

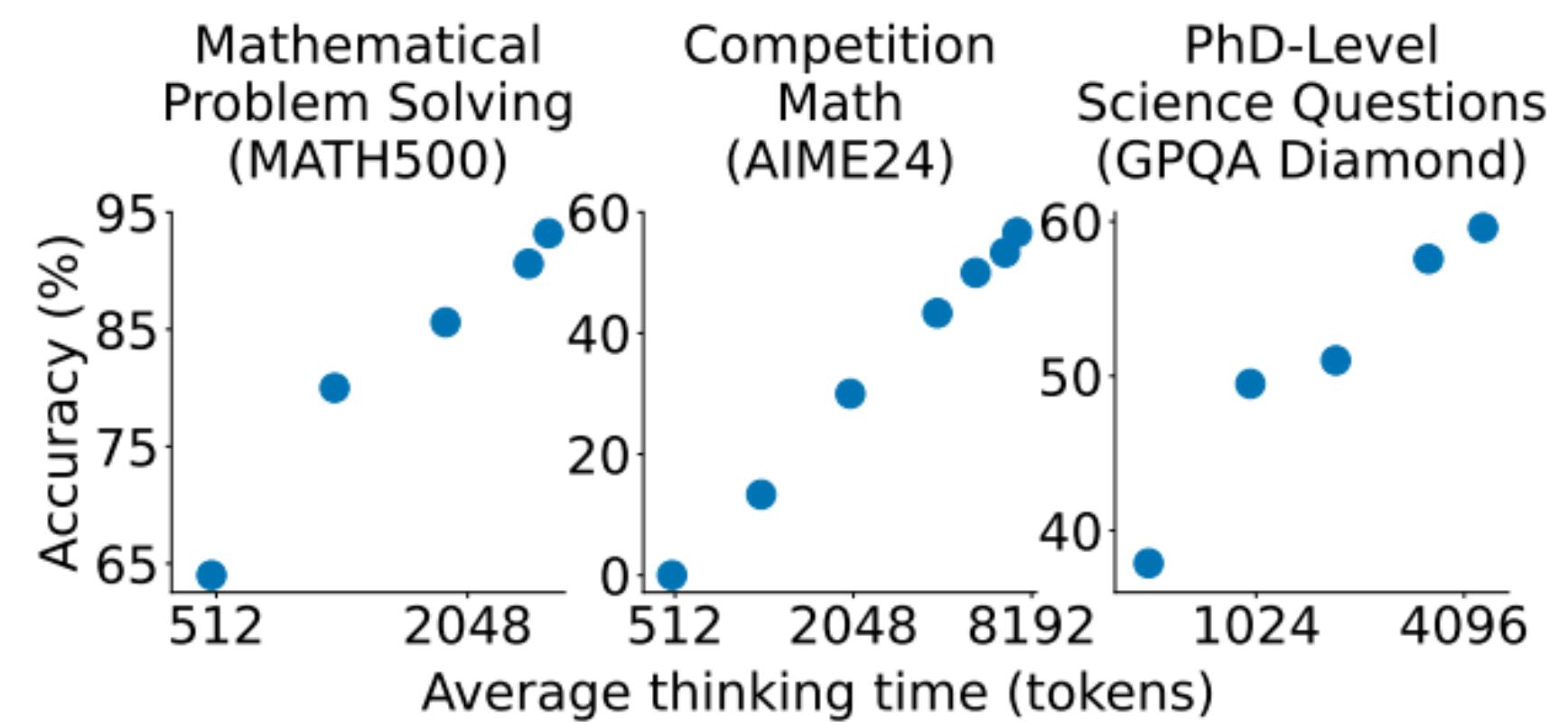
# Test-Time Optimization for Small Large Language Models (SLMs)

By - Yash, Akanksha

# Problem Statement

**Challenge:** LLMs struggle with complex reasoning despite having relevant knowledge.

**Key Factor:** Test-time compute allocation significantly affects reasoning performance but remains underexplored.



*Figure 1. Test-time scaling with s1-32B.* We benchmark s1-32B on reasoning-intensive tasks and vary test-time compute.

# Recent Work

"S1: Simple test-time scaling" (Muennighoff et al., 2025) proposed "budget forcing" as an effective strategy.

How many r in raspberry?

Question

Let's break down the process of counting the letter 'r' in the word "raspberry" ...

- \* First letter: 'r' - This is an 'r', count = 1.
- \* Second letter: 'a' - Not an 'r', count remains 1 ...
- \* Sixth letter: 'e' - Not an 'r', count remains 1.
- \* Seventh letter: 'r' - This is an 'r', count = 2.
- \* Eighth letter: 'y' - Not an 'r', count remains 2 ...

The number of 'r's in "raspberry" is 2.

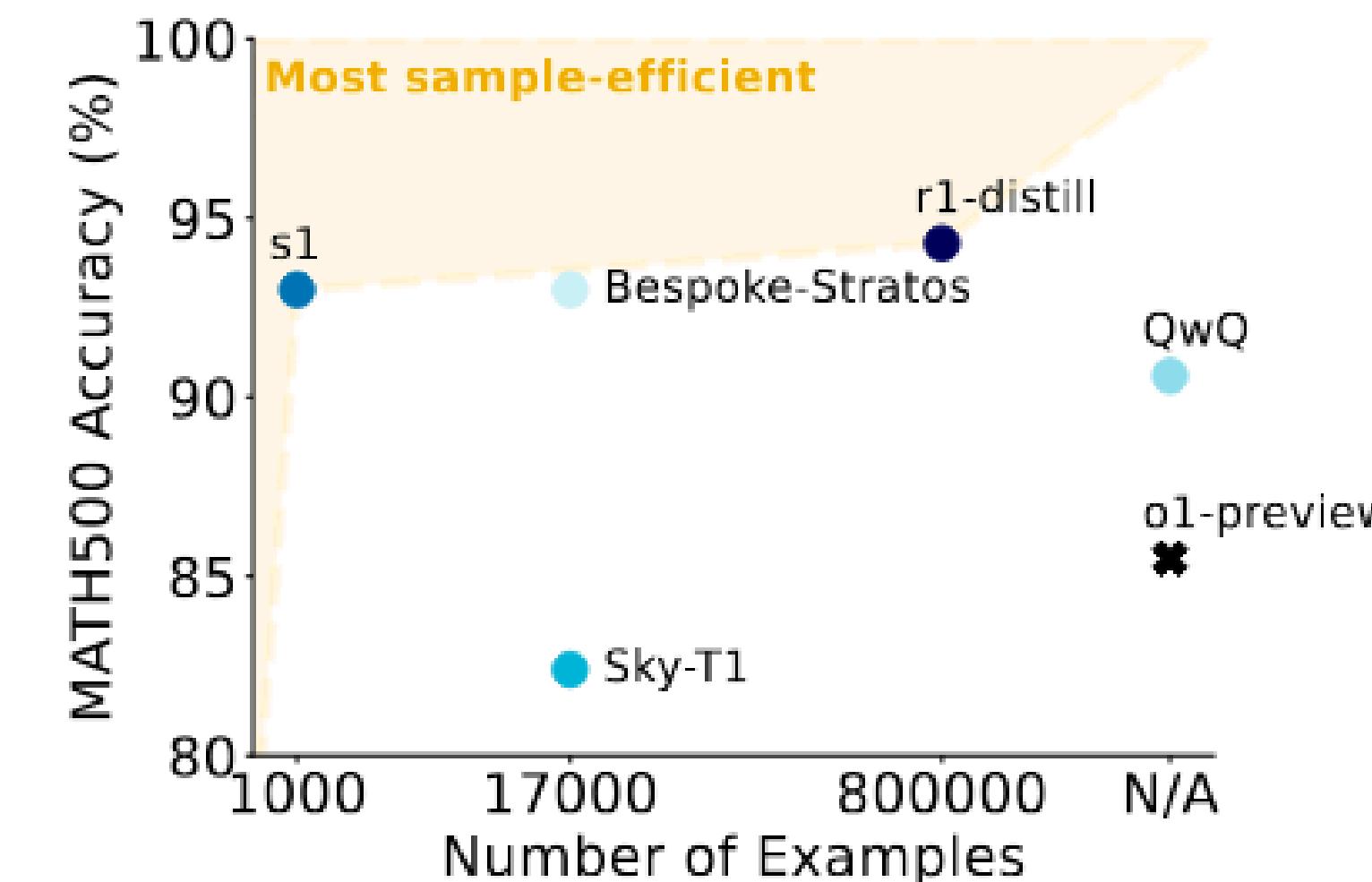
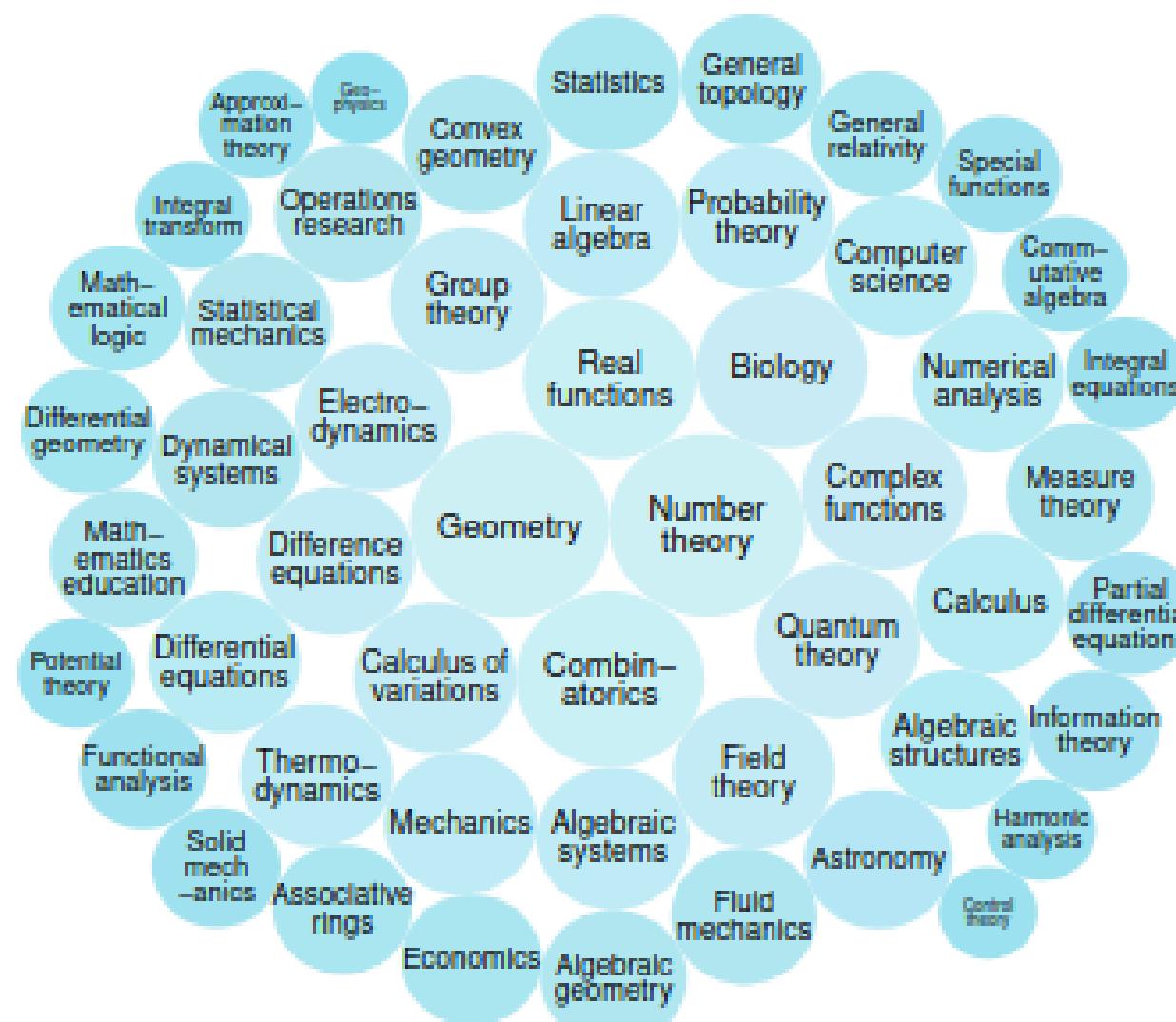
**Wait**, let's re-read the question carefully. It asks "How many r in raspberry?" ... \* r - a - s - p - b - e - r - r - y ... \* First 'r' ... \* Second 'r' ... \* Third 'r' ... Count = 3 ...

Reasoning trace

My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is **3**

Response

**"S1: Simple test-time scaling"** (Muennighoff et al., 2025) proposed **"budget forcing"** as an effective strategy.



*Figure 2. s1K and s1-32B.* (left) s1K is a dataset of 1,000 high-quality, diverse, and difficult questions with reasoning traces. (right) s1-32B, a 32B parameter model finetuned on s1K is on the sample-efficiency frontier. See Table 1 for details on other models.

# More Recent Work

**Chain of Draft (CoD)** introduced a method aimed at "Thinking Faster by Writing Less"

**Thinking-Optimal Scaling** explored the impact of scaling the length of Chain of Thoughts (CoTs) on reasoning performance

# Dataset

- **MATH-500:** Mathematical reasoning problems.
- **AIME24:** Competition-level Olympiad mathematics problems.
- **GPQA:** Graduate-level science questions and answers.

# **Background and Motivation**

# What makes a model a Reasoning Model?

The difference between Instruct and Reasoning Models lies in their output generation. Reasoning Models typically follow a fixed format: they start with <think>, generate reasoning tokens, close with </think>, and then provide the final answer within <answer> </answer>. This reasoning process is akin to a Chain-of-Thought (COT) Trajectory.

# How is Deepseek R1 different from other Reasoning Models?

First thing we agree is that the number of thinking tokens generated by Deepseek R1 is much greater than any other reasoning model present at this time. The primary reason for this is repetitive Self-Reflection in its Thinking Trace. We have evidences of this on Deepseek R1 using the prompt :

Let  $x, y$  and  $z$  be positive real numbers that satisfy the following system of equations:

$$\log_2\left(\frac{x}{yz}\right) = \frac{1}{2}$$

$$\log_2\left(\frac{y}{xz}\right) = \frac{1}{3}$$

$$\log_2\left(\frac{z}{xy}\right) = \frac{1}{4}$$

Then the value of  $\log_2(x^4y^3z^2)$  is  $\frac{m}{n}$  where  $m$  and  $n$  are relatively prime positive integers. Find  $m+n$ .

# How is Deepseek R1 different from other Reasoning Models?

Simplify that:

$$y = y * z^2 * 2^{(1/2 + 1/3)}$$

Wait, so  $y = y * z^2 * 2^{(5/6)}$ . Hmm, divide both sides by  $y$  (since  $y$  is positive and non-zero), we get:

$$1 = z^2 * 2^{(5/6)}$$

So the value is  $25/8$ , where  $m = 25$  and  $n=8$ . Since  $25$  and  $8$  are coprime,  $m + n = 33$ .

Wait, let me double-check the calculations.

First,  $\log_2(x^4y^3z^2) = 4 \log_2x + 3 \log_2y + 2 \log_2z$ .

$\log_2x = -7/24$ , so  $4*(-7/24) = -28/24 = -7/6 \approx -1.1667$

# Budget Forcing (Forcing the Aha Moment)

In Brief, Budget Forcing is letting model generate whatever its generating till end-of-text (`<lim_endl>`) is generated and replacing by Wait letting the model to generate more tokens.

How many r in raspberry?

Question

Let's break down the process of counting the letter 'r' in the word "raspberry" ...

- \* First letter: 'r' - This is an 'r', count = 1.
- \* Second letter: 'a' - Not an 'r', count remains 1 ...
- \* Sixth letter: 'e' - Not an 'r', count remains 1.
- \* Seventh letter: 'r' - This is an 'r', count = 2.
- \* Eighth letter: 'y' - Not an 'r', count remains 2 ...

The number of 'r's in "raspberry" is 2.

**Wait**, let's re-read the question carefully. It asks "How many r in raspberry?" ... \* r - a - s - p - b - e - r - r - y ... \* First 'r' ... \* Second 'r' ... \* Third 'r' ... Count = 3 ...

Reasoning trace

My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is **3**

Response

*Figure 3. Budget forcing with s1-32B.* The model tries to stop after "...is 2.", but we suppress the end-of-thinking token delimiter instead appending "Wait" leading **s1-32B** to self-correct its answer.

# Chain of Draft: Thinking Faster by Writing Less

**Silei Xu<sup>†</sup>, Wenhao Xie, Lingxiao Zhao, Pengcheng He**  
Zoom Communications

Standard
Answer the question directly. Do not return any preamble, explanation, or reasoning.
Chain-of-Thought
Think step by step to answer the following question. Return the answer at the end of the response after a separator #####.
Chain-of-Draft
Think step by step, but only keep a minimum draft for each thinking step, with 5 words at most. Return the answer at the end of the response after a separator #####.

Model	Prompt	Accuracy	Token #	Latency
GPT-4o	Standard	53.3%	1.1	0.6 s
	CoT	95.4%	205.1	4.2 s
	CoD	91.1%	43.9	1.0 s
Claude 3.5 Sonnet	Standard	64.6%	1.1	0.9 s
	CoT	95.8%	190.0	3.1 s
	CoD	91.4%	39.8	1.6 s

Table 1: GSM8K evaluation results.

# **Methodology And Experiments**

# Baseline

- **Model :** Qwen-2.5-1.5B-Instruct
- **Prompting :**
  - Zero Shot
  - Standard
  - Chain of Thought
  - Chain of Draft
- **Model :** DeepSeek-R1-Distill-Qwen-1.5B
- **Prompting :**
  - Zero Shot (Reasoning)

# Benchmark Dataset

- **Benchmarks:**
  - **MATH-500** (Mathematical reasoning)
  - **AIME24** (Olympiad math problems)
  - **GPQA** (Graduate-level Science Q&A)

# Model Performance

Table 1: Model Accuracy Comparison with Improved Formatting<sup>a,b</sup>

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	3.33	18.97	25.00
		Zero Shot	0.00	19.20	35.80
		CoT	6.67	17.86	38.00
		CoD	0.00	20.76	26.60
	DeepSeek-R1-Distill-Qwen-1.5B	Zero Shot	3.33	5.36	45.20
<b>Progressive Refinement Experiments</b>					
	Plain Qwen-1.5B	Standard	3.33	18.97	25.00
		Zero Shot	0.00	19.20	35.80
		CoT	6.67	17.86	38.00
		CoD	0.00	20.76	26.60
	DeepSeek-R1-Distill-Qwen-1.5B	Standard	3.33	5.36	45.20
		Zero Shot	3.33	5.36	45.20

<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings inferences

Note: All accuracy values are percentages. Shading indicates major category groups.

# Model Average Response Token Count

Table 2: Response Token Counts Comparison<sup>a,b</sup>

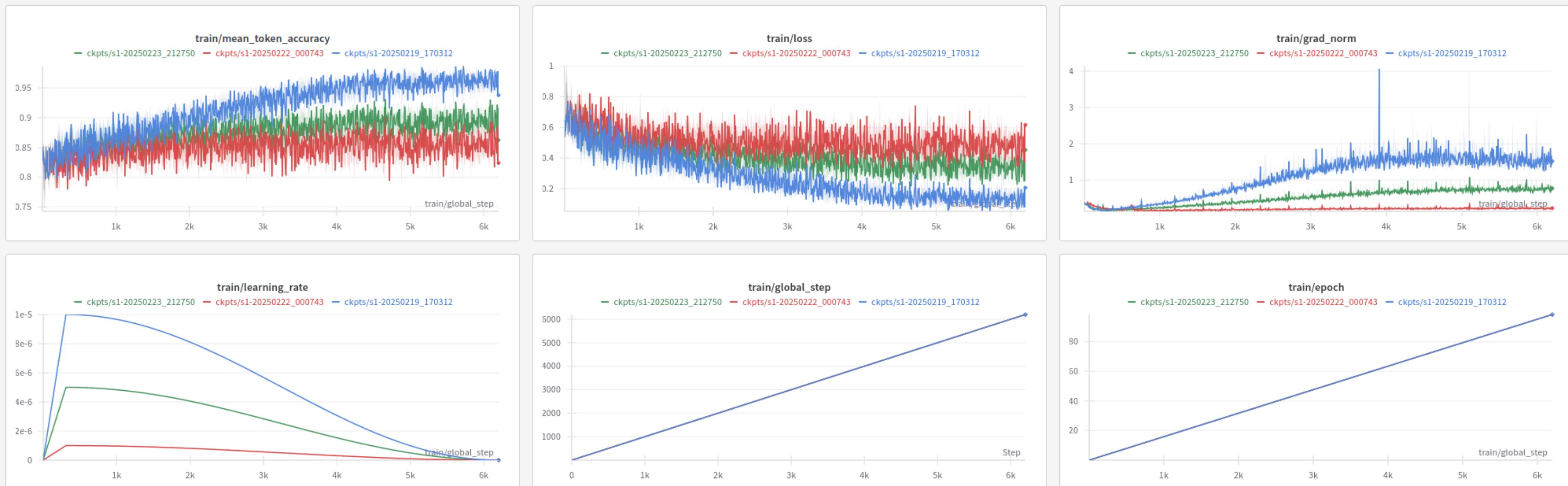
Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	762	405	852
		Zero Shot	830	807	753
		CoT	1592	775	801
		CoD	444	813	384
	DeepSeek-R1-Distill-Qwen-1.5B	R1	8308	7357	6561
<b>Expressive Reasoning Experiments</b>					
	Qwen 2.5-1.5B-instruct	Standard	762	405	852
		Zero Shot	830	807	753
		CoT	1592	775	801
		CoD	444	813	384
	DeepSeek-R1-Distill-Qwen-1.5B	R1	8308	7357	6561

<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates.

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings in inference.

# Making our own Reasoning Model using SFT

# Fine tuning Qwen 2.5-1.5B-instruct using SFT on S1K Dataset



# SFT Experiment

I used same hyper parameters as used in S1 Paper but with 4 Bit Quantization and Lora Adapters.

And Model in the paper : **Qwen-2.5-32B-instruct**

Model we used : **Qwen-2.5-1.5B-instruct**

# Number of Model Responses that either Got into Infinite loop or followed the Reasoning Format

Model (LR)	AIME24			MATH500			GPQA		
	Loops	Tokens	Format	Loops	Tokens	Format	Loops	Tokens	Format
SFT_2e4	24	7821	8	228	4418	220	133	6039	367
SFT_1e4	23	7685	7	253	6130	195	143	4623	357
SFT_1e5	14	5338	0	78	2357	0	76	2598	0
SFT_5e6	6	2840	0	51	1544	0	34	1552	0
SFT_1e6	1	1311	0	22	974	0	4	796	0

Note: 'Loops' refers to the Number of Responses that went into infinite repeating Loop. 'Tokens' refers to the Model Average Response Token Count. 'Format' refers to the Number of Model Responses following the Thinking Output Format.

Model Finetuned : Qwen-2.5-1.5B-instruct

# Model Performance

Table 1: Model Accuracy Comparison with Improved Formatting<sup>a,b</sup>

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	3.33	18.97	25.00
		Zero Shot	0.00	19.20	35.80
		CoT	6.67	17.86	38.00
		CoD	0.00	20.76	26.60
	DeepSeek-R1-Distill-Qwen-1.5B	Zero Shot	3.33	5.36	45.20
<b>Fine-tuned Models</b>					
	Blame-2.1B-instruct <sup>b</sup>	GRPO (temp: 0.8)	0.00	33.80	33.80
		GRPO (temp: 0.8)	0.00	37.86	39.80
	Qwen 2.5-1.5B-instruct <sup>a</sup>	SFT (2e-4)	3.33	12.05	35.20
		SFT (1e-4)	0.00	13.39	29.00
		SFT (1e-5)	0.00	16.74	34.00
		SFT (5e-6)	0.00	20.54	38.00
		SFT (1e-6)	3.33	17.19	35.40
<b>Progressive Reasoning Expansion</b>					
	Phi-4-mm-instruct	Zero shot	3.33	30.13	31.40
		No Wait	0.00	24.10	33.80
		1 Wait	0.00	23.88	33.00
		2 Wait	0.00	26.23	36.00
		3 Wait	0.00	27.00	34.00

<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings inferences

Note: All accuracy values are percentages. Shading indicates major category groups.

# Model Average Response Token Count

Table 2: Response Token Counts Comparison<sup>a,b</sup>

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	762	405	852
		Zero Shot	830	807	753
		CoT	1592	775	801
		CoD	444	813	384
	DeepSeek-R1-Distill-Qwen-1.5B	R1	8308	7357	6561
<b>Fine-tuned Models</b>					
	Qwen 2.5-1.5B-instruct <sup>a</sup>	SFT (2e-4)	7821	6039	4418
		SFT (1e-4)	7685	4623	6130
		SFT (1e-5)	5338	2598	2357
		SFT (5e-6)	2840	1552	1544
		SFT (1e-6)	1311	796	974
<b>Progressive Reasoning Expansion</b>					
	Qwen 2.5-1.5B-instruct	Zero Shot	692	392	592
		No 1 Way	758	567	567
		1 Way	111	946	696
		2 Way	110	110	905
		3 Way	110	1432	1221

<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates.

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings in inference.

**Why Qwen was performing that way?**

**Why it is not following the System  
Prompt properly?**

**Why despite of fine tuning, it is a little  
rigid towards its default answer  
generation pattern?**

# Evidences

yuqaf1989 opened on Sep 19, 2024

...

## What is the issue?

model: qwen2.5:14b

qwen2.5 will ignore system prompt, always return his name is Qwen.

test code:

```
from llama_index.llms.ollama import Ollama
from llama_index.core.llms import ChatMessage

sys = ChatMessage(role="system", content="your name is tom.")
c = ChatMessage(role="user", content="who are you")
llm = Ollama(model="qwen2.5:14b", request_timeout=120.0)
llm.chat([sys, c]).message.content

### returns
# I am Qwen, a large language model created by Alibaba Cloud. I am here to help with generating text, answering q

llm = Ollama(model="qwen2:7b", request_timeout=120.0)
llm.chat([sys, c]).message.content
### returns
#I am Tom, an AI assistant designed to provide information and assistance across a wide range of topics, answer q
```

# Evidences

A screenshot of a web browser window. The address bar shows the URL [qwenlm.github.io/blog/qwen2.5/](https://qwenlm.github.io/blog/qwen2.5/). The main content area displays a heading "Get ready to unlock a world of possibilities with our extensive lineup of models! We're excited to share these cutting-edge models with you, and we can't wait to see the incredible things you'll achieve with them!" Below this, there is a section titled "Takeaways" with text about the models' capabilities and resilience. A purple logo is visible on the left side of the page.

Get ready to unlock a world of possibilities with our extensive lineup of models! We're excited to share these cutting-edge models with you, and we can't wait to see the incredible things you'll achieve with them!

## Takeaways

In terms of **Qwen2.5**, the language models, all models are pretrained on our latest large-scale dataset, encompassing up to **18 trillion** tokens. Compared to Qwen2, Qwen2.5 has acquired significantly more knowledge (MMLU: 85+) and has greatly improved capabilities in coding (HumanEval 85+) and mathematics (MATH 80+). Additionally, the new models achieve significant improvements in instruction following, generating long texts (over 8K tokens), understanding structured data (e.g, tables), and generating structured outputs especially JSON.

**Qwen2.5 models are generally more resilient to the diversity of system prompts**, enhancing role-play implementation and condition-setting for chatbots. Like Qwen2, the Qwen2.5 language models support up to **128K** tokens and can generate up to **8K** tokens. They also maintain multilingual support for over **29** languages, including Chinese, English, French, Spanish, Portuguese, German, Italian, Russian, Japanese, Korean, Vietnamese, Thai, Arabic, and more. Below, we provide basic information about the models and details of the supported languages.

The specialized expert language models, namely **Qwen2.5-Coder** for coding and **Qwen2.5-Math** for mathematics, have undergone substantial enhancements compared to their predecessors, CodeQwen1.5 and Qwen2-Math. Specifically, Qwen2.5-Coder has been trained on **5.5 trillion** tokens of code-related data, enabling even smaller coding-specific models to deliver competitive performance against larger language models on



# Making our own Reasoning Model using GRPO

# GRPO

- Benefit, you don't need COT Reasoning in the labels, just input and output.
- The model will itself generate multiple solutions and the reward functions will give reward based on its generations.
- And the training proceeds, the model aligns with the rewards functions.

# GRPO

Two ways to use GRPO:

1. Train the model to follow the **<think>** Reasoning **</think>** format by first applying SFT, ensuring it learns to respond in the desired thinking format. Then, apply GRPO.
2. Use a system prompt template that specifies the required thinking format for the output.

# GRPO

---

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: **prompt**. Assistant:

---

Table 1 | Template for DeepSeek-R1-Zero. **prompt** will be replaced with the specific reasoning question during training.

**Source : Deepseek R1 Paper**

# GRPO

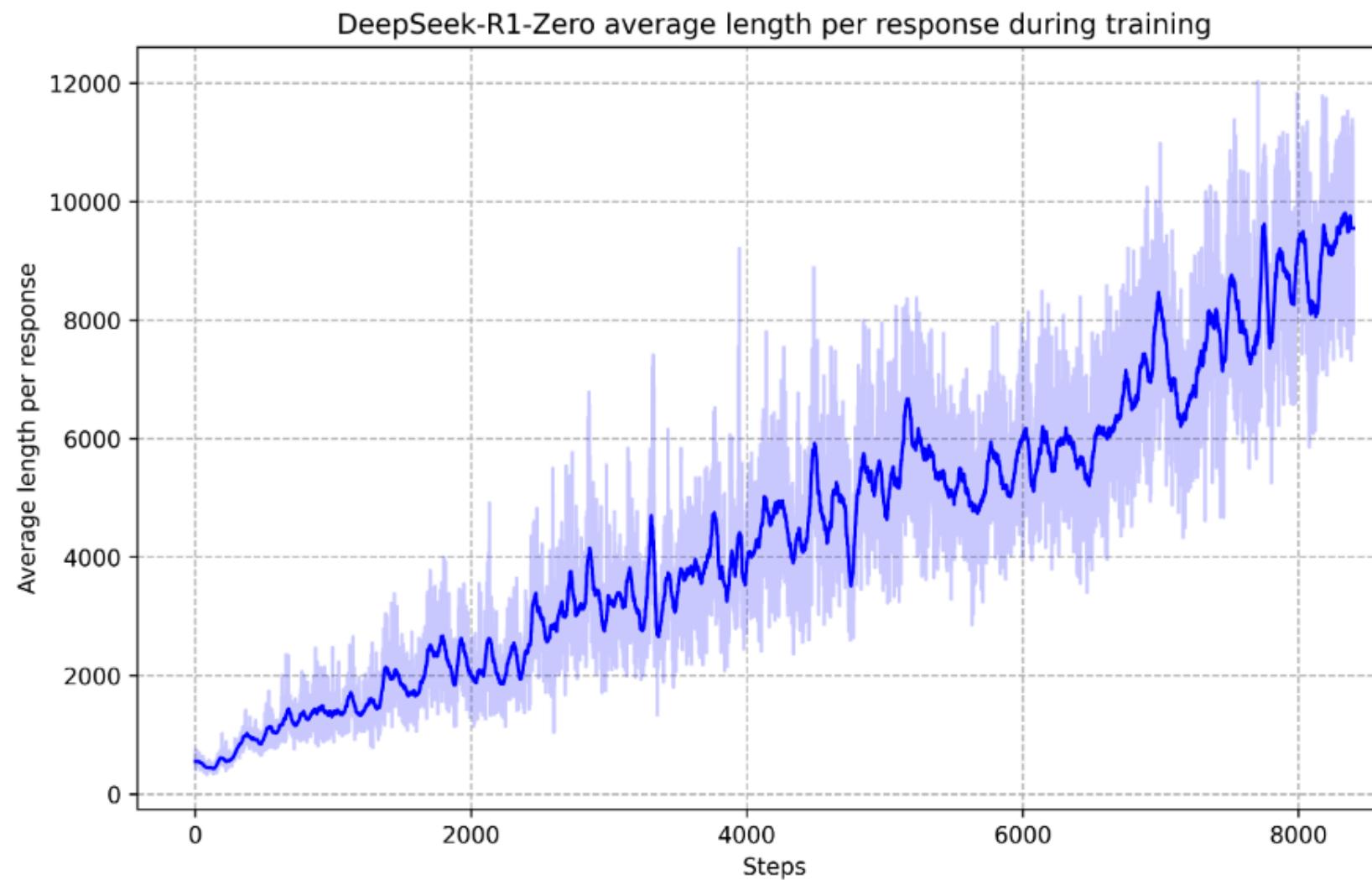
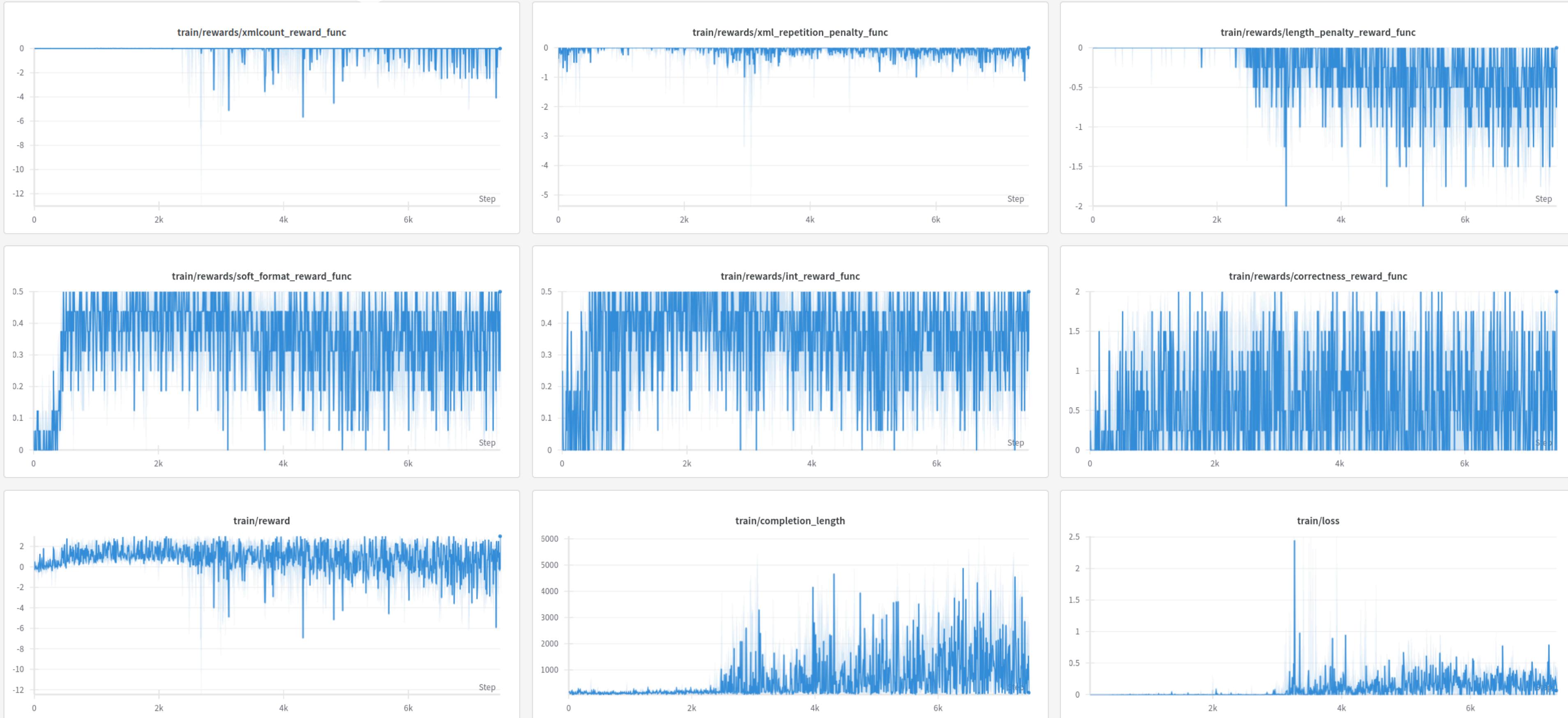


Figure 3 | The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time.

Source : Deepseek R1 Paper

# Fine tuning Llama 3.2-1B-instruct using GRPO on GSM8K Dataset



# **Reward Functions Used**

We used the following reward functions :

## **1. Penalizing Rewards**

- a. XMLCount
- b. XML Repetition Penalty
- c. Length Penalty

## **2. Positive Rewards**

- a. Soft Format Reward
- b. Int Reward
- c. Correctness Reward

# Model Performance

Table 1: Model Accuracy Comparison with Improved Formatting<sup>a,b</sup>

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	3.33	18.97	25.00
		Zero Shot	0.00	19.20	35.80
		CoT	6.67	17.86	38.00
		CoD	0.00	20.76	26.60
	DeepSeek-R1-Distill-Qwen-1.5B	Zero Shot	3.33	5.36	45.20
<b>Fine-tuned Models</b>					
	Llama-3.2-1B-instruct <sup>b</sup>	GRPO (temp 0.0)	0.00	19.87	33.80
		GRPO (temp 0.8)	0.00	17.86	29.80
	Qwen 2.5-1.5B-instruct <sup>a</sup>	SFT (2e-4)	3.33	12.05	35.20
		SFT (1e-4)	0.00	13.39	29.00
		SFT (1e-5)	0.00	16.74	34.00
		SFT (5e-6)	0.00	20.54	38.00
		SFT (1e-6)	3.33	17.19	35.40
<b>Progressive Reasoning Expansion</b>					
	Phi-4-mm-instruct	Zero shot	3.33	30.13	31.40
		No Wait	3.33	24.10	33.80
		1 Wait	3.33	23.88	33.00
		2 Wait	3.33	26.23	36.60
		3 Wait	3.33	27.00	34.00

<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings inferences

Note: All accuracy values are percentages. Shading indicates major category groups.

# Model Average Response Token Count

Table 2: Response Token Counts Comparison<sup>a,b</sup>

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	762	405	852
		Zero Shot	830	807	753
		CoT	1592	775	801
		CoD	444	813	384
	DeepSeek-R1-Distill-Qwen-1.5B	R1	8308	7357	6561

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Fine-tuned Models</b>					
	Llama-3.2-1B-instruct <sup>b</sup>	GRPO (temp 0.0)	2948	1247	1649
		GRPO (temp 0.8)	1184	567	751
	Qwen 2.5-1.5B-instruct <sup>a</sup>	SFT (2e-4)	7821	6039	4418
		SFT (1e-4)	7685	4623	6130
		SFT (1e-5)	5338	2598	2357
		SFT (5e-6)	2840	1552	1544
		SFT (1e-6)	1311	796	974



<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates.

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings in inference.

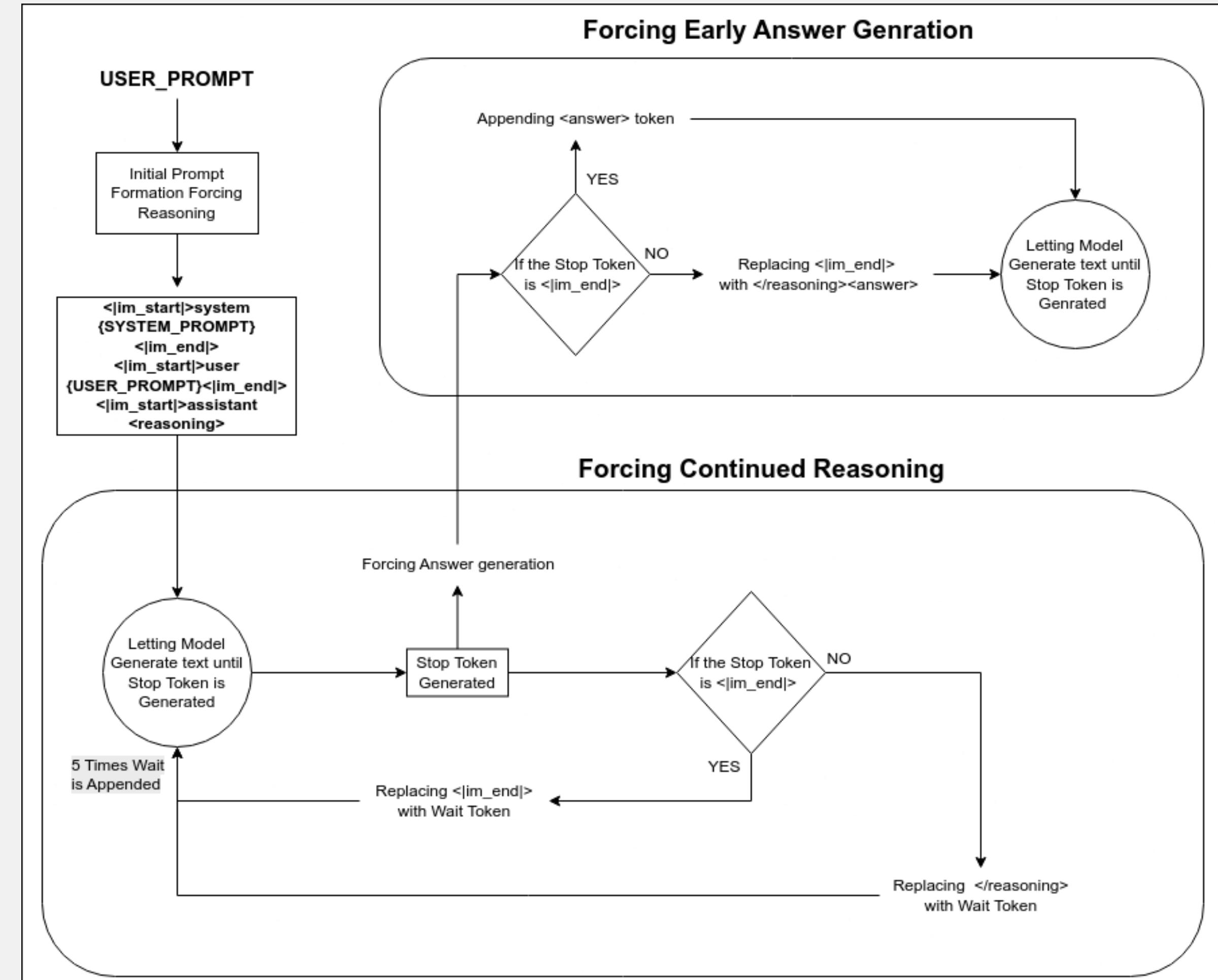
# Limitation Found: Budget Forcing on Small Models (Llama-3.2-1B)

- Forcing the 1B model to "think longer" by appending "Wait" resulted in unstable, gibberish output almost immediately (1-2 iterations).
- Suggests smaller models may lack the robustness needed for techniques that significantly alter the generation flow at test time.

# **Progressive Reasoning Expansion**

# Progressive Reasoning Expansion

- To better control test-time reasoning, I propose Progressive Reasoning Expansion. Instead of letting the model generate freely until <|im\_end|> (as in Budget Forcing), we force the first token to be <reasoning> and set the stop token as either <|im\_end|> or </reasoning>. If generation ends with <|im\_end|>, we replace it with </reasoning><answer> to force answer generation and also with Wait to encourage more reasoning. If it ends with </reasoning>, we append <answer> and again use Wait in parallel. This ensures consistent adherence to a reasoning-first generation format.



# Model Performance

Table 1: Model Accuracy Comparison with Improved Formatting<sup>a,b</sup>

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	3.33	18.97	25.00
		Zero Shot	0.00	19.20	35.80
		CoT	6.67	17.86	38.00
		CoD	0.00	20.76	26.60
	DeepSeek-R1-Distill-Qwen-1.5B	Zero Shot	3.33	5.36	45.20
<b>Fine-tuned Models</b>					
	Llama-3.2-1B-instruct <sup>b</sup>	GRPO (temp 0.0)	0.00	19.87	33.80
		GRPO (temp 0.8)	0.00	17.86	29.80
	Qwen 2.5-1.5B-instruct <sup>a</sup>	SFT (2e-4)	3.33	12.05	35.20
		SFT (1e-4)	0.00	13.39	29.00
		SFT (1e-5)	0.00	16.74	34.00
		SFT (5e-6)	0.00	20.54	38.00
		SFT (1e-6)	3.33	17.19	35.40
<b>Progressive Reasoning Expansion</b>					
	Phi-4-mini-instruct	Zero Shot	3.33	30.13	51.40
		No Wait	6.66	24.10	53.80
		1 Wait	6.66	23.88	58.00
		2 Wait	6.66	26.33	60.60
		3 Wait	10.00	27.00	64.00

<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings inferences

Note: All accuracy values are percentages. Shading indicates major category groups.

# Model Average Response Token Count

Table 2: Response Token Counts Comparison<sup>a,b</sup>

Category	Model	Setting	AIME24	GPQA	Math 500
<b>Prompting on Instruct Models</b>					
	Qwen 2.5-1.5B-instruct	Standard	762	405	852
		Zero Shot	830	807	753
		CoT	1592	775	801
		CoD	444	813	384
	DeepSeek-R1-Distill-Qwen-1.5B	R1	8308	7357	6561
<b>Fine-tuned Models</b>					
	Llama-3.2-1B-instruct <sup>b</sup>	GRPO (temp 0.0)	2948	1247	1649
		GRPO (temp 0.8)	1184	567	751
	Qwen 2.5-1.5B-instruct <sup>a</sup>	SFT (2e-4)	7821	6039	4418
		SFT (1e-4)	7685	4623	6130
		SFT (1e-5)	5338	2598	2357
		SFT (5e-6)	2840	1552	1544
		SFT (1e-6)	1311	796	974
<b>Progressive Reasoning Expansion</b>					
	Phi-4-mini-instruct	Zero Shot	687	642	462
		No Wait	784	753	567
		1 Wait	1136	916	696
		2 Wait	1418	1139	905
		3 Wait	1927	1432	1221

<sup>a</sup> SFT: Fine-tuned on S1K dataset with different learning rates.

<sup>b</sup> GRPO: Fine-tuned on GSM8K dataset with different temperature settings in inference.

# Takeaways

- 1. Format Adherence:** Small LMs (1–2 B parameters) often fail to stick to required output formats, derailing test-time optimizations like budget forcing.
- 2. Model Traits:** Core behaviors of the base model (e.g., Qwen-2.5 ignoring system prompts) can block prompt-based methods such as GRPO.
- 3. SFT Benefits:** Supervised fine-tuning can embed desired reasoning structures.
- 4. SFT Risks:** Without careful tuning, SFT on small LMs induces loops, verbosity, and instability despite format gains. Also Low learning rates don't teach <think> formats

# Takeaways (Continued)

- **GRPO Complexity:** Reward-based policy optimization demands precise reward design and is prone to instability or reward hacking in low-capacity models.
- **Instruction-Following Strength:** Models with better inherent instruction compliance (e.g. Phi-4) adapt more easily to budget forcing with minimal tuning.
- **Budget Forcing Depends on Baseline:** Encouraging longer chains (e.g. Minimum Budget) can spark self-correction but hinges on the model's initial reasoning coherence.

# Future Work

- Future work should prioritize the development of SLMs with improved inherent instruction following, reasoning capabilities, and training stability.
- These advancements would create a stronger foundation for applying test-time optimization techniques.
- Additionally, exploring simpler, potentially prompt-based, or more robust methods for controlling computational budget and inducing reflection in SLMs remains an important and promising research direction.

# **Some Extra Stuff**

# Evaluation the Runs

## LLM as a Judge

I tried the Qwen-2.5-1.5B-Instruct model.

**It literally gave a response like:**

{

Match: True,

Reasoning: "204 from the answer matches with 240  
from the model response."

}

# Total GPU Hours Used for Experiments

Used ADA for all the experiments

Total GPU Hours Used: 750+ GPU Hours

**Thank you  
very much!**