu Shaohan Huang Furu Wei  
Microsoft Research  
<https://aka.ms/GeneralAI>

### Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models

Siyao Zhao<sup>†1</sup> Zhihui Xie<sup>2</sup> Mengchen Liu<sup>3</sup> Jing Huang<sup>3</sup> Guan Pang<sup>3</sup> Feiyu Chen<sup>\*,†3</sup> Aditya Grover<sup>\*1</sup>

### Experiential Reinforcement Learning

Taiwei Shi<sup>1\*</sup>, Sihao Chen<sup>2</sup>, Bowen Jiang<sup>3\*</sup>, Linxin Song<sup>1</sup>, Longqi Yang<sup>2</sup>, Jieyu Zhao<sup>1</sup>  
<sup>1</sup>University of Southern California, <sup>2</sup>Microsoft, <sup>3</sup>University of Pennsylvania  
{taiweish@usc.edu, sihaochen@microsoft.com}

### Privileged Information Distillation for Language Models

Emiliano Penalosa<sup>1,2,3</sup>, Dheeraj Vattikonda<sup>1,2,4</sup>, Nicolas Gontier<sup>1</sup>,  
Alexandre Lacoste<sup>1</sup>, Laurent Charlin<sup>2,5</sup>, Massimo Caccia<sup>1</sup>

<sup>1</sup>ServiceNow, <sup>2</sup>Mila Quebec, <sup>3</sup>Université de Montréal, <sup>4</sup>McGill University, <sup>5</sup>HEC Montréal

### Expanding the Capabilities of Reinforcement Learning via Text Feedback

Yuda Song<sup>\*,1</sup> Lili Chen<sup>\*,1</sup> Fahim Tajwar<sup>1</sup> Rémi Munos<sup>2</sup>  
Deepak Pathak<sup>1</sup> J. Andrew Bagnell<sup>1,3</sup> Aarti Singh<sup>1</sup> Andrea Zanette<sup>1</sup>  
<sup>1</sup>Carnegie Mellon University <sup>2</sup>Inria <sup>3</sup>Aurora Innovation  
{yudas,lilic}@andrew.cmu.edu

Also in TRL now!



# Thank you

Code: <https://github.com/lasgroup/SDPO>

More: <https://self-distillation.github.io>

# On-Policy **Self-Distillation** vs On-Policy Distillation

As training signal, we need a “better” next-token distribution than the original policy.

Two ways to obtain it:

- **Different weights:** Using a stronger teacher model → on-policy distillation (used for training frontier models: Qwen, GLM-5, ...)
- **Different context:** Using *same model* as teacher, but providing “better” context → on-policy self-distillation

The rest of the algorithm is identical!

**Self-distillation is much more flexible since it doesn't require a stronger model**