

Test-Time Self-Distillation

Jonas Hübötter, Frederike Lübeck*, Lejs Behric*, Anton Baumann*,
Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld,
Thomas Kleine Buening, Carlos Guestrin, Andreas Krause

ETH zürich

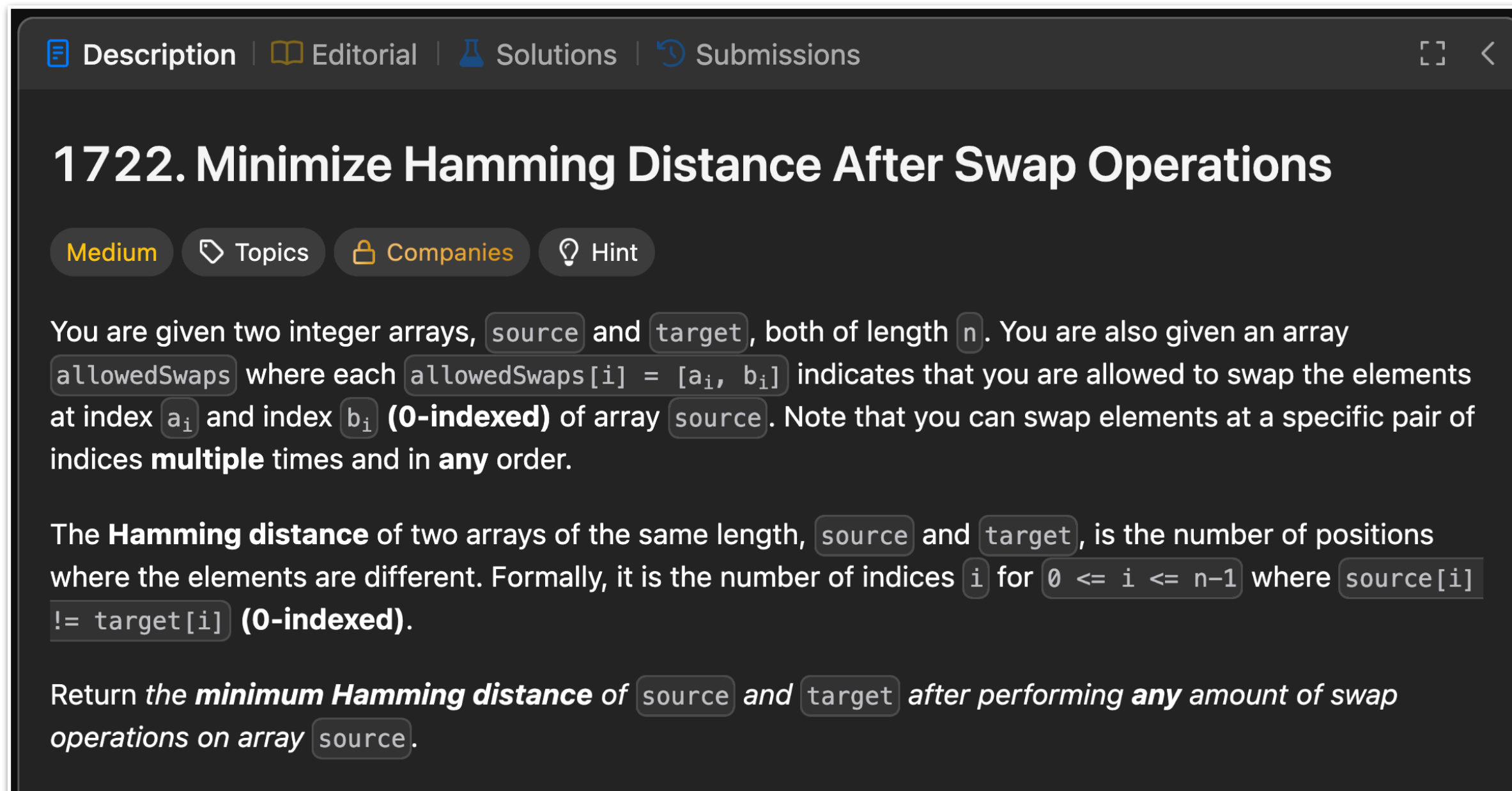
MAX PLANCK INSTITUTE
FOR INTELLIGENT SYSTEMS



Stanford
University

Learning to Solve Hard Problems

Given a hard problem at test-time, how quickly can a model learn to solve it?



The screenshot shows the LeetCode interface for problem 1722. At the top, there are navigation tabs: 'Description', 'Editorial', 'Solutions', and 'Submissions'. The problem title is '1722. Minimize Hamming Distance After Swap Operations'. Below the title, there are tags for 'Medium', 'Topics', 'Companies', and 'Hint'. The main text of the problem is as follows:

You are given two integer arrays, `source` and `target`, both of length `n`. You are also given an array `allowedSwaps` where each `allowedSwaps[i] = [ai, bi]` indicates that you are allowed to swap the elements at index `ai` and index `bi` (**0-indexed**) of array `source`. Note that you can swap elements at a specific pair of indices **multiple** times and in **any** order.

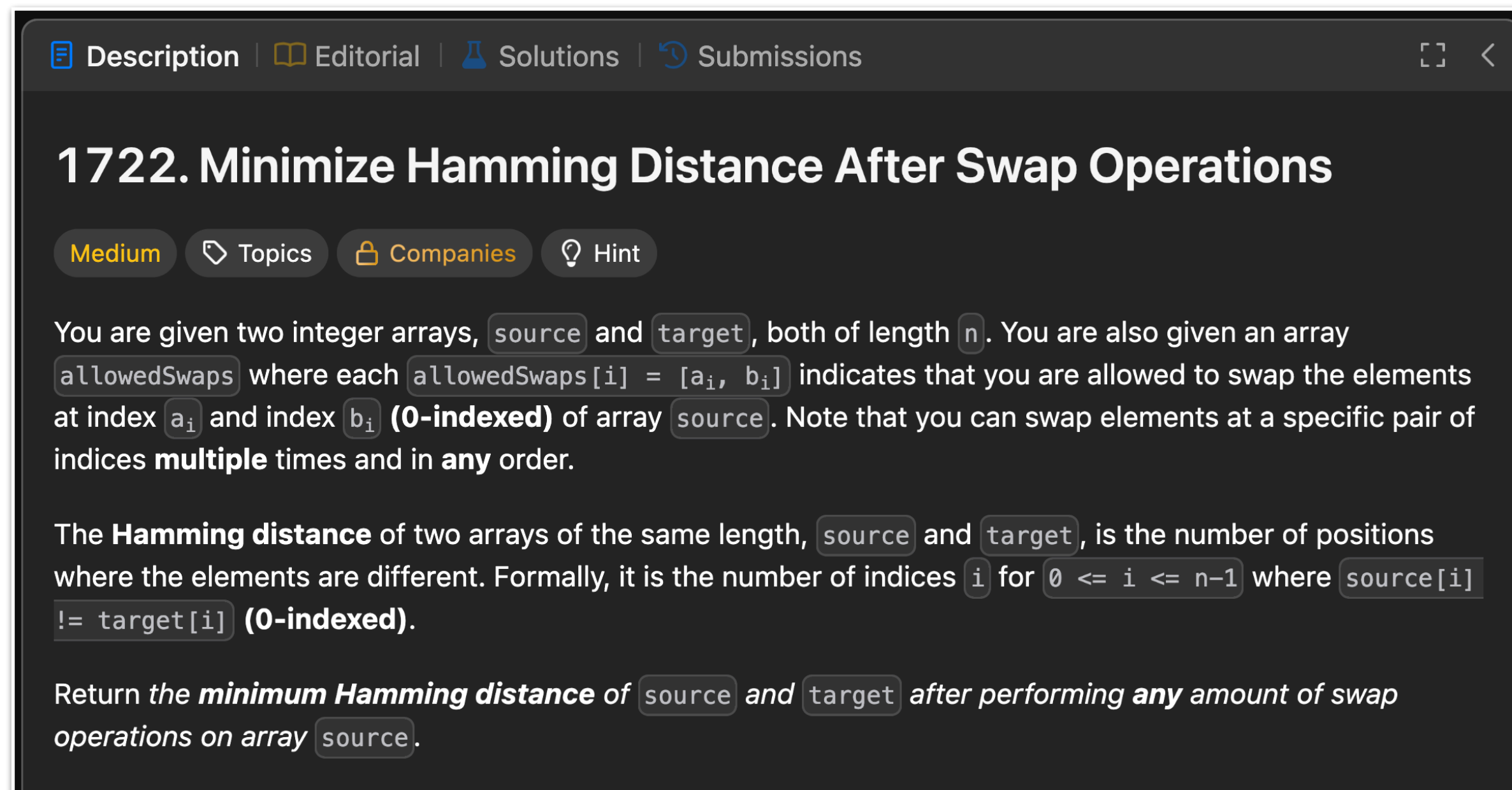
The **Hamming distance** of two arrays of the same length, `source` and `target`, is the number of positions where the elements are different. Formally, it is the number of indices `i` for `0 <= i <= n-1` where `source[i] != target[i]` (**0-indexed**).

Return the **minimum Hamming distance** of `source` and `target` after performing **any** amount of swap operations on array `source`.

Example: Leetcode

Learning to Solve Hard Problems

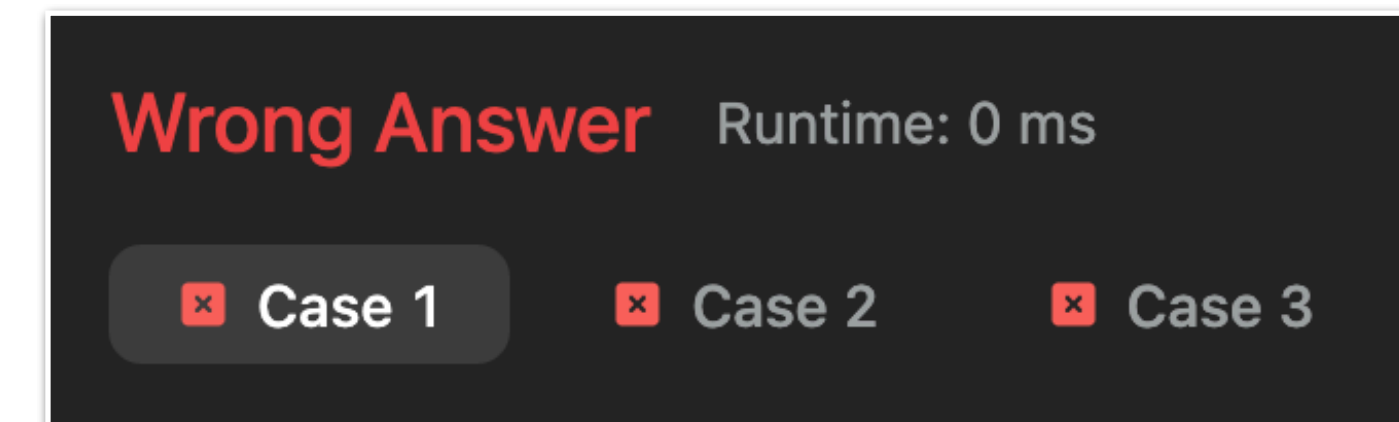
Given a hard problem at test-time, how quickly can a model learn to solve it?



The screenshot shows the LeetCode interface for problem 1722. At the top, there are navigation tabs: Description, Editorial, Solutions, and Submissions. The problem title is "1722. Minimize Hamming Distance After Swap Operations" with a "Medium" difficulty tag. Below the title are tags for "Topics", "Companies", and "Hint". The problem description states: "You are given two integer arrays, `source` and `target`, both of length `n`. You are also given an array `allowedSwaps` where each `allowedSwaps[i] = [a_i, b_i]` indicates that you are allowed to swap the elements at index `a_i` and index `b_i` (0-indexed) of array `source`. Note that you can swap elements at a specific pair of indices **multiple** times and in **any** order." The definition of Hamming distance is provided: "The **Hamming distance** of two arrays of the same length, `source` and `target`, is the number of positions where the elements are different. Formally, it is the number of indices `i` for `0 <= i <= n-1` where `source[i] != target[i]` (0-indexed)." The final instruction is: "Return the **minimum Hamming distance** of `source` and `target` after performing **any** amount of swap operations on array `source`."

Example: Leetcode

Many environments indicate *why* an attempt failed

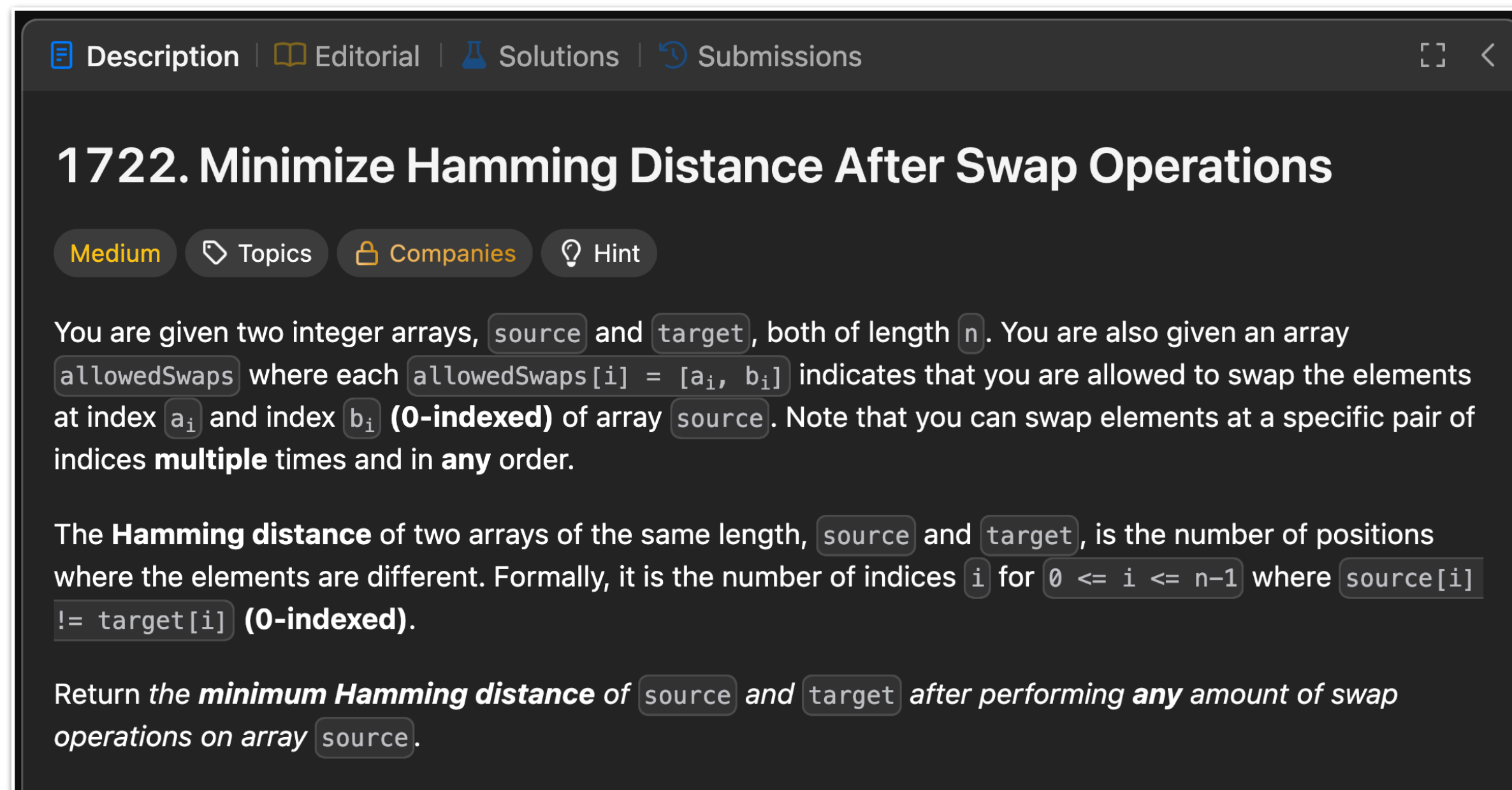


The screenshot shows a dark-themed notification box with the text "Wrong Answer" in red. To the right, it says "Runtime: 0 ms". Below this, there are three buttons, each with a red 'x' icon and the text "Case 1", "Case 2", and "Case 3" respectively, indicating which specific test cases failed.

Example: Unit tests

Learning to Solve Hard Problems

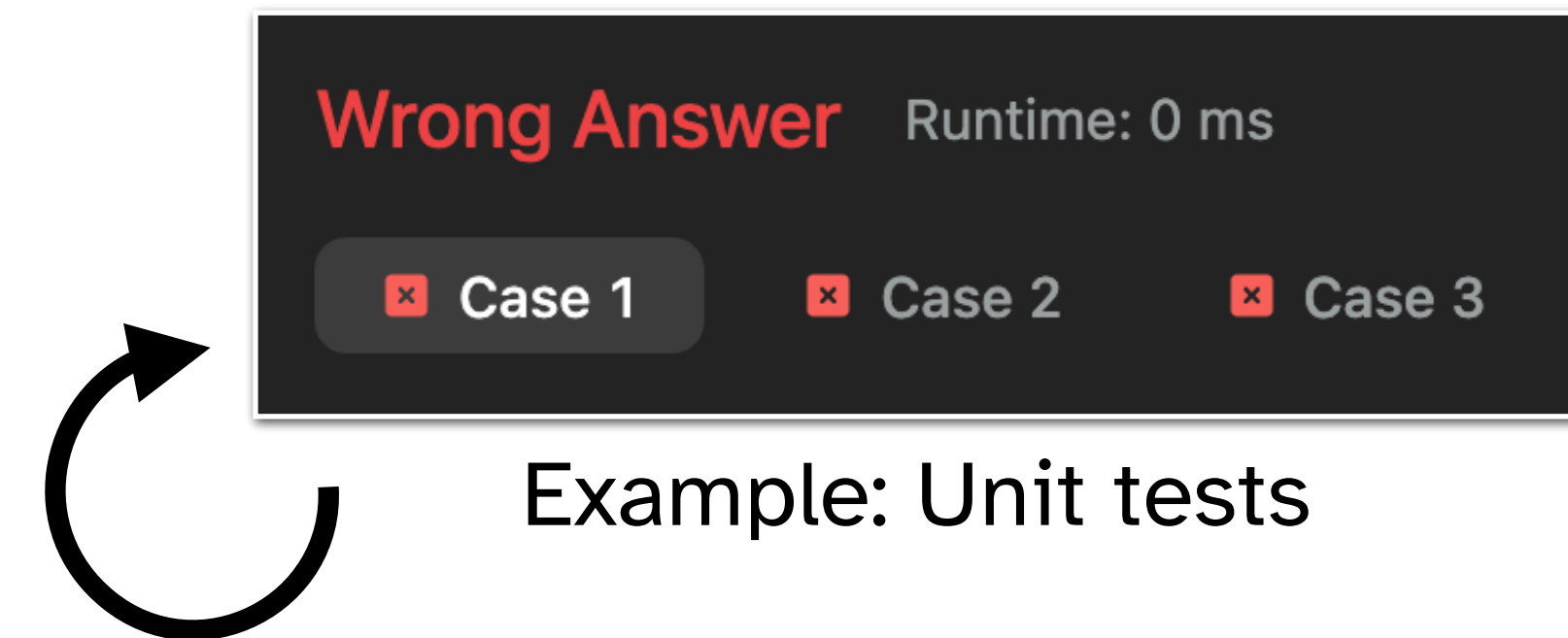
Given a hard problem at test-time, how quickly can a model learn to solve it?



The screenshot shows the LeetCode interface for problem 1722. At the top, there are navigation tabs: Description, Editorial, Solutions, and Submissions. The problem title is "1722. Minimize Hamming Distance After Swap Operations" with a "Medium" difficulty tag. Below the title are tags for "Topics", "Companies", and "Hint". The problem description states: "You are given two integer arrays, `source` and `target`, both of length `n`. You are also given an array `allowedSwaps` where each `allowedSwaps[i] = [a_i, b_i]` indicates that you are allowed to swap the elements at index `a_i` and index `b_i` (0-indexed) of array `source`. Note that you can swap elements at a specific pair of indices **multiple** times and in **any** order." The definition of Hamming distance is provided: "The **Hamming distance** of two arrays of the same length, `source` and `target`, is the number of positions where the elements are different. Formally, it is the number of indices `i` for `0 <= i <= n-1` where `source[i] != target[i]` (0-indexed)." The final instruction is: "Return the **minimum Hamming distance** of `source` and `target` after performing **any** amount of swap operations on array `source`."

Example: Leetcode

Many environments indicate *why* an attempt failed

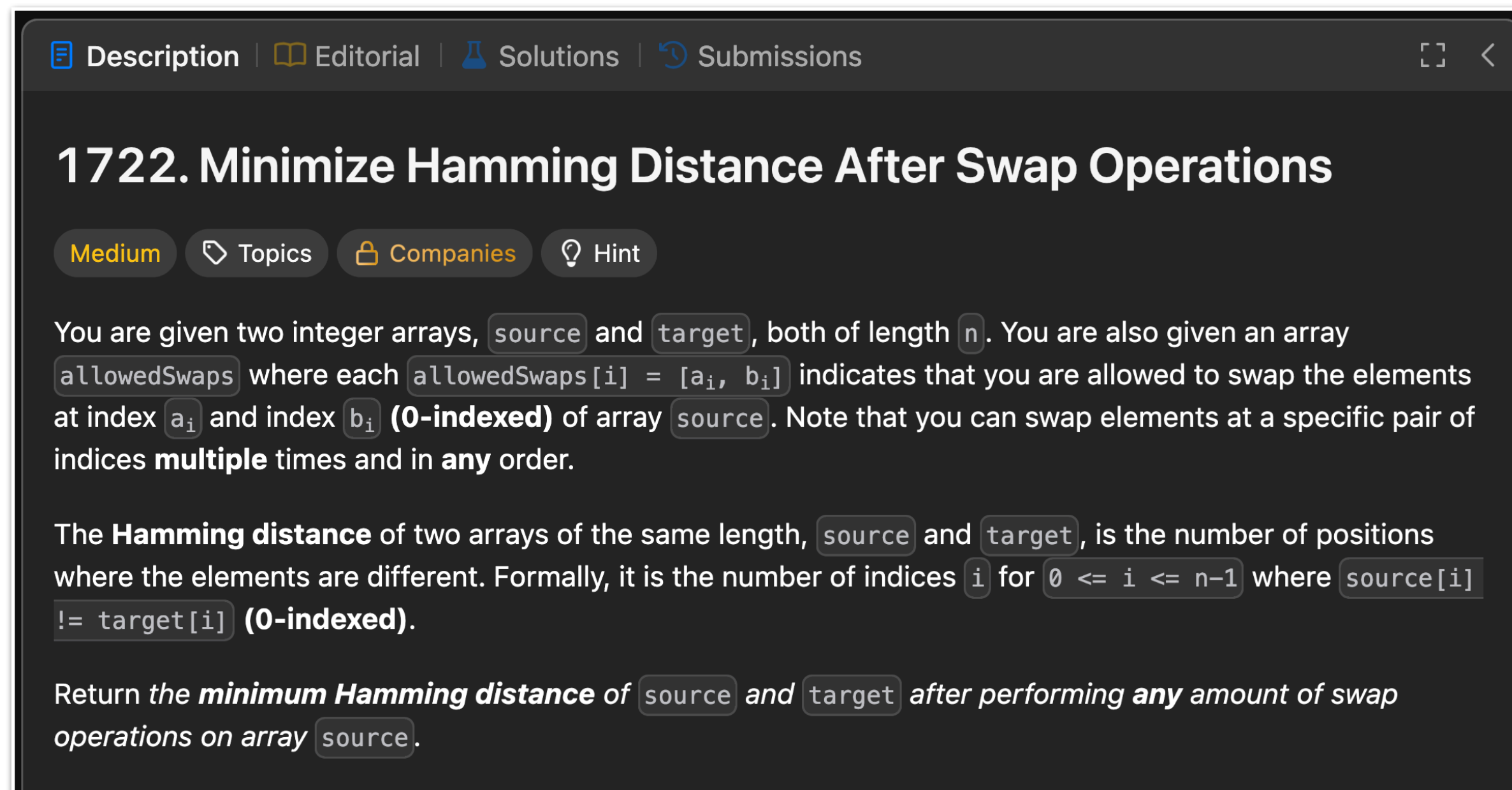


The screenshot shows a "Wrong Answer" message in red text, with "Runtime: 0 ms" to its right. Below the message are three buttons, each with a red 'x' icon and the text "Case 1", "Case 2", and "Case 3" respectively, indicating that the solution failed on all three test cases. A curved arrow points from the "Wrong Answer" message towards the "Example: Unit tests" text below.

Example: Unit tests

Learning to Solve Hard Problems

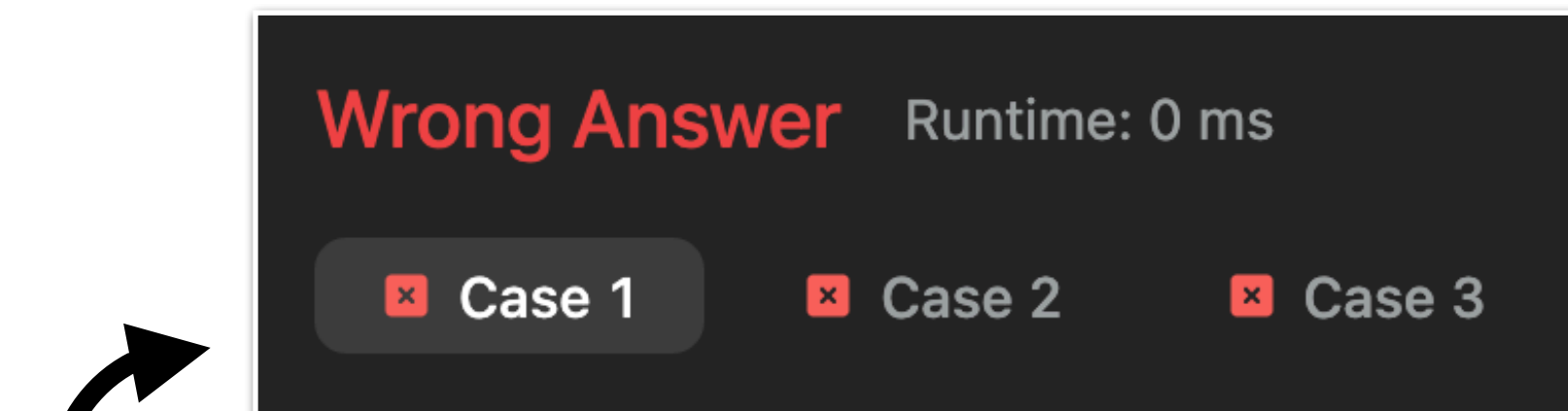
Given a hard problem at test-time, how quickly can a model learn to solve it?



The screenshot shows the LeetCode interface for problem 1722. At the top, there are navigation tabs: Description, Editorial, Solutions, and Submissions. The problem title is "1722. Minimize Hamming Distance After Swap Operations" with a "Medium" difficulty tag. The description states: "You are given two integer arrays, source and target, both of length n. You are also given an array allowedSwaps where each allowedSwaps[i] = [a_i, b_i] indicates that you are allowed to swap the elements at index a_i and index b_i (0-indexed) of array source. Note that you can swap elements at a specific pair of indices multiple times and in any order." It then defines the Hamming distance and asks for the minimum Hamming distance after any amount of swaps.

Example: Leetcode

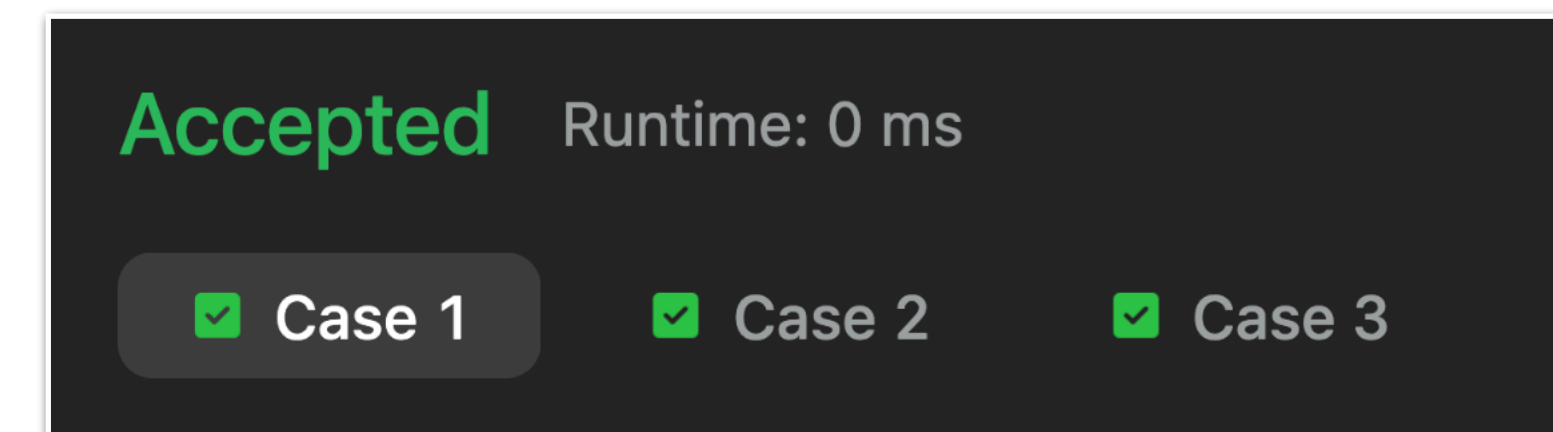
Many environments indicate *why* an attempt failed



This screenshot shows a "Wrong Answer" status in red text, with "Runtime: 0 ms" in grey. Below it, three test cases are listed, each with a red 'x' icon: "Case 1", "Case 2", and "Case 3".

Example: Unit tests

until...



This screenshot shows an "Accepted" status in green text, with "Runtime: 0 ms" in grey. Below it, three test cases are listed, each with a green checkmark icon: "Case 1", "Case 2", and "Case 3".

Recent work accelerates discovery for dense reward problems

Google DeepMind

AlphaEvolve: A coding agent for scientific and algorithmic discovery

Alexander Novikov*, Ngan Vū*, Marvin Eisenberger*, Emilien Dupont*, Po-Sen Huang*, Adam Zsolt Wagner*, Sergey Shirobokov*, Borislav Kozlovskii*, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abiga See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli and Matej Balog
Google DeepMind¹

Learning to Discover at Test Time

Mert Yuksekogul^{*1}, Daniel Kocaja^{*1}, Xinhao Li^{*4}, Federico Bianchi^{*5}
Jed McCaleb³, Xiaolong Wang⁴, Jan Kautz², Yejin Choi², James Zou^{+1,5}, Carlos Guestrin⁺¹, Yu Sun^{*1,2}
¹ Stanford University ² NVIDIA ³ Astera Institute ⁴ UC San Diego ⁵ Together AI

This work: How can we accelerate discovery in sparse reward problems?

Question:

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

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

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

incorrect!

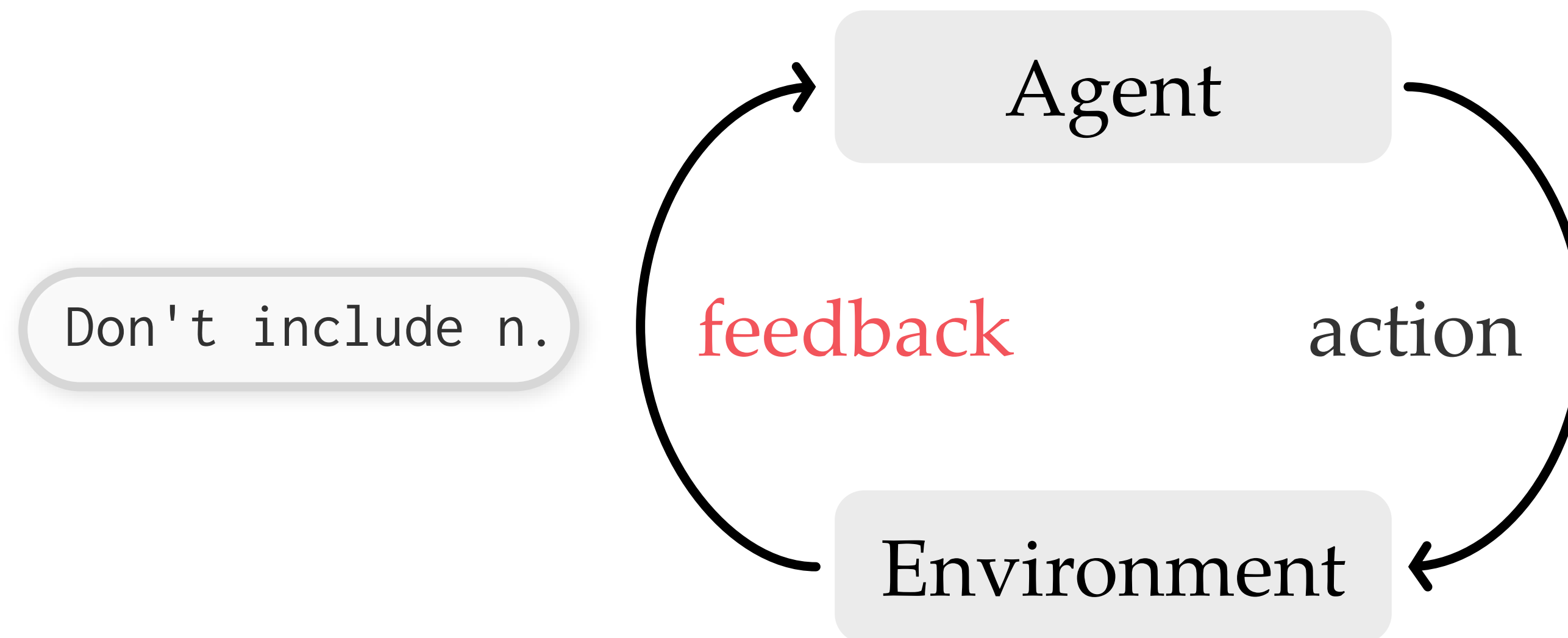
Question:

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

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

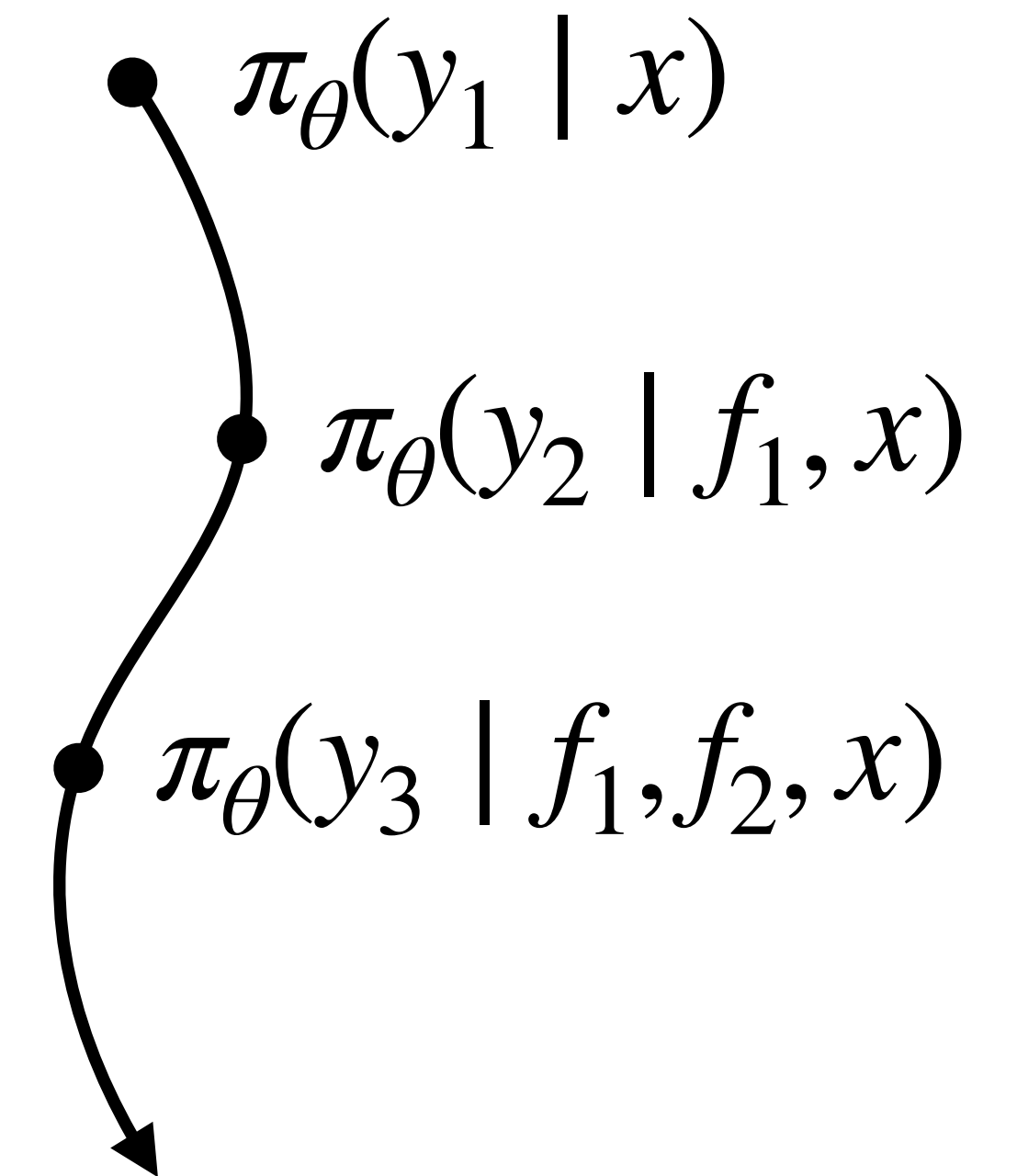
```
```python
def numbers_up_to_n(n):
 return list(range(1, n + 1))
```
```

incorrect!



In-context learning

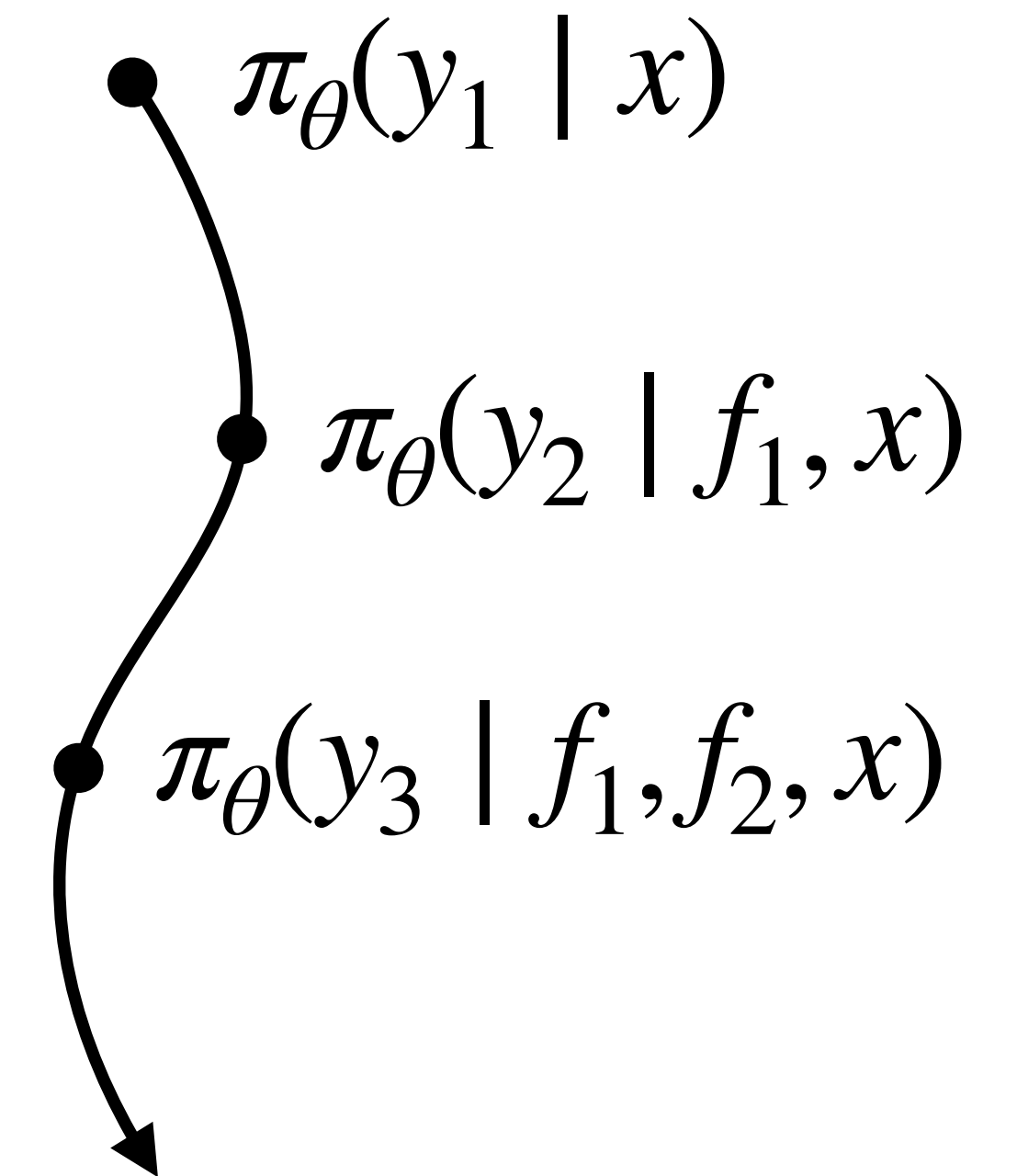
- Enables models to learn from rich feedback, hints, examples, and more.



Prompt x , Responses y , Feedback f

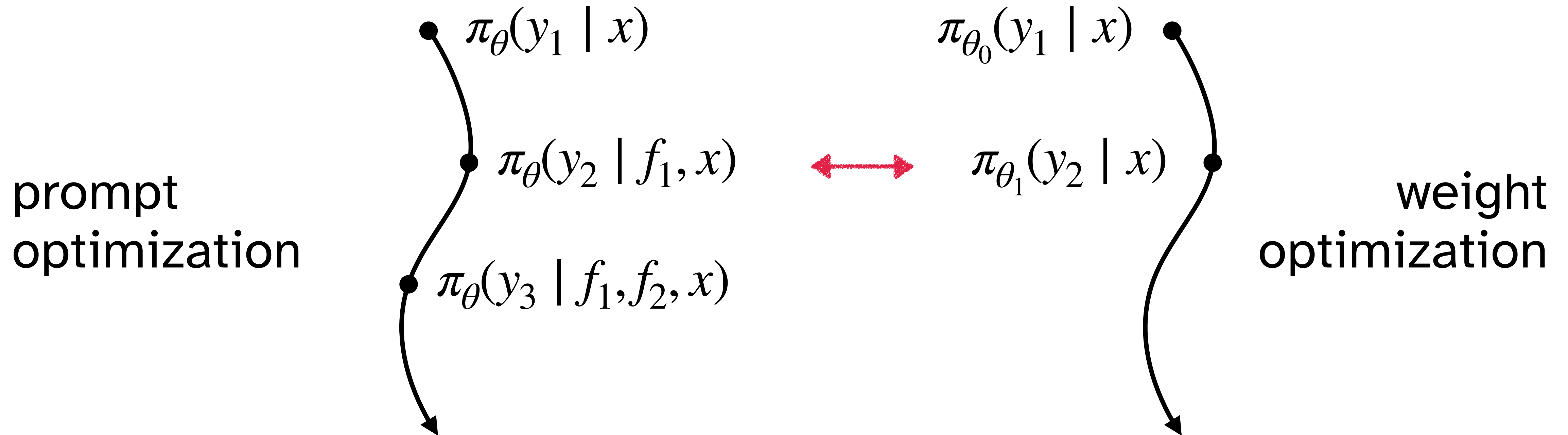
In-context learning

- Enables models to learn from rich feedback, hints, examples, and more.
- **BUT** context grows and learning is transient.



Prompt x , Responses y , Feedback f

In-context learning → Self-distillation



Core idea: Use in-context learning to turn the model into its own teacher, then distill it.

Self-distillation

Prompt x , Response y , Feedback f

Input prompt x

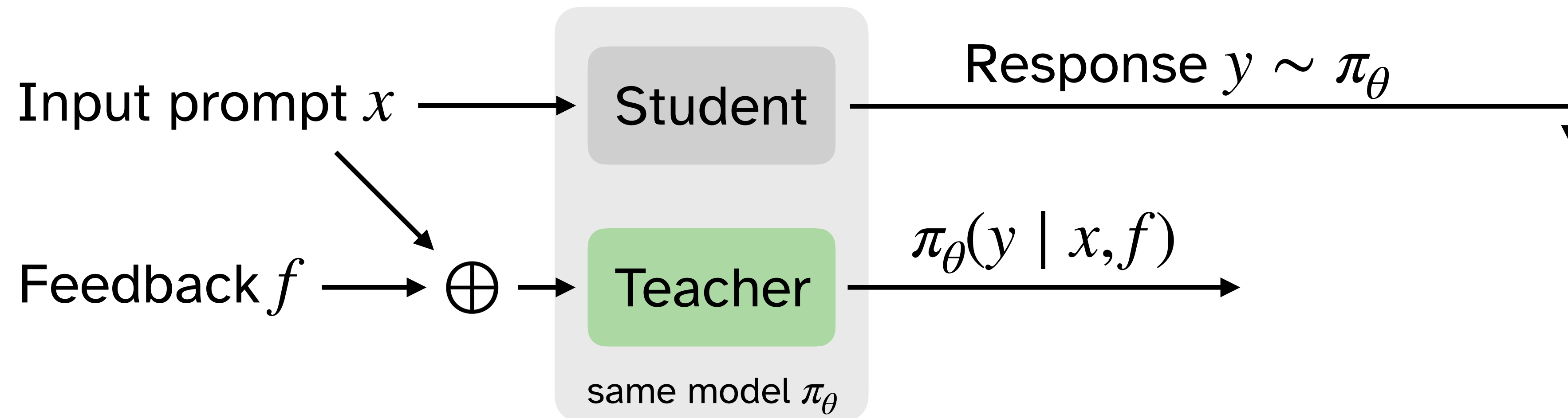
Self-distillation

Prompt x , Response y , Feedback f



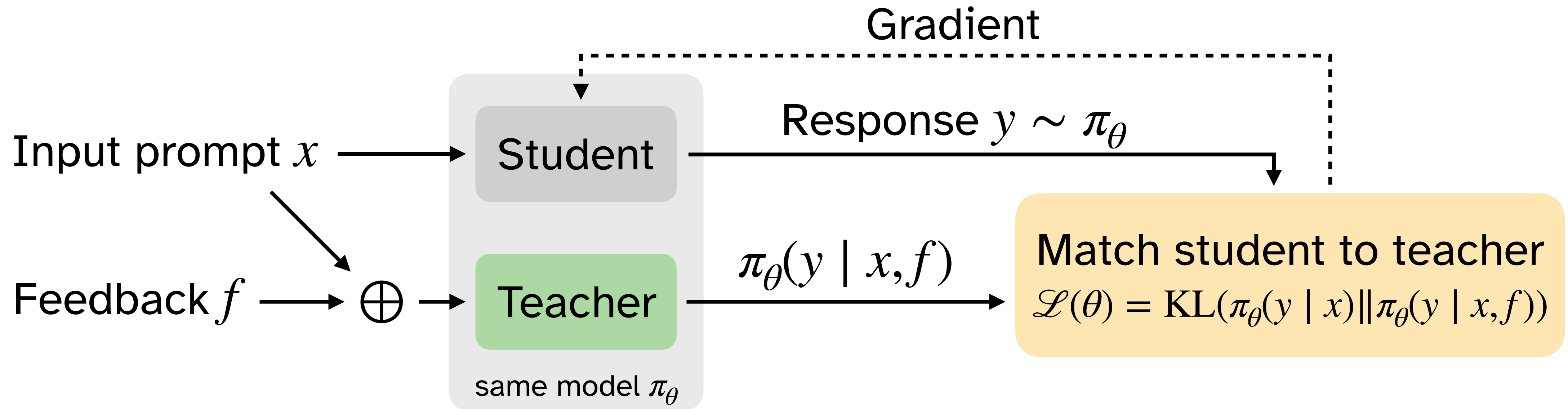
Self-distillation

Prompt x , Response y , Feedback f



Self-distillation

Prompt x , Response y , Feedback f



Two equivalent perspectives

Match student to teacher
 $\mathcal{L}(\theta) = \text{KL}(\pi_\theta(y | x) \| \pi_\theta(y | x, f))$



Token-level advantages
 $A_i(x, y, f) := \log \frac{\pi_\theta(y_i | x, f, y_{<i})}{\pi_\theta(y_i | x, y_{<i})}$

On-policy distillation

On-policy RL

Credit assignment in self-distillation

Token-level advantages

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

Question:

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

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

## SDPO

Generated tokens →

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

# Credit assignment in self-distillation

Token-level advantages

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

Question:

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

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

## SDPO

Generated tokens →

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

Vocabulary ↓

+  
+  
))\n

⋮

# How to measure improvement?

- **Goal:** Accelerate time-to-discovery
- **Discovery time** is the number of trials until a solution is found
- We want to increase the probability of discovery:

$$\text{discovery}@k := \mathbb{P}(\text{discovery time} \leq k)$$

# How to measure improvement?

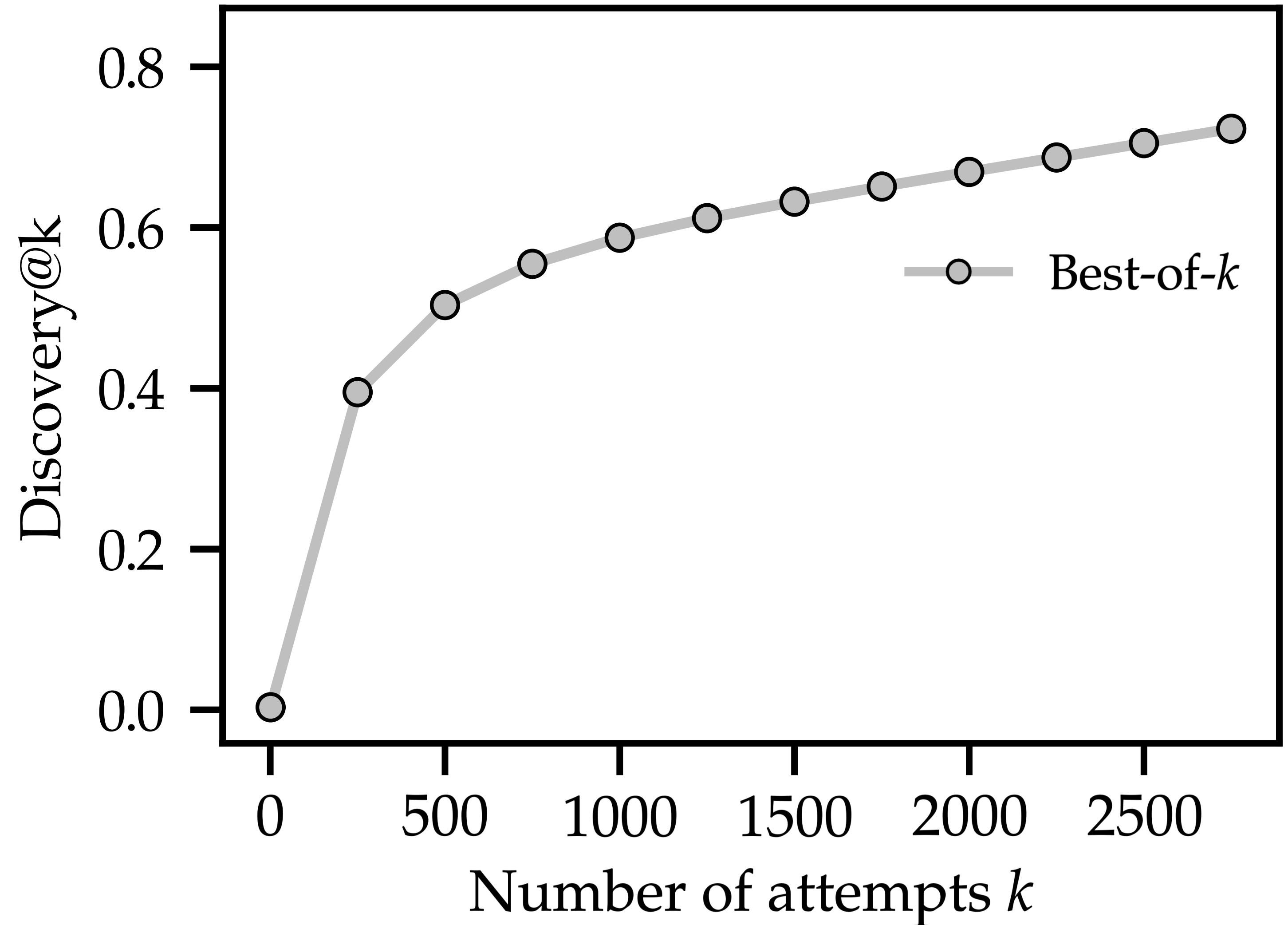
- **Goal:** Accelerate time-to-discovery
- **Discovery time** is the number of trials until a solution is found
- We want to increase the probability of discovery:

$$\text{discovery}@k := \mathbb{P}(\text{discovery time} \leq k)$$

*Note: pass@k = discovery@k for best-of-k sampling*

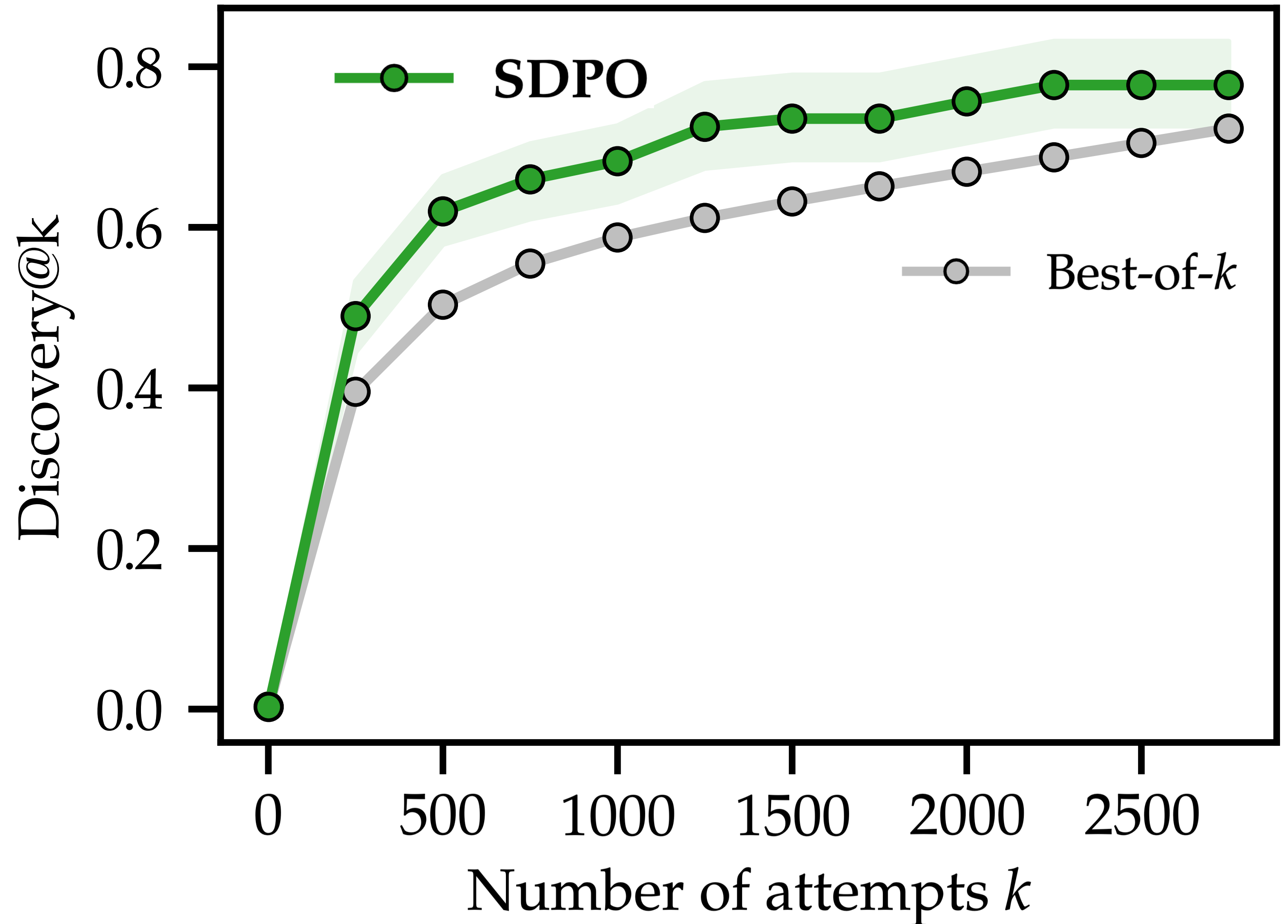
# Results

- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*:  $\text{pass}@64 < 0.5$



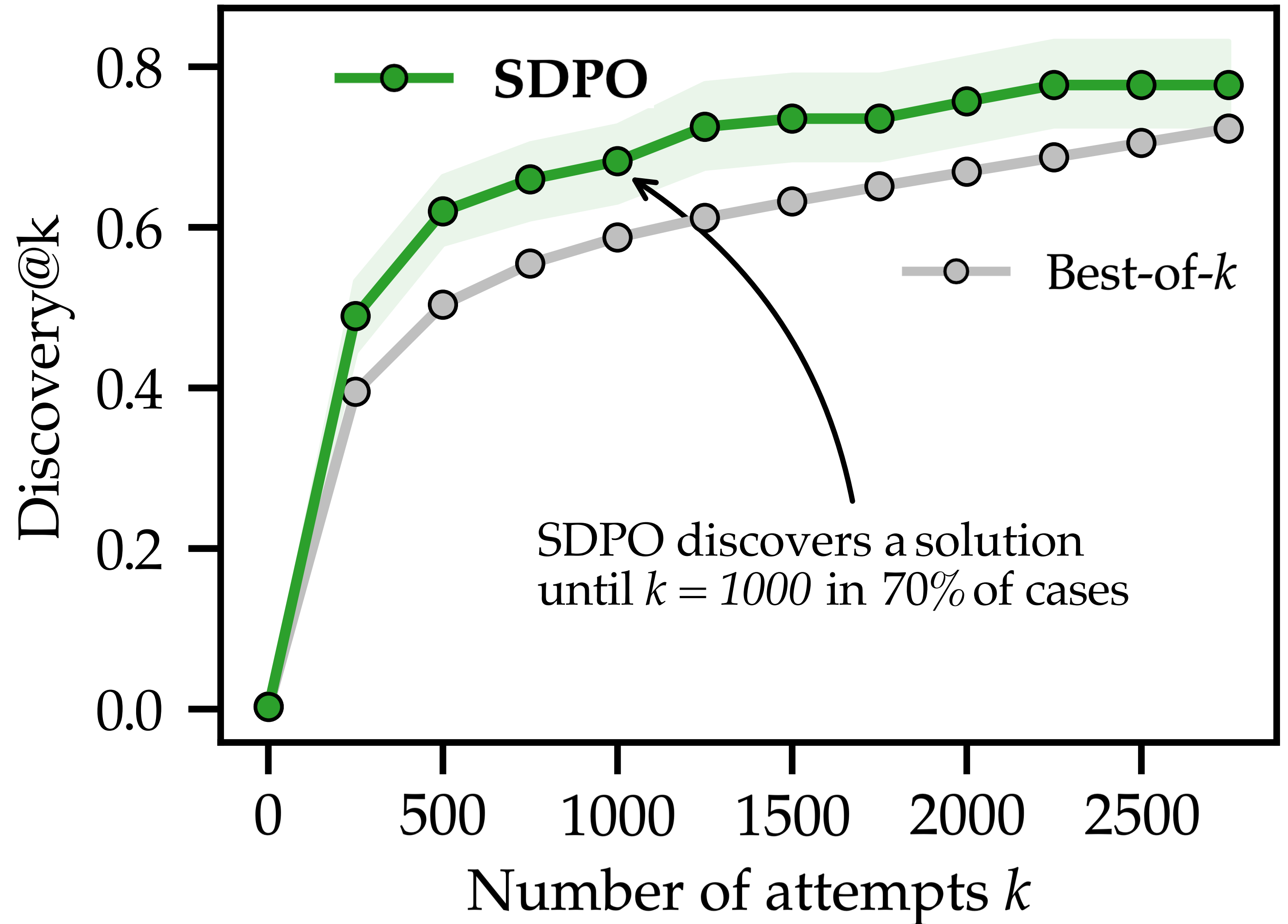
# Results

- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*:  $\text{pass}@64 < 0.5$



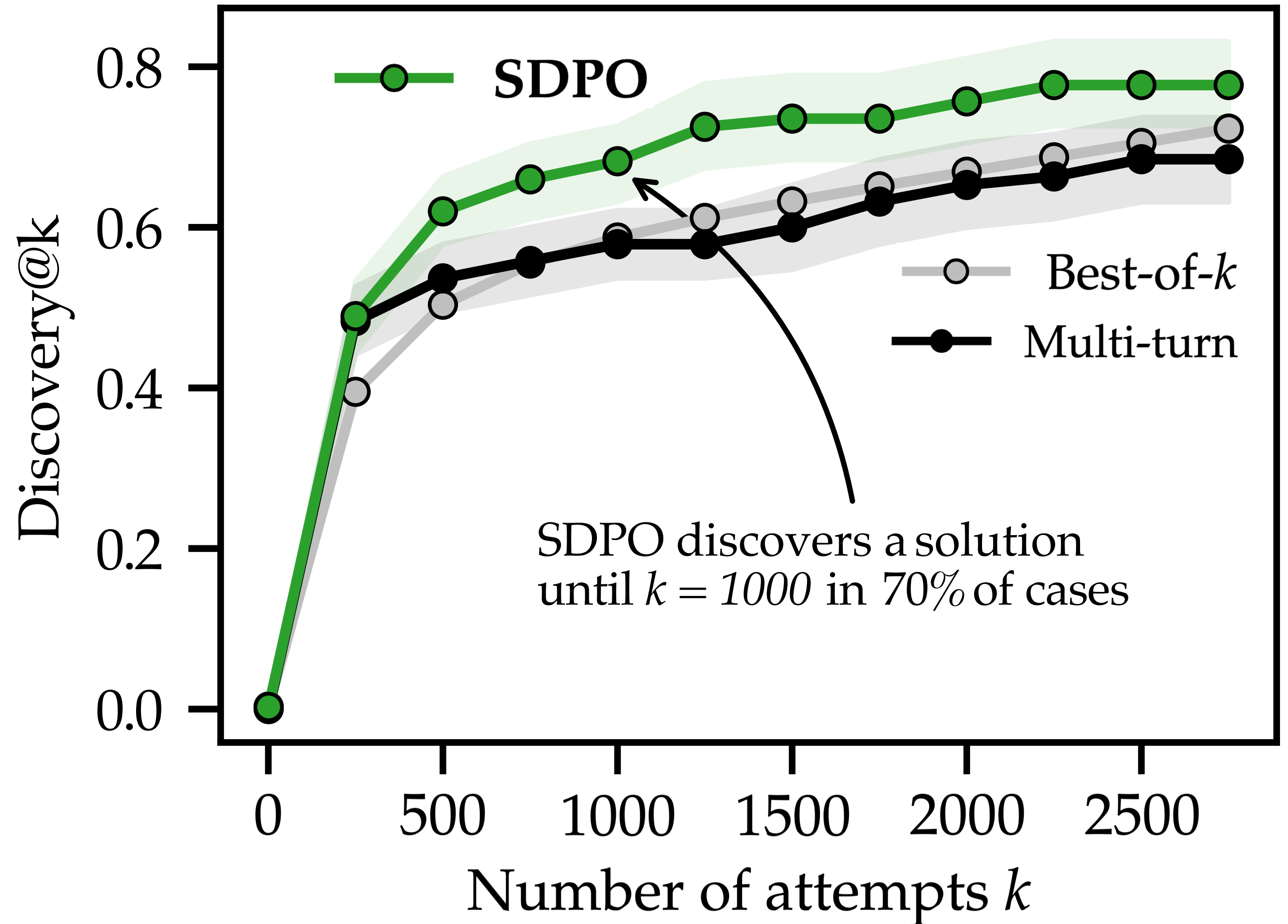
# Results

- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*:  $\text{pass}@64 < 0.5$

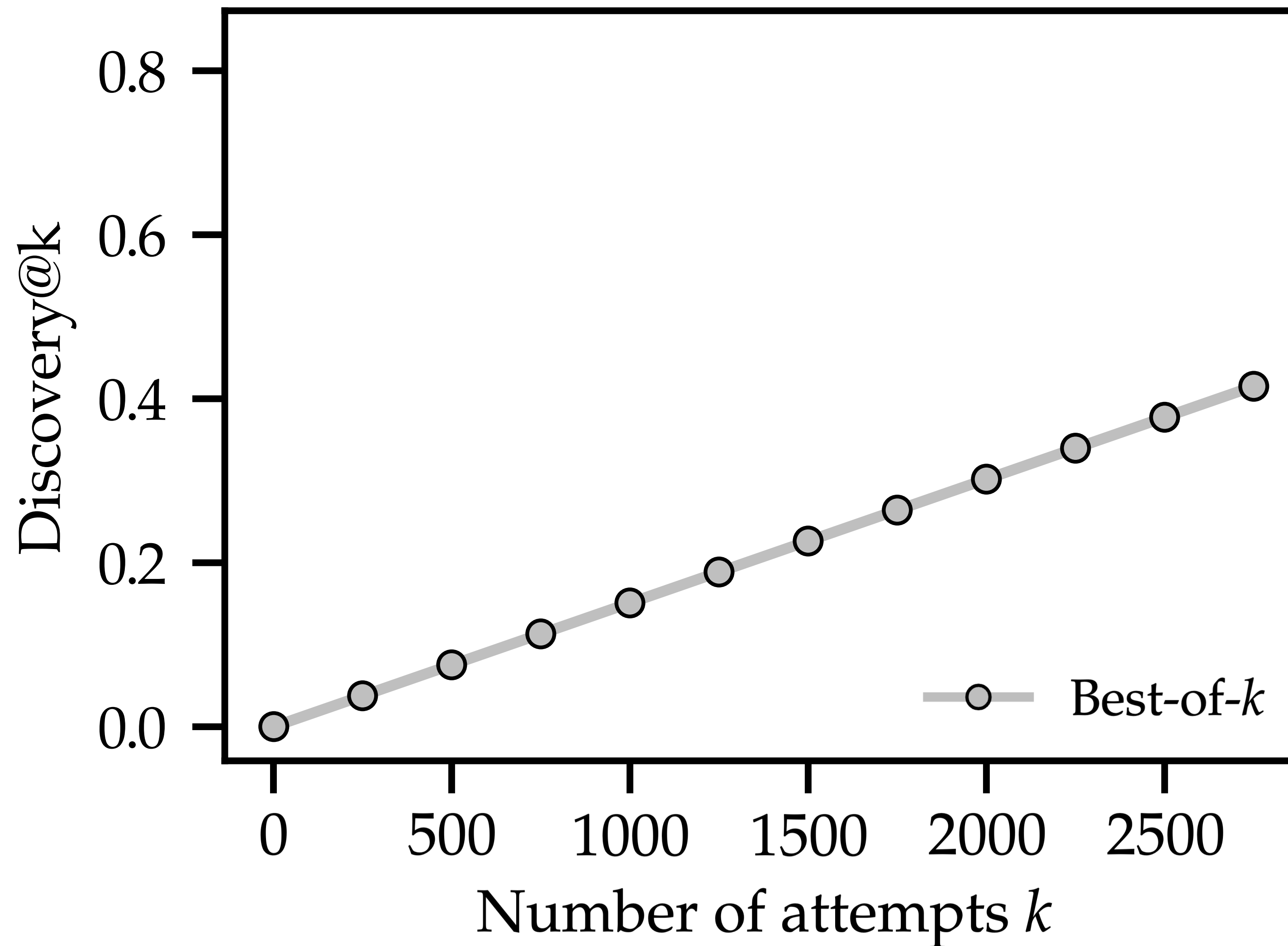


# Results

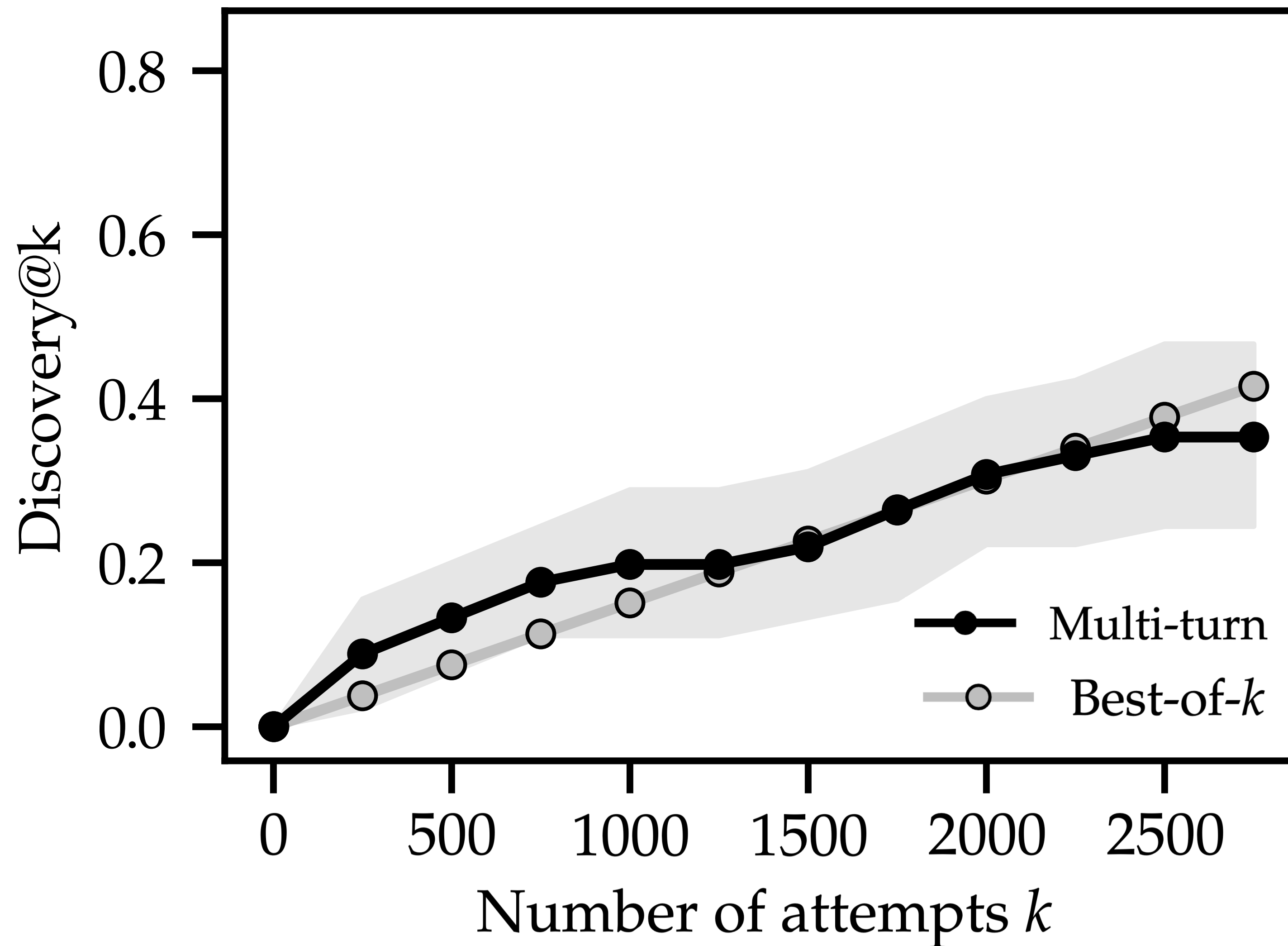
- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*:  $\text{pass}@64 < 0.5$



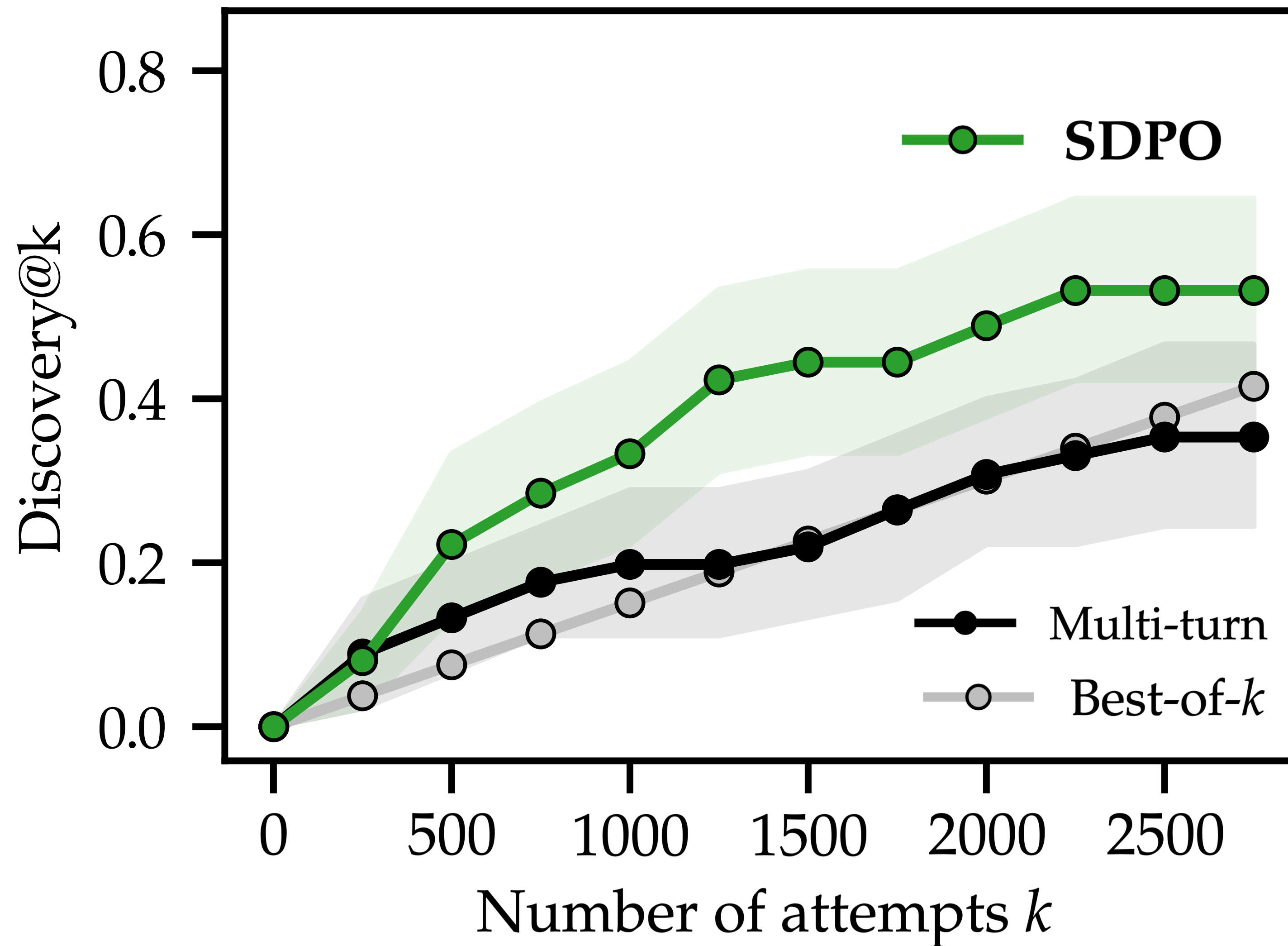
# Results on very hard tasks ( $\text{pass@64} < 0.03$ )



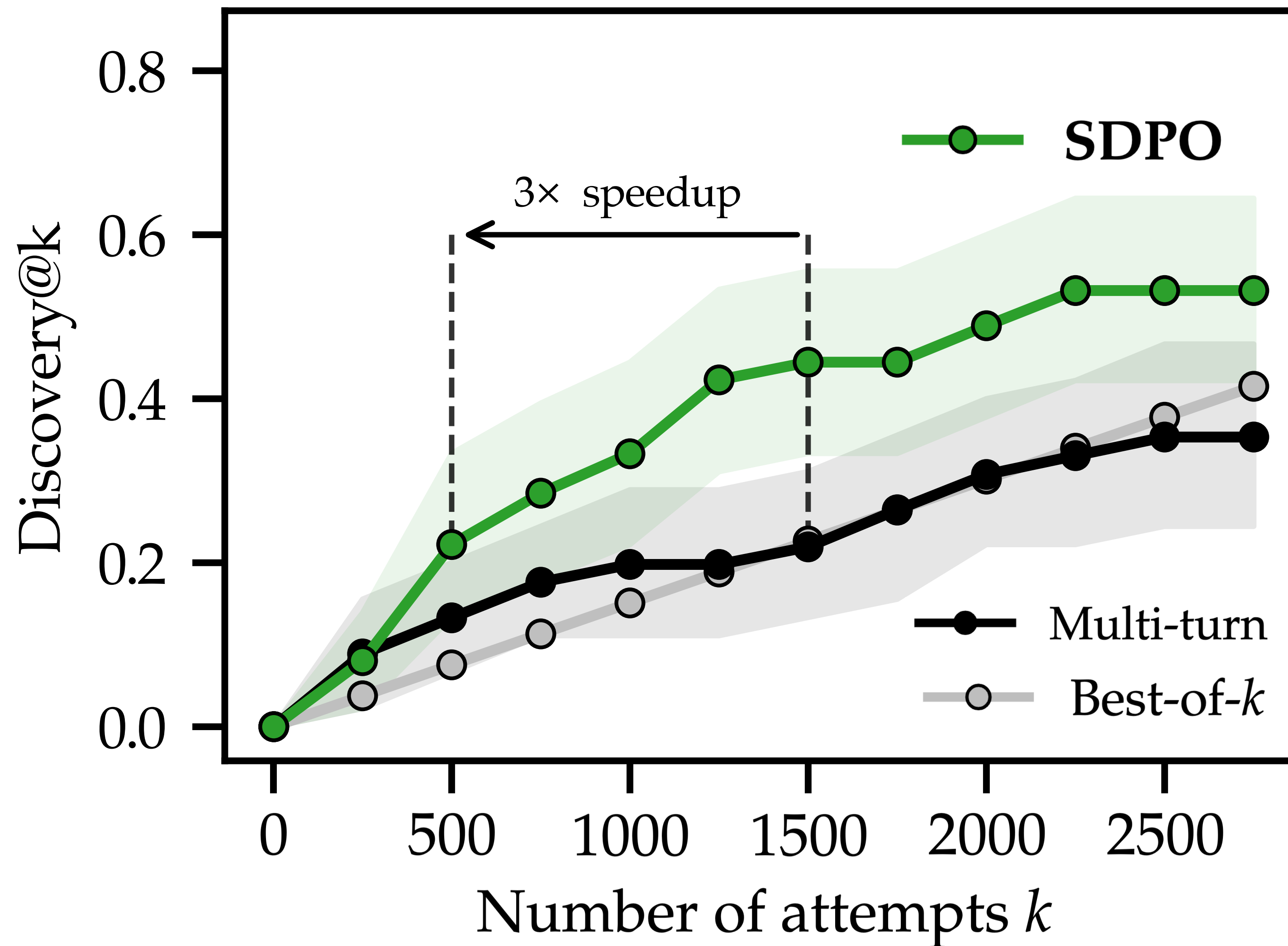
# Results on very hard tasks ( $\text{pass}@64 < 0.03$ )



# Results on very hard tasks ( $\text{pass@64} < 0.03$ )



# Results on very hard tasks ( $\text{pass}@64 < 0.03$ )



# Takeaways

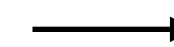
- Self-distillation solves problems not solved by the base model.
- The initial teacher does not yet solve questions: **Teacher feedback is directionally helpful.**
- Multi-turn baseline runs out of context (40k tokens) at k=800 to k=1000 tokens, but underperforms at k=500: **Multi-turn underperforms even prior to running out of context.**

# 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

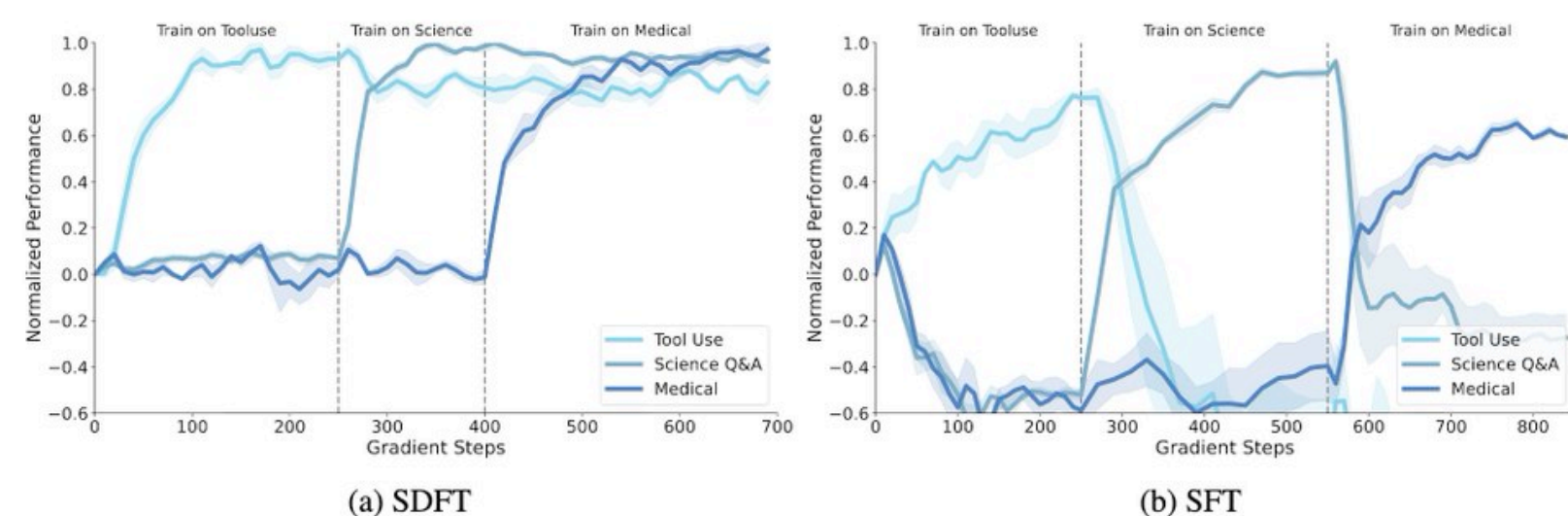
# Self-distillation leverages in-context improvement for parametric learning

Check out other applications of self-distillation:

- Shenfeld et al. *Self-Distillation enables Continual Learning*, 2026.
- Buening et al. *Aligning Language Models from User Interactions*, 2026.

## 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

