

Teaching Language Models to Reason with Tools

Chengpeng Li^{*1,2}, Zhengyang Tang^{*2,3}, Ziniu Li^{*3,4}, Mingfeng Xue², Keqin Bao^{1,2}, Tian Ding⁴, Ruoyu Sun^{3,4},
Benyou Wang³, Xiang Wang¹, Junyang Lin², Dayiheng Liu²

University of Science and Technology of China¹

Qwen Team, Alibaba Inc.²

The Chinese University of Hong Kong, Shenzhen³

Shenzhen International Center for Industrial and Applied Mathematics, Shenzhen Research Institute of Big
Data⁴

^{*}: Equal Contribution

- 1. Background**
2. Motivation
3. Method
4. Experiments

1.1 Tool-integrated Reasoning

Problem: Suppose that the sum of the squares of two complex numbers x and y is 7 and the sum of their cubes is 10. List all possible values for $x + y$, separated by commas.

We are given that $x^2 + y^2 = 7$ and $x^3 + y^3 = 10$. We can factor $x^3 + y^3$ to get $(x + y)(x^2 - xy + y^2)$.

Thus, we can write $10 = (x + y)(7)$. It follows that $x + y$ must divide 10.

The only possibilities are that $x + y$ is 1, 2, 5 or 10. ❌

```
import math
```

```
def solution():
```

```
    x = symbols('x'), y = symbols('y')
    x**2 + y**2 = 7
    x**3 + y**3 = 10
```

```
    x = (-1 + math.sqrt(17)) / 2
    y = (-1 - math.sqrt(17)) / 2
```

```
    # Get all possible values for x+y
    result = [x + y, -x - y]
```

```
>>> SyntaxError: cannot assign to expression. ❌
```

Let's use `SymPy` to calculate and print all possible values for $x + y$,

```
from sympy import symbols, Eq, solve, simplify
```

```
def possible_values():
```

```
    x, y = symbols('x y')
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve((eq1, eq2), (x, y))
    sums = [simplify(sol[0] + sol[1]) for sol in solutions]
    return sums
print(possible_values())
```

```
>>> [-5, -5, 1, 1, 4, 4]
```

Removing duplicates, the possible values for $x + y$ are $\boxed{-5, 1, 4}$. ✅



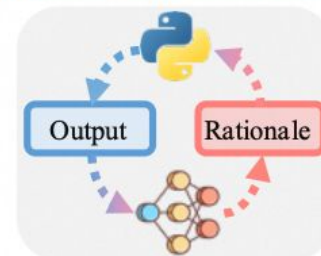
Rationale

(a) Rationale-based



Program

(b) Program-based



(c) Tool-integrated Reasoning

(Format used by ToRA)

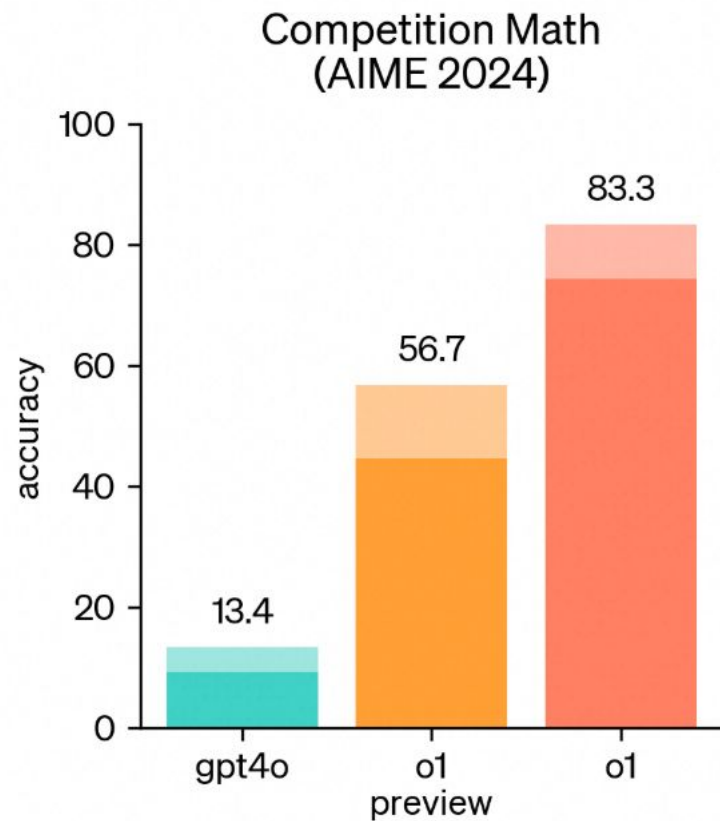
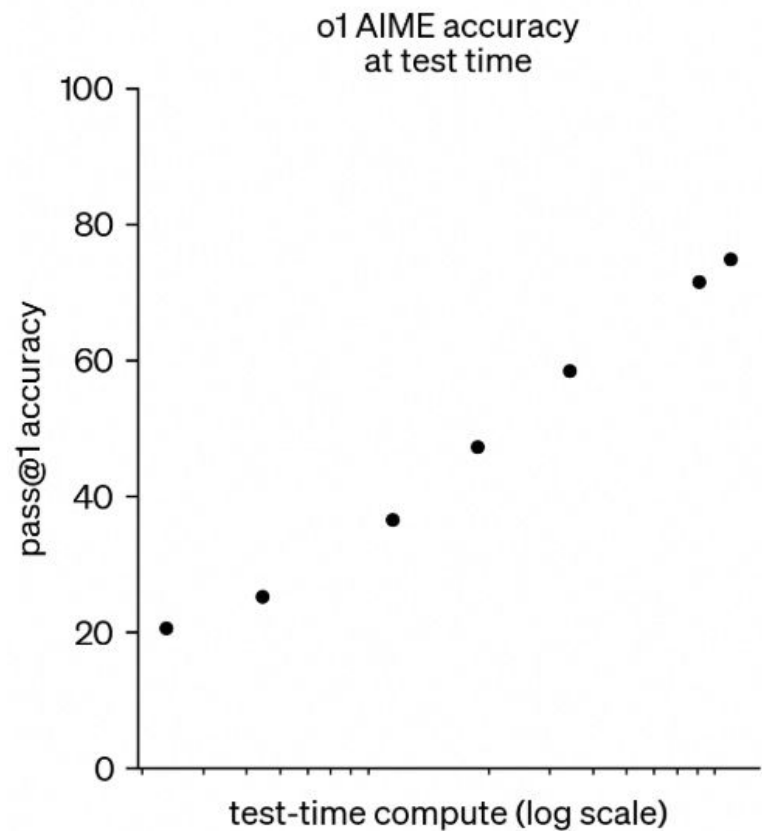
r_1

a_1

o_1

r_2

1.2 Long CoT Reasoning



1. Background
- 2. Motivation**
3. Method
4. Experiments

2.1 Open question

- How can we synthesize high-quality training data when models like o3 and o4-mini do not expose their detailed reasoning traces?
- How can the model effectively coordinate a CI's computational precision with its abstract CoT reasoning capabilities?
- How can the self-reflection mechanisms inherent to LRMs be reconciled with the exact external knowledge provided by CIs?

1. Background
2. Motivation
- 3. Method**
4. Experiments

3.1 Cold Start – Prompt-Hint

Each face of two noncongruent parallelepipeds is a rhombus whose diagonals have lengths $\sqrt{21}$ and $\sqrt{31}$. The ratio of the volume of the larger of the two polyhedra to the volume of the smaller is m/n , where m and n are relatively prime positive integers. Find $m + n$.

<think>

Okay, let's try to solve this problem step by step using multiple python code calls.

...

[skip 8524 tokens]

...

Thus, the answer is 125.

Now, to confirm using Python.

```
import numpy as np
```

...

```
>>> 125
```

So $m=63$, $n=62$, $m+n=125$.

</think>

...

The sum $m + n = 63 + 62 = \boxed{125}$.

(a) Prompt-hint

<think>

Okay, let's try to solve this problem step by step using multiple python code calls.

[skip 3261 tokens]

Let's compute V1 and V2 with Python for accuracy.

```
import sympy as sp
```

...

```
>>> Volume 1: 63*sqrt(2)/2
      Volume 2: 31*sqrt(2)
```

Hint-1:

Complex calculations

We don't need to doubt the accuracy of python calculations.

[skip 216 tokens]

</think>...

The answer is $\boxed{125}$.

Hint-2:

Confirm python calculation

Without Hint-1

Let's compute V1 and V2 for these two cases.

[skip 5057 tokens]

V1= 63sqrt(2)/2,V2= 31sqrt(2)

Without Hint-2

Wait, but let me check These calculation...

[skip 2215 tokens]

</think>...

The answer is $\boxed{125}$.

(b) Hint-engineering

3.1 Cold Start – Hint-engineering

Each face of two noncongruent parallelepipeds is a rhombus whose diagonals have lengths $\sqrt{21}$ and $\sqrt{31}$. The ratio of the volume of the larger of the two polyhedra to the volume of the smaller is m/n , where m and n are relatively prime positive integers. Find $m + n$.

<think>

Okay, let's try to solve this problem step by step using multiple python code calls.

...

[skip 8524 tokens]

...

Thus, the answer is 125.

Now, to confirm using Python.

```
import numpy as np
```

...

```
>>> 125
```

So $m=63$, $n=62$, $m+n=125$.

</think>

...

The sum $m + n = 63 + 62 = \boxed{125}$.

(a) Prompt-hint

<think>

Okay, let's try to solve this problem step by step using multiple python code calls.

[skip 3261 tokens]

Let's compute V1 and V2 with Python for accuracy.

```
import sympy as sp
```

...

```
>>> Volume 1: 63*sqrt(2)/2
      Volume 2: 31*sqrt(2)
```

Hint-1:

Complex calculations

We don't need to doubt the accuracy of python calculations.

[skip 216 tokens]

</think>...

The answer is $\boxed{125}$.

Hint-2:

Confirm python calculation

Without Hint-1

Let's compute V1 and V2 for these two cases.

[skip 5057 tokens]

V1= 63sqrt(2)/2,V2= 31sqrt(2)

Without Hint-2

Wait, but let me check These calculation...

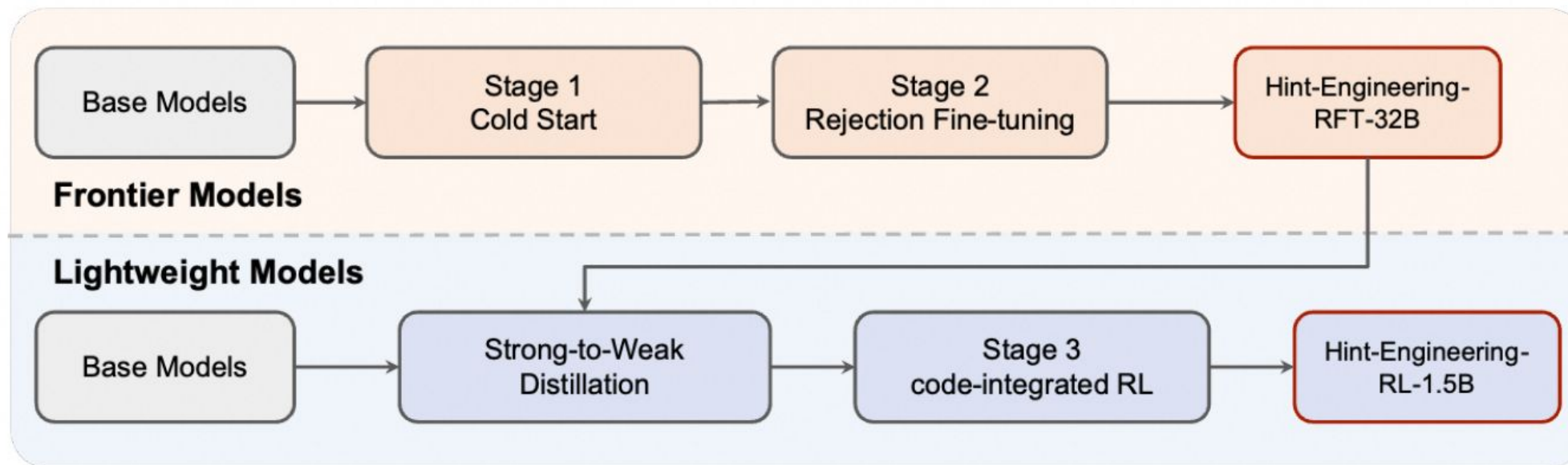
[skip 2215 tokens]

</think>...

The answer is $\boxed{125}$.

(b) Hint-engineering

3.2 Strong-to-weak Distillation



3.2 Code-integrated Reinforcement Learning

- Rollout with Code Interpreter
- Persistent Execution Environment
- Output Masking
- Reward Design

$$R_a = \begin{cases} 1 & \text{if answers match} \\ 0 & \text{otherwise} \end{cases}$$

$$R_c = \begin{cases} -1 & \text{if all codes fail} \\ 0 & \text{otherwise} \end{cases}$$

$$R = R_a + \omega R_c$$

1. Background
2. Motivation
3. Method
- 4. Experiments**

4.1 Setup

□ Model

- Base model: R1-distill-Qwen-1.5B, R1-distill-Qwen-32B

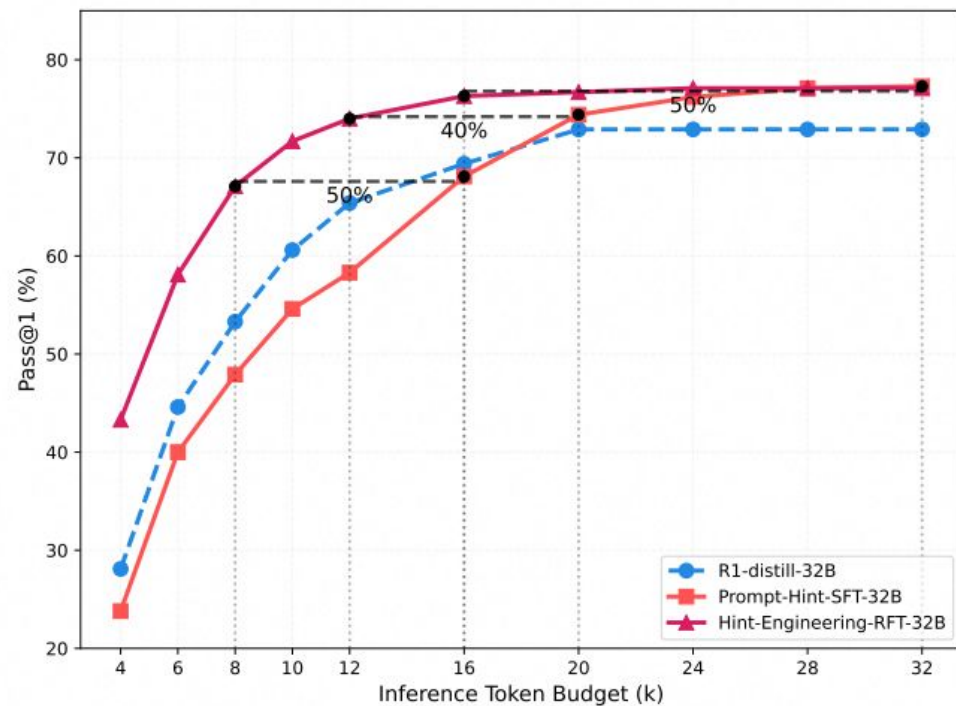
□ Data

- Training: previous AIME problems (before 2024), MATH and Numina-MATH,
Evaluation: AIME24, AIME25, AMC23, MATH500, Olympiad

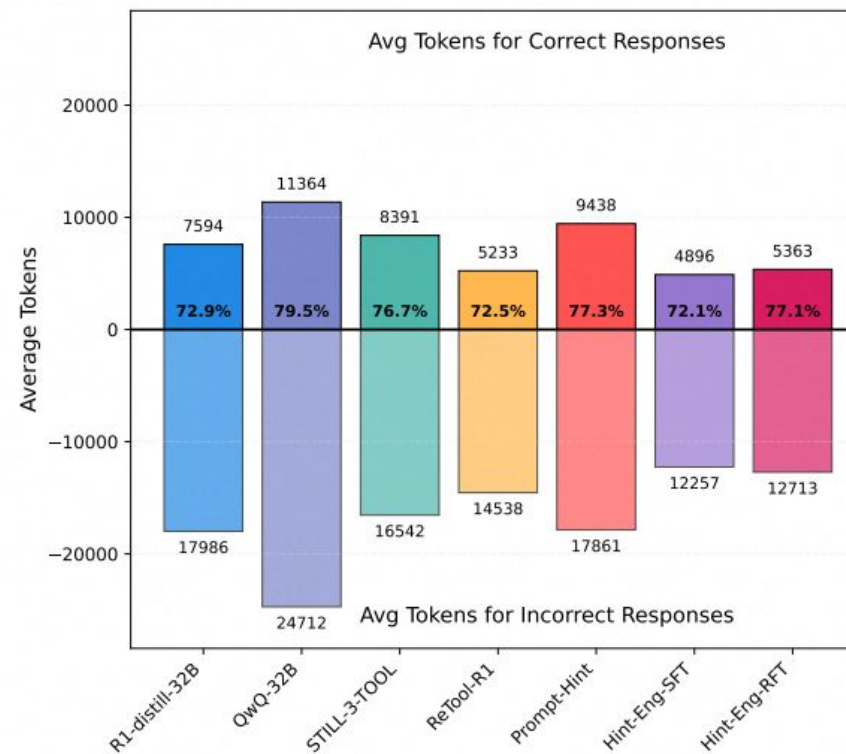
4.2 Main Result

Model	Tool-Use	Stage	AIME24	AIME25	AMC23	MATH500	Olympiad	Avg
<i>SOTA Models</i>								
o1	✗	RL	74.3	79.2	-	<u>96.4</u>	-	-
o3	✓	RL	95.2	98.7	-	-	-	-
o4-mini	✓	RL	98.4	99.5	-	-	-	-
DeepSeek-R1	✗	RL	79.8	<u>70.0</u>	-	97.3	-	-
QwQ-32B	✗	RL	<u>79.5</u>	65.3	94.3	92.3	79.7	82.2
<i>Frontier Models (32B)</i>								
DeepSeek-R1-32B	✗	SFT	72.9	59.0	88.8	94.3	72.5	77.5
START-32B	✓	SFT	66.7	47.1	95.0	94.4	-	-
STILL-3-TOOL-32B	✓	SFT	<u>76.7</u>	64.4	91.3	96.6	75.9	81.0
ReTool-R1-32B	✓	RL	72.5	54.3	92.9	94.3	69.2	76.6
Prompt-Hint-SFT-32B	✓	SFT	77.3	<u>65.0</u>	95.0	96.6	<u>75.1</u>	81.8
Hint-Engineering-SFT-32B	✓	SFT	72.1	60.2	91.3	94.4	71.2	77.8
Hint-Engineering-RFT-32B	✓	RFT	<u>76.7</u>	67.1	<u>94.4</u>	<u>95.1</u>	73.4	<u>81.3</u>
<i>Lightweight Models (1.5B)</i>								
DeepSeek-R1-1.5B	✗	SFT	28.8	21.8	62.9	83.9	43.3	48.1
DeepScaleR-1.5B-Preview	✗	RL	40.0	<u>30.0</u>	<u>73.6</u>	87.8	50.0	56.3
ToRL-1.5B	✓	RL	26.7	26.7	67.5	77.8	44.0	48.5
Prompt-Hint-1.5B-SFT	✓	SFT	30.6	25.0	63.1	83.3	50.4	50.5
Prompt-Hint-1.5B-RL	✓	RL	43.1	30.2	73.8	<u>87.3</u>	57.1	58.3
Hint-Engineering-1.5B-SFT	✓	SFT	34.0	23.5	64.6	84.2	49.8	51.2
Hint-Engineering-1.5B-RL	✓	RL	<u>41.0</u>	29.4	70.0	85.8	<u>55.6</u>	<u>56.4</u>

4.3 Token Efficiency Analysis

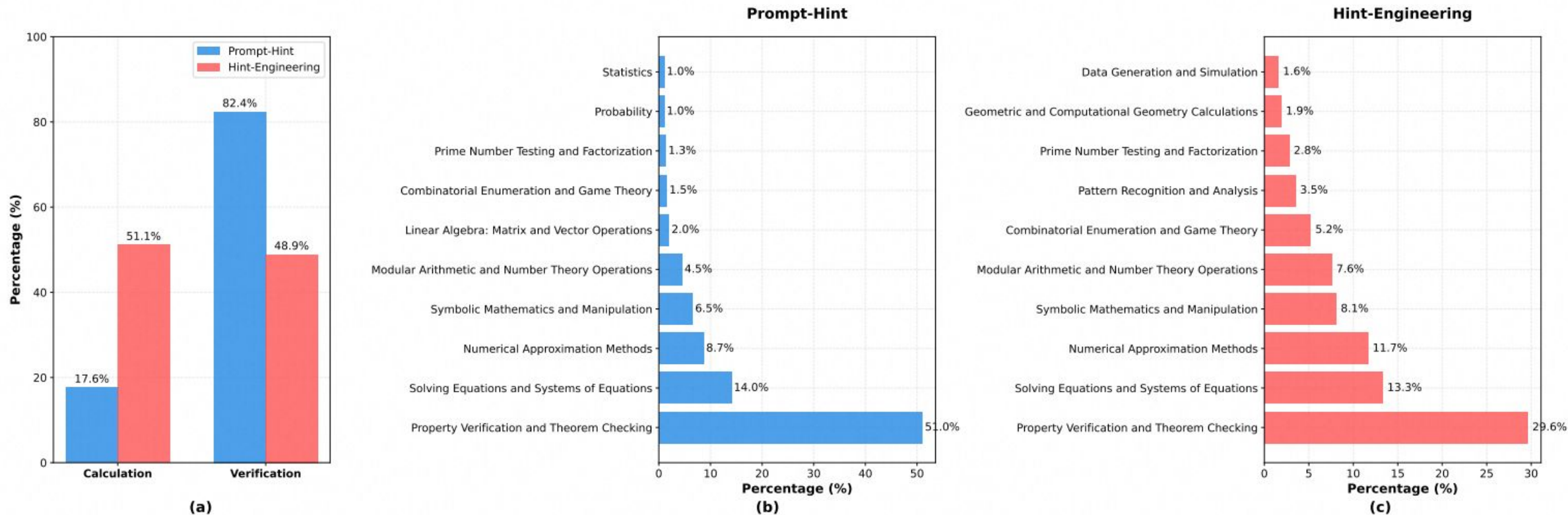


(a) Performance on AIME24 with different token budget

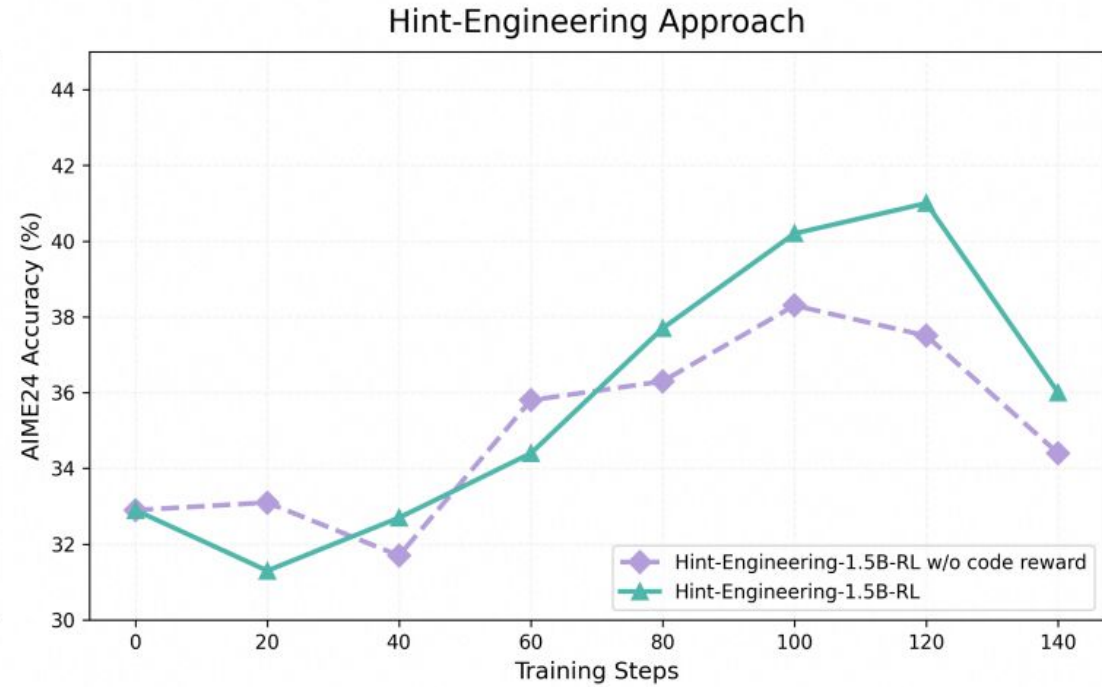
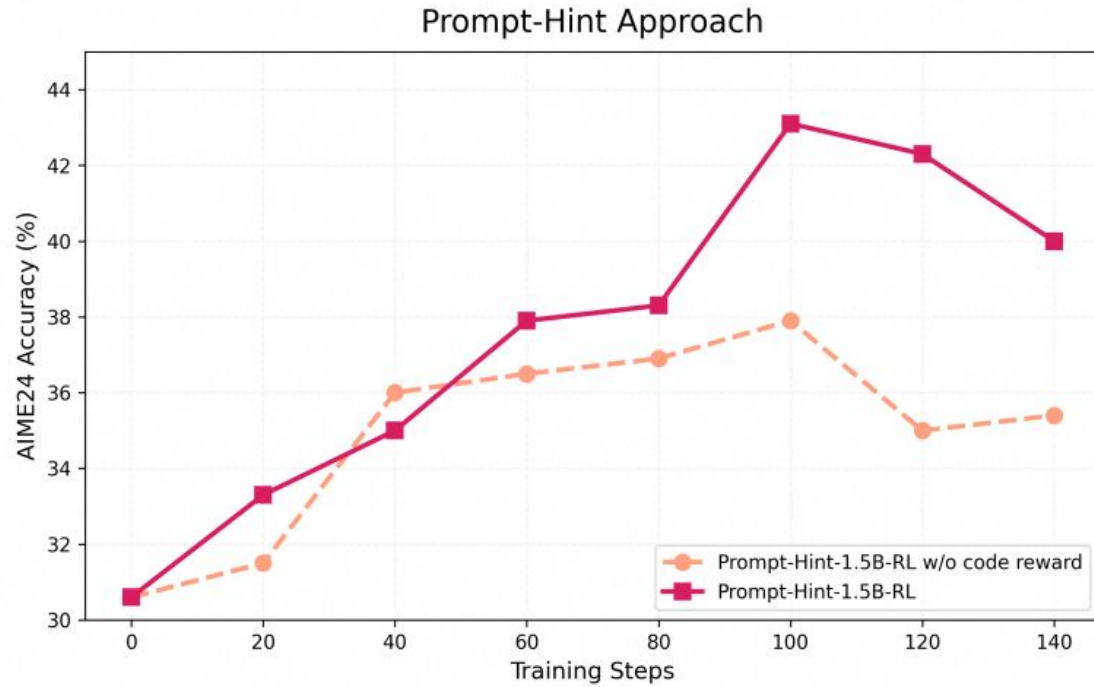


(b) Average token usage on AIME24

4.3 Code Behavior Analysis



4.4 Code Reward in RL



□ Conclusion

- High-quality data with optimal code behavior patterns can match or exceed the performance of larger datasets, while reinforcement learning significantly improves performance beyond SFT, particularly for smaller models.
- The Hint-Engineering approach achieves remarkable efficiency, reducing token usage by 30-50% while maintaining competitive performance.
- Moreover, RL shapes code usage behavior toward either efficiency or increased integration.

Thanks