s1: Simple test-time scaling
Improving Reasoning by Spending More Time at Test-Time

Authors: Based in China (DeepSeek-AI)

Paper: https://arxiv.org/pdf/2501.19393
Code: https://github.com/simplescaling/s1

Presenter: Fae Gaze

Machine Learning Researcher in Bioinformatics

Presented by FG

Why This Paper Matters
A Sample-Efficient Reasoning Model

- **Problem:** Most large language models (LLMs) rely on **static reasoning** — what they know is fixed after training.
- **Goal:** Can we improve a model's reasoning just by letting it **think longer at test-time**, without retraining?
- **Solution:** This paper introduces **Budget Forcing**, a simple and effective method to do exactly that.
- **Efficiency:** s1-32B is trained on only **1,000 high-quality examples**, yet performs comparably to models trained on **800K+ samples**.
- **Significance:** Fully open-source, unlike OpenAI's o1 series.
- **Result:** Achieves **state-of-the-art performance** on reasoning benchmarks like AIME and GPQA.

Motivation

- Traditional LLMs are trained once → static outputs
- Can we get better performance at test-time without retraining?
- Inspired by OpenAI's o1 series — but fully open!

Key Points: Can you Run s1-32B Yourself?

- **It's open weight** – anyone can download and use it
- **But it's large** – needs ~60GB VRAM (not for regular laptops)
- **Alternatives:**
  - Run **quantized versions** (4-bit) on Mac using llama.cpp
  - Use **cloud services** (e.g. Colab, Hugging Face, Replicate)
- **Good for learning and research** – even if you can't run it fully
- **Great example of accessible AI** — open models help everyone explore reasoning

Core Idea

[**Training Phase**]

1000 Examples
(e.g., AIME, Science)

[s1-32B Model]

[Test-Time Reasoning]

Short Thinking Time
(Few Steps, Fast)

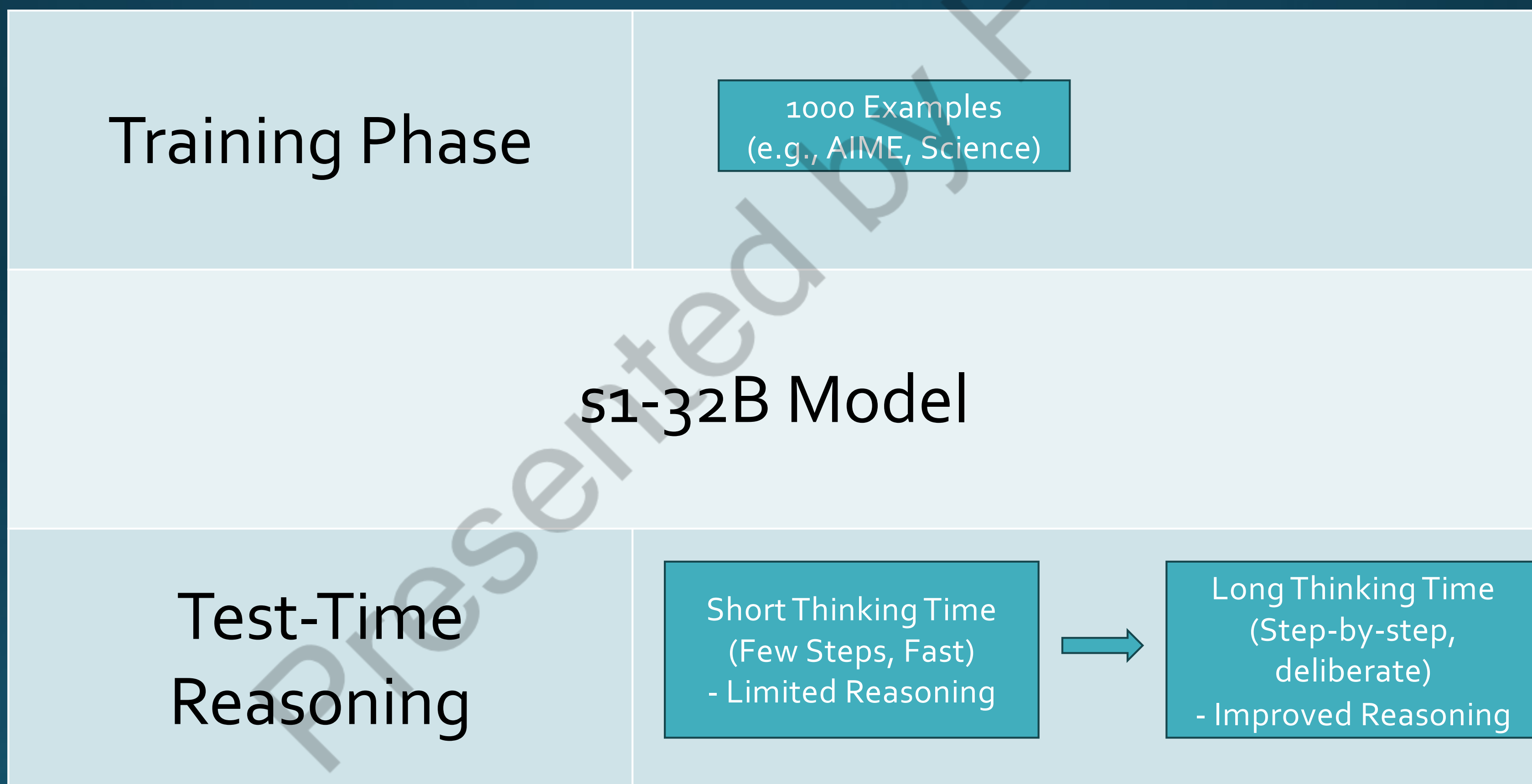
Long Thinking Time
(Step-by-step, deliberate)

[Limited Reasoning]

[Improved Reasoning]

by **controlling test-time thinking duration**

Core Idea


Training Phase
1000 Examples
(e.g., AIME, Science)
s1-32B Model
Test-Time
Reasoning
Short Thinking Time
(Few Steps, Fast)
- Limited Reasoning
Long Thinking Time
(Step-by-step,
deliberate)
- Improved Reasoning

What is s1-32B?

Inside s1-32B: A Reasoning-Optimized Language Model

Component	Description
Base Model	Qwen2.5-32B-Instruct(released by Alibaba/Qwen team)
Finetuning Samples	1,000 (s1K dataset) **carefully selected question–answer pairs**, each with a reasoning trace.
Distillation Source	**Gemini 2.0 Flash Thinking**
Finetuning Type	Supervised (next-token prediction)
Time to Train	16 NVIDIA H100 GPUs in parallel 26 minutes (7 GPU-hours)[(26*16 GPU)/(60)]=7GPU hours

Figure 1 Overview

Section	Content
Figure 1 Overview	Accuracy improves with more thinking token
	Tasks: MATH500, AIME24, GPQA Diamond
	Shows that more compute = more reasoning = higher scores
Figure 1 Benchmarks Used	30 elite math competition problems
	500 math problems from past contests
	198 PhD-level science questions

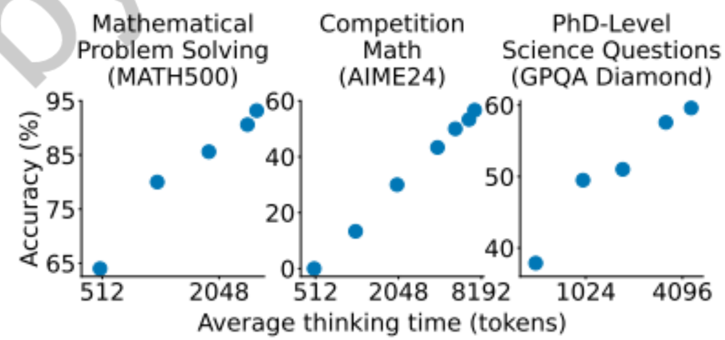


Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

Figure 2: Compared Table, S1K vs 59 vs r1 vs O1

Model	Training Data Size	Uses RL?	Compute Budget	Notes
s1K	1,000 hand-picked examples	No	**26 minutes** on 16×H100 GPUs	Best efficiency; curated reasoning examples; nearly matches 59K/full
59K	~59,000 examples	No	Not specified (likely a few hours on same hardware)	Good performance; brute-force data; no RL used
r1	~800,000+ examples (varies by stage); trained on **14.8 trillion tokens**	Yes	**Massive** — multi-stage training (pretrain + SFT + RL)	State-of-the-art open model; matches o1-level performance
o1	Unknown (closed)	Unknown	Unknown (assumed massive)	Closed benchmark; introduced test-time scaling; no training details

Presented by FG

Understanding the Core Metrics

Control:
100% = perfect control ✅
< 100% = model skips or overshoots ❌,

- Scaling:
- Positive slope = model gets better with more compute ✅
- A negative slope means accuracy gets worse — bad sign ❌.

$$\text{Control} = \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} \mathbb{I}(a_{\min} \le a \le a_{\max}) \quad (1)$$

$$\text{Scaling} = \frac{1}{\binom{|\mathcal{A}|}{2}} \sum_{\substack{a,b\in\mathcal{A}\\ b>a}} \frac{f(b)-f(a)}{b-a} \quad (2)$$

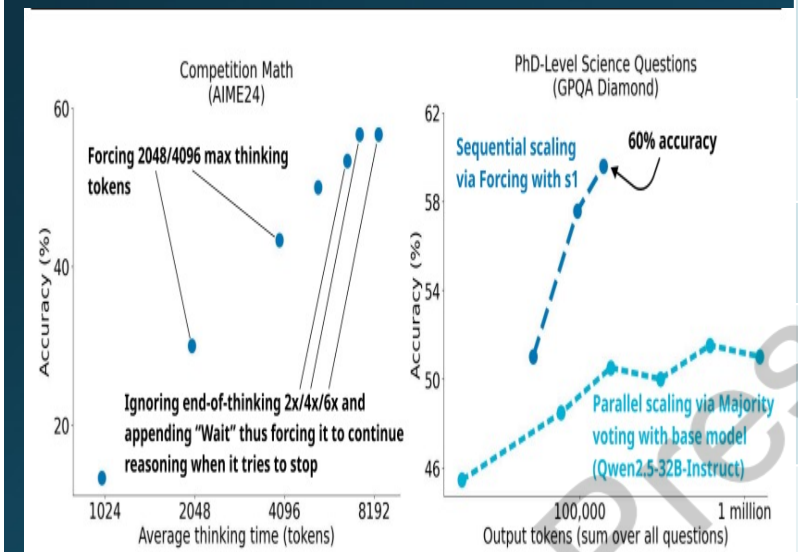
$$\text{Performance} = \max_{a\in\mathcal{A}} f(a) \quad (3)$$

Control: Does model stay within compute budget?
•A = different compute settings (e.g., 1000, 2000, 3000 tokens)
• a_{min}, a_{max} refer to a pre-specified minimum and maximum amount of test-time compute; in our case thinking tokens.
•f(b) and f(a) = accuracy at budget b and a
Scaling: Does accuracy improve with more tokens?
Performance: Max accuracy achieved on benchmark.

Budget Forcing (BF)

- Force the model to keep thinking
- Example: append “Wait” to delay final answer and expand its reasoning.
- Simple but powerful to control test-time compute.

Comparison of Test-Time Scaling Methods (Figure 4)



(a) Sequential scaling via budget forcing

(b) Parallel scaling via majority voting

Aspect	Sequential Scaling (s1 + Budget Forcing)	Parallel Scaling (Majority Voting)
Benchmark	AIME24 (Math) & GPQA (Science)	GPQA (Science)
Method	Forces model to keep thinking by adding "Wait"	Runs model multiple times and picks majority output
Tokens Used	1024 → 2048 → 4096 → up to 8192	2, 4, 8, 16, 32, 64 runs (total ~100K–1M output tokens)
Mechanism	Single, deeper chain of thought	Multiple shallow thoughts
Performance Trend	Steady improvement with more tokens	Slight improvement but plateaus
Peak Accuracy	**60% on GPQA**	~52–54% on GPQA
Strengths	Strong for logical, multi-step reasoning	Simple to implement, uses ensemble behavior
Limitations	May hit context window limit (e.g., 8K tokens)	Can't go deeper, even with more outputs
Example Prompt	"Let's think step-by-step... Wait."	"What's the answer?" (run many times)

Models Compared of Table 1

Category	Model Examples	LLM Type	Test-Time Scaled?	Open Weights?	Open Data?	Training Size	Remarks
API Only	o1, o1-mini, Gemini	Yes (LLM)	NA	NO	NO	NA	Powerful but closed. Test-time scaling results not fully public.
Open Weights	Qwen2.5-32B, r1	Yes	Qwen: No R1: Yes	Yes	NO	r1: 800K+ reasoning samples	r1 is a reasoning-optimized LLM; Qwen is general-purpose.
Open + Open Data	s1-32B, Bespoke-32B	Yes	Yes	Yes	Yes	s1: 1K curated examples	Fully reproducible

Overview of Table 2

- Compared other training setups:
- **Random 1K**: Grab 1,000 questions from anywhere — some easy, some useless
- **Diverse 1K**: Mix topics well, but not all are hard
- **Longest 1K**: Choose the most complicated-looking ones
- **Full 59K dataset:** From the same data pool that s1K was sampled from
- s1K (hand-curated) **outperformed** all 1K variants
- 59K model is stronger, but **much more expensive**

Model	AIME 2024	MATH 500	GPQA Diamond
1K-random	36.7 [-26.7%, -3.3%]	90.6 [-4.8%, 0.0%]	52.0 [-12.6%, 2.5%]
1K-diverse	26.7 [-40.0%, -10.0%]	91.2 [-4.0%, 0.2%]	54.6 [-10.1%, 5.1%]
1K-longest	33.3 [-36.7%, 0.0%]	90.4 [-5.0%, -0.2%]	59.6 [-5.1%, 10.1%]
59K-full	53.3 [-13.3%, 20.0%]	92.8 [-2.6%, 2.2%]	58.1 [-6.6%, 8.6%]
s1K	50.0	93.0	57.6

Figure 5 (Model Output Examples)

Benchmark	Reasoning Type	Final Answer
AIME24	Game theory	809
MATH500	Vector algebra	(16/49,...)
GPQA	Quantum Physics	h$\sqrt{\frac{k}{m}}(2nx + ny + 3/2)$

Main Methods Compared (Table 3)

Method	Description
BF (Budget Forcing)	Force the model to reason longer using prompts like "Wait"
TCC (Token Conditional Control)	Stop or continue reasoning based on token count
SCC (Step Conditional Control)	Control based on number of reasoning steps
CCC (Class Conditional Control)	Condition reasoning on task class/type
RS (Rejection Sampling)	Sample multiple outputs and keep the best

Test Time Scaling Methods of Table 3

Table 3. **Ablations on methods to scale test-time compute on AIME24.** $|\mathcal{A}|$ refers to the number of evaluation runs used to estimate the properties; thus a higher value indicates more robustness. **Bold** indicates our chosen method and the best values. BF = budget forcing, TCC/SCC/CCC = token/step/class-conditional control, RS = rejection sampling.

Method	Control	Scaling	Performance	$\|\mathcal{A}\|$
BF	**100%**	15	**56.7**	5
TCC	40%	-24	40.0	5
TCC + BF	**100%**	13	40.0	5
SCC	60%	3	36.7	5
SCC + BF	**100%**	6	36.7	5
CCC	50%	**25**	36.7	2
RS	**100%**	-35	40.0	5

	Method	Good at	Verdict
Budget Forcing	BF	Control, Scaling, Accuracy	Best Overall
Token Conditional Control	TCC	Nothing	Poor control, negative scaling, (Fails)
	TCC+BF	Control & Good Scaling	Still bad
Step Conditional Control	SCC	Small Scaling	Weak Scaling
	SCC+BF	Control	Still Weak
Class Conditional Control	CCC	Best Scaling	Good Scaling, Low score
Rejection Sampling	RS	Control	Inefficient(Wates Compute)

Budget Forcing vs. Soft Budget Forcing

Model	AIME 2024	MATH 500	GPQA Diamond
No extrapolation	50.0	**93.0**	57.6
2x without string	50.0	90.2	55.1
2x "Alternatively"	50.0	92.2	**59.6**
2x "Hmm"	50.0	**93.0**	**59.6**
2x "Wait"	**53.3**	**93.0**	**59.6**

- "Wait" (hard BF) is best for logic-heavy tasks like AIME
- "Hmm?" and "Alternatively" (soft BF) work just as well on science/math tasks like GPQA and MATH500
- No extrapolation = the model is allowed to naturally stop thinking when it wants — no additional prompting is added to force longer reasoning.
  "2x without string": You force the model to generate 2x its normal length

Why Rejection Sampling (RS) Fails

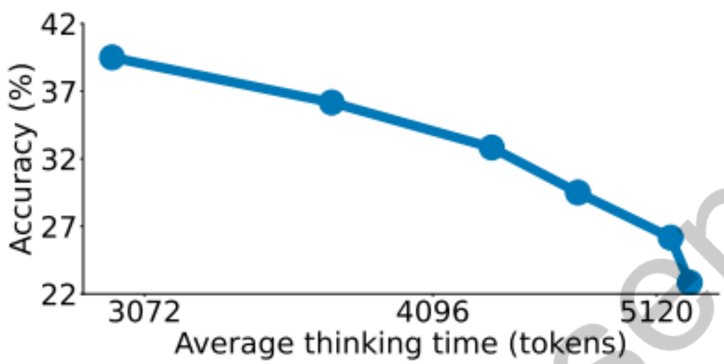


Figure 6. **Rejection sampling on AIME24 with s1-32B.** We sample with a temperature of 1 until all generations have less than (from left to right) 3500, 4000, 5000, 8000, and 16000 thinking tokens requiring an average of 655, 97, 8, 3, 2, and 1 tries per sample.

- We keep generating responses **while** they are **less than 3500**.
  Once we get one **greater than or equal to 3500**, we stop.

Target Tokens (N)	Average Thinking Time (tokens)	Average Tries Per Sample
3500	3072	**655 tries**
4000	4096	**97 tries**
5000	5120	**8 tries**
8000	8000	**3 tries**
3500	16000	**1 tries**

Two Paths — Parallel and Sequential Scaling

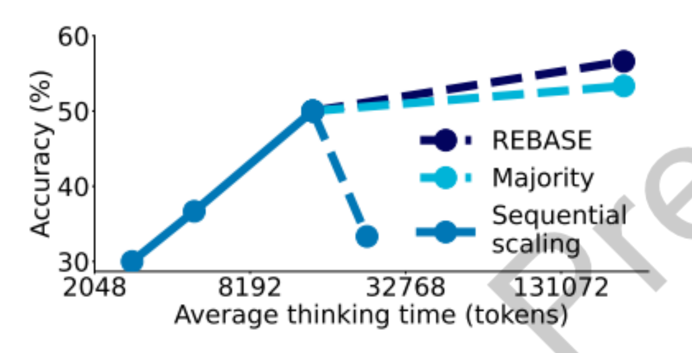
Method	What It Does	What It Does	What It Does	What It Does
Parallel Scaling	Runs multiple generations in parallel, then selects the best answer	- Majority Voting - REBASE((reward-based search)	Higher accuracy Robust to noise	Expensive (multiple forward passes)
Sequential Scaling	Generates and refines answer step-by-step in a single pass	- Budget Forcing ("Wait")	More efficient Competitive accuracy	Flattens at long token lengths (e.g., 32K+)

Presented by FG

Limits of Test-Time Scaling

- The **model's architecture limits** (e.g. context window)
- Lack of more reasoning paths
- Possible **infinite loops** when the model "waits" forever

Why Budget Forcing Eventually Drops? Why Doesn't More Thinking Always Help?



Method	Behavior with More Tokens	Accuracy Trend	Why It Drops or Flattens
Sequential Scaling	- Begins at 2048 tokens, goes up to ~32K+ - Runs reasoning in one long path	Improves until ~56–57%, then drops	1. **Model Saturation**: Runs out of useful knowledge 2. **Context Overflow**: Earlier logic gets forgotten 3. **"Wait" Trap**: Model loops or stalls
Majority Voting	- Aggregates 16 reasoning paths - Slight improvement over sequential	Slight boost over Sequential	Same issues as Sequential, but **more robust** due to path diversity
REBASE	- Uses a reward model to score 512 reasoning paths - Picks highest-quality answer	Peaks ~60% on AIME24	Flattens after 32K due to **diminishing returns** — more tokens yield less gain per token

Visualization from Fig 7

Method	Behavior with More Tokens	Accuracy Trend	Why It Drops or Flattens
Sequential Scaling	- Begins at 2048 tokens, goes up to ~32K+ - Runs reasoning in one long path	Improves until ~56–57%, then drops	1. **Model Saturation**: Runs out of useful knowledge 2. **Context Overflow**: Earlier logic gets forgotten 3. **"Wait" Trap**: Model loops or stalls
Majority Voting	- Aggregates 16 reasoning paths - Slight improvement over sequential	Slight boost over Sequential	Same issues as Sequential, but **more robust** due to path diversity
REBASE	- Uses a reward model to score 512 reasoning paths - Picks highest-quality answer	Peaks ~60% on AIME24	Flattens after 32K due to **diminishing returns** — more tokens yield less gain per token

Example Workflow:

1. **Generate 512 reasoning attempts** per question:
   - Some short, some long
   - Some correct, some incorrect

2. For each attempt G_i, compute:

$R(G_i)$=Reward model score

(High score = better alignment with expert reasoning)

3.Pick:

$$G^{*}=\arg\max_i R(G_i)$$

→ This becomes the final answer.

- **Why is REBASE Better?**
- It selects the *best quality answer*, not just the most frequent one.
- Combines **reward-guided search** + **aggregation**
- Helps avoid:
  - Circular reasoning
  - Overconfidence in wrong answers
  - Verbose fluff from Budget Forcing

Comparison of Strategies on Same Question

Attempt	Final Answer	Reasoning Steps	Reward Score	Notes
G_1	404	3	0.45	Fast, guessed
G_2	409	6	0.41	Wrong logic
G_{12}	809	9	0.92	Correct and aligned
G_{45}	809	20	0.75	Too long, but right

Limits of Test-Time Scaling

- Even with Budget Forcing, we hit a ceiling: performance **flattens out.**
- This is due to:
- The **model's architecture limits** (e.g. context window)
- Lack of more reasoning paths
- Possible **infinite loops** when the model "waits" forever

Scaling, Control, and Performance — The Big Picture

Metric	Why It Matters	Best Method
Scaling	Accuracy should increase with tokens	BF, CCC
Control	Keep thinking time within budget	BF, RS
Performance	Final accuracy on real benchmarks	REBASE, BF

Ranking Summary

Method	Scaling	Control	Accuracy	Notes
BF	✅	✅	✅ 56.7%	Efficient, simple, best all-rounder
Soft BF	✅	✅	~56%	Good on science, weaker on logic
REBASE	✅⭐	NA	✅⭐60%	Best accuracy, higher compute
Majority Vote	✅	NA	✅ 58%	Simpler than REBASE
CCC	✅	❌	❌ 36%	Cheats token limit
RS	❌	✅	❌ 27%	Inverse scaling, wasted compute

Presented by FG

Core Contributions of the Paper:

- **s1-32B model** trained on only **1,000 samples**
- Introduced **Budget Forcing** — controls test-time compute with "Wait" prompts
- Strong results on **math (AIME24)**, **science (GPQA)**, and **algebra (MATH500)**
- Defined **3 evaluation metrics**: Control, Scaling, Performance
- Showed failures of traditional methods (RS, TCC, CCC)
- Introduced **REBASE** — reward-based answer selection that reaches **60% AIME24 accuracy**


THANK YOU
PLEASE REACH ME AT
FOR ANY QUESTION