Test-Time Self-Distillation

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

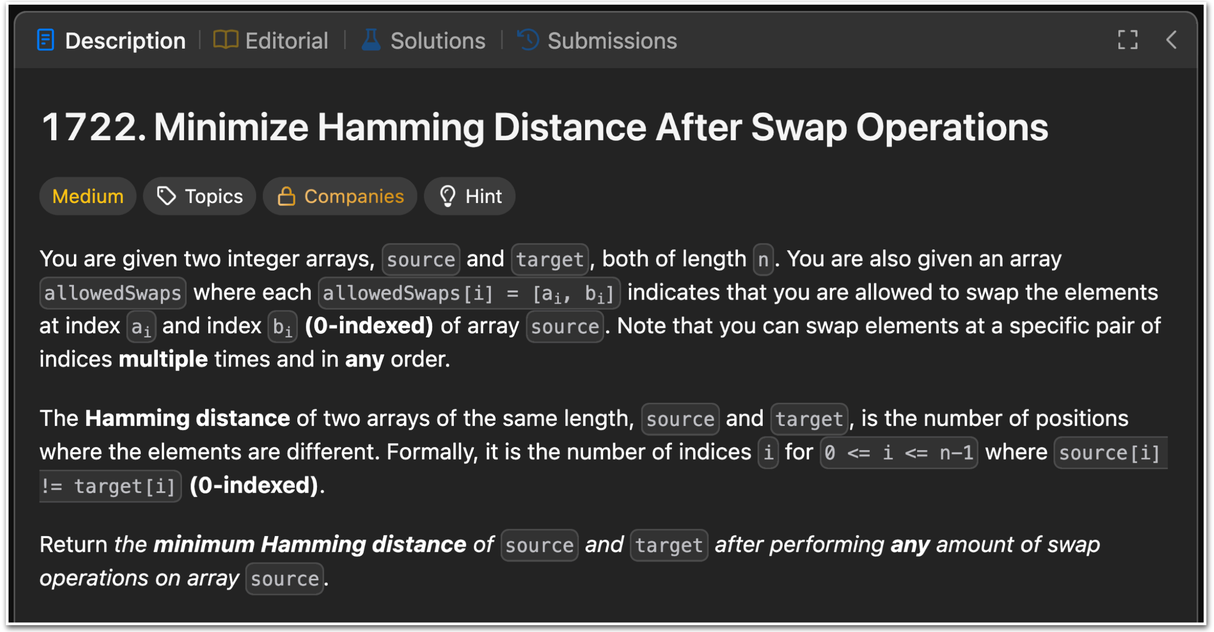
ETH zürich

MAX PLANCK INSTITUTE
FOR INTELLIGENT SYSTEMS

Stanford
University

Learning to Solve Hard Problems

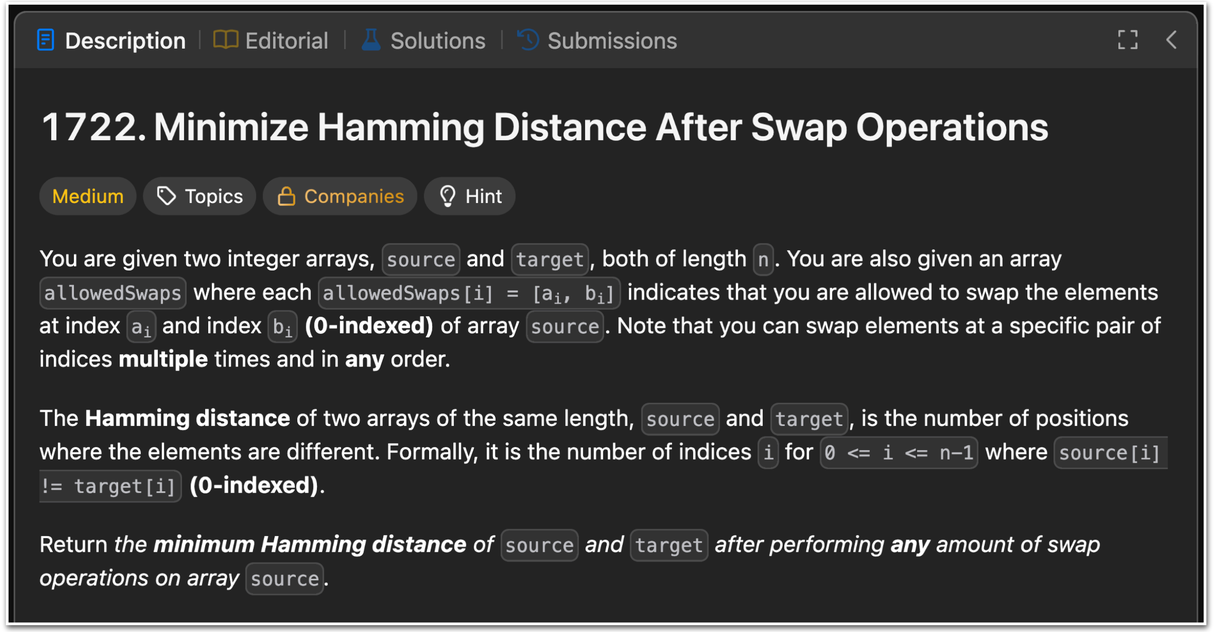
Given a hard problem at test-time, how quickly can a model learn to solve it?



Example: Leetcode

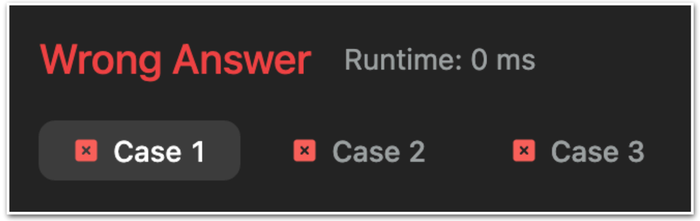
Learning to Solve Hard Problems

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



Example: Leetcode

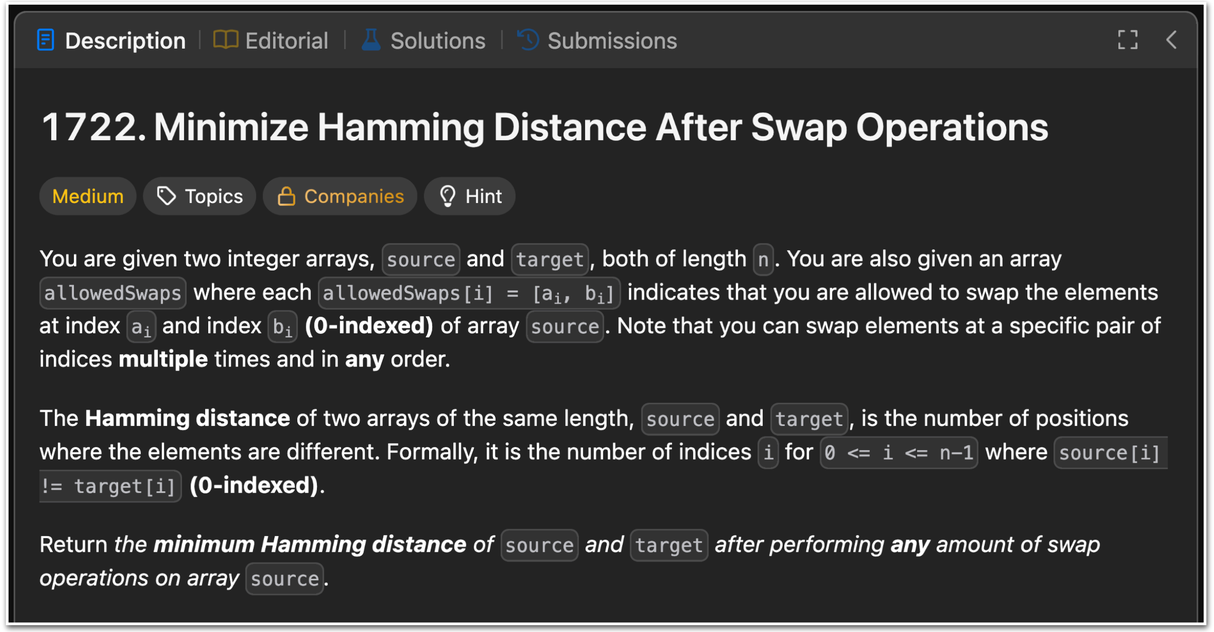
Many environments indicate *why* an attempt failed



Example: Unit tests

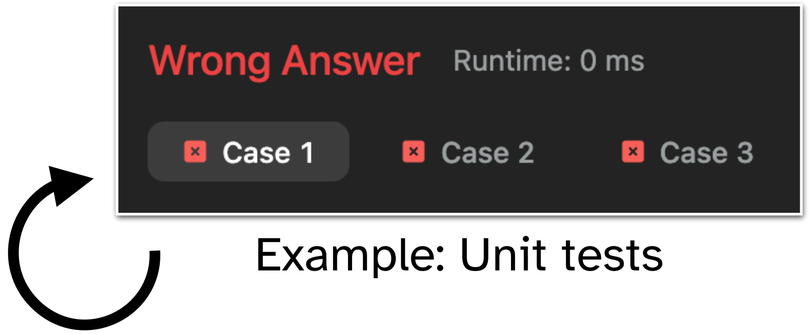
Learning to Solve Hard Problems

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



Example: Leetcode

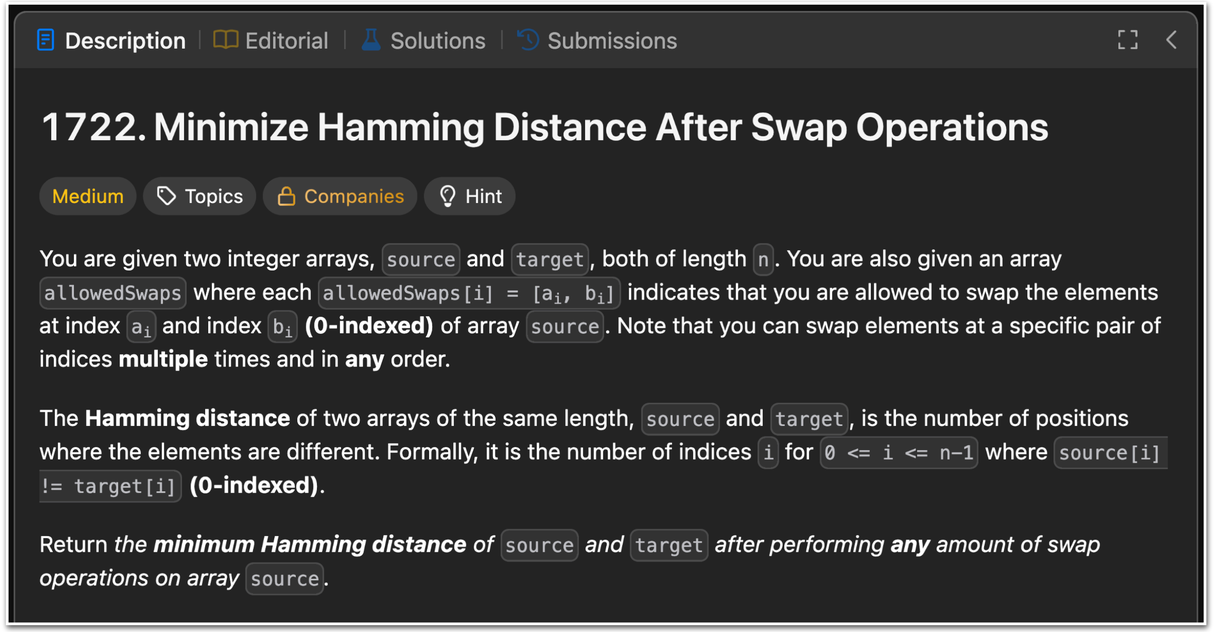
Many environments indicate *why* an attempt failed



Example: Unit tests

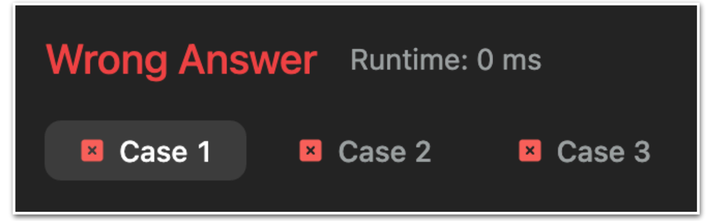
Learning to Solve Hard Problems

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



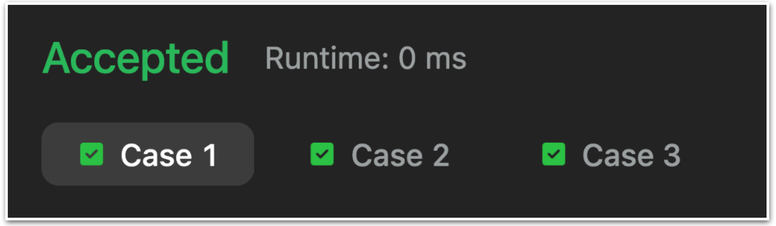
Example: Leetcode

Many environments indicate *why* an attempt failed



Example: Unit tests

until...



Recent work accelerates discovery for dense reward problems

Google DeepMind

AlphaEvolve: A coding agent for scientific and algorithmic discovery

Alexander Novikov*, Ngân Vũ*, 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 Balo
Google DeepMind[1]

Learning to Discover at Test Time

Mert Yuksekgonul*[1], Daniel Koceja*[1], Xinhao Li*[4], Federico Bianchi*[5]
Jed McCaleb[3], Xiaolong Wang[4], Jan Kautz[2], Yejin Choi[2], 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 $y \sim \pi_\theta(\cdot \mid x)$:

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

incorrect!

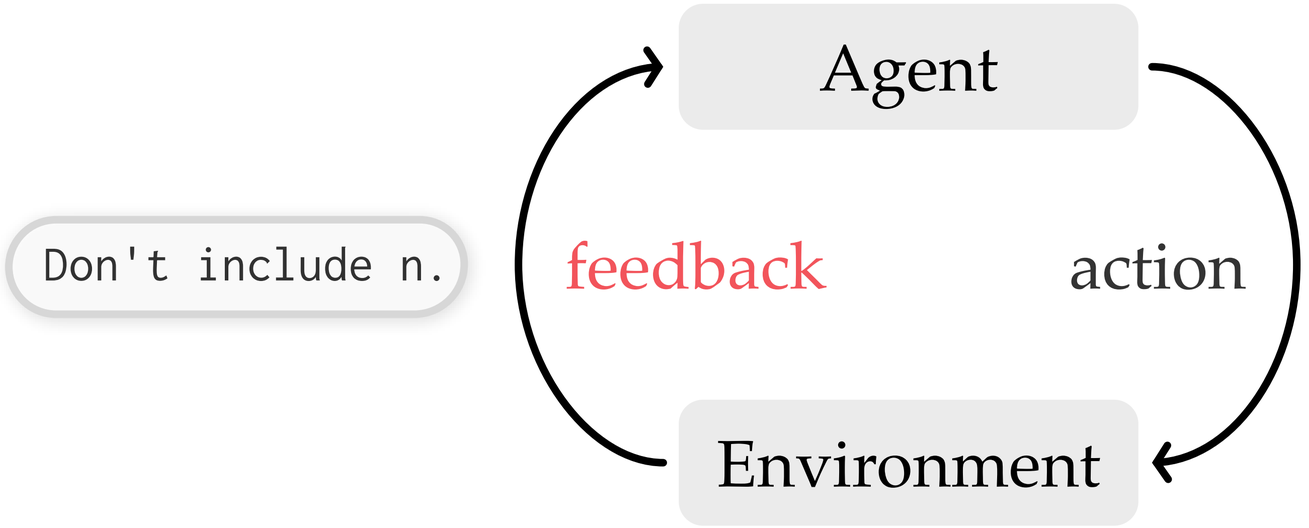
Question:

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

Answer $y \sim \pi_\theta(\cdot \mid x)$:

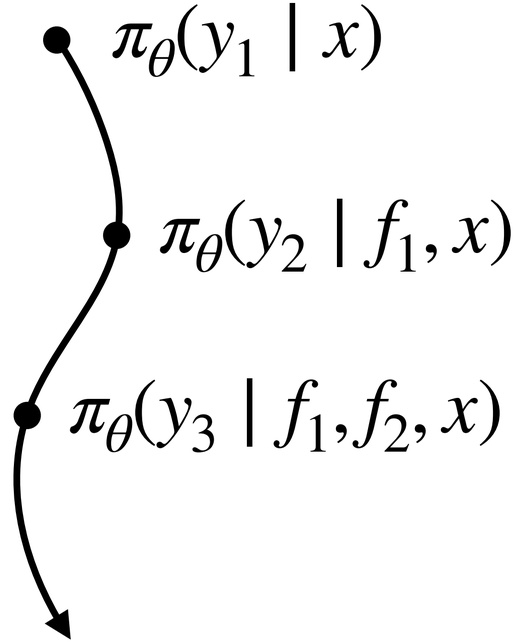
```python
def numbers_up_to_n(n):
    return list(range(1, n + 1))
```

incorrect!



In-context learning

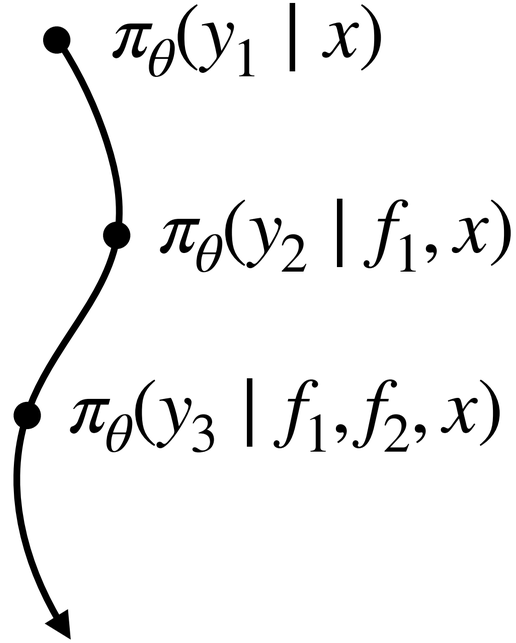
- Enables models to learn from rich feedback, hints, examples, and more.



Prompt x, Responses y, Feedback f

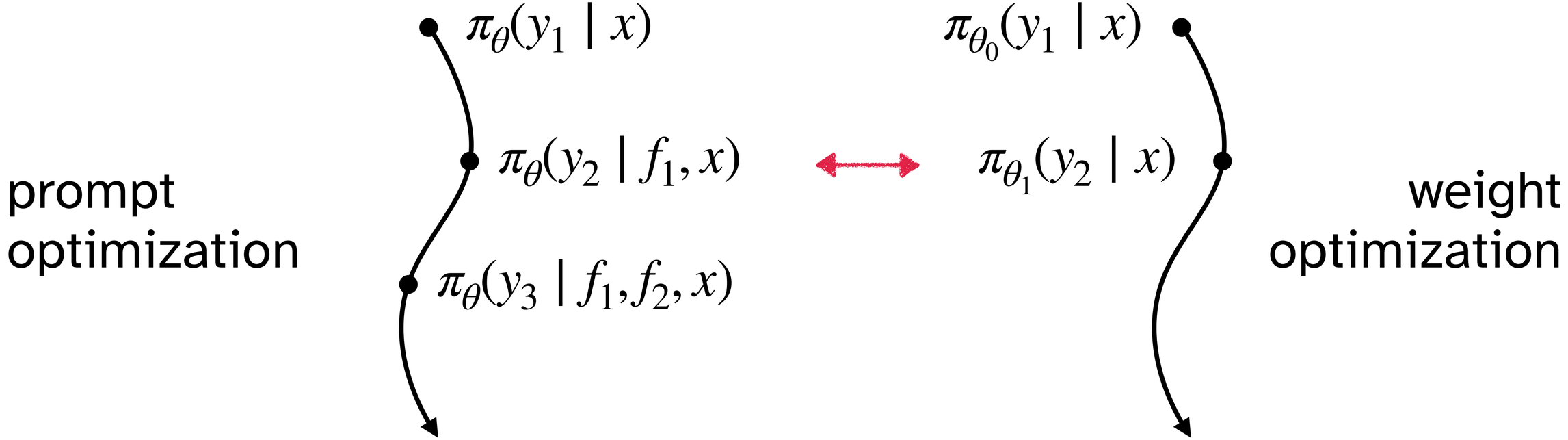
In-context learning

- 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.

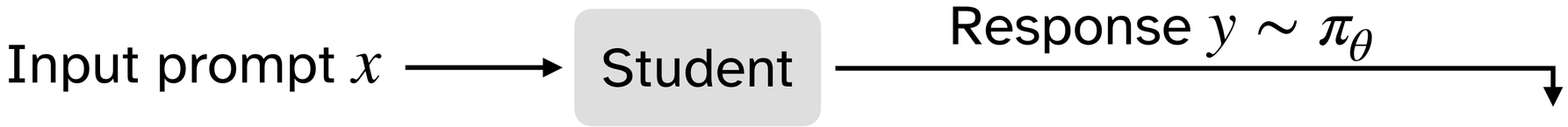
Self-distillation

Prompt x, Response y, Feedback f

Input prompt x

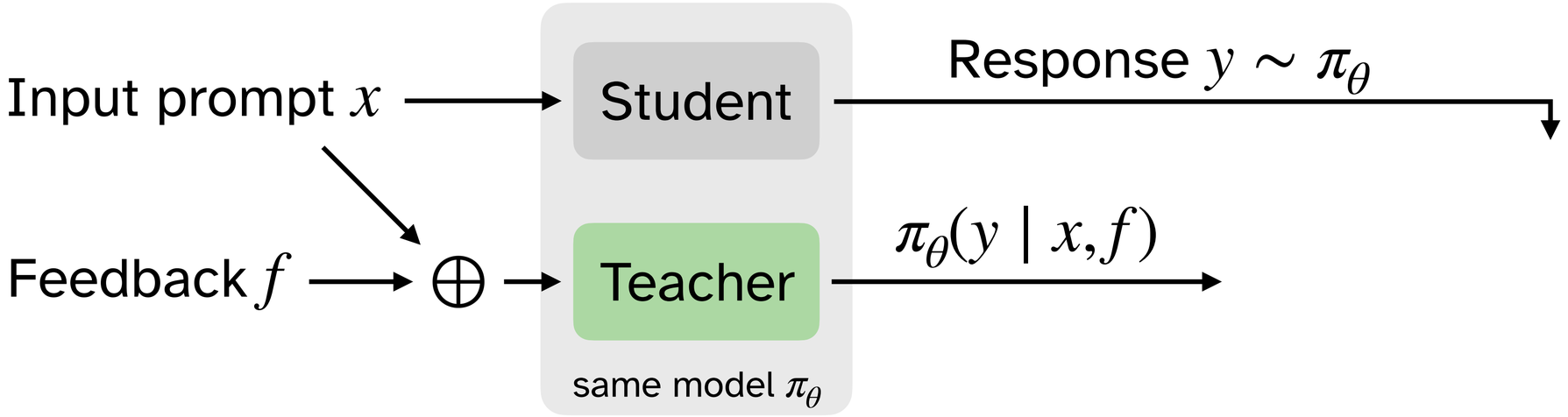
Self-distillation

Prompt x, Response y, Feedback f



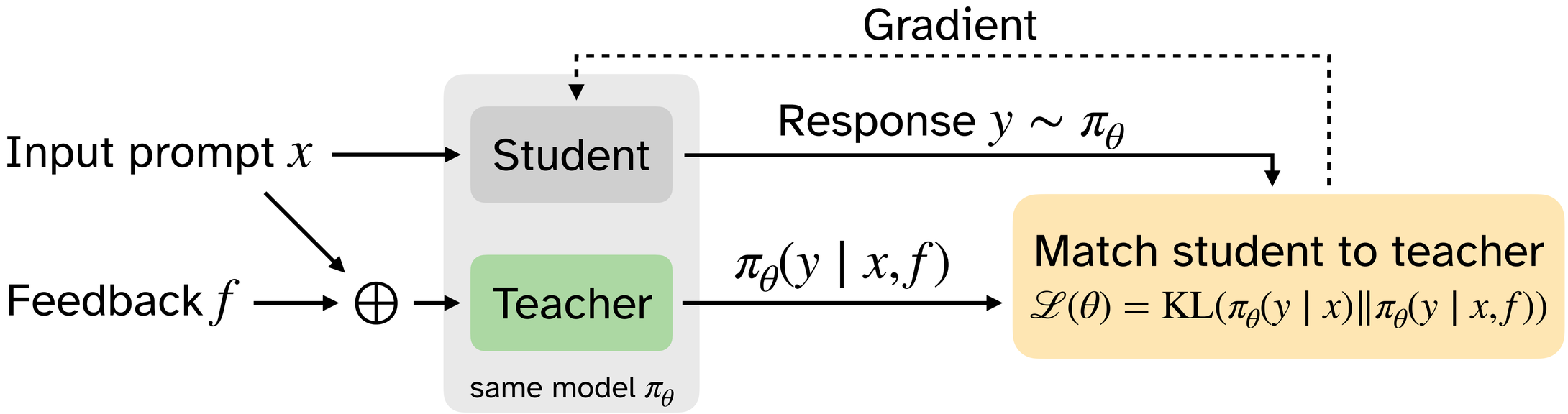
Self-distillation

Prompt x, Response y, Feedback f



Self-distillation

Prompt x, Response y, Feedback f



Two equivalent perspectives

Match student to teacher

$$\mathscr{L}(\theta) = \mathrm{KL}(\pi_\theta(y \mid x) \| \pi_\theta(y \mid x, f))$$

$\Longleftrightarrow$

Token-level advantages

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

On-policy distillation

On-policy RL

Credit assignment in self-distillation

Token-level advantages

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

Question:

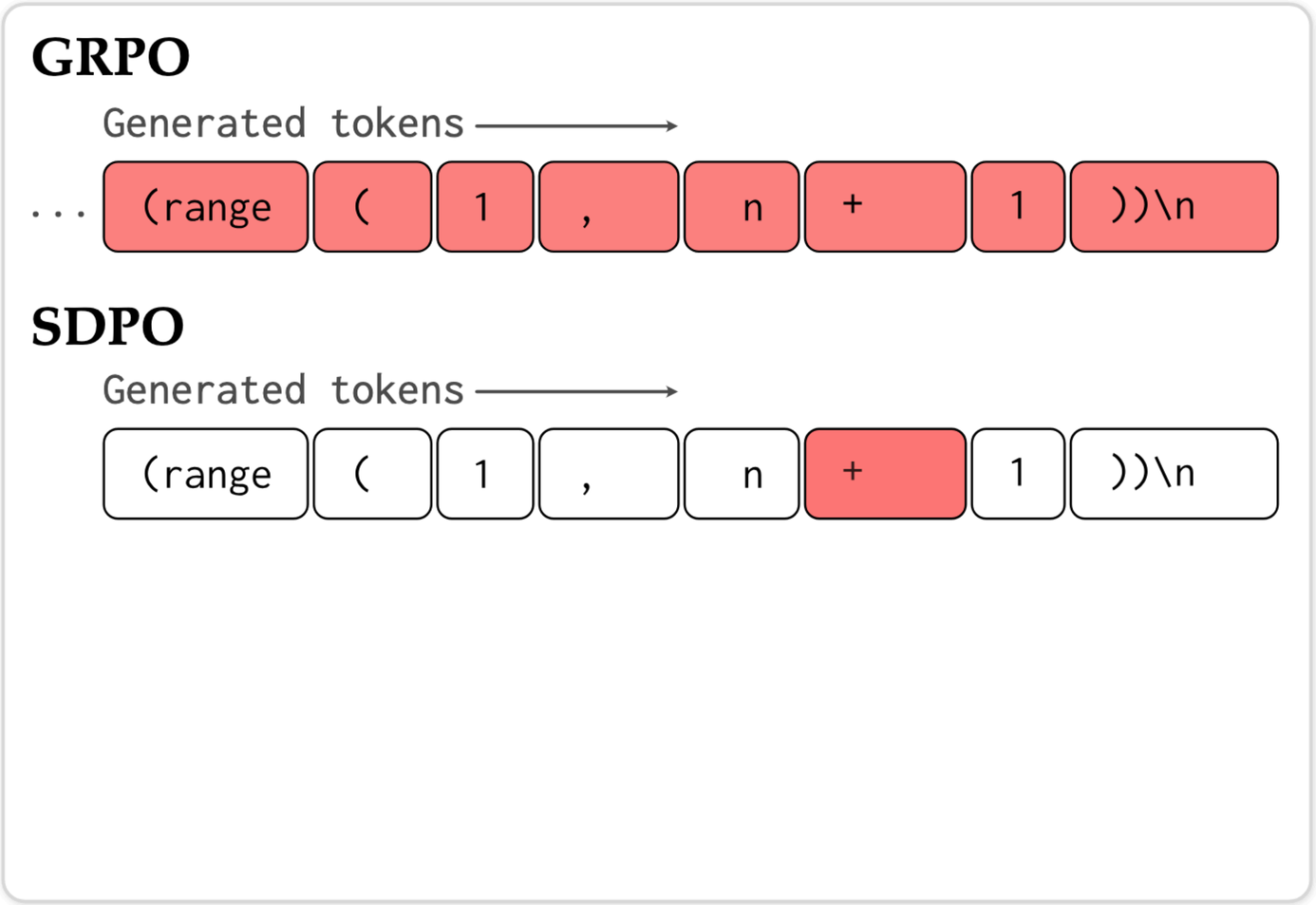
Write a python function that returns all numbers from 1 to n. Answer briefly.

Answer $y \sim \pi_\theta(\cdot \mid x)$:

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

Feedback:

Don't include n.



Credit assignment in self-distillation

Token-level advantages

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

Question:

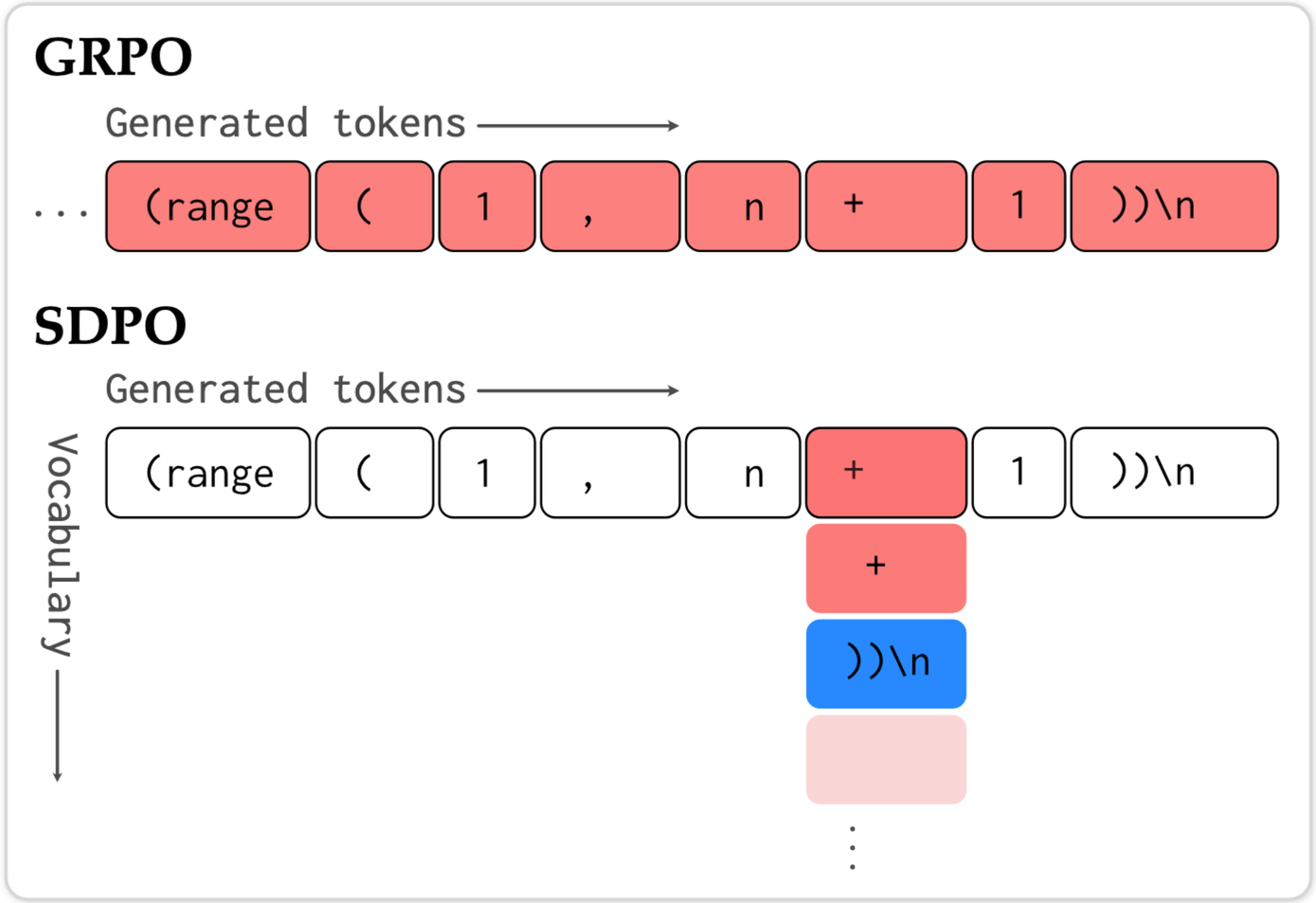
Write a python function that returns all numbers from 1 to n. Answer briefly.

Answer $y \sim \pi_\theta(\cdot \mid x)$:

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

Feedback:

Don't include 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?

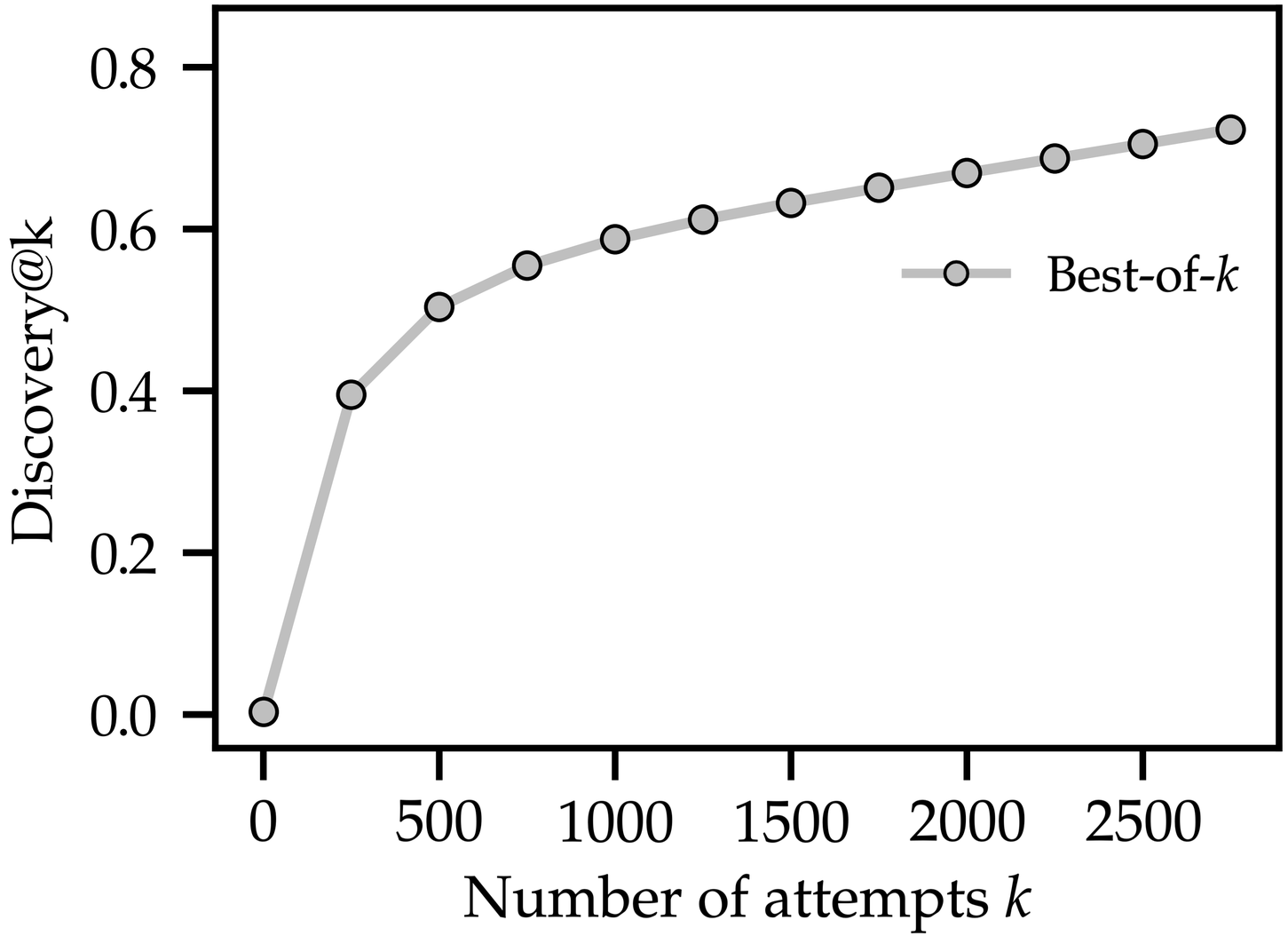
- **Goal:** Accelerate time-to-discovery
- **Discovery time** is the number of trials until a solution is found
- We want to increase the probability of discovery:

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

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

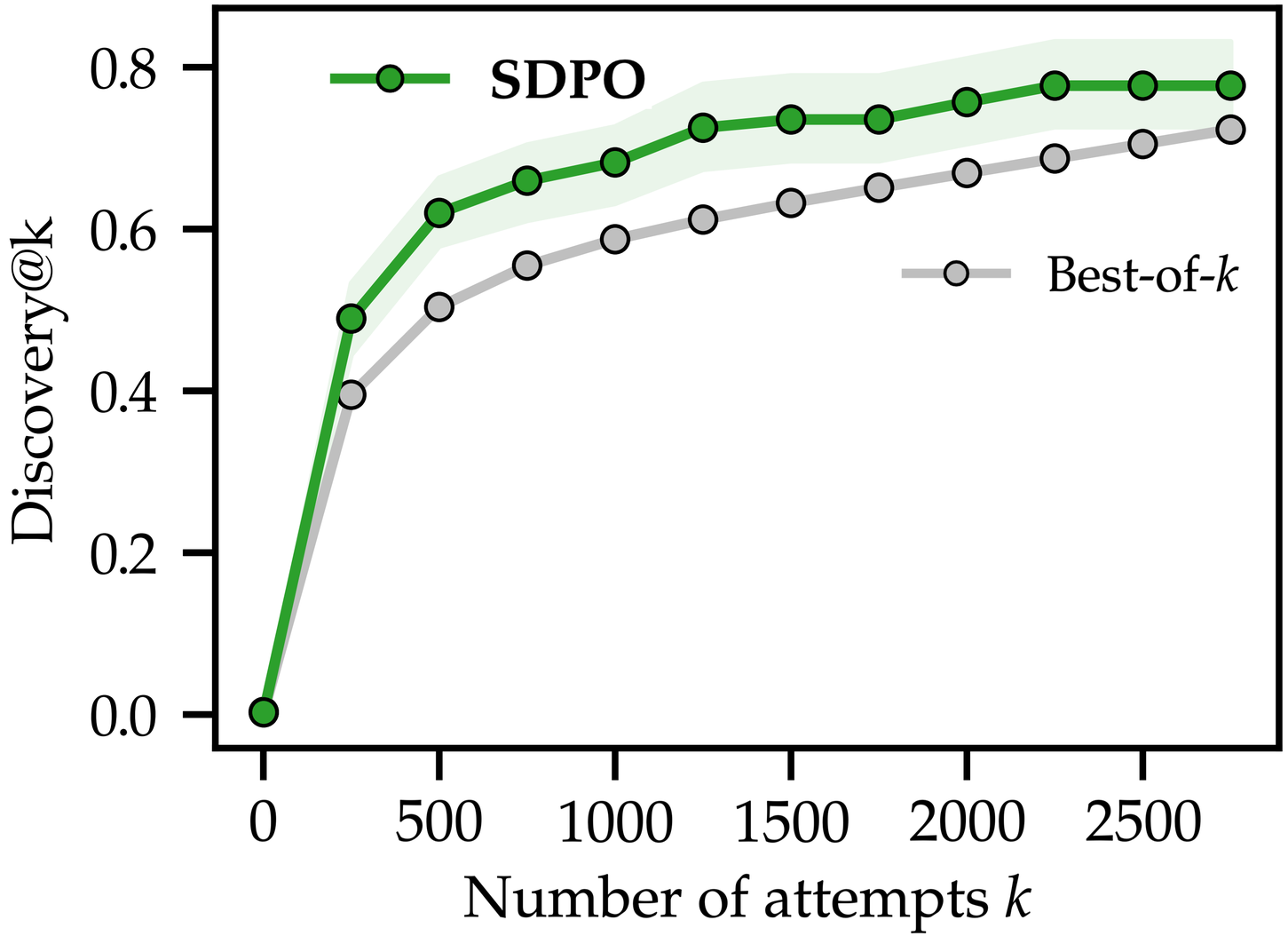
Results

- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*: pass@64 < 0.5



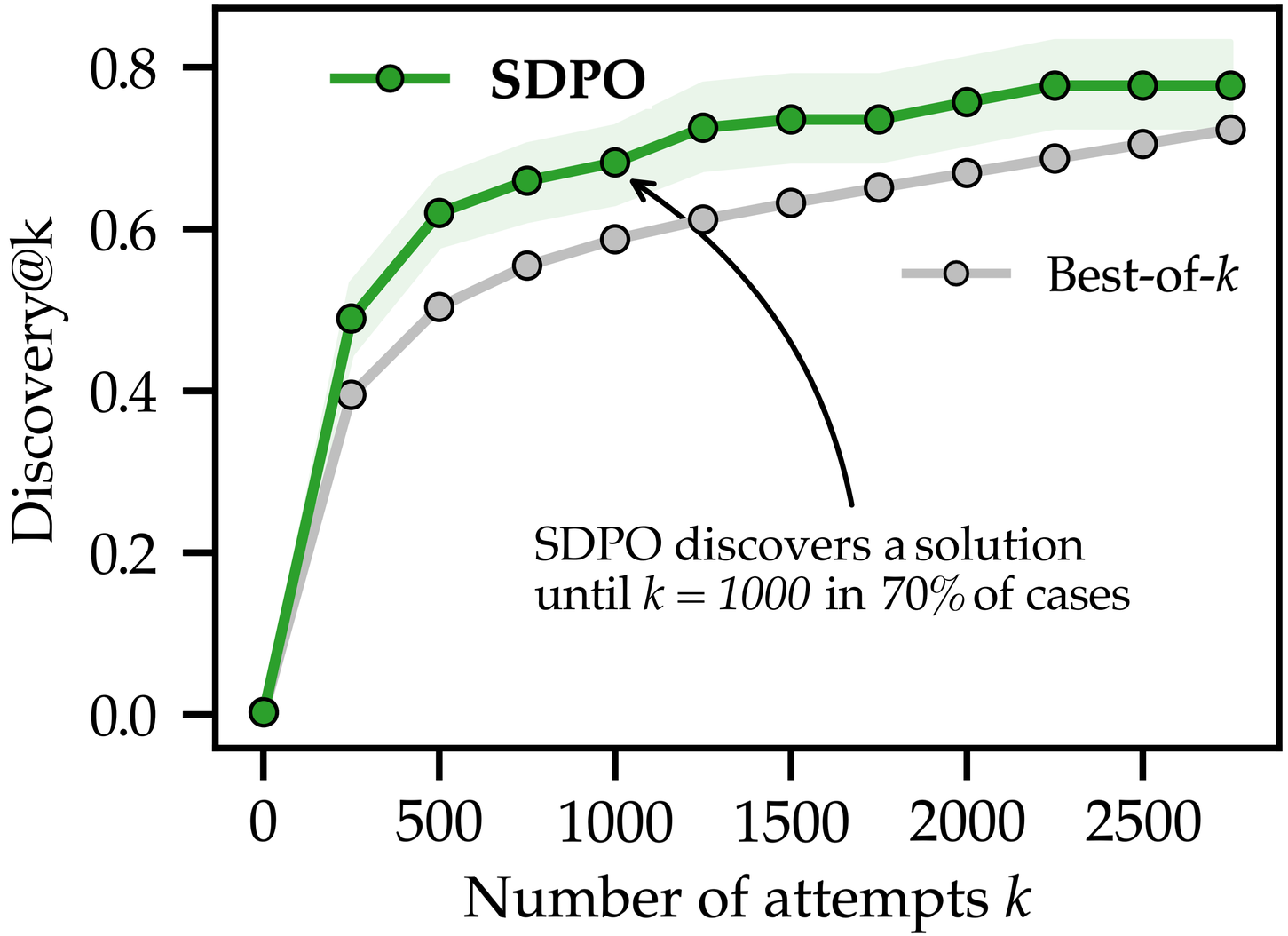
Results

- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*: pass@64 < 0.5



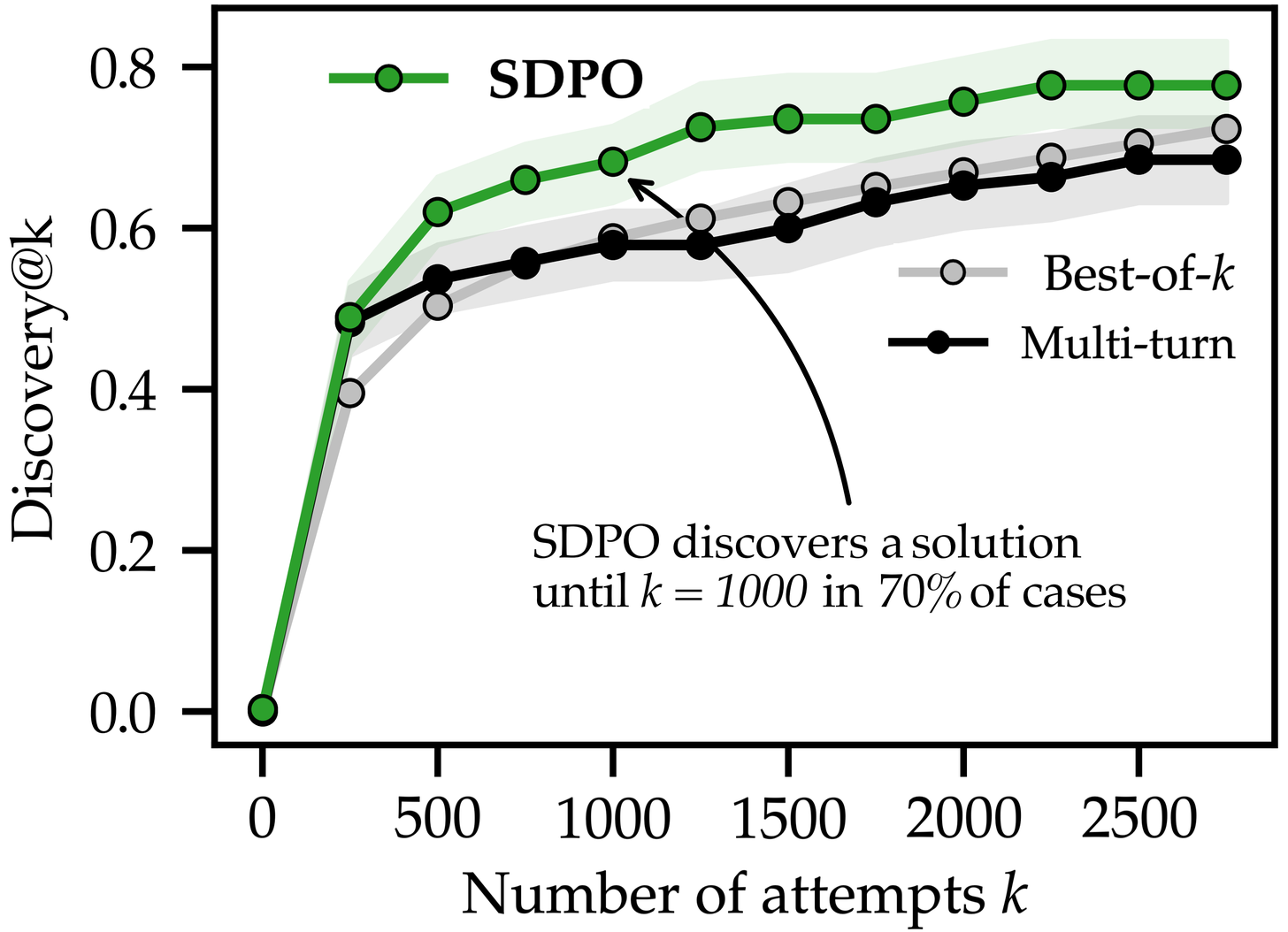
Results

- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*: pass@64 < 0.5

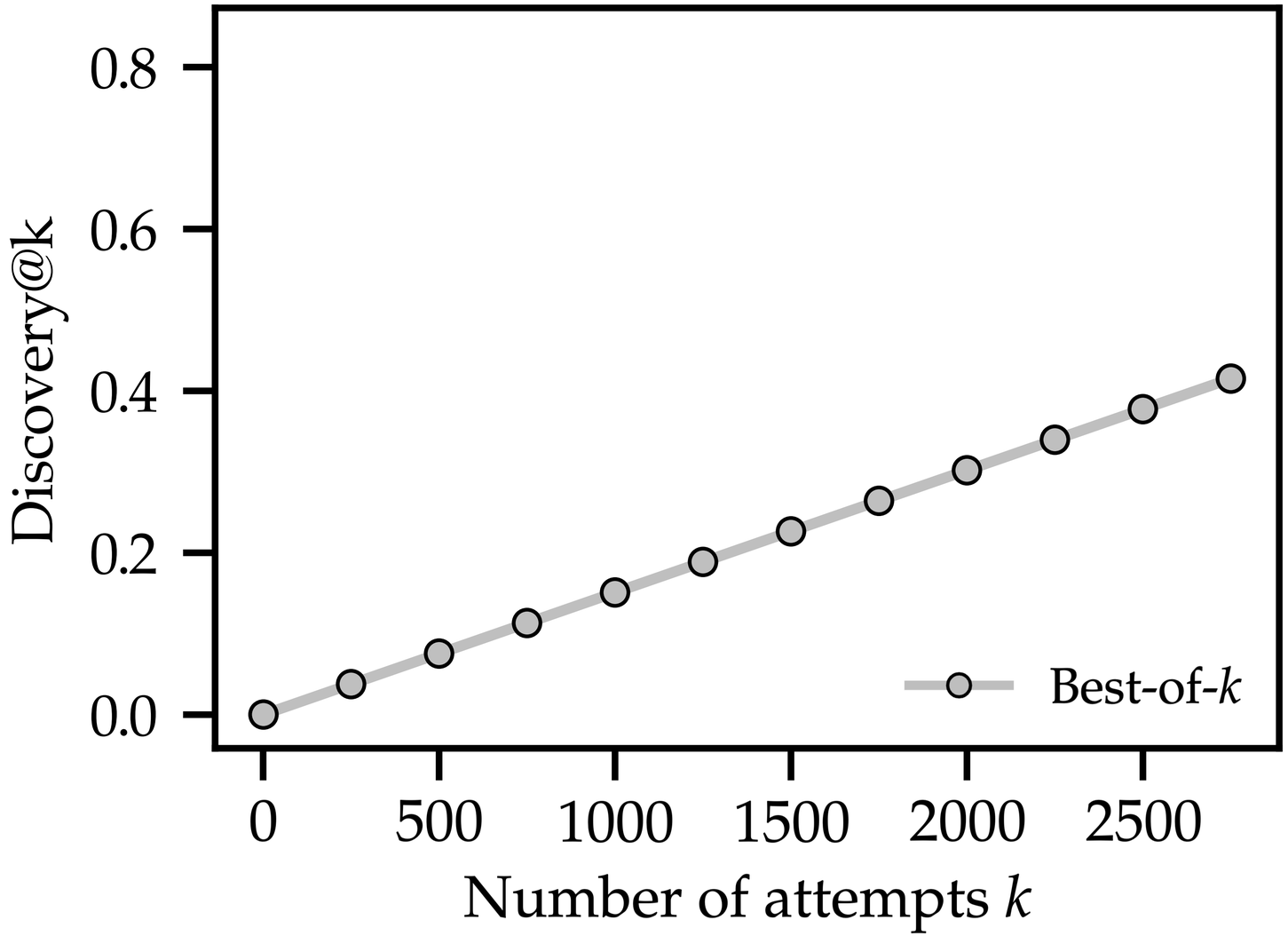


Results

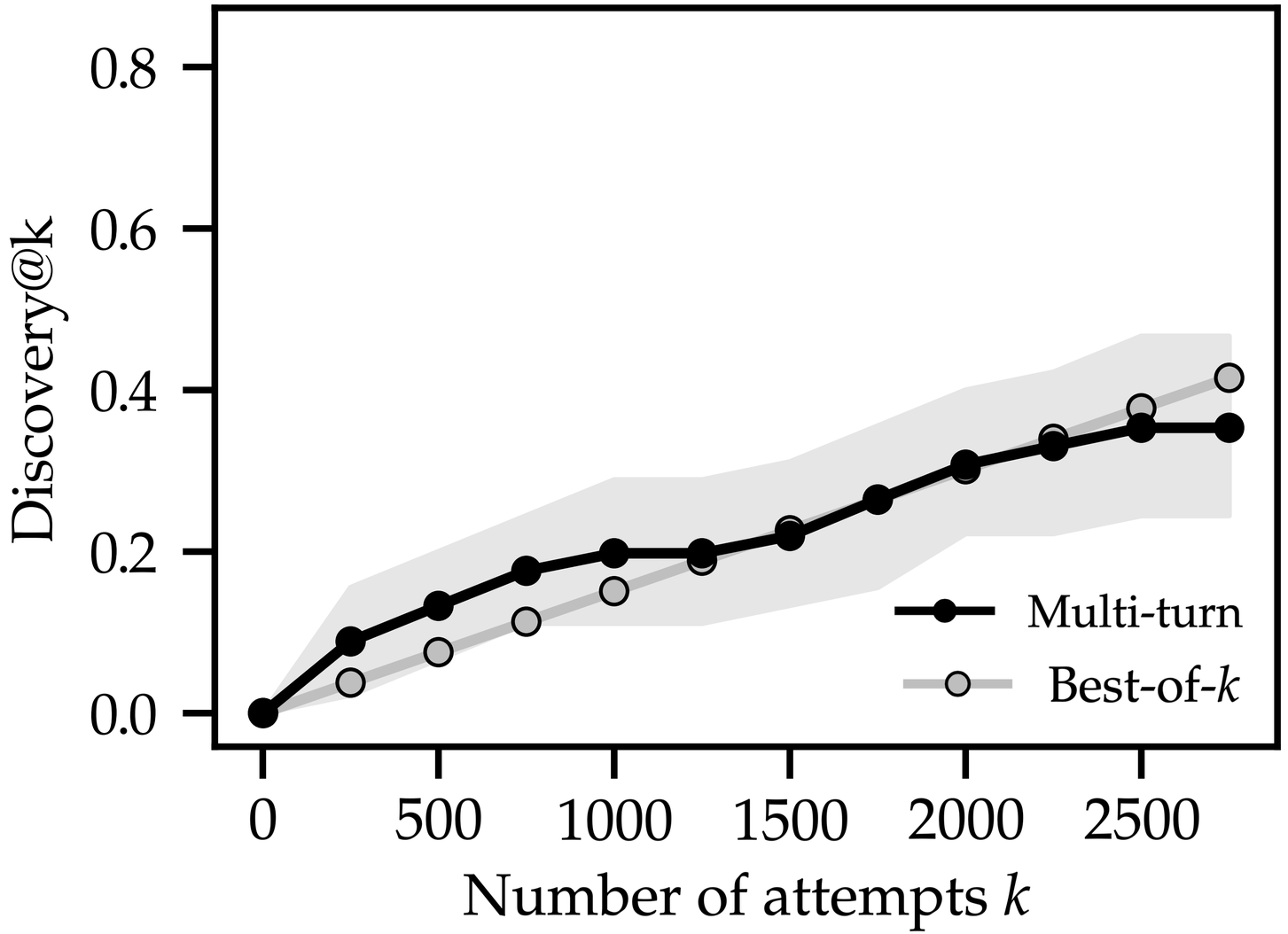
- Qwen3-8B on LiveCodeBench-v6
- Subset of *hard tasks*: pass@64 < 0.5



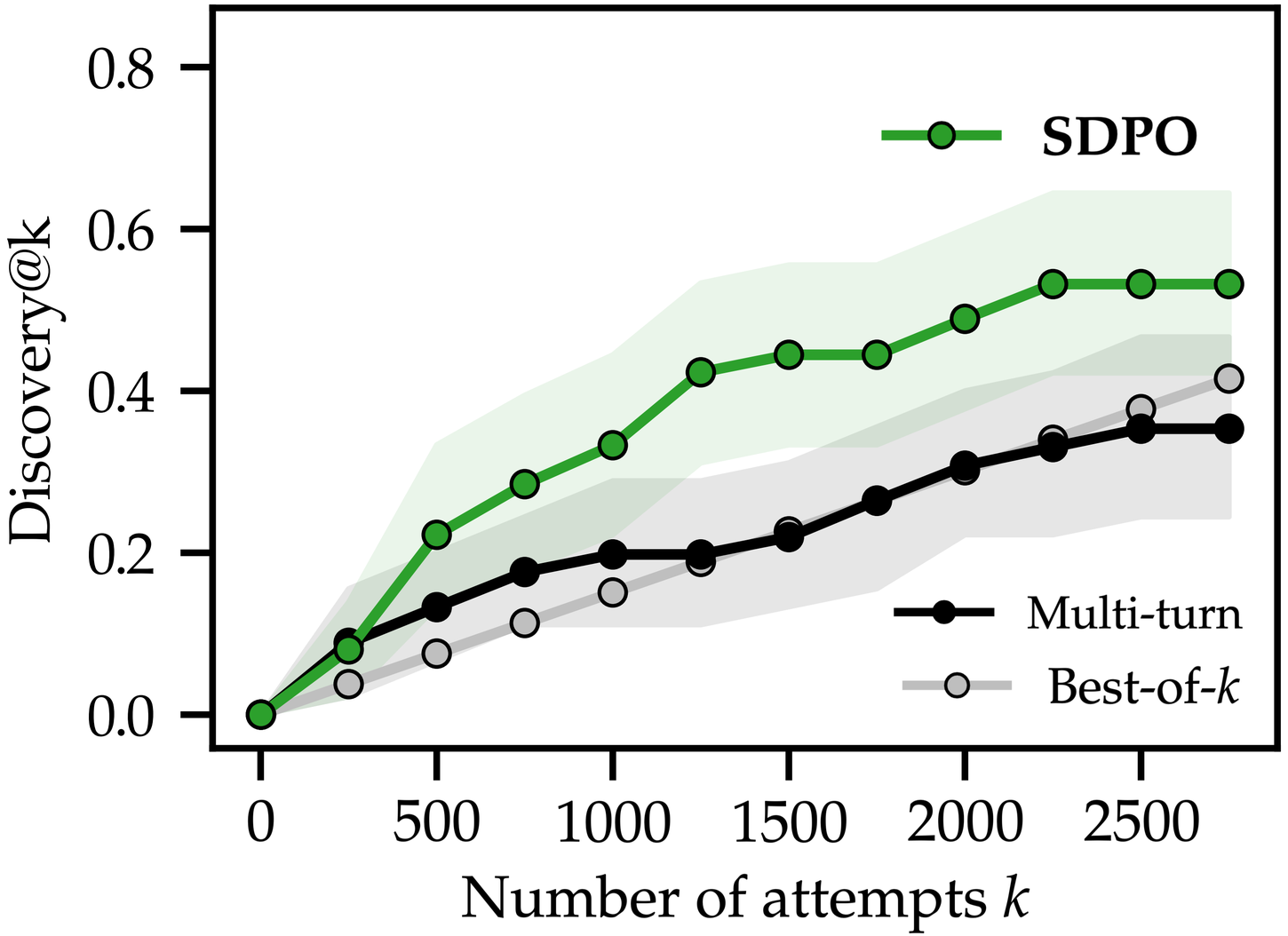
Results on very hard tasks (pass@64<0.03)



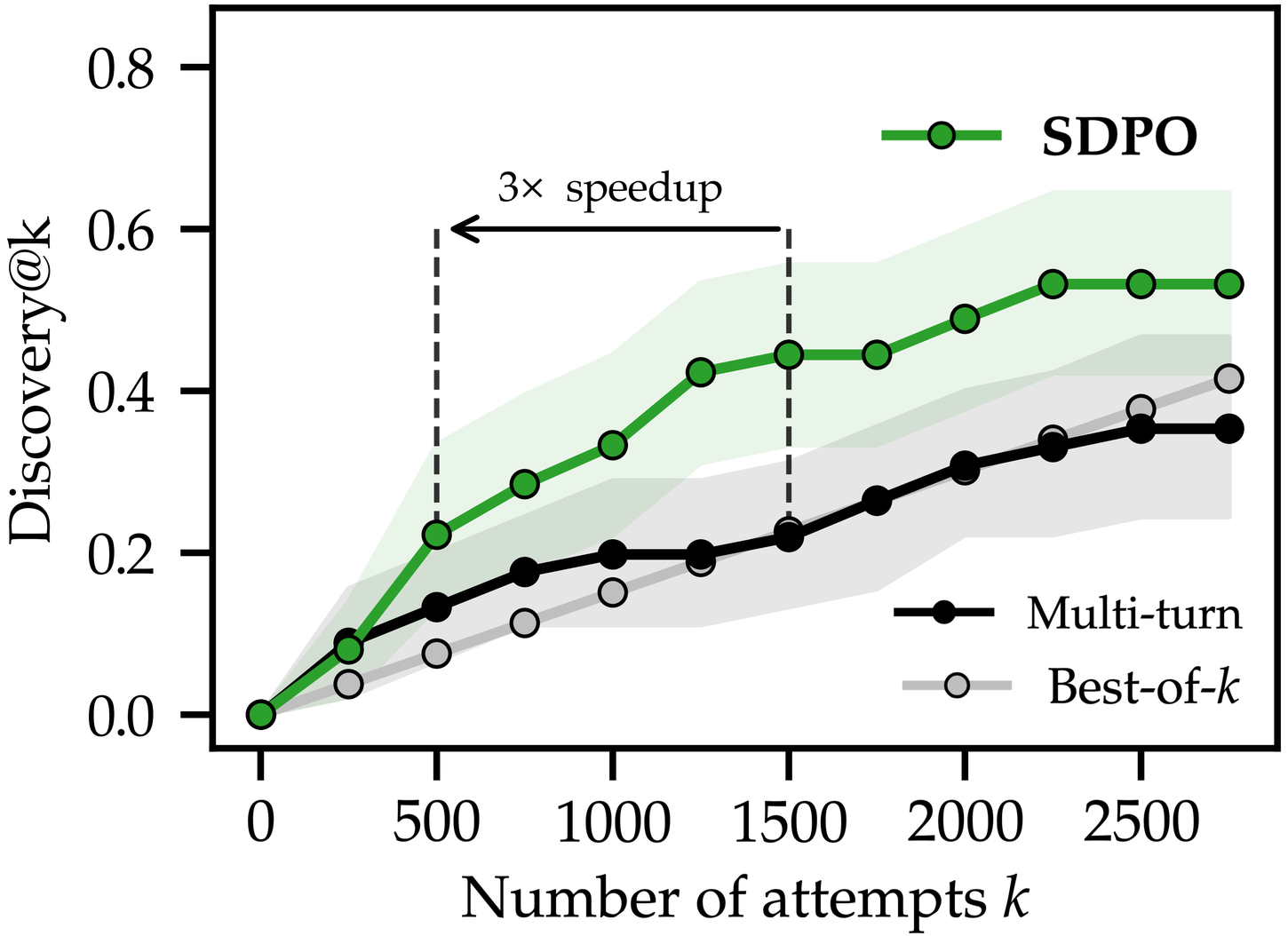
Results on very hard tasks (pass@64<0.03)



Results on very hard tasks (pass@64<0.03)



Results on very hard tasks (pass@64<0.03)

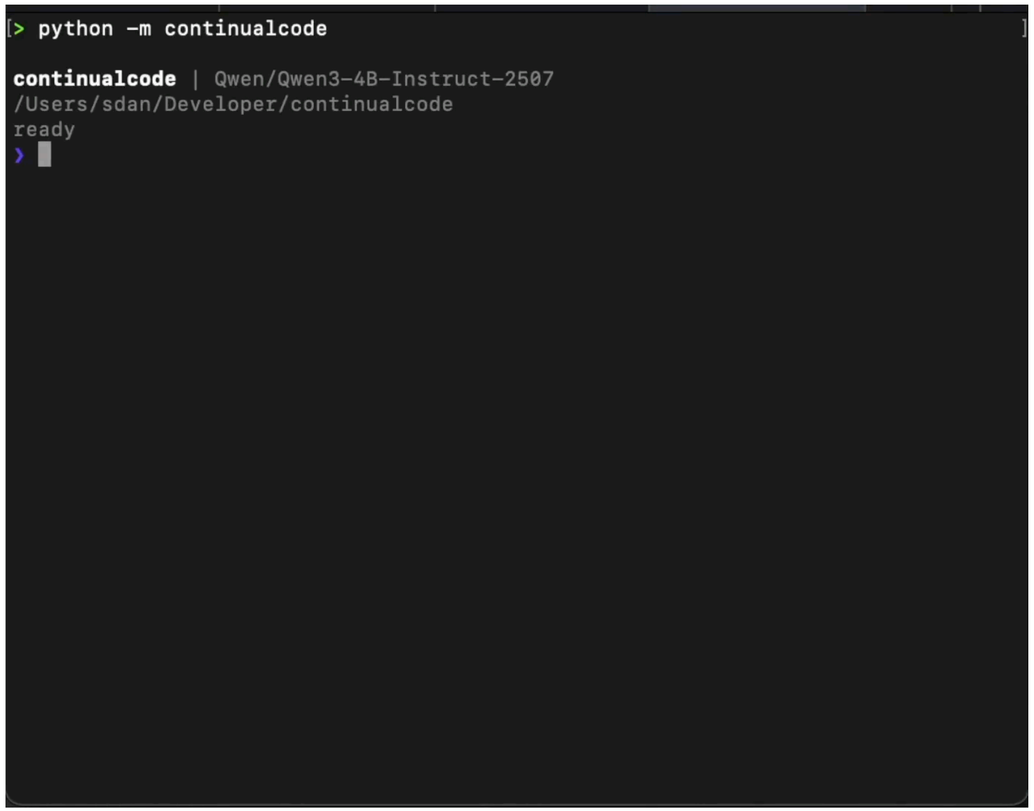


Takeaways

- 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



OpenClaw-RL

>5k stars on GitHub!

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 × $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.