Inference Test Time Scaling Law

TEAM 5:

DANIEL SLYEPICHEV, ANANYA ANANDA, AADITYA GHOSALKAR , AKIRA DURHAM, SAHLAR SALEHI

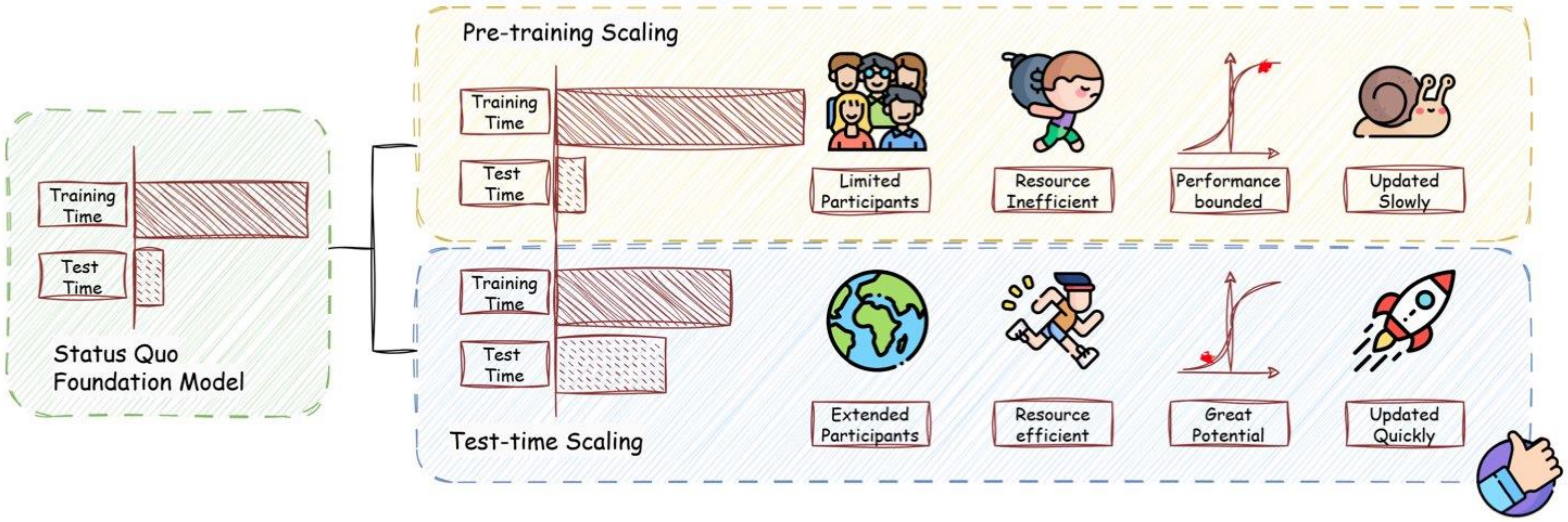
Daniel Slyepichev
dos8nw

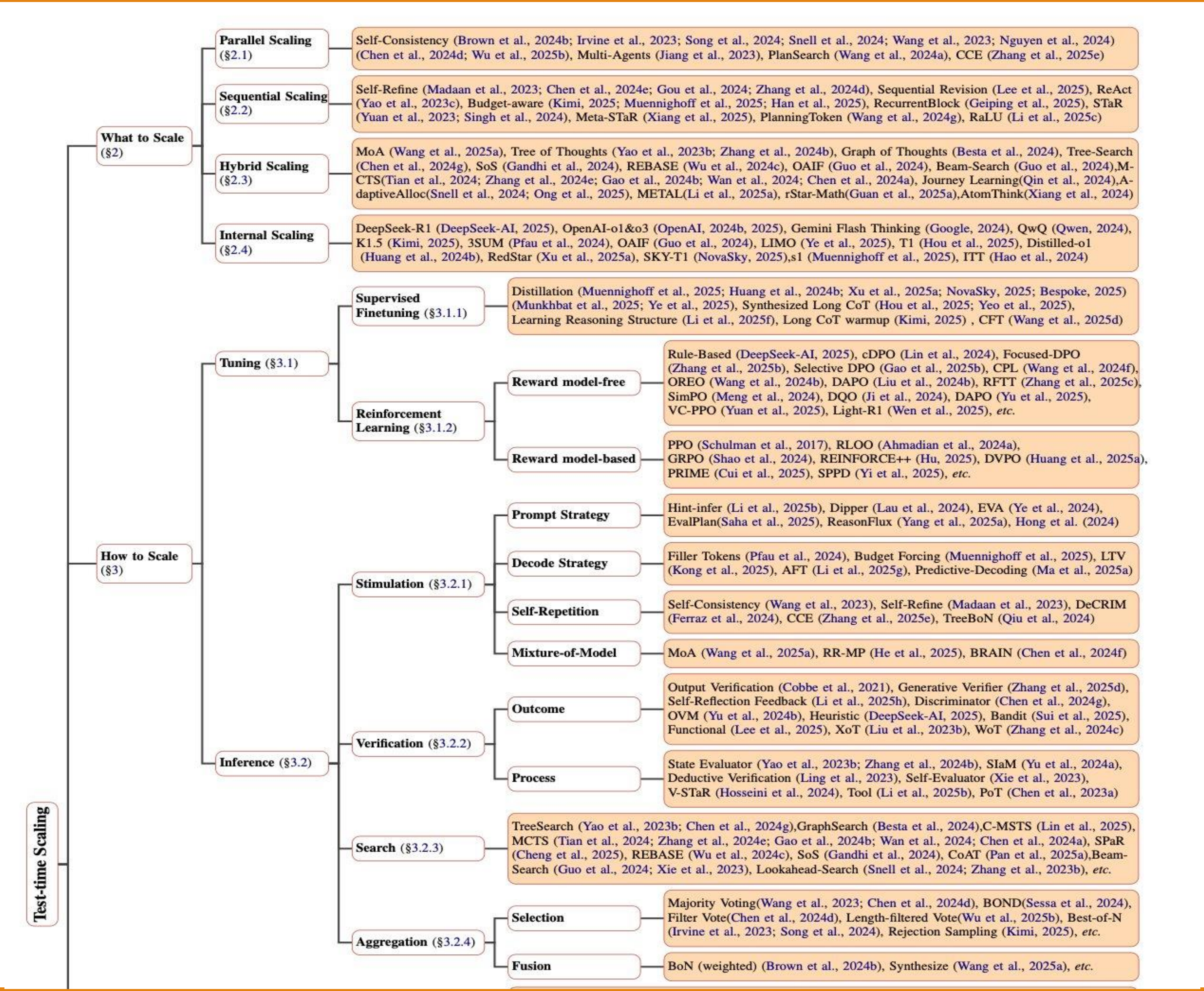
What, How, Where, and How Well? A Survey on Test-Time Scaling in Large Language Models

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, Chen Ma

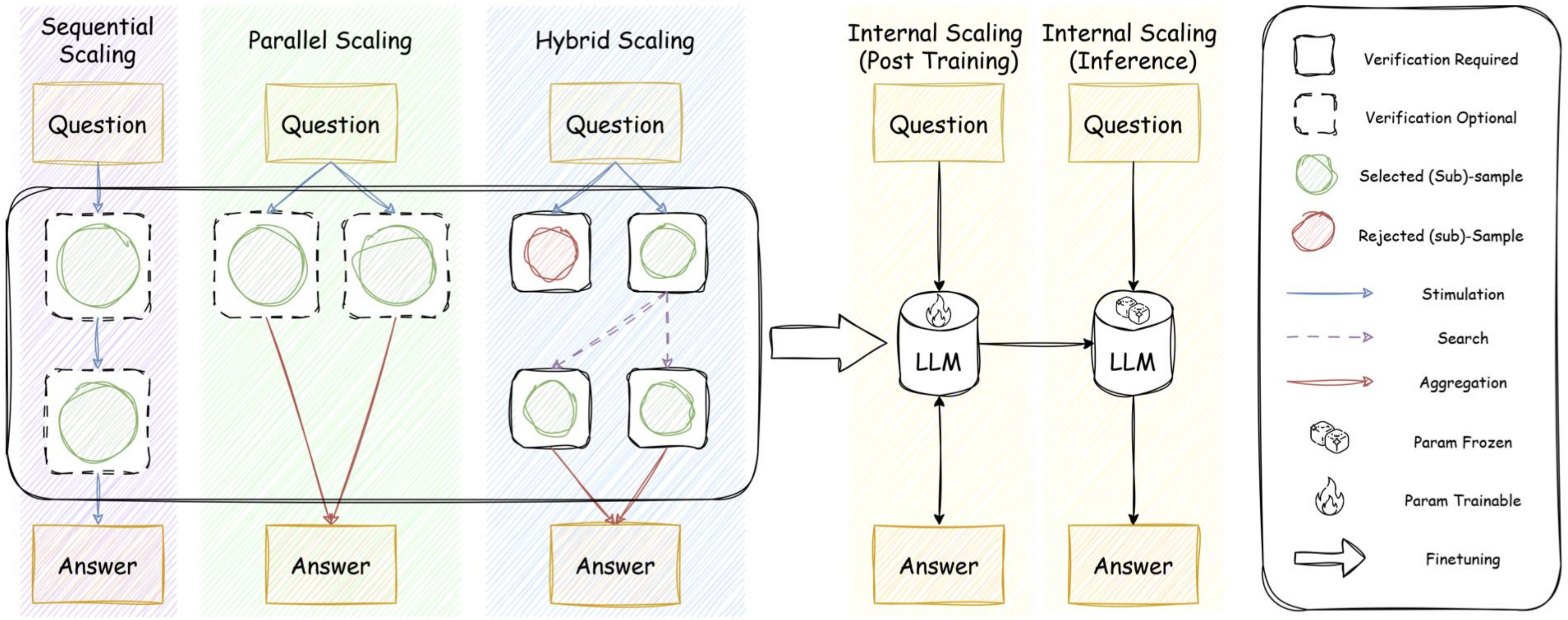
- Test-time scaling emerges as prominent research focus
- What is Test Time Scaling?
  - Allocate additional resources during inference (kind of like humans)
- Enabling breakthroughs in specialized and general tasks
  - OpenAI o1, DeepSeek's R1
- Need for comprehensive survey for systemic understanding
  - What to scale
  - How to scale
  - Where to scale
  - How well to scale

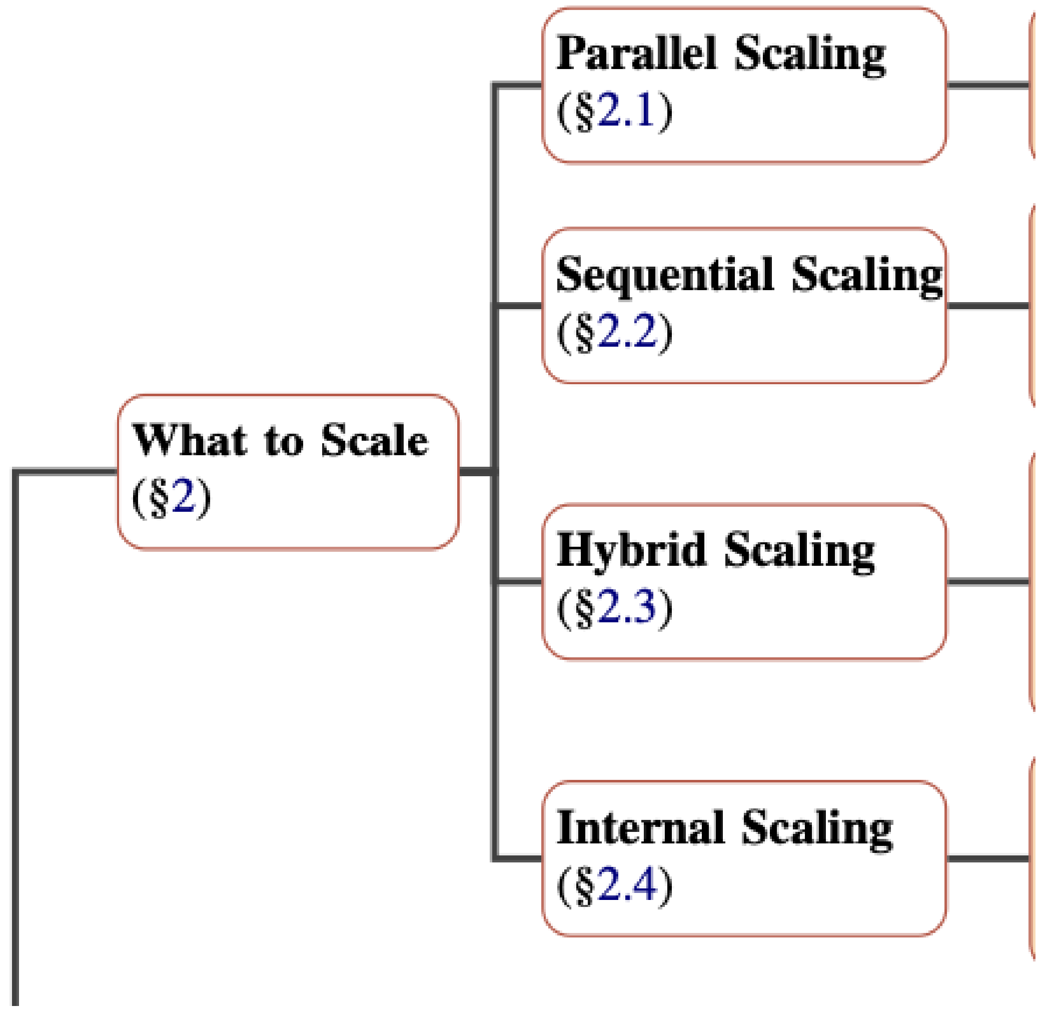
Test Time Scaling vs. Pre Training Scaling




Test-time Scaling
What to Scale (§2)
Parallel Scaling (§2.1)
Self-Consistency (Brown et al., 2024b; Irvine et al., 2023; Song et al., 2024; Snell et al., 2024; Wang et al., 2023; Nguyen et al., 2024) (Chen et al., 2024d; Wu et al., 2025b), Multi-Agents (Jiang et al., 2023), PlanSearch (Wang et al., 2024a), CCE (Zhang et al., 2025e)
Sequential Scaling (§2.2)
Self-Refine (Madaan et al., 2023; Chen et al., 2024e; Gou et al., 2024; Zhang et al., 2024d), Sequential Revision (Lee et al., 2025), ReAct (Yao et al., 2023c), Budget-aware (Kimi, 2025; Muennighoff et al., 2025; Han et al., 2025), RecurrentBlock (Geiping et al., 2025), STaR (Yuan et al., 2023; Singh et al., 2024), Meta-STaR (Xiang et al., 2025), PlanningToken (Wang et al., 2024g), RaLU (Li et al., 2025c)
Hybrid Scaling (§2.3)
MoA (Wang et al., 2025a), Tree of Thoughts (Yao et al., 2023b; Zhang et al., 2024b), Graph of Thoughts (Besta et al., 2024), Tree-Search (Chen et al., 2024g), SoS (Gandhi et al., 2024), REBASE (Wu et al., 2024c), OAIF (Guo et al., 2024), Beam-Search (Guo et al., 2024),M-CTS(Tian et al., 2024; Zhang et al., 2024e; Gao et al., 2024b; Wan et al., 2024; Chen et al., 2024a), Journey Learning(Qin et al., 2024),A-daptiveAlloc(Snell et al., 2024; Ong et al., 2025), METAL(Li et al., 2025a), rStar-Math(Guan et al., 2025a),AtomThink(Xiang et al., 2024)
Internal Scaling (§2.4)
DeepSeek-R1 (DeepSeek-AI, 2025), OpenAI-o1&o3 (OpenAI, 2024b, 2025), Gemini Flash Thinking (Google, 2024), QwQ (Qwen, 2024), K1.5 (Kimi, 2025), 3SUM (Pfau et al., 2024), OAIF (Guo et al., 2024), LIMO (Ye et al., 2025), T1 (Hou et al., 2025), Distilled-o1 (Huang et al., 2024b), RedStar (Xu et al., 2025a), SKY-T1 (NovaSky, 2025),s1 (Muennighoff et al., 2025), ITT (Hao et al., 2024)
How to Scale (§3)
Tuning (§3.1)
Supervised Finetuning (§3.1.1)
Distillation (Muennighoff et al., 2025; Huang et al., 2024b; Xu et al., 2025a; NovaSky, 2025; Bespoke, 2025) (Munkhbat et al., 2025; Ye et al., 2025), Synthesized Long CoT (Hou et al., 2025; Yeo et al., 2025), Learning Reasoning Structure (Li et al., 2025f), Long CoT warmup (Kimi, 2025) , CFT (Wang et al., 2025d)
Reinforcement Learning (§3.1.2)
Reward model-free
Rule-Based (DeepSeek-AI, 2025), cDPO (Lin et al., 2024), Focused-DPO (Zhang et al., 2025b), Selective DPO (Gao et al., 2025b), CPL (Wang et al., 2024f), OREO (Wang et al., 2024b), DAPO (Liu et al., 2024b), RFTT (Zhang et al., 2025c), SimPO (Meng et al., 2024), DQO (Ji et al., 2024), DAPO (Yu et al., 2025), VC-PPO (Yuan et al., 2025), Light-R1 (Wen et al., 2025), etc.
Reward model-based
PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024a), GRPO (Shao et al., 2024), REINFORCE++ (Hu, 2025), DVPO (Huang et al., 2025a), PRIME (Cui et al., 2025), SPPD (Yi et al., 2025), etc.
Inference (§3.2)
Stimulation (§3.2.1)
Prompt Strategy
Hint-infer (Li et al., 2025b), Dipper (Lau et al., 2024), EVA (Ye et al., 2024), EvalPlan(Saha et al., 2025), ReasonFlux (Yang et al., 2025a), Hong et al. (2024)
Decode Strategy
Filler Tokens (Pfau et al., 2024), Budget Forcing (Muennighoff et al., 2025), LTV (Kong et al., 2025), AFT (Li et al., 2025g), Predictive-Decoding (Ma et al., 2025a)
Self-Repetition
Self-Consistency (Wang et al., 2023), Self-Refine (Madaan et al., 2023), DeCRIM (Ferraz et al., 2024), CCE (Zhang et al., 2025e), TreeBoN (Qiu et al., 2024)
Mixture-of-Model
MoA (Wang et al., 2025a), RR-MP (He et al., 2025), BRAIN (Chen et al., 2024f)
Verification (§3.2.2)
Outcome
Output Verification (Cobbe et al., 2021), Generative Verifier (Zhang et al., 2025d), Self-Reflection Feedback (Li et al., 2025h), Discriminator (Chen et al., 2024g), OVM (Yu et al., 2024b), Heuristic (DeepSeek-AI, 2025), Bandit (Sui et al., 2025), Functional (Lee et al., 2025), XoT (Liu et al., 2023b), WoT (Zhang et al., 2024c)
Process
State Evaluator (Yao et al., 2023b; Zhang et al., 2024b), SIaM (Yu et al., 2024a), Deductive Verification (Ling et al., 2023), Self-Evaluator (Xie et al., 2023), V-STaR (Hosseini et al., 2024), Tool (Li et al., 2025b), PoT (Chen et al., 2023a)
Search (§3.2.3)
TreeSearch (Yao et al., 2023b; Chen et al., 2024g),GraphSearch (Besta et al., 2024),C-MSTS (Lin et al., 2025), MCTS (Tian et al., 2024; Zhang et al., 2024e; Gao et al., 2024b; Wan et al., 2024; Chen et al., 2024a), SPaR (Cheng et al., 2025), REBASE (Wu et al., 2024c), SoS (Gandhi et al., 2024), CoAT (Pan et al., 2025a),Beam-Search (Guo et al., 2024; Xie et al., 2023), Lookahead-Search (Snell et al., 2024; Zhang et al., 2023b), etc.
Aggregation (§3.2.4)
Selection
Majority Voting(Wang et al., 2023; Chen et al., 2024d), BOND(Sessa et al., 2024), Filter Vote(Chen et al., 2024d), Length-filtered Vote(Wu et al., 2025b), Best-of-N (Irvine et al., 2023; Song et al., 2024), Rejection Sampling (Kimi, 2025), etc.
Fusion
BoN (weighted) (Brown et al., 2024b), Synthesize (Wang et al., 2025a), etc.

What & How: Summary




What to Scale (§2)
Parallel Scaling (§2.1)
Sequential Scaling (§2.2)
Hybrid Scaling (§2.3)
Internal Scaling (§2.4)

What to Scale: Parallel Scaling

- What is being scaled at the inference stage?
- Parallel Scaling
  - Generating multiple outputs in parallel and then aggregating them into a final answer.
  - Can be from multiple models, or the same model run repeatedly
  - Same model adjustment from hyperparameter adjustment or prompt rephrasing

- Effectiveness derives from:
  - Coverage: the likelihood of generating at least one correct response
  - Aggregation Quality: if a correct response is successfully identified
  - Idea that complex solutions have multiple pathways to the answer

What to Scale: Sequential & Hybrid

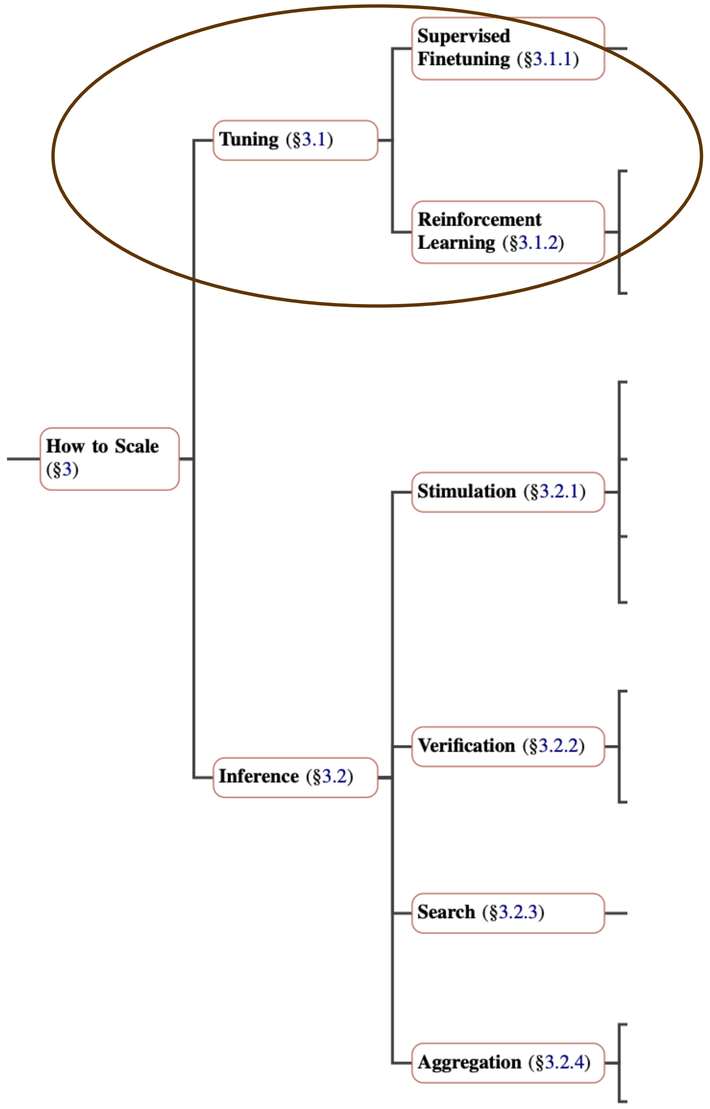
- Sequential Scaling
  - Involves explicitly directing later computations based on intermediate steps.
  - Has several states that involve previous states and problem context
  - Chain-of-Thought (CoT) Prompting, Step-by-Step, Refine
  - Iterations create self-correction, improving accuracy
- Hybrid Scaling
  - Combines Sequential and Parallel Scaling
  - Generate multiple hypotheses and then refine/evaluate them
  - Early work: Tree of Thoughts, Graph of Thoughts
  - More advanced: Monte Carlo Tree Search, Multi-Agent Reasoning (debate)

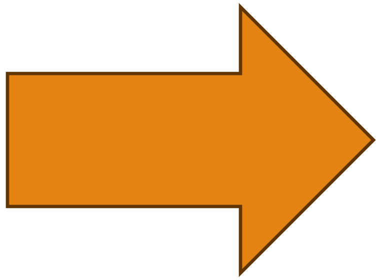
What to Scale: Internal Scaling

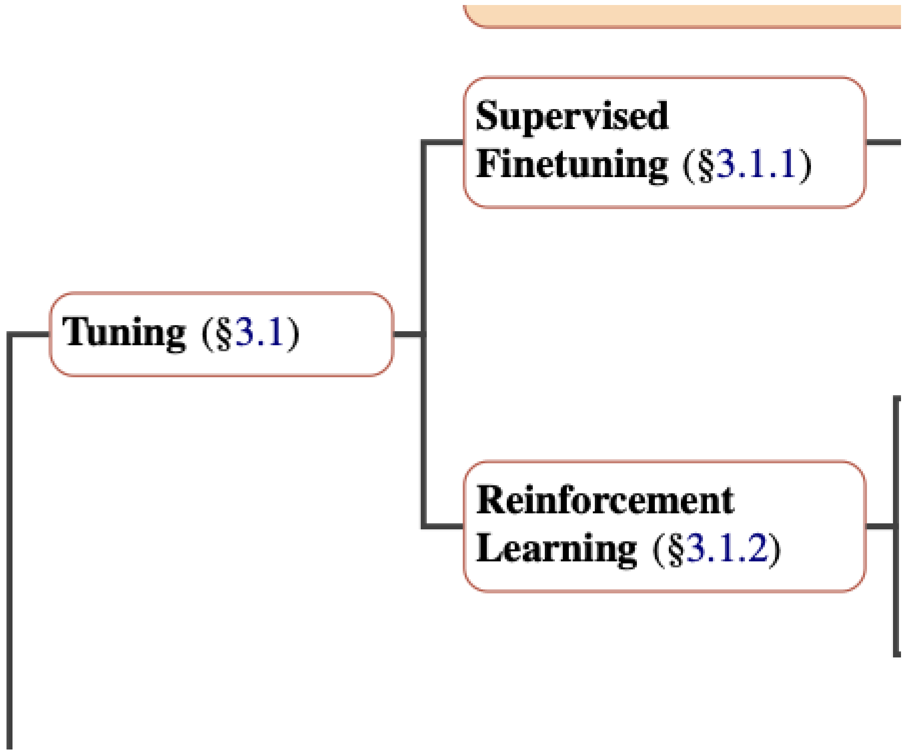
- Internal Scaling
  - Model chooses how much scaling to do for the problem instead of human strategy
- Param Trainable Model
  - Continuously update the model based on reasoning tasks via some training procedure
    - long CoT examples produced by external scaling
  - Outcome-oriented reward modeling for RL (DeepSeek)
- Frozen Model
  - At Test Time, the model generates a sequence of internal states (z)

$$z_{t+1} = f_\theta(z_t), \quad \text{stop}(z_t) = \pi_\theta(z_t).$$

  - Controls when to stop via learned policy
  - Leads to emergent thinking without external prompting


How to Scale (§3)
Tuning (§3.1)
Supervised Finetuning (§3.1.1)
Reinforcement Learning (§3.1.2)
Inference (§3.2)
Stimulation (§3.2.1)
Verification (§3.2.2)
Search (§3.2.3)
Aggregation (§3.2.4)




Tuning (§3.1)
Supervised Finetuning (§3.1.1)
Reinforcement Learning (§3.1.2)

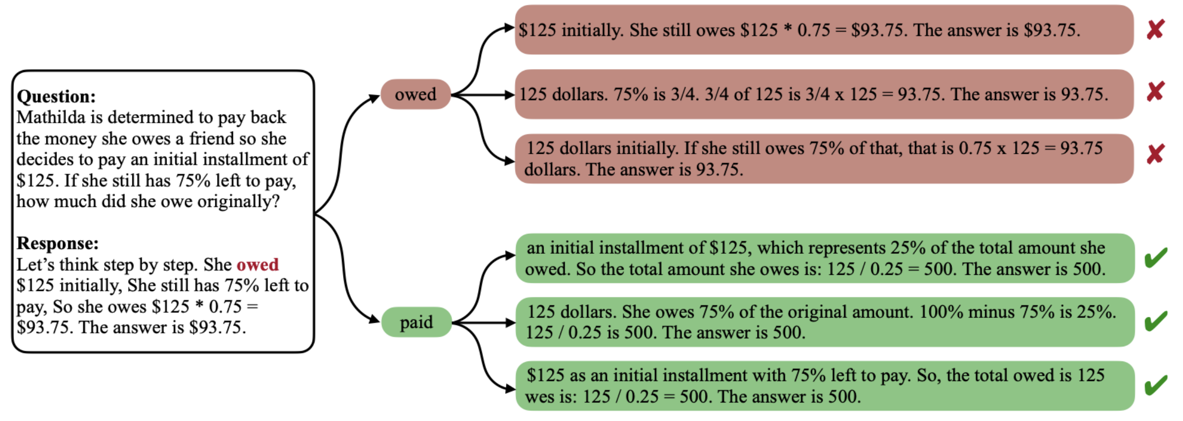
How to Scale: Tuning Based Approaches

- Directly tuning the LLM's parameters with 2 approaches: SFT and RL
- Supervised Fine Tuning (SFT)
  - Train LLM to mimic the rationale/structure to prompt the model to think through complex problems
  - 2 main approaches
- SFT Imitation
  - Generate long CoT demonstrations using test-time “planner” algorithms and then fine-tune the model to imitate those demonstrations
  - STaR: Can be guided by the model itself (generates step-by-step solutions with filtering/verification)
  - ReST-MCTS: use MCTS planner to model itself to reasoning steps
- SFT Distillation
  - Use responses of "stronger" models for supervised learning
  - Can lead to smaller models answer questions just as well as the teacher model

How to Scale: Tuning Based Approaches

- RL: Reward Model-Free
  - Verifiable reward by DeepSeek R1: rule-based reward mechanisms to optimize accuracy in large models
    - SimpleR1: Open-source reproduction of R1
  - OpenR1: HuggingFace's open-source tool for RL
  - cDPO: preference-based optimizer, utilizing critical tokens (base for many other expansions)
  - OREO: value-based optimizer for mathematical reasoning

- RL: Reward Model-Based
  - PPO: Using Human Based model for optimization
    - ReMax takes PPO and reduces hyperparameters, compute time, and need for additional value models
      - Reinforce/Reinforce++ also do this, ReMax more greedy
  - UGDA: refines reward model with (previously) uncertain data.



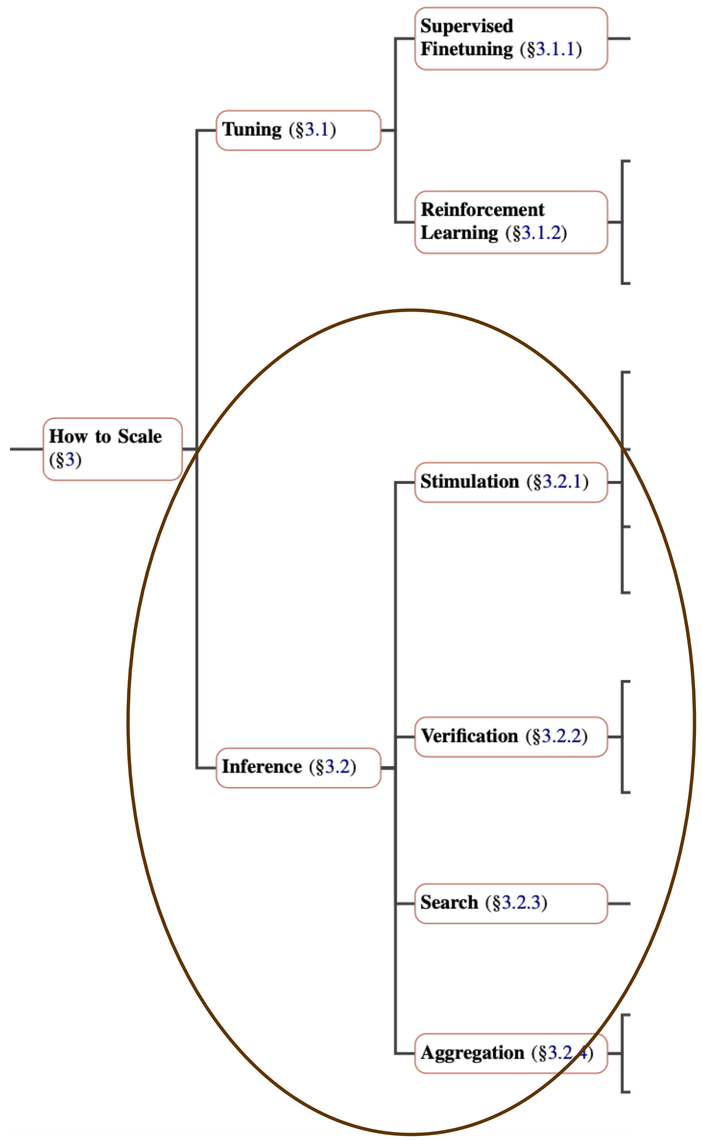
Question:
Mathilda is determined to pay back the money she owes a friend so she decides to pay an initial installment of $125. If she still has 75% left to pay, how much did she owe originally?

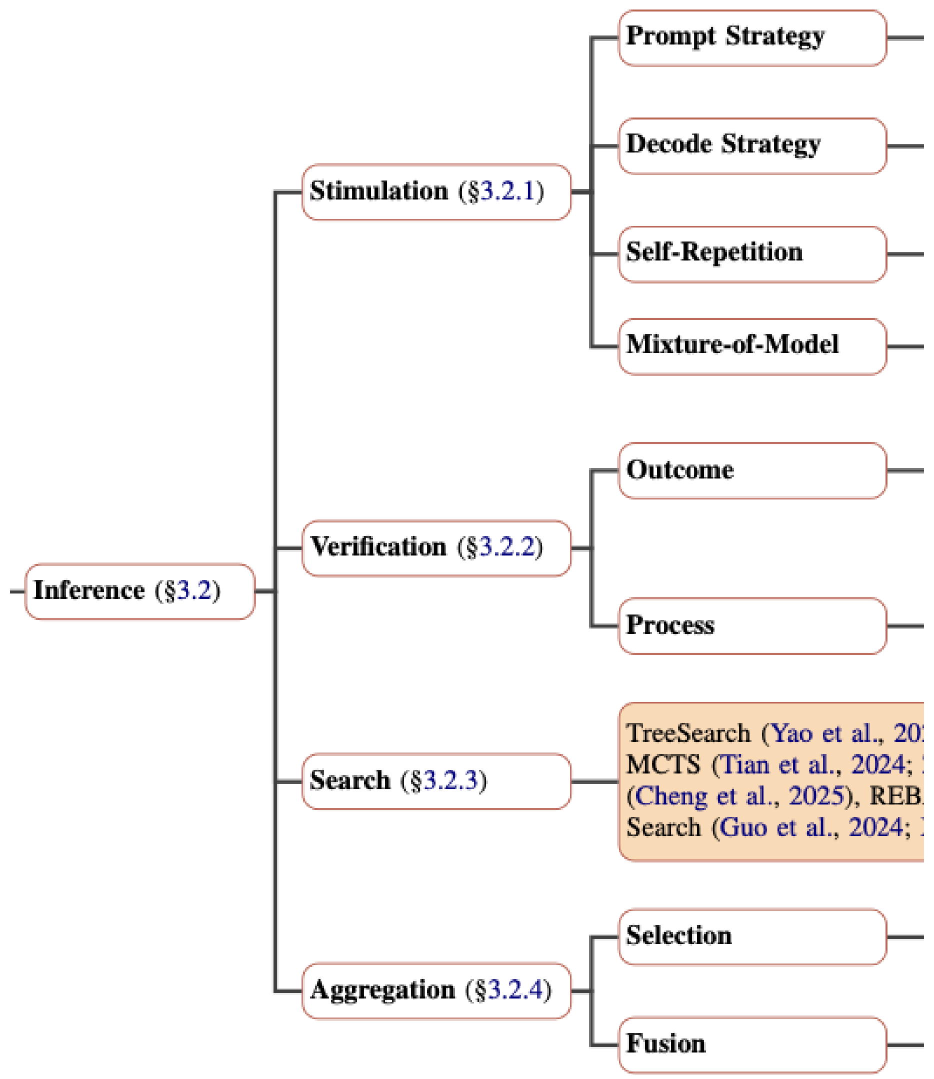
Response:
Let's think step by step. She **owed** $125 initially, She still has 75% left to pay, So she owes $125 * 0.75 = $93.75. The answer is $93.75.

owed
- $125 initially. She still owes $125 * 0.75 = $93.75. The answer is $93.75. ✘
- 125 dollars. 75% is 3/4. 3/4 of 125 is 3/4 x 125 = 93.75. The answer is 93.75. ✘
- 125 dollars initially. If she still owes 75% of that, that is 0.75 x 125 = 93.75 dollars. The answer is 93.75. ✘

paid
- an initial installment of $125, which represents 25% of the total amount she owed. So the total amount she owes is: 125 / 0.25 = 500. The answer is 500. ✔
- 125 dollars. She owes 75% of the original amount. 100% minus 75% is 25%. 125 / 0.25 is 500. The answer is 500. ✔
- $125 as an initial installment with 75% left to pay. So, the total owed is 125 wes is: 125 / 0.25 = 500. The answer is 500. ✔

Figure 1: An illustration of the critical token "*owed*" shows that it fails to lead to the correct answer in any case. Replacing it with an alternative can significantly increase model accuracy.


How to Scale (§3)
Tuning (§3.1)
Supervised Finetuning (§3.1.1)
Reinforcement Learning (§3.1.2)
Inference (§3.2)
Stimulation (§3.2.1)
Verification (§3.2.2)
Search (§3.2.3)
Aggregation (§3.2.4)


Inference (§3.2)
Stimulation (§3.2.1)
Prompt Strategy
Decode Strategy
Self-Repetition
Mixture-of-Model
Verification (§3.2.2)
Outcome
Process
Search (§3.2.3)
TreeSearch (Yao et al., 20
MCTS (Tian et al., 2024;
(Cheng et al., 2025), REB
Search (Guo et al., 2024;
Aggregation (§3.2.4)
Selection
Fusion

How to Scale: Inference Based Approach

- Dynamically adjust parameters during deployment
- Stimulation
  - Getting LLM to think more and allocate longer samples
  - Prompting Strategies
    - "Think Step by Step," and listing requirements to stimulate more samples
  - Decoding Strategies
    - Input more filler phrases or tokens, enforcing intermediate generation (drafts), enforcing prior distributions of latent vectors
  - Self-Repetition Strategies
    - Prompt LLM repeatedly during decoding stage, another is to mimic refinement process
  - Mixture of Model Strategies
    - Ask different models about what they think. Can be all the same or different perspectives

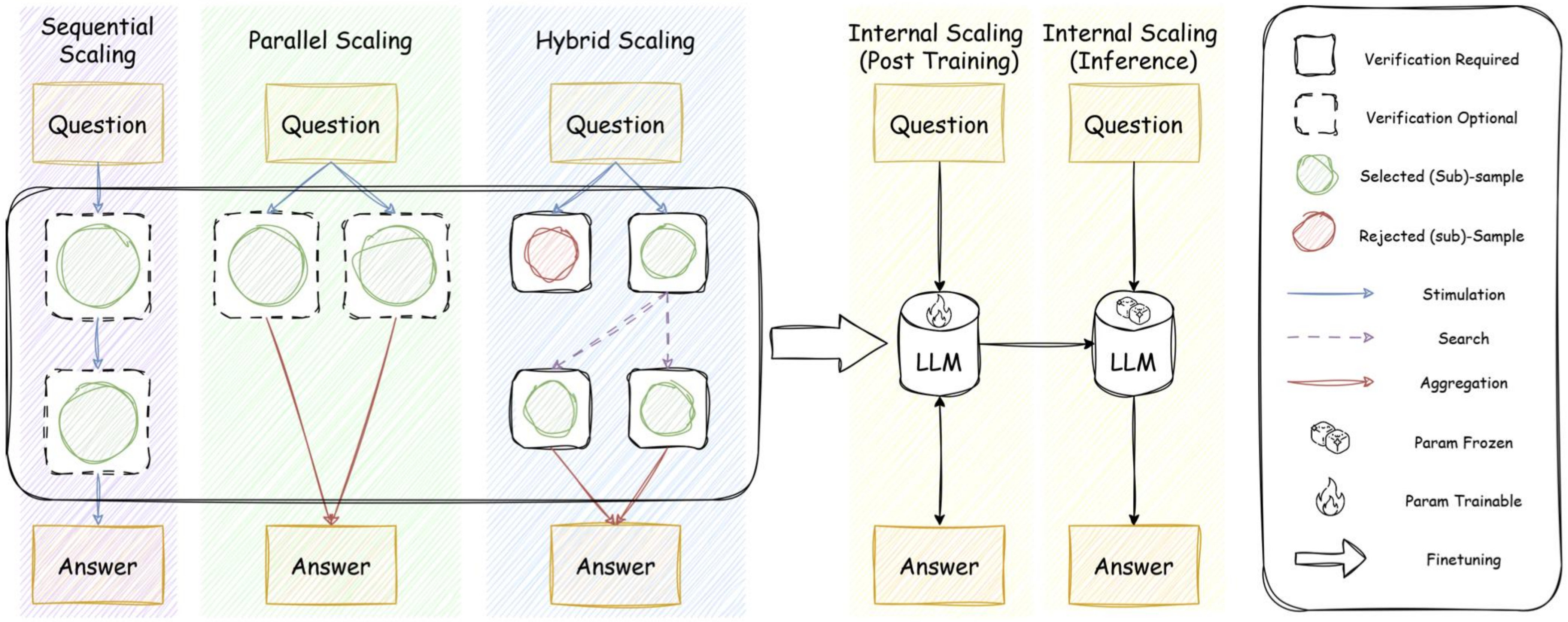
How to Scale: Inference Based Approach

- Verification
  - How do we make sure the LLM is generating a correct response?
    - Can be used for Parallel Scaling, Sequential (to know when to stop), Aggregation or Searching Process (we'll get back to this)
  - Outcome Verification
    - Model Voting
    - Self-consistency
    - Separate algorithms/functions (verifiers)
    - Code generation checks
    - Separate LLM verifier (Judges), Agents
    - RAG
  - Process Verification
    - AKA: Process Reward Model, State Verification
    - Evaluating if the process is correct: Is it actually using CoT? Do the steps to reach the outcome make sense?
    - Process Verification harder for LLMs to evaluate if too complex or long context, decomposition needed
    - Used mostly in Code Generation or Mathmatical Reasoning

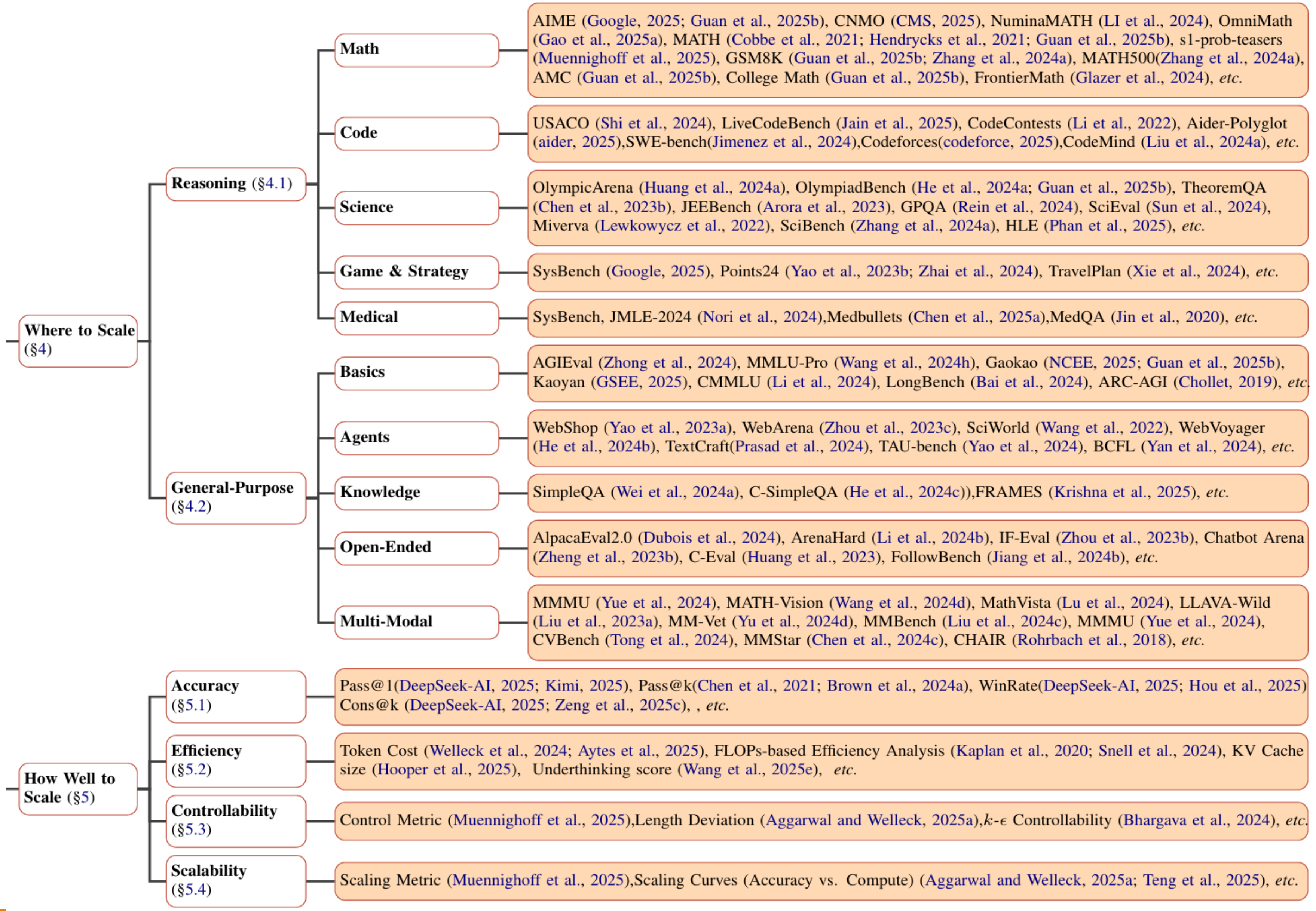
How to Scale: Inference Based Approach

- Search
  - Make sure LLM is utilizing its vast database of knowledge to ensure accuracy
  - Can organize thoughts into a tree and utilize BFS or DFS
  - Utilize Monte Carlo Tree Search during decoding to guide planning
  - Graph Search is also experimented, utilizing stochastic beam search
- Aggregation
  - How to consolidate multiple answers
  - Selection
    - Self-consistency of different routes (most common answer, but sometimes filtering required), Selection Agent
    - Best-of-N (score based on external verifier)
  - Fusion
    - Combine Best-of-N (based on external verifier)
    - Have LLM summarize

What & How: Summary



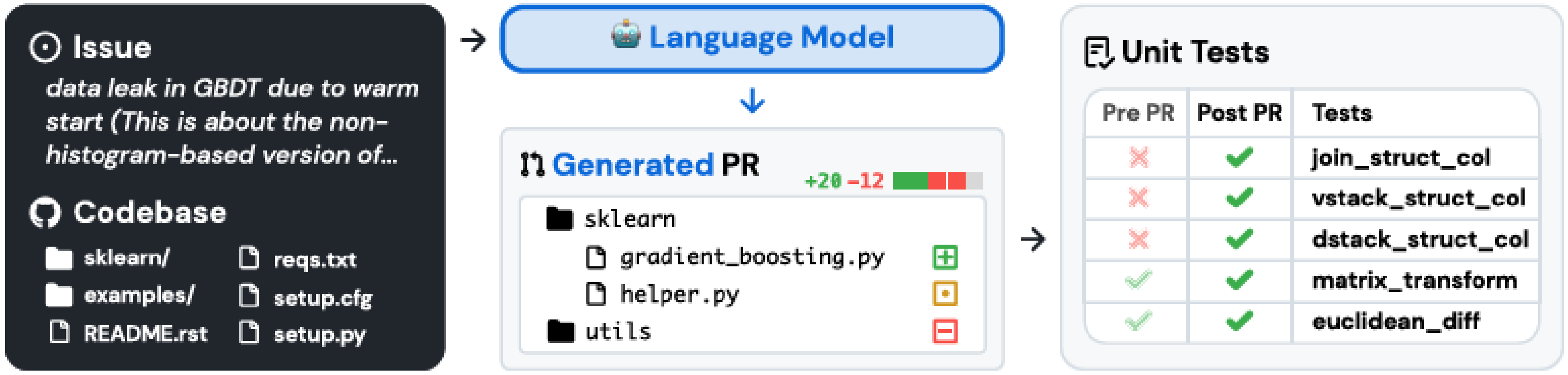
Akira Durham
zup9su



- **Where to Scale** (§4)
  - **Reasoning** (§4.1)
    - **Math**
      - AIME (Google, 2025; Guan et al., 2025b), CNMO (CMS, 2025), NuminaMATH (LI et al., 2024), OmniMath (Gao et al., 2025a), MATH (Cobbe et al., 2021; Hendrycks et al., 2021; Guan et al., 2025b), s1-prob-teasers (Muennighoff et al., 2025), GSM8K (Guan et al., 2025b; Zhang et al., 2024a), MATH500(Zhang et al., 2024a), AMC (Guan et al., 2025b), College Math (Guan et al., 2025b), FrontierMath (Glazer et al., 2024), *etc.*
    - **Code**
      - USACO (Shi et al., 2024), LiveCodeBench (Jain et al., 2025), CodeContests (Li et al., 2022), Aider-Polyglot (aider, 2025),SWE-bench(Jimenez et al., 2024),Codeforces(codeforce, 2025),CodeMind (Liu et al., 2024a), *etc.*
    - **Science**
      - OlympicArena (Huang et al., 2024a), OlympiadBench (He et al., 2024a; Guan et al., 2025b), TheoremQA (Chen et al., 2023b), JEEBench (Arora et al., 2023), GPQA (Rein et al., 2024), SciEval (Sun et al., 2024), Miverva (Lewkowycz et al., 2022), SciBench (Zhang et al., 2024a), HLE (Phan et al., 2025), *etc.*
    - **Game & Strategy**
      - SysBench (Google, 2025), Points24 (Yao et al., 2023b; Zhai et al., 2024), TravelPlan (Xie et al., 2024), *etc.*
    - **Medical**
      - SysBench, JMLE-2024 (Nori et al., 2024),Medbullets (Chen et al., 2025a),MedQA (Jin et al., 2020), *etc.*
  - **General-Purpose** (§4.2)
    - **Basics**
      - AGIEval (Zhong et al., 2024), MMLU-Pro (Wang et al., 2024h), Gaokao (NCEE, 2025; Guan et al., 2025b), Kaoyan (GSEE, 2025), CMMLU (Li et al., 2024), LongBench (Bai et al., 2024), ARC-AGI (Chollet, 2019), *etc.*
    - **Agents**
      - WebShop (Yao et al., 2023a), WebArena (Zhou et al., 2023c), SciWorld (Wang et al., 2022), WebVoyager (He et al., 2024b), TextCraft(Prasad et al., 2024), TAU-bench (Yao et al., 2024), BCFL (Yan et al., 2024), *etc.*
    - **Knowledge**
      - SimpleQA (Wei et al., 2024a), C-SimpleQA (He et al., 2024c)),FRAMES (Krishna et al., 2025), *etc.*
    - **Open-Ended**
      - AlpacaEval2.0 (Dubois et al., 2024), ArenaHard (Li et al., 2024b), IF-Eval (Zhou et al., 2023b), Chatbot Arena (Zheng et al., 2023b), C-Eval (Huang et al., 2023), FollowBench (Jiang et al., 2024b), *etc.*
    - **Multi-Modal**
      - MMMU (Yue et al., 2024), MATH-Vision (Wang et al., 2024d), MathVista (Lu et al., 2024), LLAVA-Wild (Liu et al., 2023a), MM-Vet (Yu et al., 2024d), MMBench (Liu et al., 2024c), MMMU (Yue et al., 2024), CVBench (Tong et al., 2024), MMStar (Chen et al., 2024c), CHAIR (Rohrbach et al., 2018), *etc.*
- **How Well to Scale** (§5)
  - **Accuracy** (§5.1)
    - Pass@1(DeepSeek-AI, 2025; Kimi, 2025), Pass@k(Chen et al., 2021; Brown et al., 2024a), WinRate(DeepSeek-AI, 2025; Hou et al., 2025) Cons@k (DeepSeek-AI, 2025; Zeng et al., 2025c), , *etc.*
  - **Efficiency** (§5.2)
    - Token Cost (Welleck et al., 2024; Aytes et al., 2025), FLOPs-based Efficiency Analysis (Kaplan et al., 2020; Snell et al., 2024), KV Cache size (Hooper et al., 2025), Underthinking score (Wang et al., 2025e), *etc.*
  - **Controllability** (§5.3)
    - Control Metric (Muennighoff et al., 2025),Length Deviation (Aggarwal and Welleck, 2025a),k-ϵ Controllability (Bhargava et al., 2024), *etc.*
  - **Scalability** (§5.4)
    - Scaling Metric (Muennighoff et al., 2025),Scaling Curves (Accuracy vs. Compute) (Aggarwal and Welleck, 2025a; Teng et al., 2025), *etc.*

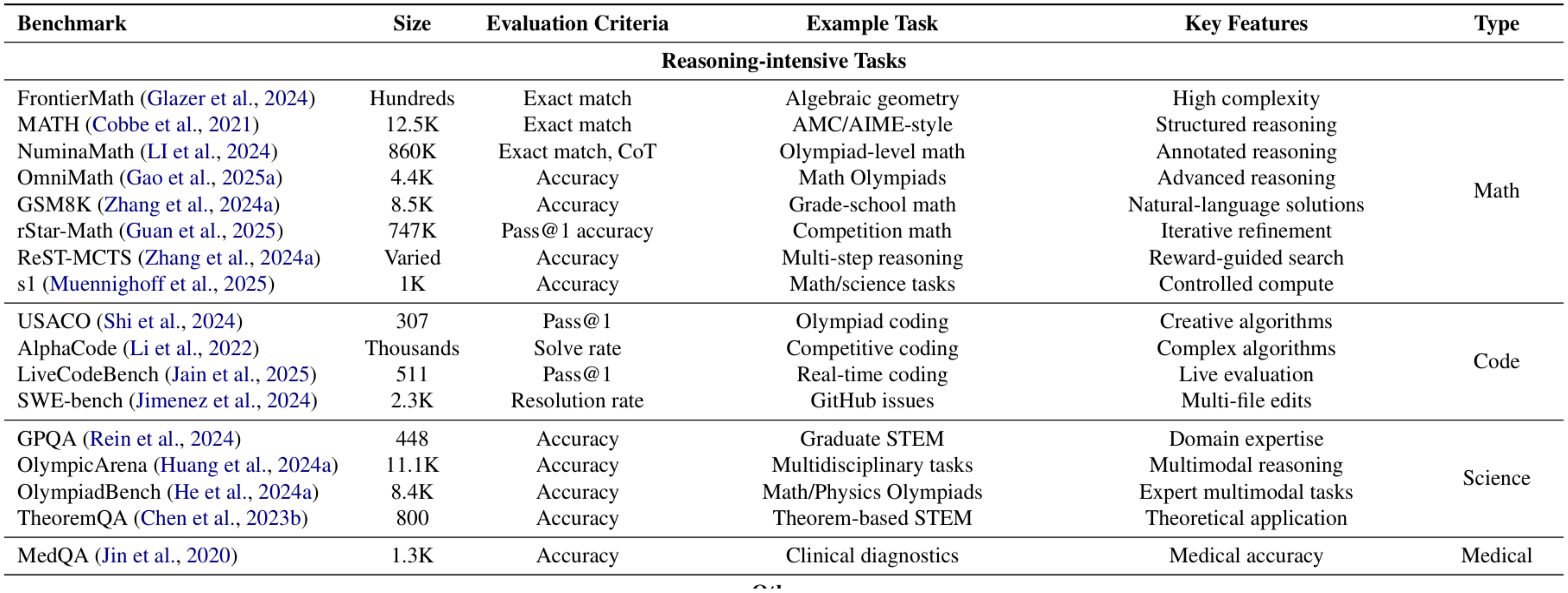
Where to Scale: Reasoning Tasks

- Challenging tasks that require structured, explicit, and precise reasoning
- Mathematical Reasoning
  - Complex computations, logical inference, and iterative verification
- Programming and Code Generation
  - Syntactic accuracy, executable correctness, and iterative debugging



data leak in GBDT due to warm start (This is about the non-histogram-based version of...

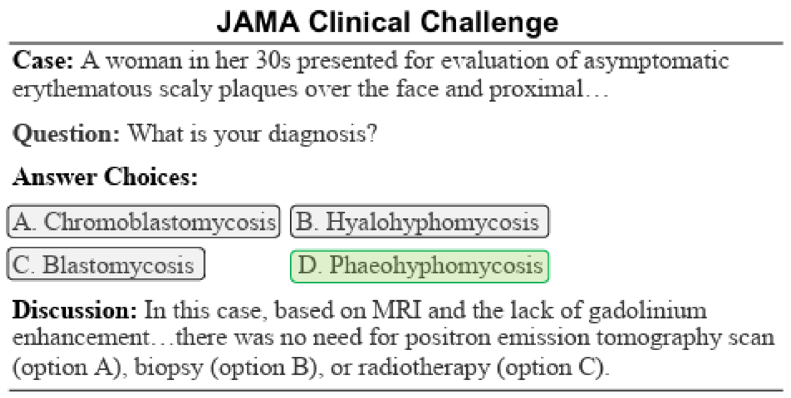
Pre PR	Post PR	Tests
✗	✓	join_struct_col
✗	✓	vstack_struct_col
✗	✓	dstack_struct_col
✓	✓	matrix_transform
✓	✓	euclidean_diff



Benchmark	Size	Evaluation Criteria	Example Task	Key Features	Type
Reasoning-intensive Tasks					
FrontierMath (Glazer et al., 2024)	Hundreds	Exact match	Algebraic geometry	High complexity	Math
MATH (Cobbe et al., 2021)	12.5K	Exact match	AMC/AIME-style	Structured reasoning	
NuminaMath (LI et al., 2024)	860K	Exact match, CoT	Olympiad-level math	Annotated reasoning	
OmniMath (Gao et al., 2025a)	4.4K	Accuracy	Math Olympiads	Advanced reasoning	
GSM8K (Zhang et al., 2024a)	8.5K	Accuracy	Grade-school math	Natural-language solutions	
rStar-Math (Guan et al., 2025)	747K	Pass@1 accuracy	Competition math	Iterative refinement	
ReST-MCTS (Zhang et al., 2024a)	Varied	Accuracy	Multi-step reasoning	Reward-guided search	
s1 (Muennighoff et al., 2025)	1K	Accuracy	Math/science tasks	Controlled compute	
USACO (Shi et al., 2024)	307	Pass@1	Olympiad coding	Creative algorithms	Code
AlphaCode (Li et al., 2022)	Thousands	Solve rate	Competitive coding	Complex algorithms	
LiveCodeBench (Jain et al., 2025)	511	Pass@1	Real-time coding	Live evaluation	
SWE-bench (Jimenez et al., 2024)	2.3K	Resolution rate	GitHub issues	Multi-file edits	
GPQA (Rein et al., 2024)	448	Accuracy	Graduate STEM	Domain expertise	Science
OlympicArena (Huang et al., 2024a)	11.1K	Accuracy	Multidisciplinary tasks	Multimodal reasoning	
OlympiadBench (He et al., 2024a)	8.4K	Accuracy	Math/Physics Olympiads	Expert multimodal tasks	
TheoremQA (Chen et al., 2023b)	800	Accuracy	Theorem-based STEM	Theoretical application	
MedQA (Jin et al., 2020)	1.3K	Accuracy	Clinical diagnostics	Medical accuracy	Medical

Where to Scale: Reasoning Tasks

- Game Playing and Strategic Reasoning
  - Adaptive planning, interactive decision-making, and complex multi-round reasoning
- Scientific Reasoning
  - Multi-domain knowledge integration
- Medical Reasoning
  - Diagonostic decision-making, clinical reasoning, precise medical knowledge



JAMA Clinical Challenge

Case: A woman in her 30s presented for evaluation of asymptomatic erythematous scaly plaques over the face and proximal…

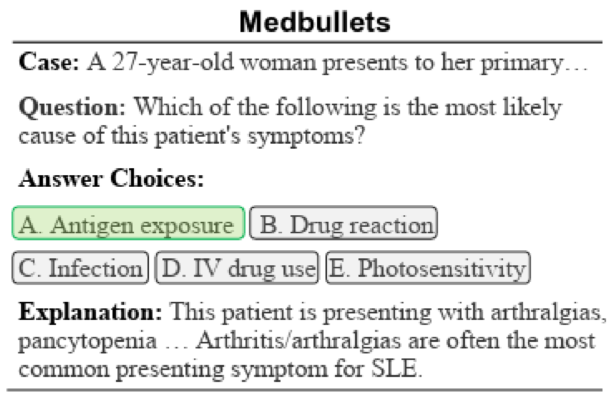
Question: What is your diagnosis?

Answer Choices:

A. Chromoblastomycosis B. Hyalohyphomycosis
C. Blastomycosis [✓ D. Phaeohyphomycosis]

Discussion: In this case, based on MRI and the lack of gadolinium enhancement…there was no need for positron emission tomography scan (option A), biopsy (option B), or radiotherapy (option C).

(a)



Medbullets

Case: A 27-year-old woman presents to her primary…

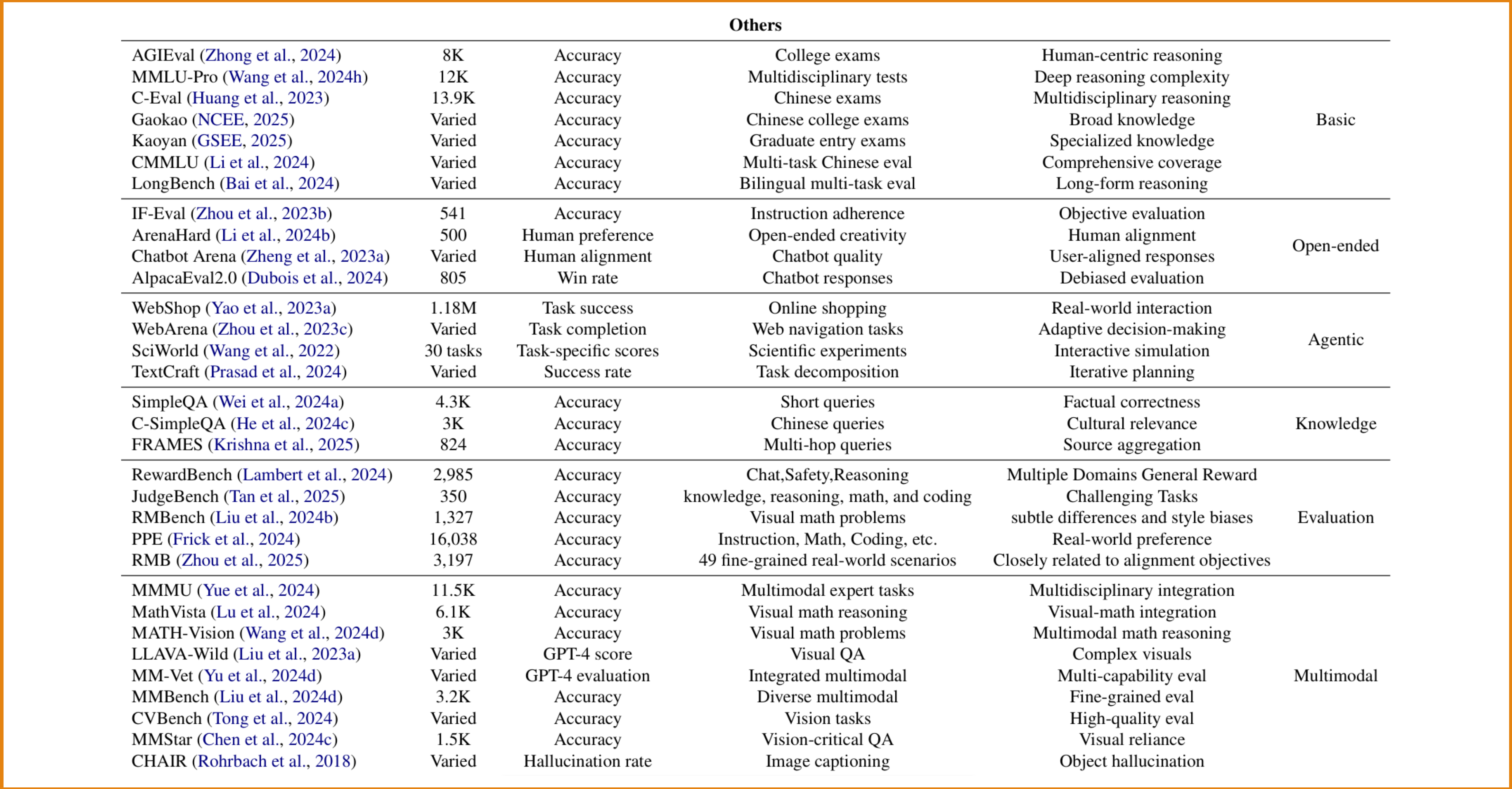
Question: Which of the following is the most likely cause of this patient's symptoms?

Answer Choices:

[✓ A. Antigen exposure] B. Drug reaction
C. Infection D. IV drug use E. Photosensitivity

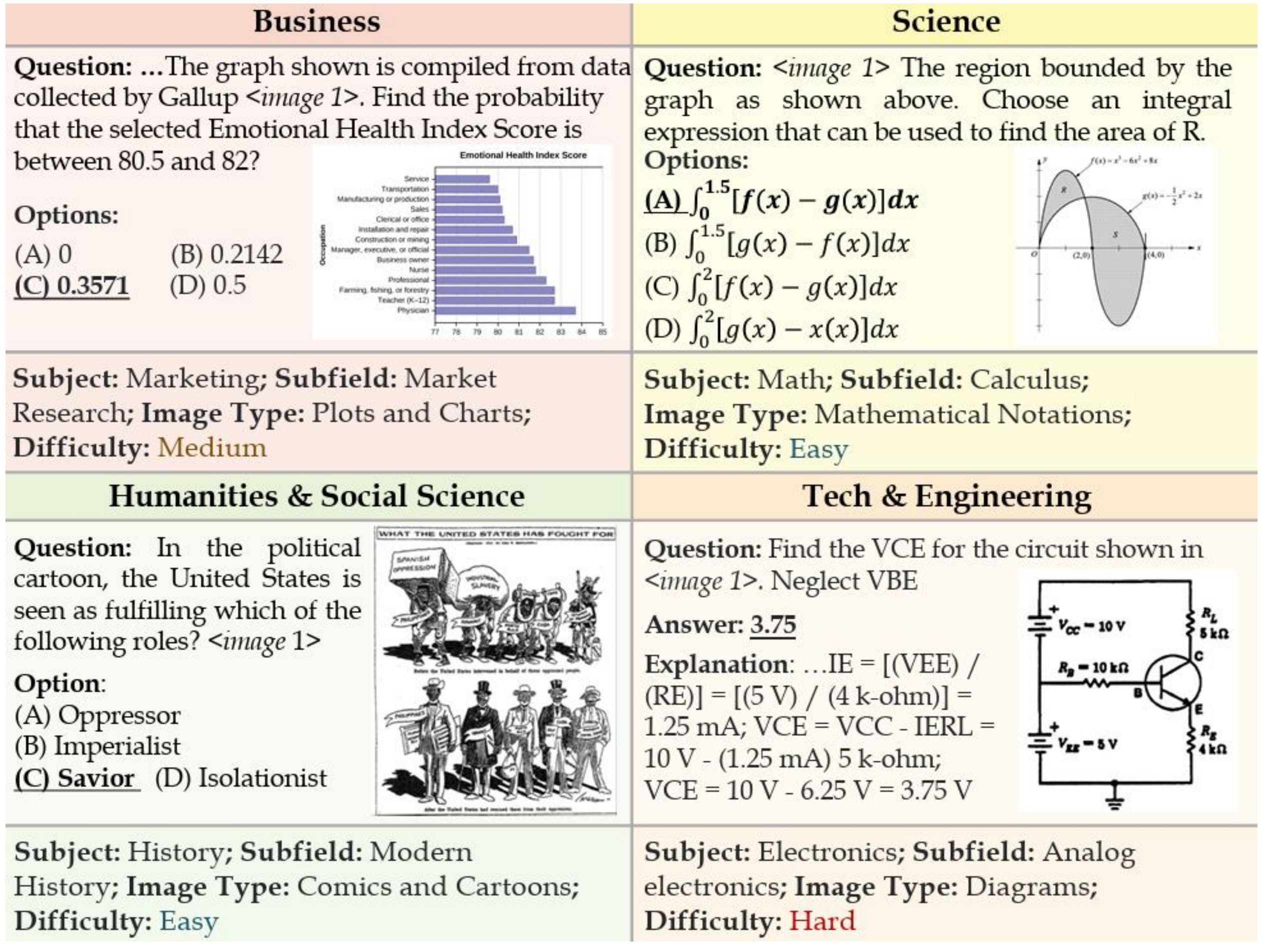
Explanation: This patient is presenting with arthralgias, pancytopenia … Arthritis/arthralgias are often the most common presenting symptom for SLE.

(b)



Others					
AGIEval (Zhong et al., 2024)	8K	Accuracy	College exams	Human-centric reasoning	Basic
MMLU-Pro (Wang et al., 2024h)	12K	Accuracy	Multidisciplinary tests	Deep reasoning complexity	
C-Eval (Huang et al., 2023)	13.9K	Accuracy	Chinese exams	Multidisciplinary reasoning	
Gaokao (NCEE, 2025)	Varied	Accuracy	Chinese college exams	Broad knowledge	
Kaoyan (GSEE, 2025)	Varied	Accuracy	Graduate entry exams	Specialized knowledge	
CMMLU (Li et al., 2024)	Varied	Accuracy	Multi-task Chinese eval	Comprehensive coverage	
LongBench (Bai et al., 2024)	Varied	Accuracy	Bilingual multi-task eval	Long-form reasoning	
IF-Eval (Zhou et al., 2023b)	541	Accuracy	Instruction adherence	Objective evaluation	Open-ended
ArenaHard (Li et al., 2024b)	500	Human preference	Open-ended creativity	Human alignment	
Chatbot Arena (Zheng et al., 2023a)	Varied	Human alignment	Chatbot quality	User-aligned responses	
AlpacaEval2.0 (Dubois et al., 2024)	805	Win rate	Chatbot responses	Debiased evaluation	
WebShop (Yao et al., 2023a)	1.18M	Task success	Online shopping	Real-world interaction	Agentic
WebArena (Zhou et al., 2023c)	Varied	Task completion	Web navigation tasks	Adaptive decision-making	
SciWorld (Wang et al., 2022)	30 tasks	Task-specific scores	Scientific experiments	Interactive simulation	
TextCraft (Prasad et al., 2024)	Varied	Success rate	Task decomposition	Iterative planning	
SimpleQA (Wei et al., 2024a)	4.3K	Accuracy	Short queries	Factual correctness	Knowledge
C-SimpleQA (He et al., 2024c)	3K	Accuracy	Chinese queries	Cultural relevance	
FRAMES (Krishna et al., 2025)	824	Accuracy	Multi-hop queries	Source aggregation	
RewardBench (Lambert et al., 2024)	2,985	Accuracy	Chat,Safety,Reasoning	Multiple Domains General Reward	Evaluation
JudgeBench (Tan et al., 2025)	350	Accuracy	knowledge, reasoning, math, and coding	Challenging Tasks	
RMBench (Liu et al., 2024b)	1,327	Accuracy	Visual math problems	subtle differences and style biases	
PPE (Frick et al., 2024)	16,038	Accuracy	Instruction, Math, Coding, etc.	Real-world preference	
RMB (Zhou et al., 2025)	3,197	Accuracy	49 fine-grained real-world scenarios	Closely related to alignment objectives	
MMMU (Yue et al., 2024)	11.5K	Accuracy	Multimodal expert tasks	Multidisciplinary integration	Multimodal
MathVista (Lu et al., 2024)	6.1K	Accuracy	Visual math reasoning	Visual-math integration	
MATH-Vision (Wang et al., 2024d)	3K	Accuracy	Visual math problems	Multimodal math reasoning	
LLAVA-Wild (Liu et al., 2023a)	Varied	GPT-4 score	Visual QA	Complex visuals	
MM-Vet (Yu et al., 2024d)	Varied	GPT-4 evaluation	Integrated multimodal	Multi-capability eval	
MMBench (Liu et al., 2024d)	3.2K	Accuracy	Diverse multimodal	Fine-grained eval	
CVBench (Tong et al., 2024)	Varied	Accuracy	Vision tasks	High-quality eval	
MMStar (Chen et al., 2024c)	1.5K	Accuracy	Vision-critical QA	Visual reliance	
CHAIR (Rohrbach et al., 2018)	Varied	Hallucination rate	Image captioning	Object hallucination	

Where to Scale: General Tasks



Business

Question: …The graph shown is compiled from data collected by Gallup <*image* 1>. Find the probability that the selected Emotional Health Index Score is between 80.5 and 82?

Options:

(A) 0 (B) 0.2142

(C) 0.3571 (D) 0.5

Emotional Health Index Score
Occupation
Service
Transportation
Manufacturing or production
Sales
Clerical or office
Installation and repair
Construction or mining
Manager, executive, or official
Business owner
Nurse
Professional
Farming, fishing, or forestry
Teacher (K–12)
Physician
77
78
79
80
81
82
83
84
85

Subject: Marketing; **Subfield:** Market Research; **Image Type:** Plots and Charts; **Difficulty:** Medium

Science

Question: <*image* 1> The region bounded by the graph as shown above. Choose an integral expression that can be used to find the area of R.

Options:

(A) $\int_0^{1.5}[f(x) - g(x)]dx$

(B) $\int_0^{1.5}[g(x) - f(x)]dx$

(C) $\int_0^{2}[f(x) - g(x)]dx$

(D) $\int_0^{2}[g(x) - x(x)]dx$

Subject: Math; **Subfield:** Calculus; **Image Type:** Mathematical Notations; **Difficulty:** Easy

Humanities & Social Science

Question: In the political cartoon, the United States is seen as fulfilling which of the following roles? <*image* 1>

Option:

(A) Oppressor

(B) Imperialist

(C) Savior (D) Isolationist

Subject: History; **Subfield:** Modern History; **Image Type:** Comics and Cartoons; **Difficulty:** Easy

Tech & Engineering

Question: Find the VCE for the circuit shown in <*image* 1>. Neglect VBE

Answer: 3.75

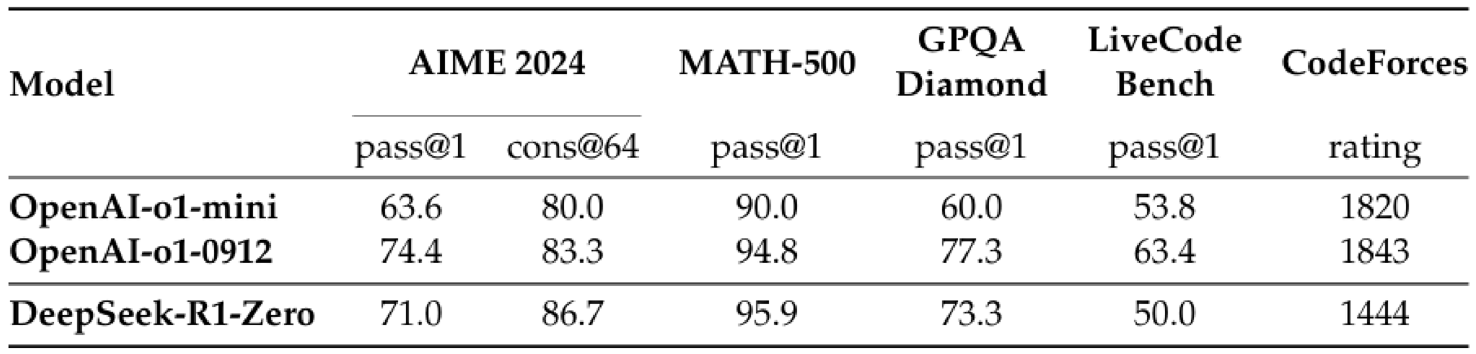
Explanation: …IE = [(VEE) / (RE)] = [(5 V) / (4 k-ohm)] = 1.25 mA; VCE = VCC - IERL = 10 V - (1.25 mA) 5 k-ohm; VCE = 10 V - 6.25 V = 3.75 V

Subject: Electronics; **Subfield:** Analog electronics; **Image Type:** Diagrams; **Difficulty:** Hard

- Require broad, general-purpose reasoning capabilities
- Open-ended Tasks
  - Enhance output diversity, quality, and coherence
- Agentic Tasks
  - Realistic and interactive environments, complex planning, tool utilization, and iterative reasoning
- Knowledge-intensive Tasks
  - Retrieve and synthesize factual knowledge from external sources
- Multimodal Tasks
  - Cross-modal integration, iterative reasoning between modalities, and robust verification

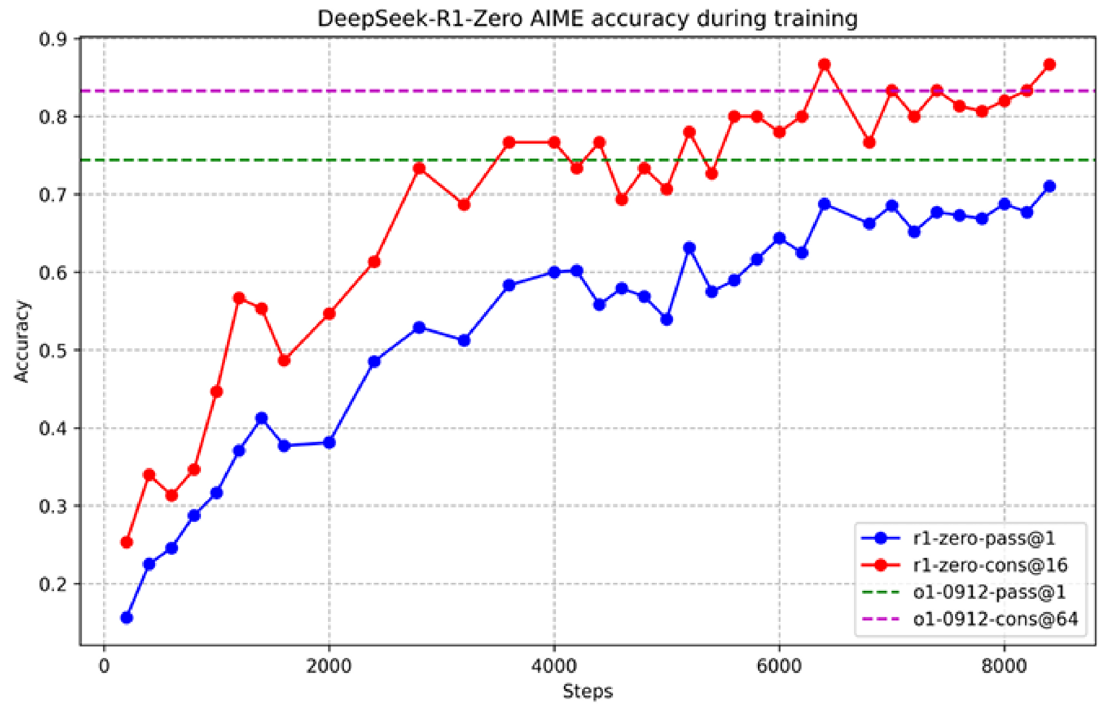
How Well to Scale

- Classify metrics used in evaluating TTS methods
- Performance: Assess correctness of generated solutions
  - Pass@1 – Widely used, proportion of problems where first response was correct
  - Pass@k – Extends Pass@1, at least one of *k* responses is correct
  - Consensus@k – Majority-voted answer through k responses, was it correct
  - Arena-based Evaluation – Paired with additional output metrics, ex. shorter answers
- Efficiency: Assess computational and resource costs
  - Token Cost – Total number of tokens generated during inference, intermediate + final
  - FLOPs-based Efficiency Analysis – Quantify computational cost, comparison for similar compute models
  - Underthinking Score – Initial correct thought but fails to follow through, measures time + length



Model	**AIME 2024**		**MATH-500**	**GPQA Diamond**	**LiveCode Bench**	**CodeForces**
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444

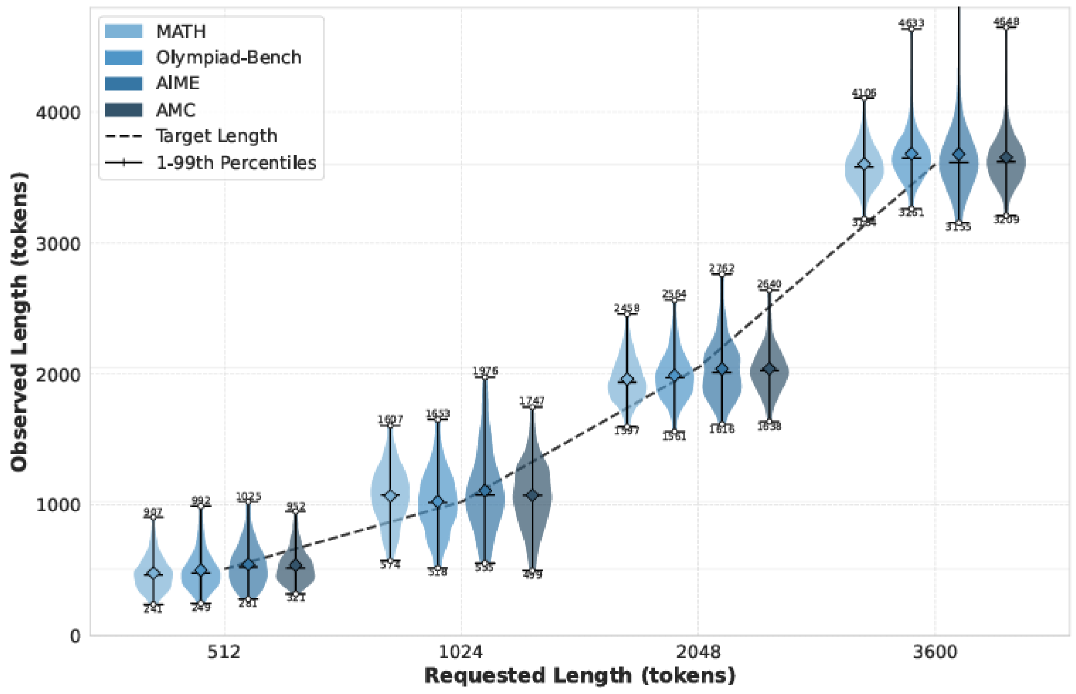
Table 2 | Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

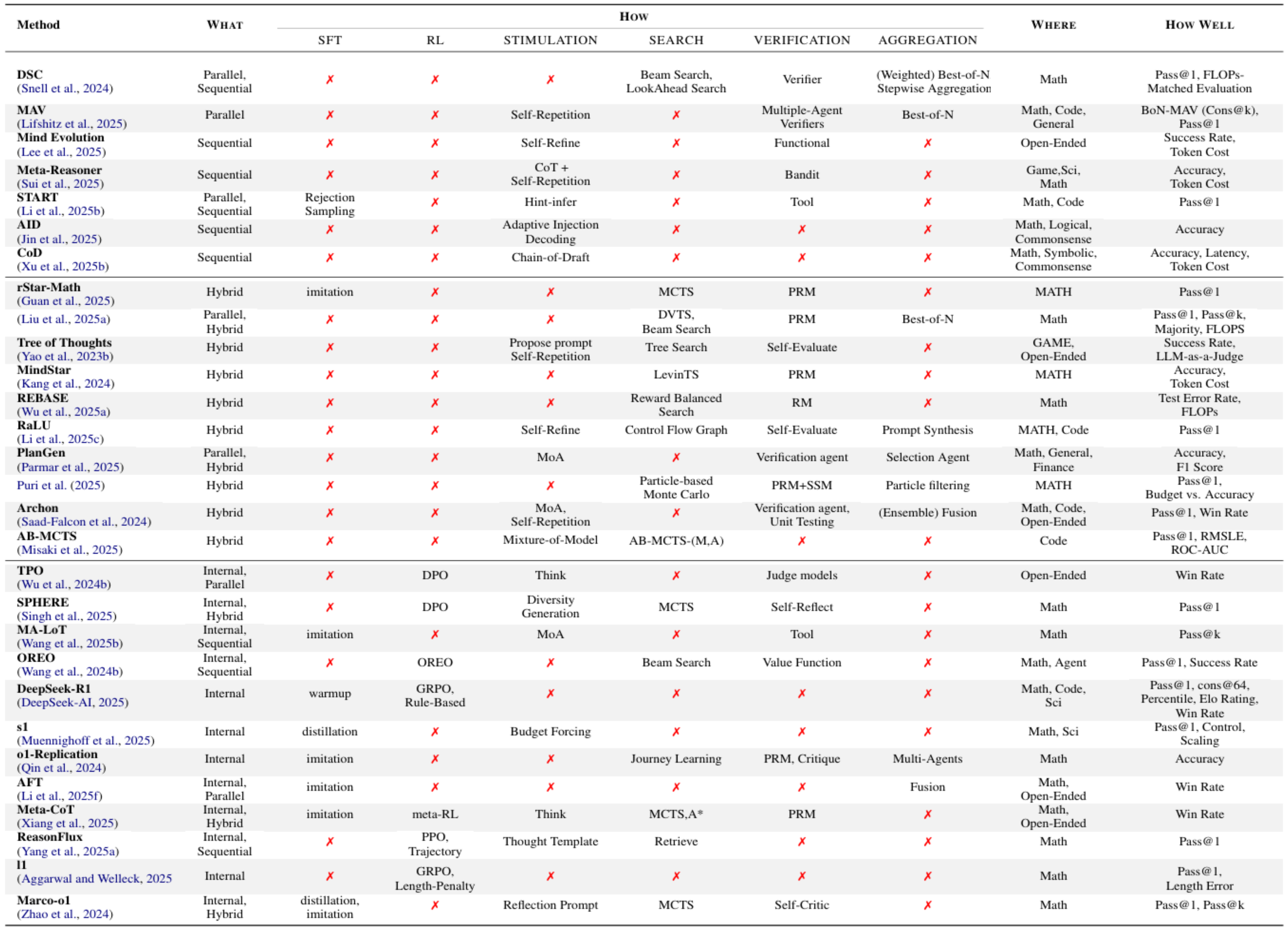


https://arxiv.org/abs/2501.12948

How Well to Scale

- Controllability: Assess if inference-time methods can consistently align to constraints
  - Control Metric – Quantify adherence to specific compute budget range
  - Length-Deviation – Quantify model's ability to control output length
  - K-ϵ Controllability – Prompt-based steerability, achieve some specific output
- Scalability: Assess TTS methods leverage increased compute to improve performance
  - Scaling Metric – Average slope of performance gains as compute increases
  - Scaling Curves – Visualize diminishing returns at higher compute budgets (accuracy, pass rate, etc)





Method	What	How						Where	How Well
		SFT	RL	Stimulation	Search	Verification	Aggregation		
DSC (Snell et al., 2024)	Parallel, Sequential	✗	✗	✗	Beam Search, LookAhead Search	Verifier	(Weighted) Best-of-N Stepwise Aggregation	Math	Pass@1, FLOPs-Matched Evaluation
MAV (Lifshitz et al., 2025)	Parallel	✗	✗	Self-Repetition	✗	Multiple-Agent Verifiers	Best-of-N	Math, Code, General	BoN-MAV (Cons@k), Pass@1
Mind Evolution (Lee et al., 2025)	Sequential	✗	✗	Self-Refine	✗	Functional	✗	Open-Ended	Success Rate, Token Cost
Meta-Reasoner (Sui et al., 2025)	Sequential	✗	✗	CoT + Self-Repetition	✗	Bandit	✗	Game,Sci, Math	Accuracy, Token Cost
START (Li et al., 2025b)	Parallel, Sequential	Rejection Sampling	✗	Hint-infer	✗	Tool	✗	Math, Code	Pass@1
AID (Jin et al., 2025)	Sequential	✗	✗	Adaptive Injection Decoding	✗	✗	✗	Math, Logical, Commonsense	Accuracy
CoD (Xu et al., 2025b)	Sequential	✗	✗	Chain-of-Draft	✗	✗	✗	Math, Symbolic, Commonsense	Accuracy, Latency, Token Cost
rStar-Math (Guan et al., 2025)	Hybrid	imitation	✗	✗	MCTS	PRM	✗	MATH	Pass@1
(Liu et al., 2025a)	Parallel, Hybrid	✗	✗	✗	DVTS, Beam Search	PRM	Best-of-N	Math	Pass@1, Pass@k, Majority, FLOPS
Tree of Thoughts (Yao et al., 2023b)	Hybrid	✗	✗	Propose prompt Self-Repetition	Tree Search	Self-Evaluate	✗	GAME, Open-Ended	Success Rate, LLM-as-a-Judge
MindStar (Kang et al., 2024)	Hybrid	✗	✗	✗	LevinTS	PRM	✗	MATH	Accuracy, Token Cost
REBASE (Wu et al., 2025a)	Hybrid	✗	✗	✗	Reward Balanced Search	RM	✗	Math	Test Error Rate, FLOPs
RaLU (Li et al., 2025c)	Hybrid	✗	✗	Self-Refine	Control Flow Graph	Self-Evaluate	Prompt Synthesis	MATH, Code	Pass@1
PlanGen (Parmar et al., 2025)	Parallel, Hybrid	✗	✗	MoA	✗	Verification agent	Selection Agent	Math, General, Finance	Accuracy, F1 Score
Puri et al. (2025)	Hybrid	✗	✗	✗	Particle-based Monte Carlo	PRM+SSM	Particle filtering	MATH	Pass@1, Budget vs. Accuracy
Archon (Saad-Falcon et al., 2024)	Hybrid	✗	✗	MoA, Self-Repetition	✗	Verification agent, Unit Testing	(Ensemble) Fusion	Math, Code, Open-Ended	Pass@1, Win Rate
AB-MCTS (Misaki et al., 2025)	Hybrid	✗	✗	Mixture-of-Model	AB-MCTS-(M,A)	✗	✗	Code	Pass@1, RMSLE, ROC-AUC
TPO (Wu et al., 2024b)	Internal, Parallel	✗	DPO	Think	✗	Judge models	✗	Open-Ended	Win Rate
SPHERE (Singh et al., 2025)	Internal, Hybrid	✗	DPO	Diversity Generation	MCTS	Self-Reflect	✗	Math	Pass@1
MA-LoT (Wang et al., 2025b)	Internal, Sequential	imitation	✗	MoA	✗	Tool	✗	Math	Pass@k
OREO (Wang et al., 2024b)	Internal, Sequential	✗	OREO	✗	Beam Search	Value Function	✗	Math, Agent	Pass@1, Success Rate
DeepSeek-R1 (DeepSeek-AI, 2025)	Internal	warmup	GRPO, Rule-Based	✗	✗	✗	✗	Math, Code, Sci	Pass@1, cons@64, Percentile, Elo Rating, Win Rate
s1 (Muennighoff et al., 2025)	Internal	distillation	✗	Budget Forcing	✗	✗	✗	Math, Sci	Pass@1, Control, Scaling
o1-Replication (Qin et al., 2024)	Internal	imitation	✗	✗	Journey Learning	PRM, Critique	Multi-Agents	Math	Accuracy
AFT (Li et al., 2025f)	Internal, Parallel	imitation	✗	✗	✗	✗	Fusion	Math, Open-Ended	Win Rate
Meta-CoT (Xiang et al., 2025)	Internal, Hybrid	imitation	meta-RL	Think	MCTS,A*	PRM	✗	Math, Open-Ended	Win Rate
ReasonFlux (Yang et al., 2025a)	Internal, Sequential	✗	PPO, Trajectory	Thought Template	Retrieve	✗	✗	Math	Pass@1
l1 (Aggarwal and Welleck, 2025	Internal	✗	GRPO, Length-Penalty	✗	✗	✗	✗	Math	Pass@1, Length Error
Marco-o1 (Zhao et al., 2024)	Internal, Hybrid	distillation, imitation	✗	Reflection Prompt	MCTS	Self-Critic	✗	Math	Pass@1, Pass@k

Method	WHAT	HOW			
		SFT	RL	STIMULATION	SEARCH
rStar-Math (Guan et al., 2025)	Hybrid	imitation	✗	✗	MCTS

		WHERE	HOW WELL
VERIFICATION	AGGREGATION		
PRM	✗	MATH	Pass@1

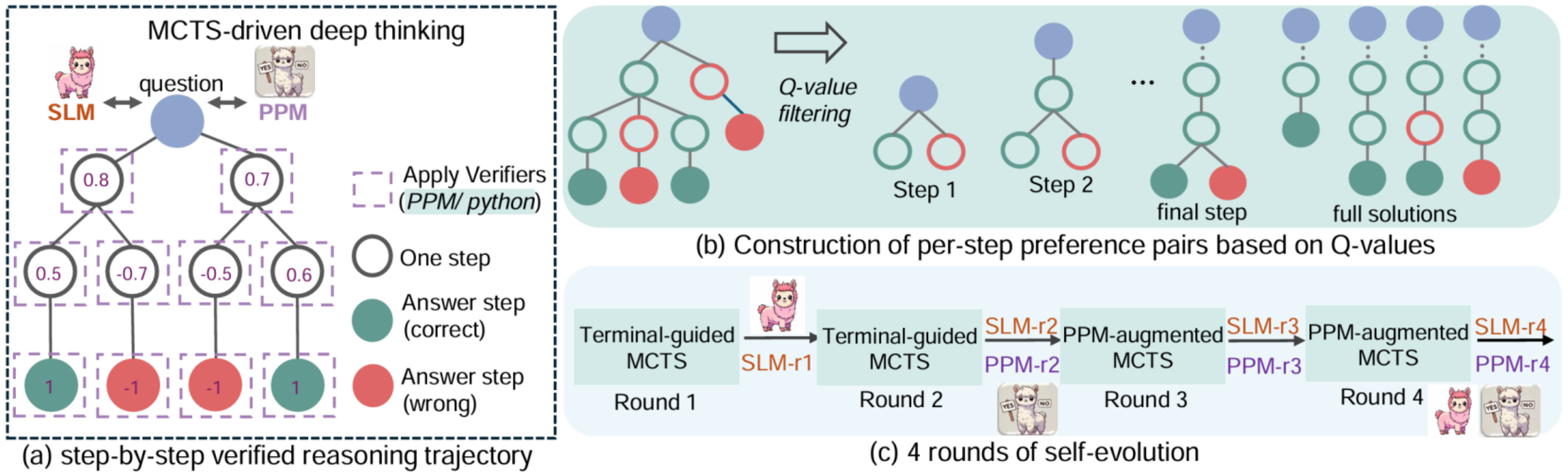


Figure 1: The overview of rStar-Math.

Organization and Trends

- 2022 – 2023
  - Emphasized structured inference to guide LLMs
- 2024
  - Methods like PRM and MCTS enabled automatic supervision of reasoning
- 2025
  - Pure RL can also elicit comprehensive, sound reasoning

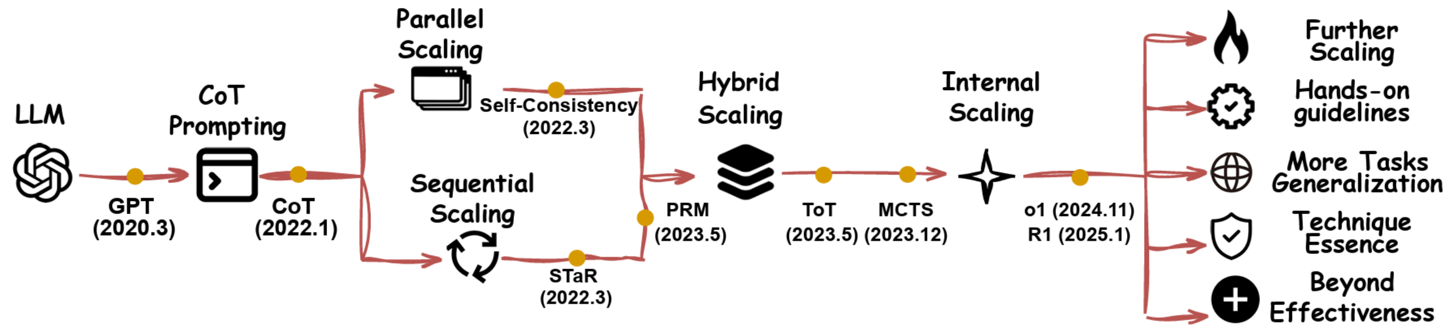
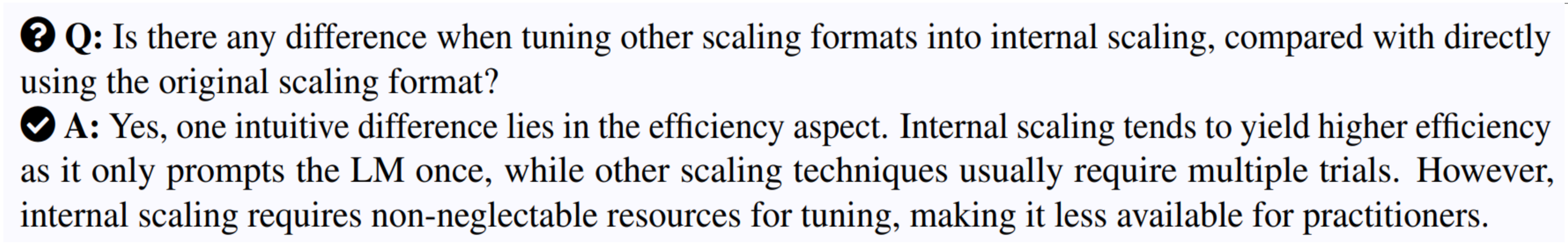
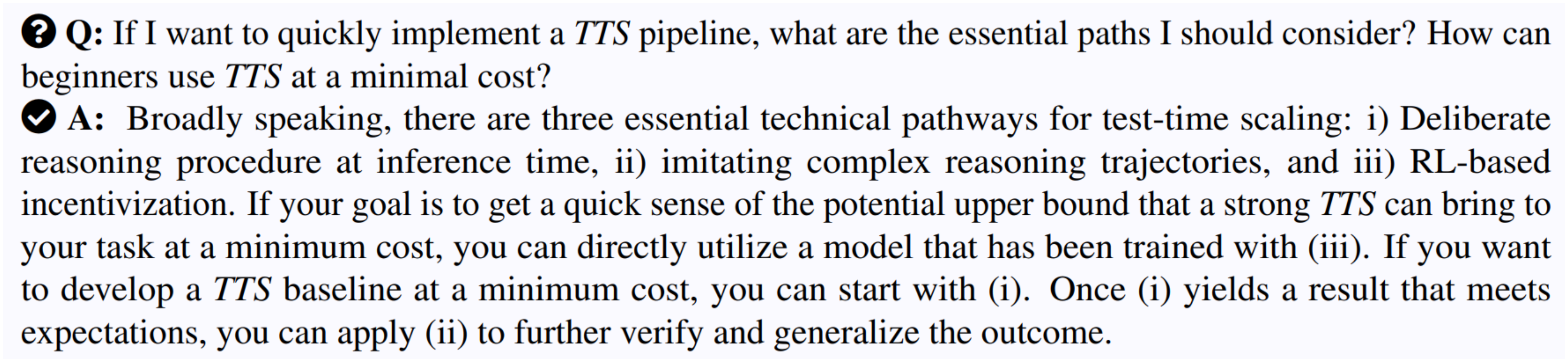


Figure 4: From Emergence to the Next Frontier, the Evolutionary Path of Test-Time Scaling.

Hands-on Guidelines



❓ **Q:** Is there any difference when tuning other scaling formats into internal scaling, compared with directly using the original scaling format?
✔ **A:** Yes, one intuitive difference lies in the efficiency aspect. Internal scaling tends to yield higher efficiency as it only prompts the LM once, while other scaling techniques usually require multiple trials. However, internal scaling requires non-neglectable resources for tuning, making it less available for practitioners.



❓ **Q:** If I want to quickly implement a *TTS* pipeline, what are the essential paths I should consider? How can beginners use *TTS* at a minimal cost?
✔ **A:** Broadly speaking, there are three essential technical pathways for test-time scaling: i) Deliberate reasoning procedure at inference time, ii) imitating complex reasoning trajectories, and iii) RL-based incentivization. If your goal is to get a quick sense of the potential upper bound that a strong *TTS* can bring to your task at a minimum cost, you can directly utilize a model that has been trained with (iii). If you want to develop a *TTS* baseline at a minimum cost, you can start with (i). Once (i) yields a result that meets expectations, you can apply (ii) to further verify and generalize the outcome.

Challenges and Opportunities

- More scaling is the frontier
- Clarifying the techniques
- Optimizing Scaling
- Domain generalization

More Scaling

- Transformative impact on reasoning-intensive tasks – as seen in o1 and R1
- Parallel
  - Generating multiple responses and selecting best answer, leads to diminishing returns
- Sequential
  - Maintaining coherence and preventing error accumulation
- Hybrid
  - Blends parallel and sequential, more adaptive and practical, more specialized and less generalizable
- Internal
  - On the fly computation modulation without external intervention, unique challenges

Techniques & Generalization

- Clarifying Techniques
  - Gaps in scaling techniques, improving reward modeling, CoT reasoning priorities, and adaptive TTS
- Optimizing Scaling
  - Comprehensive and comparable measurements of different strategies
- Generalization
  - Balancing cost + accuracy, ensure domain-specific interpretability, and integrate external knowledge

Ananya Ananda
jaf5rp

s1: Simple test-time scaling

NIKLAS MUENNIGHOFF*[1,3,4] ZITONGYANG*[1] WEIJIA SHI*[2,3] XIANGLISALI*[1] LI FEI-FEI[1] HANNANEH HAJISHIRZI[2,3]

LUKE ZETTLEMOYER[2] PERCY LIANG[1] EMMANUEL CANDÈS[1] TATSUNORIHASHIMOTO[1]

Test-Time Scaling

- Increase compute at test time for better results
- OpenAI o1 – validated test-time scaling
  - Using large scale RL (implying sizable amounts of data)
- DeepSeek R1 – replicated o1-level performance
  - Employing RL w/ millions of samples and multiple training stages

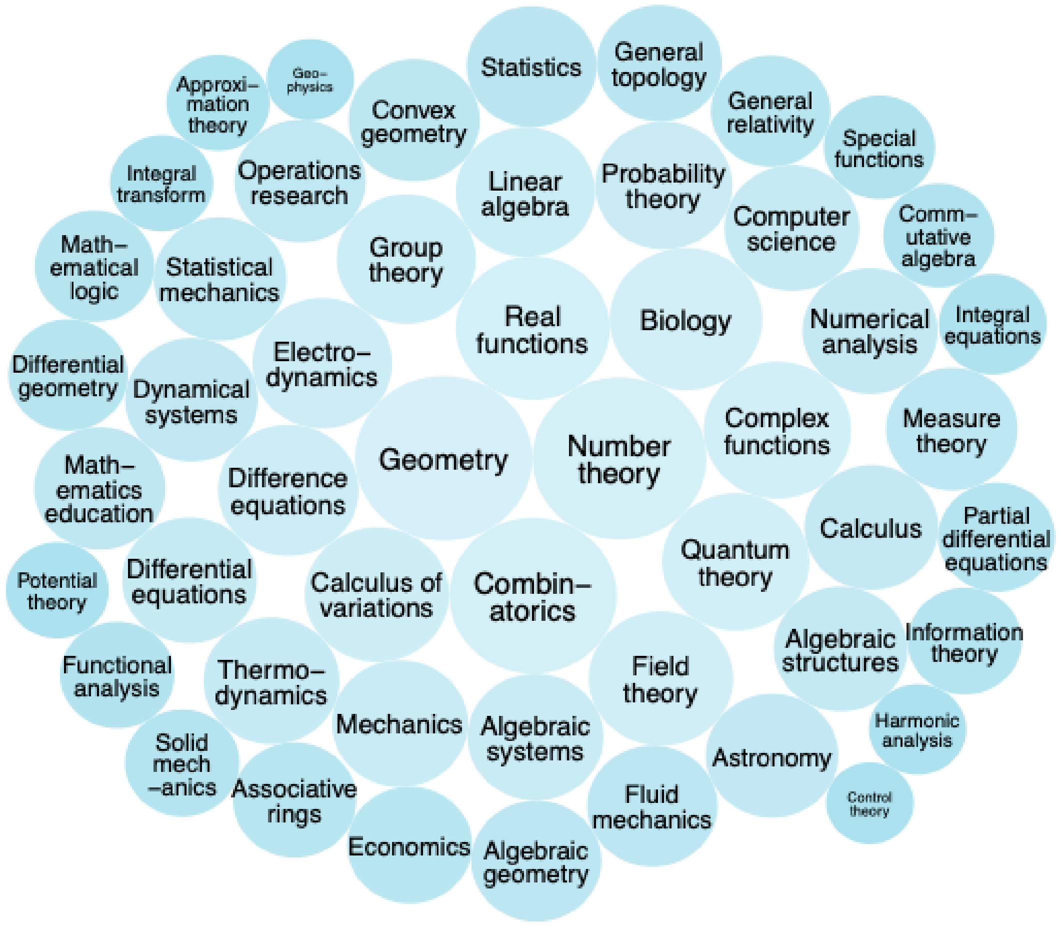
BUT What's the simplest approach to achieve both test-time scaling and strong reasoning performance?

s1-32B

- Trained on 1000 samples (from MATH, GPQA, AIME24)
- Sample test-time technique called budget forcing that controls thinking duration
- SFT on off the shelf pretrained model (26 minutes & 16 H100 GPUs)
  - Qwen2.5-32B-Instruct
- Competitive with OpenAI's o1-preview

1K Dataset

- 3 well known reasoning benchmarks: MATH, GPQA, AIME24
- 3 main principles
  - Quality
  - Difficulty
  - Diversity
- 3 parts to each sample
  - Prompt
  - Reasoning trace
    - Google Gemini Flash Thinking API
  - Answer

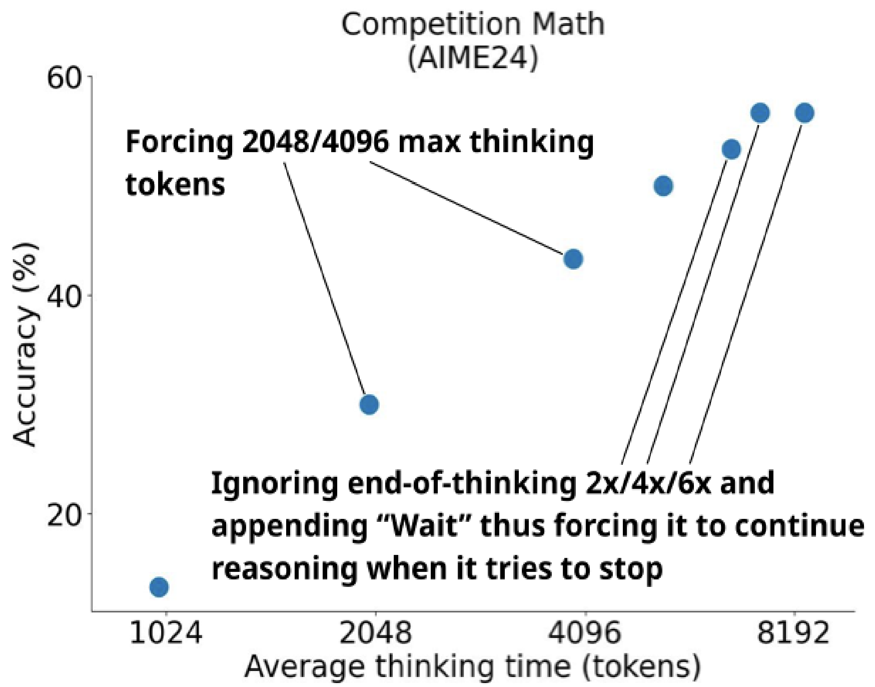


Data Filtering

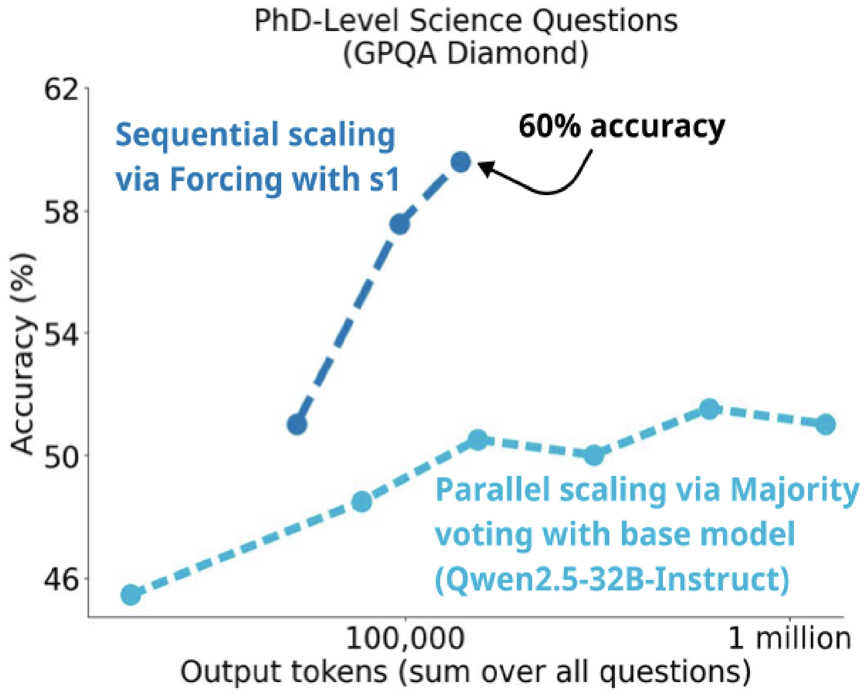
- 59K -> 1K
- Quality
  - Exclude API errors, formatting issues, inconsistent question numbering
- Difficulty
  - Evaluate 2 models on each question: Qwen2.5-7B-Instruct & Qwen2.5-32B-Instruct
    - Correctness assessed by Claude 3.5 Sonnet against reference solution
  - Token length
- Diversity

Test Time Scaling: Sequential scaling

- **Sequential** – scales better because computations build on intermediate results
  - Allow for deeper reasoning and iterative refinement



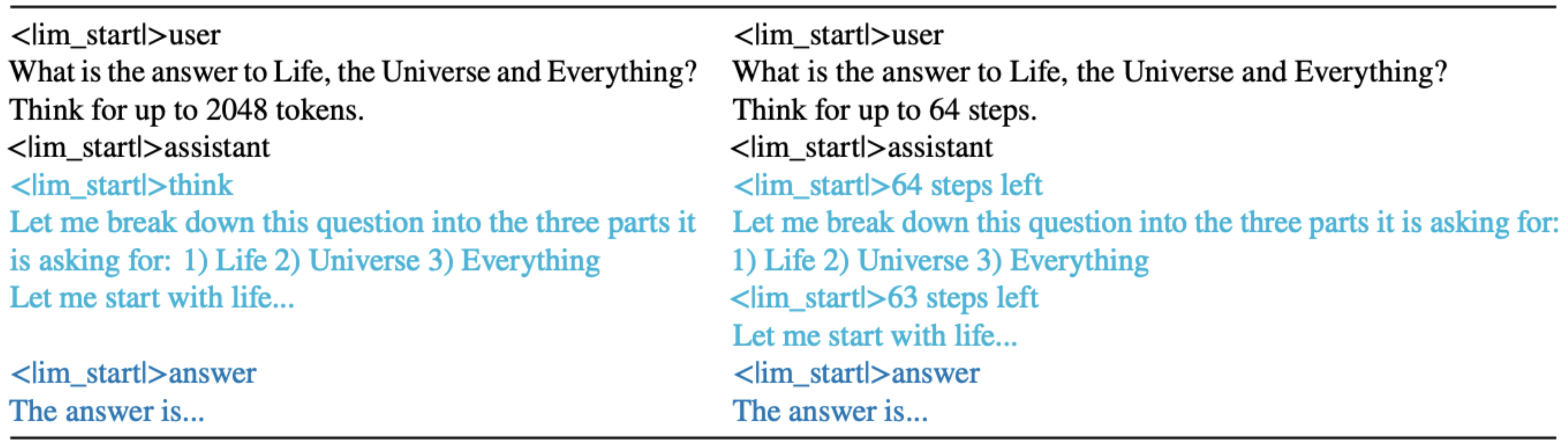
(a) Sequential scaling via budget forcing



(b) Parallel scaling via majority voting

Test-Time Control Methods

- **Token-conditional control**
- Step-conditional control
- Class-conditional control



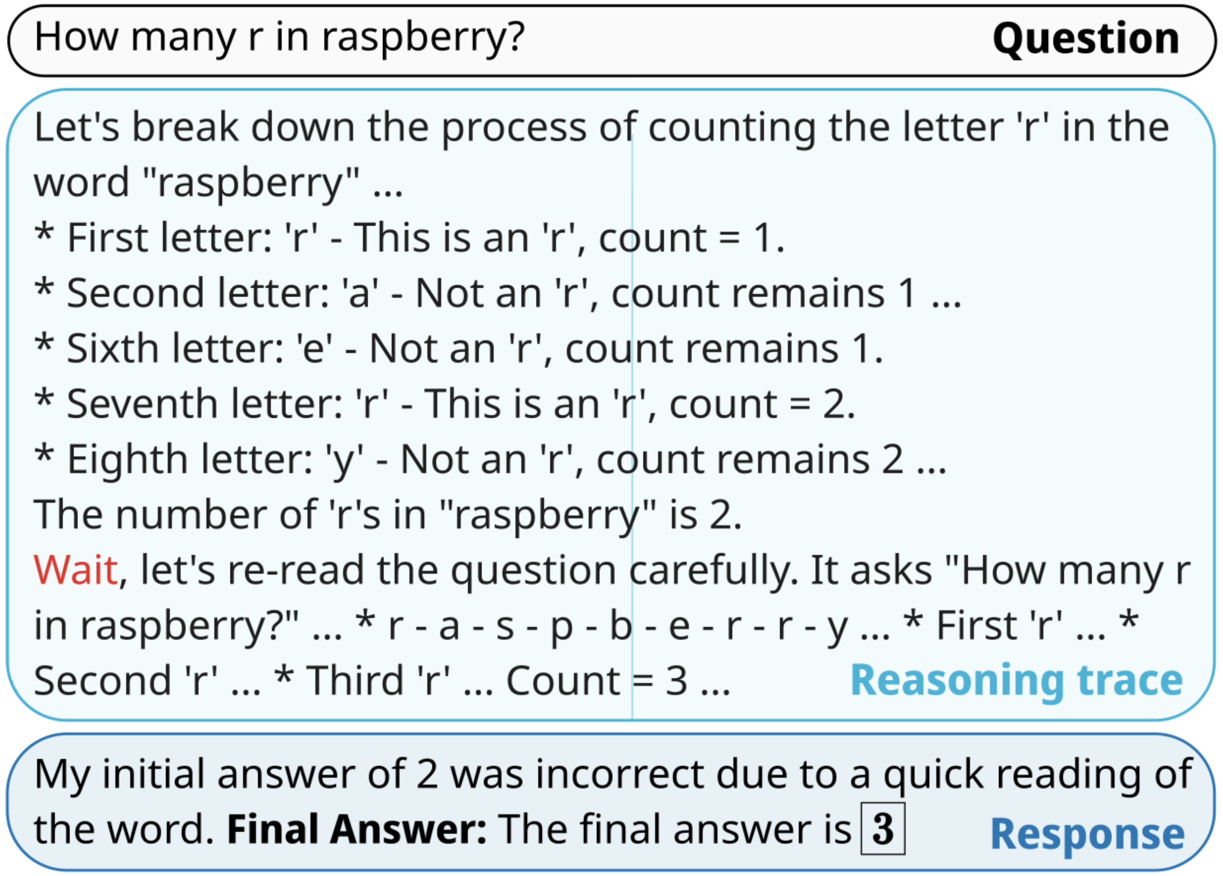
```
<|im_start|>user
What is the answer to Life, the Universe and Everything?
Think for up to 2048 tokens.
<|im_start|>assistant
<|im_start|>think
Let me break down this question into the three parts it
is asking for: 1) Life 2) Universe 3) Everything
Let me start with life...

<|im_start|>answer
The answer is...
```

```
<|im_start|>user
What is the answer to Life, the Universe and Everything?
Think for up to 64 steps.
<|im_start|>assistant
<|im_start|>64 steps left
Let me break down this question into the three parts it is asking for:
1) Life 2) Universe 3) Everything
<|im_start|>63 steps left
Let me start with life...
<|im_start|>answer
The answer is...
```

Figure 10. **Token and step instruction data formats for controlling test-time compute.** We only train our model on the reasoning trace and the answer.

Test Time Scaling: Budget Forcing

- **Budget Forcing** – simple decoding time intervention by forcing a max/min number of thinking tokens
  - End-of-thinking token delimiter ("Final Answer")
  - "Wait" to encourage model to reflect


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 $\boxed{3}$ **Response**

Performance

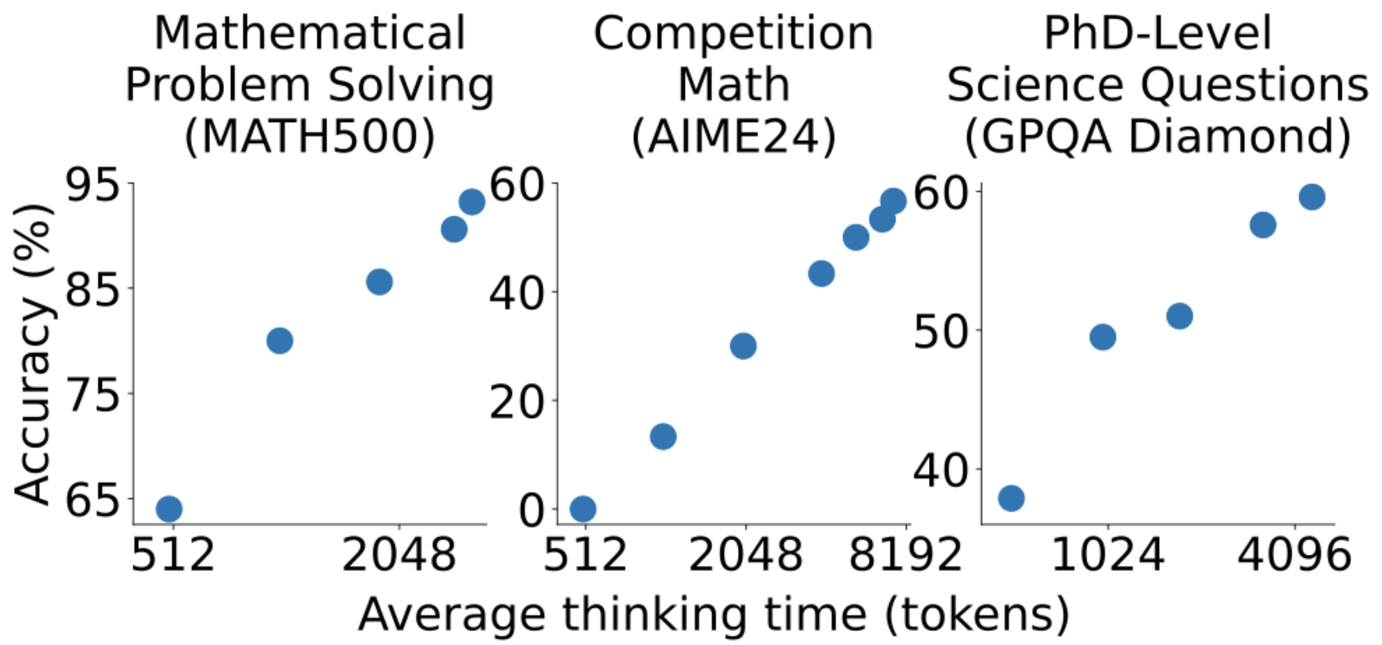
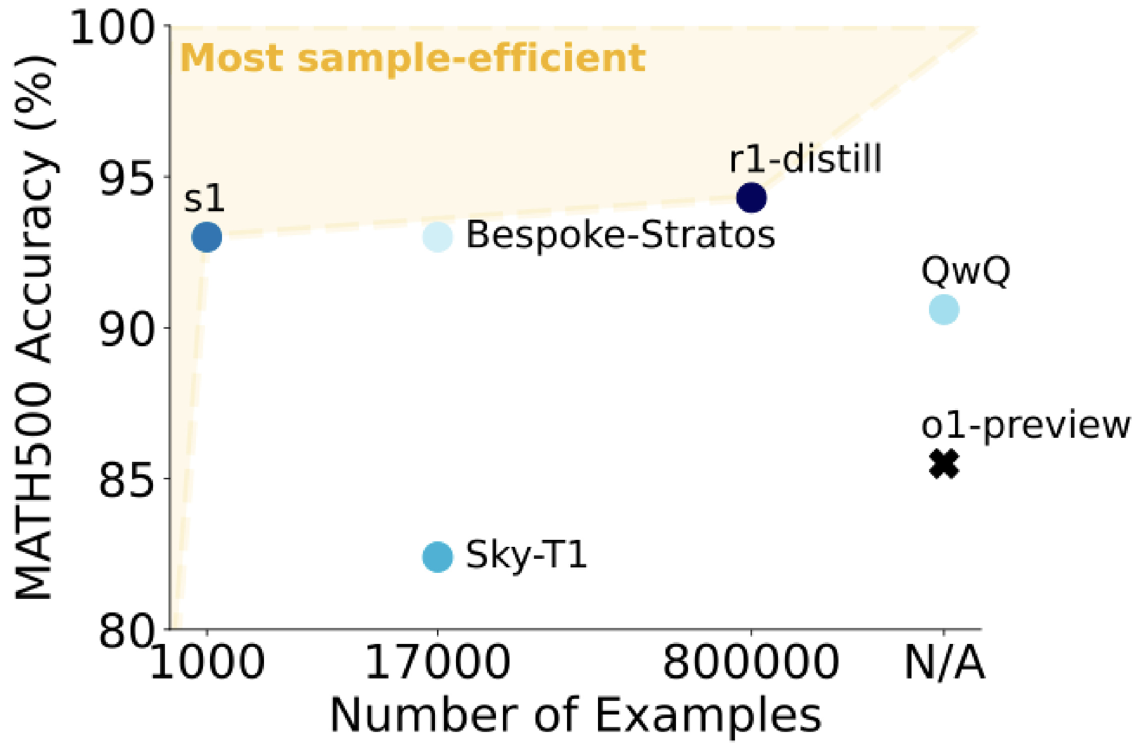


Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

Sample Efficiency

- S1-32B as the most sample efficient open data reasoning model
  - Model nearly matches Gemini 2.0
  - r1-32B has stronger performance
    - But also trained on 800x more reasoning samples



Sahlar Salehi
rmh7ce

Recent Paper: Scaling LLM Test-Time Compute Optimally canbe More Effective than Scaling Model Parameters

- Charlie Snell, Jaehoon Lee, Kelvin Xu, 2 Aviral Kumar
  - Google DeepMind, UC Berkeley

- **If we give a model more inference time, can it improve accuracy enough that we can decrease the model size and get comparable results to LLM?**
  - Powerful lightweight models

- How do models best use additional inference time?
- What is the tradeoff between test-time compute and pretraining compute?

Method	What	How						Where	How Well
		SFT	RL	STIMULATION	SEARCH	VERIFICATION	AGGREGATION		
DSC (Snell et al., 2024)	Parallel, Sequential	✗	✗	✗	Beam Search, LookAhead Search	Verifier	(Weighted) Best-of-N, Stepwise Aggregation	Math	Pass@1, FLOPs-Matched Evaluation

Method	What			
		SFT	RL	STIMULATION
DSC (Snell et al., 2024)	Parallel, Sequential	✗	✗	✗

How			Where	How Well
SEARCH	VERIFICATION	AGGREGATION		
Beam Search, LookAhead Search	Verifier	(Weighted) Best-of-N, Stepwise Aggregation	Math	Pass@1, FLOPs-Matched Evaluation

Use of Additional Test Time

- How is additional time used to improve model accuracy?

- Two ways we can improve model accuracy
  - Modify the model's proposal distribution
    - Proposal distribution : probability distribution of predicted tokens
  - Use a post-hoc verifier to modify/select outputs

Modifying the Proposal Distribution

- Could augment prompt with tokens, but not effective at test time
- Better: use RL inspired finetuning, iteratively improve outputs
  - Model produces output
  - Self-critique technique to evaluate output
  - Use evaluation to improve proposal distribution->output again
- Question: is it better to use additional test time to iteratively revise a single output, or should it be used for model to generate multiple independent responses and select the best one?

Optimizing the Verifier

- Verifier selects best answer from proposal distribution
  - Best-of-N sampling: sample N complete solutions, use verifier to select best
- Need to train process-based verifier, or process reward model (PRM)
  - Idea: predict correctness of intermediate steps in solution
  - Use step-wise predictions to tree search over solutions, find best process
- Question: what search technique (best-of-N, beam search, lookahead search) performs best with PRM

Allocating Test-Time Budget

- Several hyperparameters
  - How much time generating independent samples vs revising samples
  - Which search algorithm for verifier
- Test time compute optimal scaling strategy
  - Optimal hyperparameter configuration to maximize performance benefits on a specific prompt
  - Strategy that works best will depend on the prompt/difficulty

$$\theta^*_{q,a^*(q)}(N) = \operatorname{argmax}_\theta \left(\mathbb{E}_{y\sim\mathrm{Target}(\theta,N,q)}\left[\mathbb{1}_{y=y^*(q)}\right]\right),$$

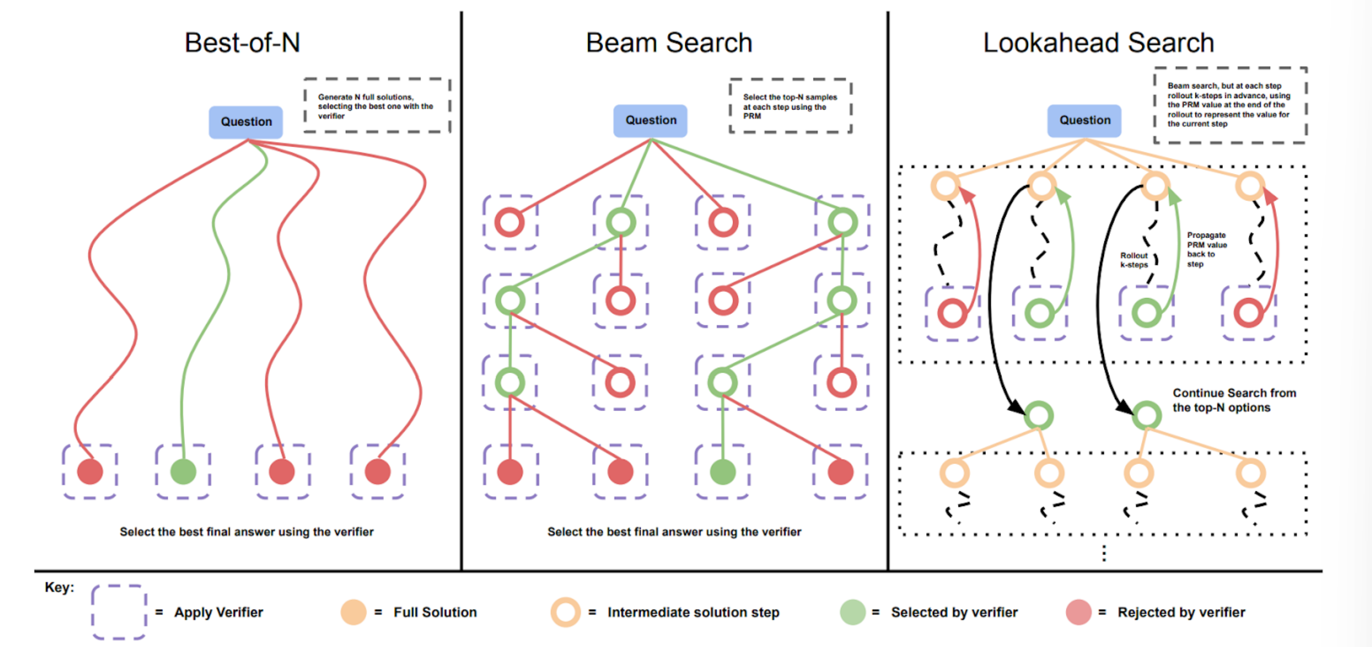
Estimating Question Difficulty

- Need to approximate difficulty to determine optimal strategy
  - Put pass@1 rate estimated from 2048 samples in bins of increasing difficulty
- No ground truth difficulty-> rely on model-predicted notion of difficulty
  - Use learned verifier to bin samples, adds additional one-time cost during inference

Experiments

- Scaling test time compute via verifiers
  - Best of N vs Beam Search vs Lookahead Search
- Scaling test time compute via refining proposal distribution
  - How to train and use revision models
  - Parallel vs sequential sampling to optimize proposal distribution
- Ratios of pretraining vs inference time
- PaLM 2 base model, evaluated on MATH benchmark

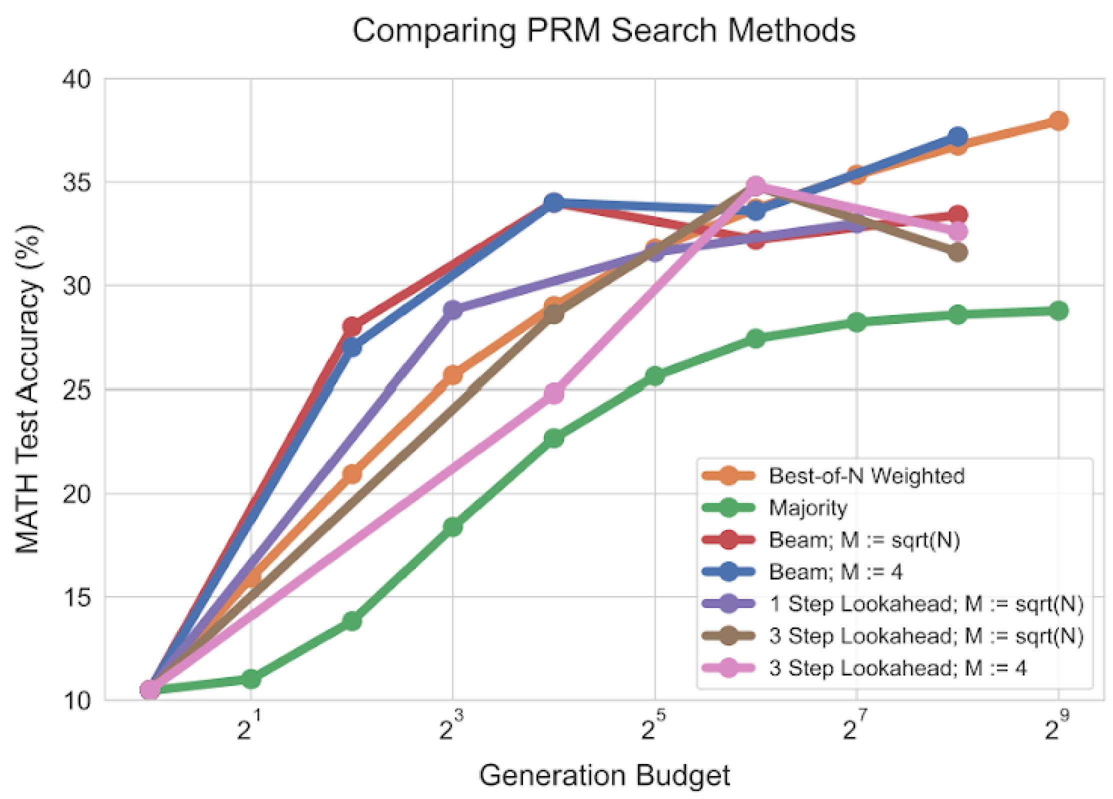
Scaling via Verifiers: Search Methods

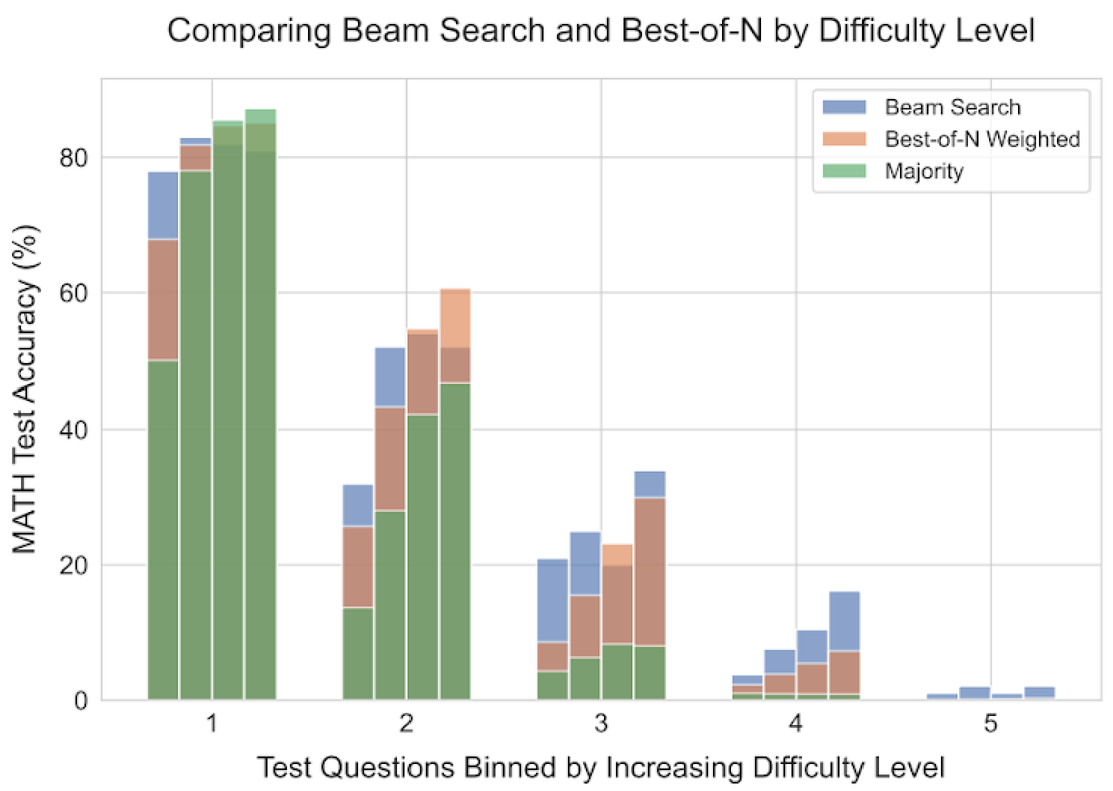


Best-of-N
Generate N full solutions, selecting the best one with the verifier
Question
Select the best final answer using the verifier
Beam Search
Select the top-N samples at each step using the PRM
Question
Select the best final answer using the verifier
Lookahead Search
Beam search, but at each step rollout k-steps in advance, using the PRM value at the end of the rollout to represent the value for the current step
Question
Rollout k-steps
Propagate PRM value back to step
Continue Search from the top-N options
Key:
= Apply Verifier
= Full Solution
= Intermediate solution step
= Selected by verifier
= Rejected by verifier

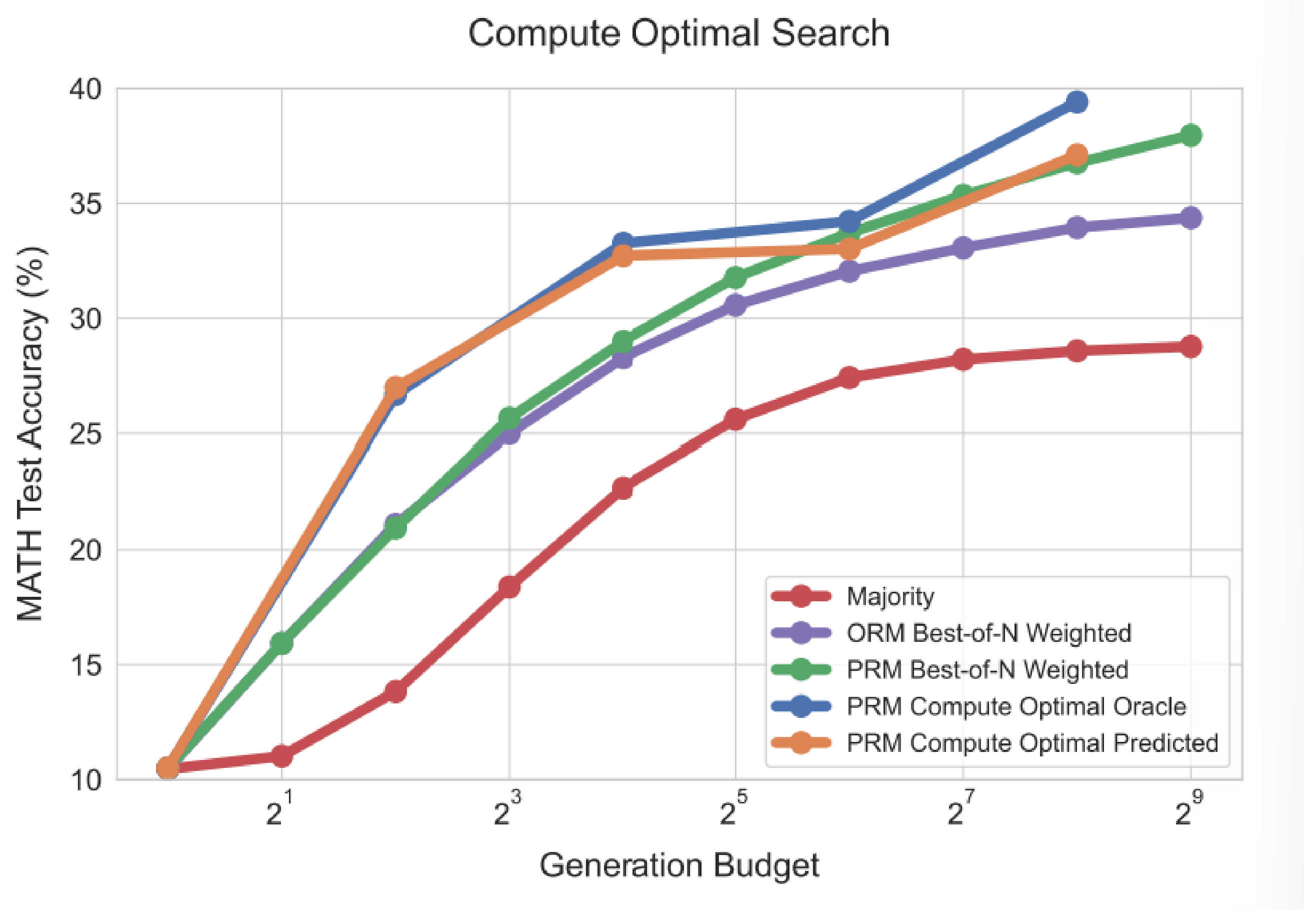
Figure 2 | ***Comparing different PRM search methods.*** **Left:** Best-of-N samples N full answers and then selects the best answer according to the PRM final score. **Center:** Beam search samples N candidates at each step, and selects the top M according to the PRM to continue the search from. **Right:** lookahead-search extends each step in beam-search to utilize a k-step lookahead while assessing which steps to retain and continue the search from. Thus lookahead-search needs more compute.

Scaling via Verifiers: Results





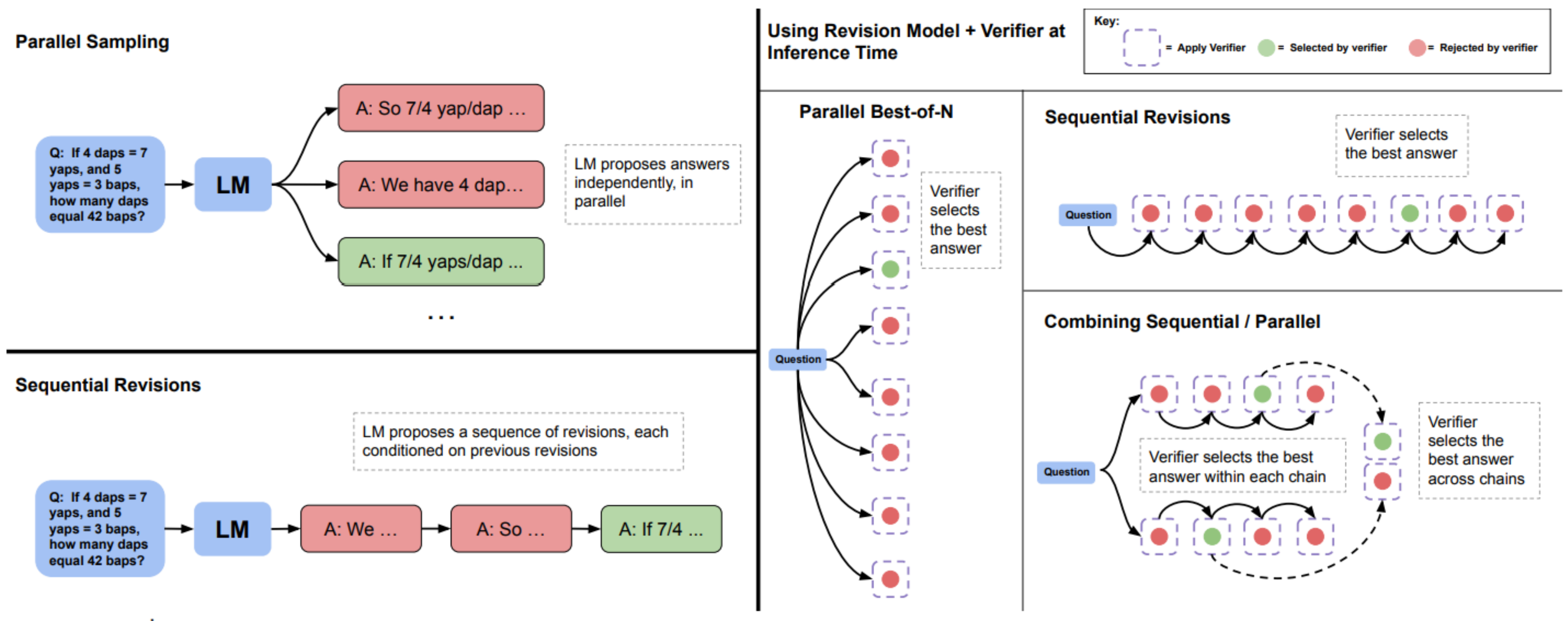
Scaling via Verifiers: Results



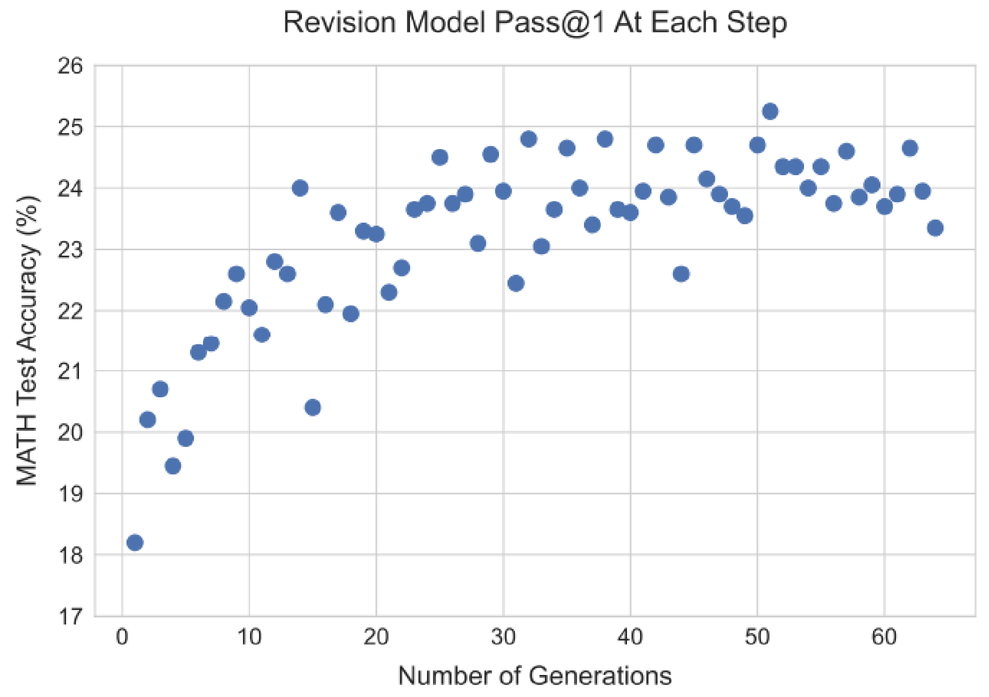
Scaling via Refining Proposal Distribution: Training and Using Revision Models

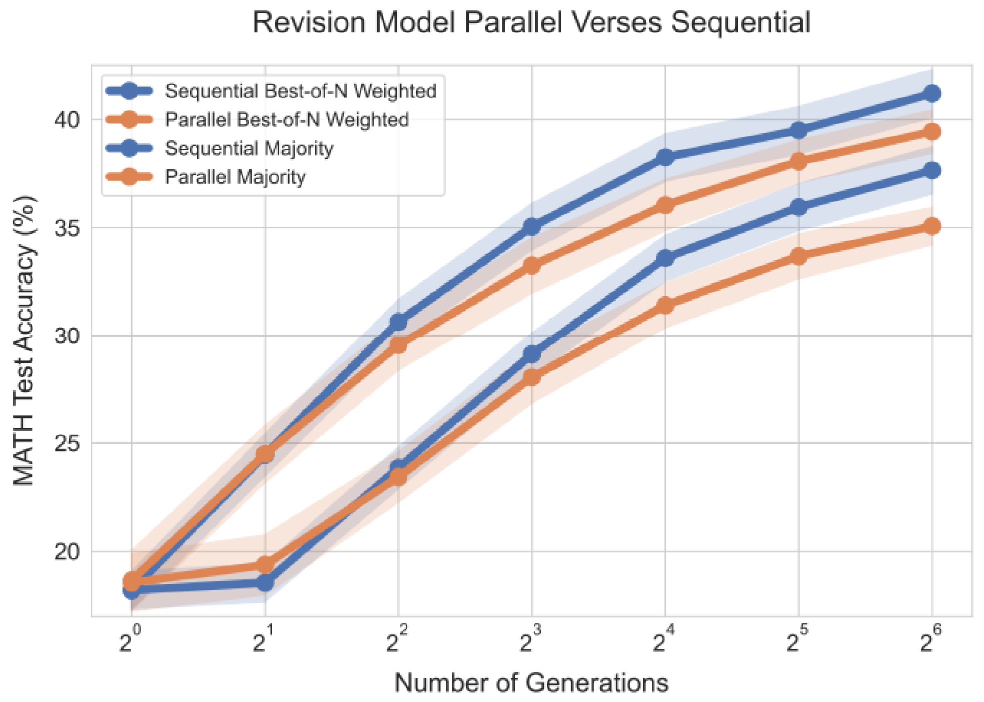
- Model trajectory of incorrect answers approaching and arriving at a correct answer
  - Want to correlate incorrect and correct answers to teach model to point out mistakes
- For each question
  - Sampled 64 prallel responses, pairing correct answers with sequence of up to 4 incorrect answers in context
  - Select incorrect answer more closely related to final correct answer->trajectory from incorrect to correct

Scaling via Refining Proposal Distribution: Parallel Sampling vs Sequential Revisions

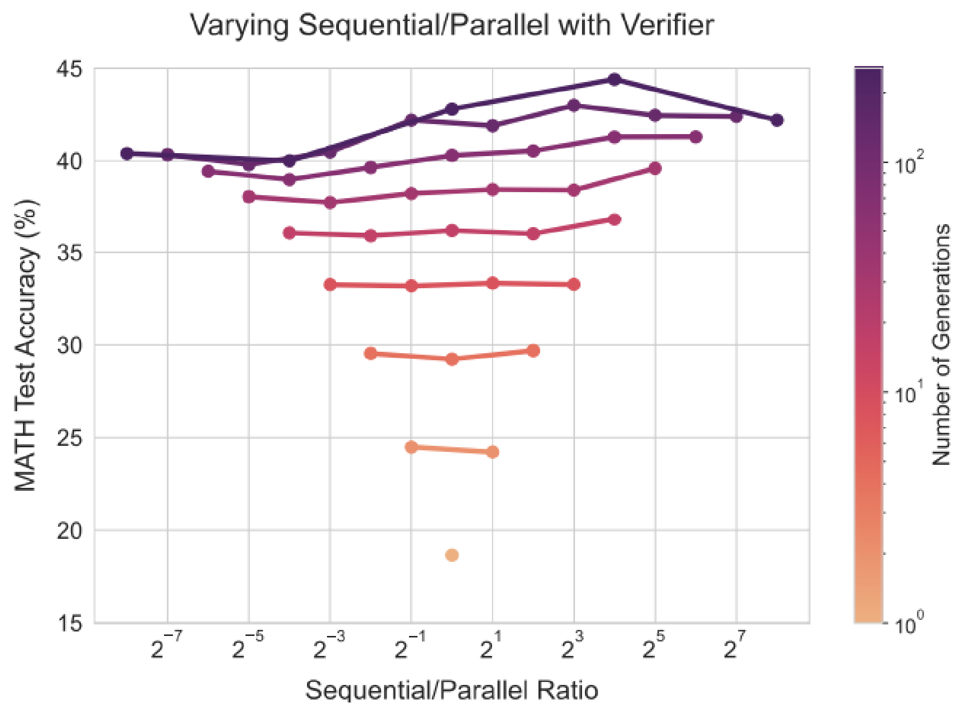

Parallel Sampling
Q: If 4 daps = 7 yaps, and 5 yaps = 3 baps, how many daps equal 42 baps?
LM
A: So 7/4 yap/dap …
A: We have 4 dap…
A: If 7/4 yaps/dap ...
LM proposes answers independently, in parallel
...
Sequential Revisions
LM proposes a sequence of revisions, each conditioned on previous revisions
Q: If 4 daps = 7 yaps, and 5 yaps = 3 baps, how many daps equal 42 baps?
LM
A: We ...
A: So ...
A: If 7/4 ...
Using Revision Model + Verifier at Inference Time
Key:
= Apply Verifier
= Selected by verifier
= Rejected by verifier
Parallel Best-of-N
Question
Verifier selects the best answer
Sequential Revisions
Question
Verifier selects the best answer
Combining Sequential / Parallel
Question
Verifier selects the best answer within each chain
Verifier selects the best answer across chains

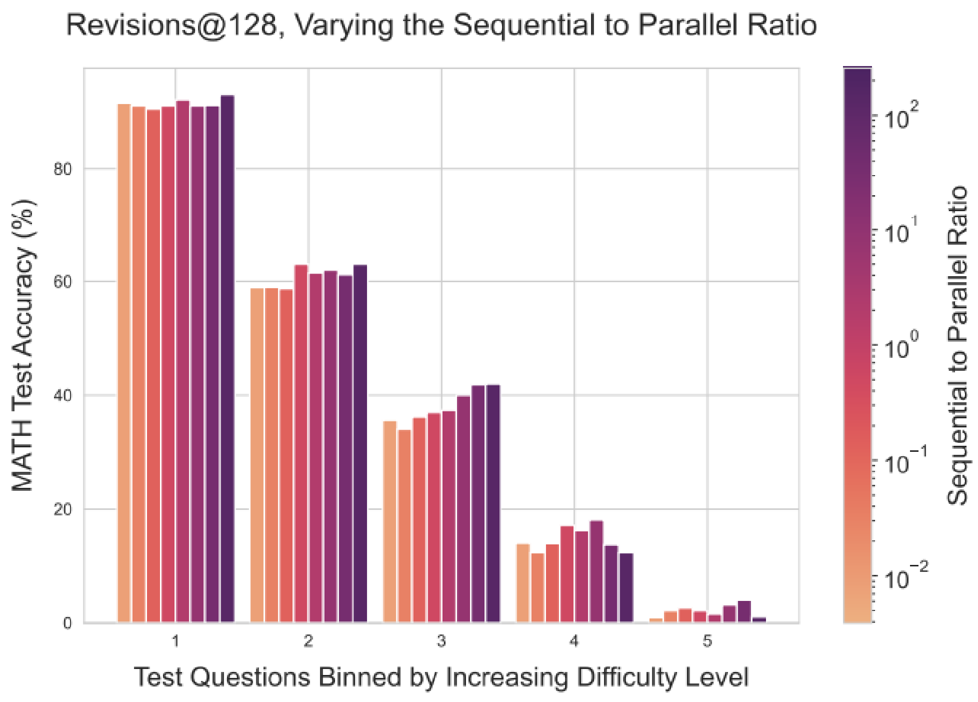
Scaling via Refining Proposal Distribution: Results



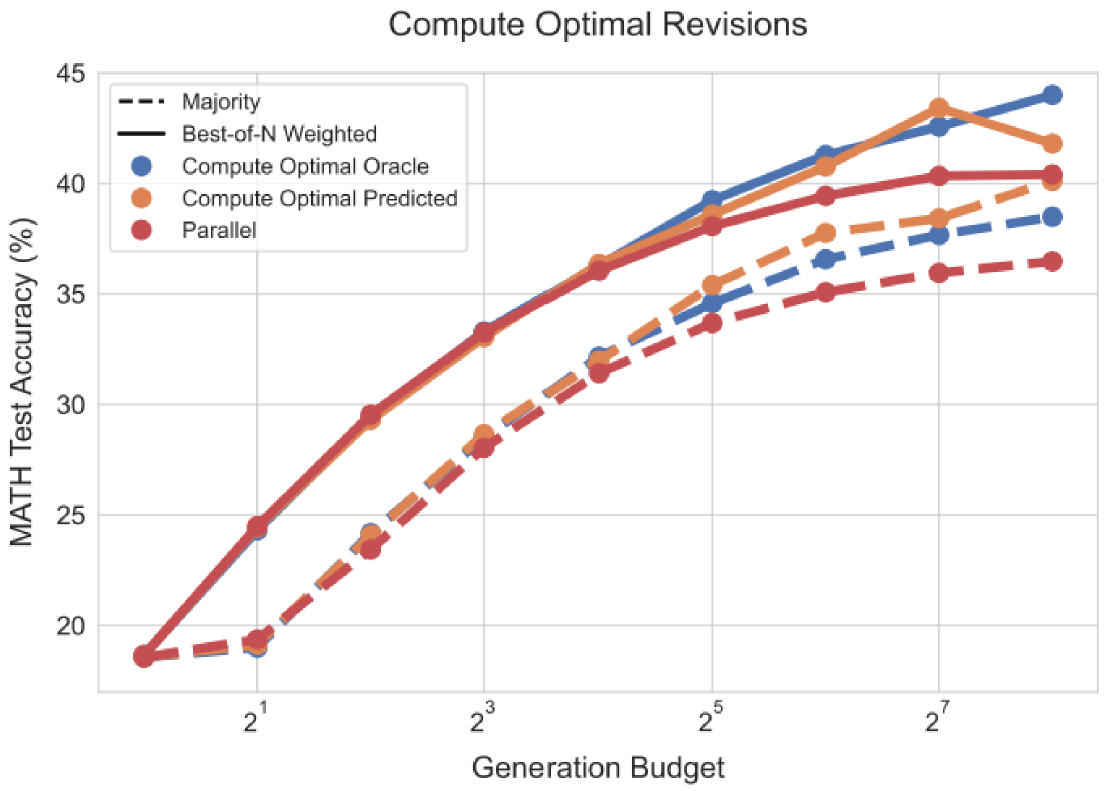


Scaling via Refining Proposal Distribution: Parallel vs Sequential Ratio Results





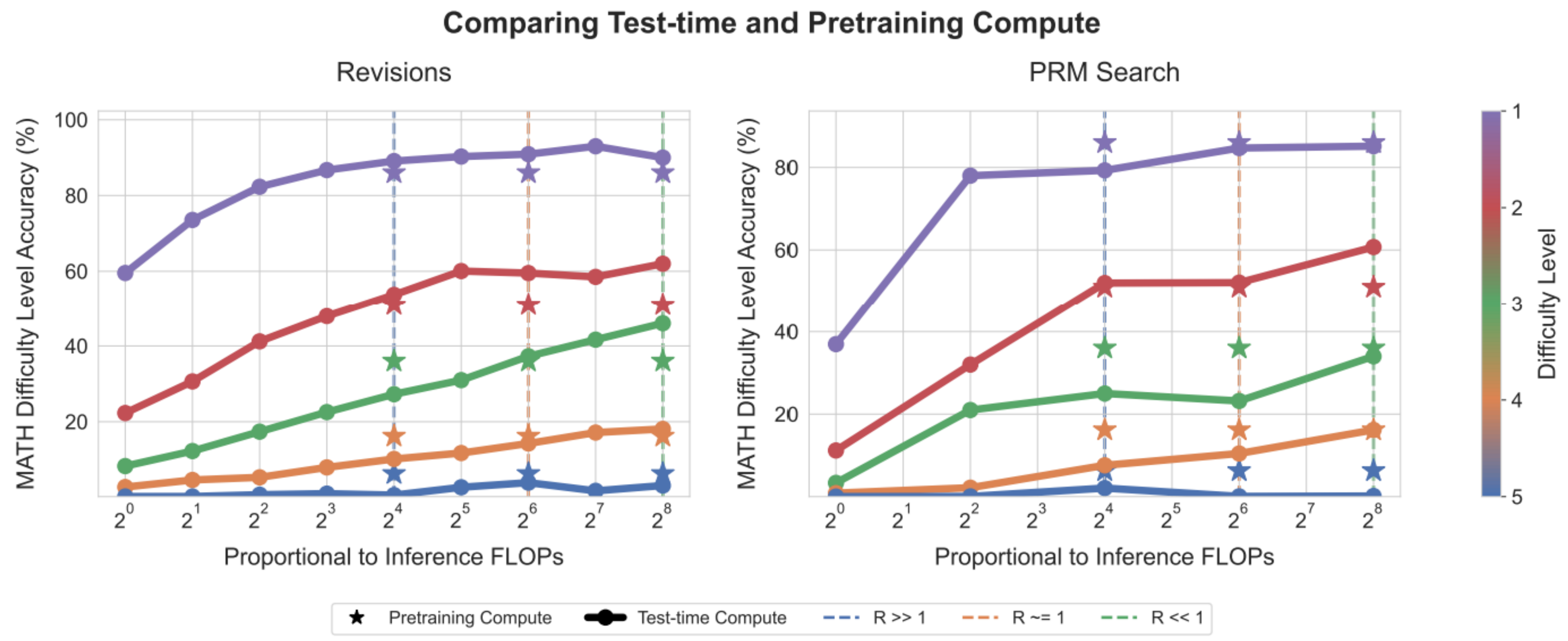
Scaling via Refining Proposal Distribution: Compute Optimal Results



Exchanging Pretraining and Test Time Compute

- Model pretrained with X FLOPs, we want to run Y FLOPs of inference on the model
- We want to improve performance by increasing total FLOP budget by a factor of M
  - That is M(X+Y) total FLOPs across pretraining and inference
- Should we spend additional FLOPs on increased pretraining compute or increased test-time compute?
- Need to define exchange rate between pretraining and inference FLOPs
  - X=6ND_pretrain, Y=2ND_inference
  - Amount of inference compute we can use to match the FLOPs of the larger pretrained model depends on ratio R=D_inference/D_pretrain

Results: Comparing Test-Time and Pretraining Compute



Discussions and Future Work

- Test time compute and Pretraining compute not 1-to-1 exchangeable, depends on the prompt
- Difficulty assessment requires a non-trivial amount of additional test time compute, potentially taking away from performance
- Study focused purely on test time compute scaling and trading off for additional pretraining
  - Potential direction for putting test-time compute into the base LLM to enable self-improvement during inference

References

- https://openreview.net/forum?id=VTF8yNQM66
- https://arxiv.org/abs/2402.18060
- https://arxiv.org/abs/2311.16502
- https://arxiv.org/abs/2501.12948
- https://arxiv.org/abs/2503.04697
- https://arxiv.org/abs/2503.04697
-

Aaditya Ghosalkar ag5jk

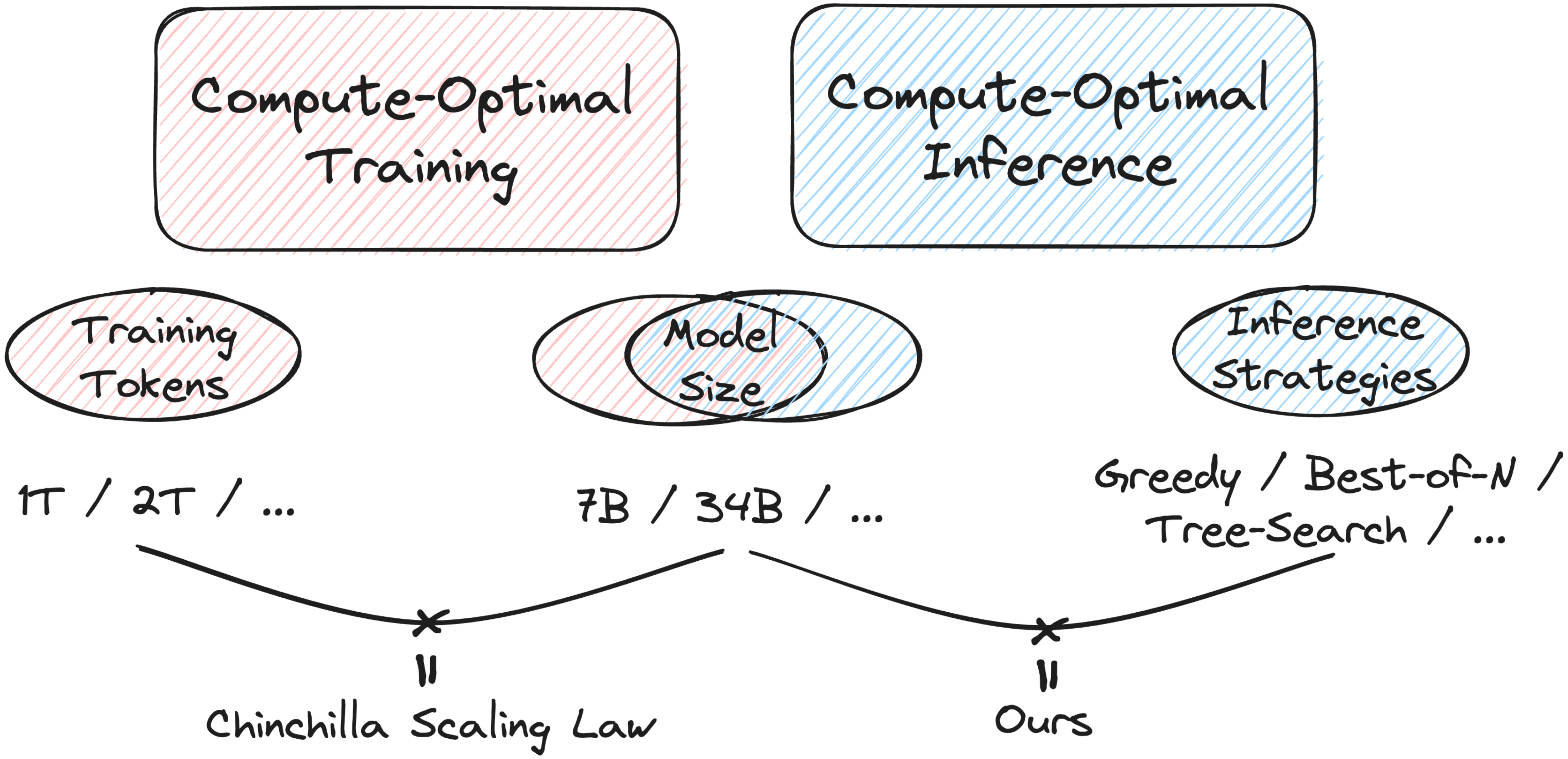
INFERENCE SCALING LAWS: AN EMPIRICAL ANALYSIS OF COMPUTE-OPTIMAL INFERENCE FOR LLM PROBLEM-SOLVING

- Yangzhen Wu , Zhiqing Sun , Shanda Li , Sean Welleck , Yiming Yang
  - Institute for Interdisciplinary Information Sciences, Tsinghua University
  - School of Computer Science, Carnegie Mellon University
- Inference scaling laws
  - Can we use better strategies to make smaller models perform as well as larger ones?
    - Example: LLemme-7B + tree search > Llemma-34-B
- Strategies researched
  - Greedy search, Best of n, Majority & Weighted voting, Tree search

Motivation and Background

- Current issue:
  - Big models are more powerful but require more computing
  - Smaller models are cheaper, but less capable

- Most research is based on optimizing training scaling laws, like Chinchilla Scaling Laws, and how one would optimize a budget based on training size.
- Goal of this paper: Is it possible to make smaller models perform the same as larger ones by reducing the overhead of a trained model generating answers
  - The phase where a trained model is used to generate answers is called the **Inference Phase**


Compute-Optimal Training
Compute-Optimal Inference
Training Tokens
Model Size
Inference Strategies
1T / 2T / ...
7B / 34B / ...
Greedy / Best-of-N / Tree-Search / ...
Chinchilla Scaling Law
Ours

Problem Statement – Compute Optimal Inference

Question: Given a fixed FLOPs budget, how should one select an optimal model size for the policy model, and an effective inference strategy to maximize performance (i.e., accuracy)?

N is the Model size, T is the number of tokens generated, and S is the inference strategy

The goal is to minimize the Error rate E under the test time compute constraint of FLOPS(N,T,S) = C

$$(N_{\text{opt}}(C), T_{\text{opt}}(C); \mathcal{S}) = \underset{(N,T,\mathcal{S}) \text{ s.t. } \mathrm{FLOPs}(N,T,\mathcal{S})=C}{\arg\min} E(N, T; \mathcal{S})$$

Nopt(C) and Topt(C) denote the optimal allocation of a computational budget C

Inference Strategies

The paper aims to examine between different inference strategies on the performance and the cost using the metrics discussed.

Sampling-based methods

- Greedy Decoding – fastest, picks most likely token, doesn't explore alternatives
- Majority voting – generates multiple completions, chooses the most common answer
- Weighted majority voting – like majority voting, but tanks completion by confidence or reward

Tree-based methods

- MCTS (Monte Carlo Tree Search): based on game playing AI, it simulates multiple paths
- REBASE : proposed inference strategy by the paper

Voting-Based Inference

Majority Voting:

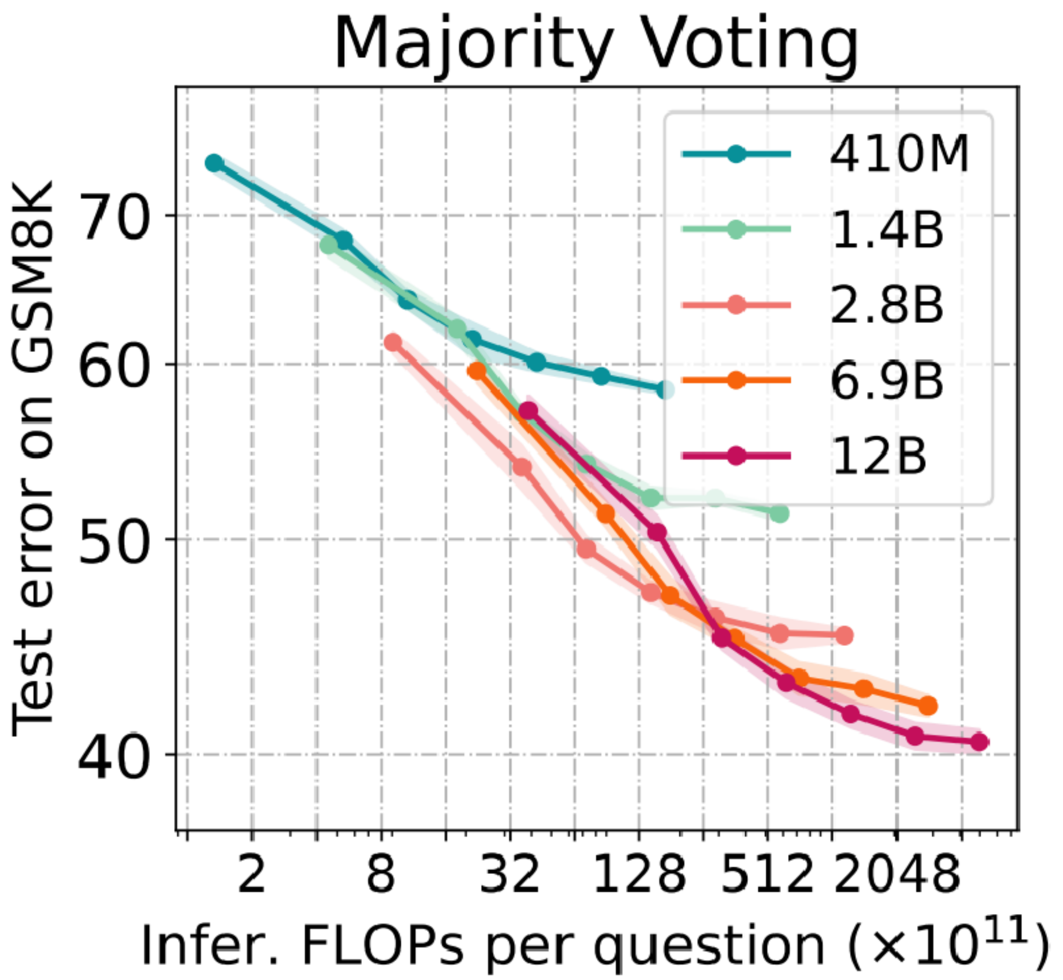
- Run the model multiple times with the same prompt
- Collect the outputs and pick the **most frequent** answer
- Assumes that common answers are more likely to be correct

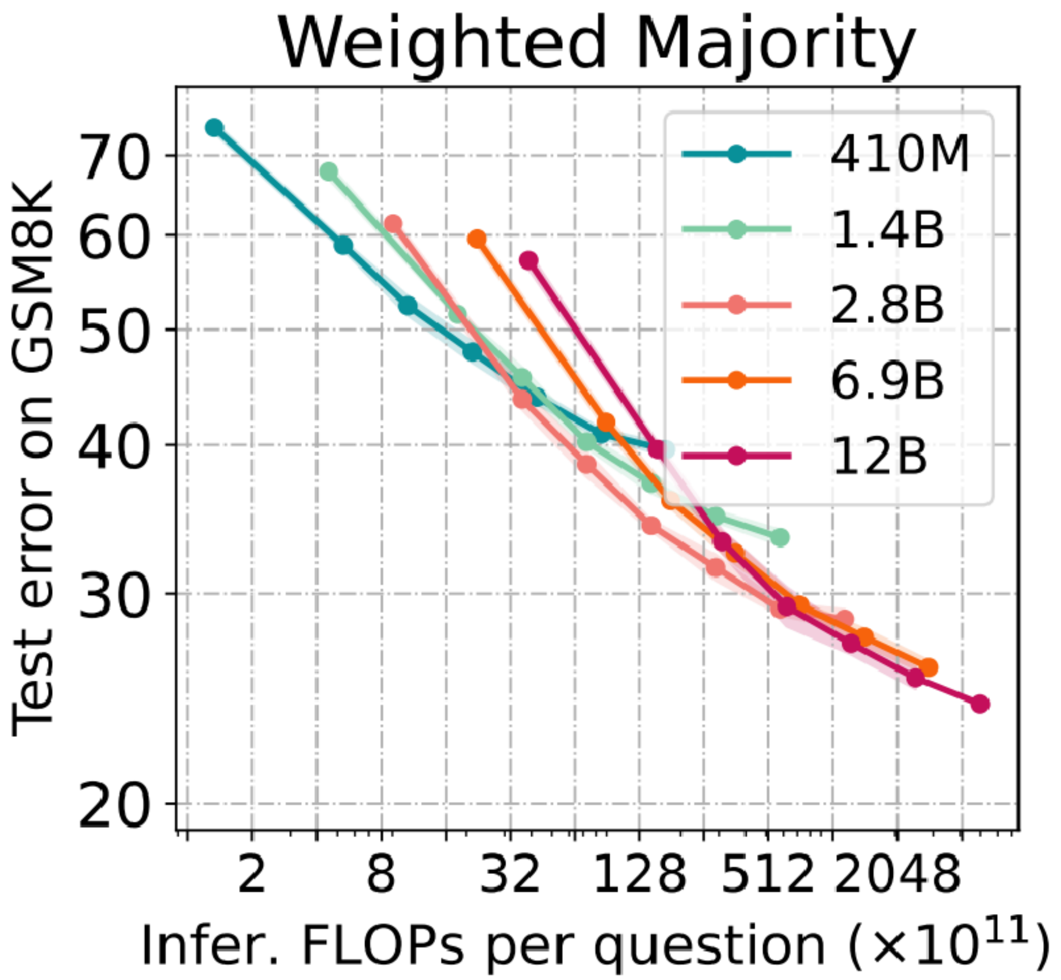
Weighted Majority Voting:

- Like majority voting, but each answer is **scored** (e.g., by a reward model)
- Select the answer with the **highest total weighted score**

Limitations:

- Performance **depends on number of samples**
- More samples = more compute → diminishing returns
- Eventually, sampling more does **not yield better results**
- These methods are simple, but they reach a plateau. To go further, we need structured search.


Majority Voting
Test error on GSM8K
Infer. FLOPs per question ($\times 10^{11}$)
410M
1.4B
2.8B
6.9B
12B
70
60
50
40
2
8
32
128
512
2048


Weighted Majority
Test error on GSM8K
Infer. FLOPs per question ($\times 10^{11}$)
410M
1.4B
2.8B
6.9B
12B
70
60
50
40
30
20
2
8
32
128
512
2048

Tree-Based Inference

Why sampling isn't enough:

- Sampling generates full completions blindly
- It lacks structure and wastes compute on bad outputs

Enter tree search:

- Builds solutions incrementally (step-by-step)
- Allows dynamic allocation of compute to promising paths

Example: MCTS (Monte Carlo Tree Search)

- Simulates many possible completions (rollouts)
- Assigns rewards based on full solution outcomes
- Backpropagates rewards to improve search decisions

Drawback:

- MCTS is compute-heavy – expensive rollouts for every path
- Doesn’t scale well for LLM inference, especially with long solutions

The goal is to find a tree search that's lightweight, greedy, and guided.

REBASE

Step-by-step generation:

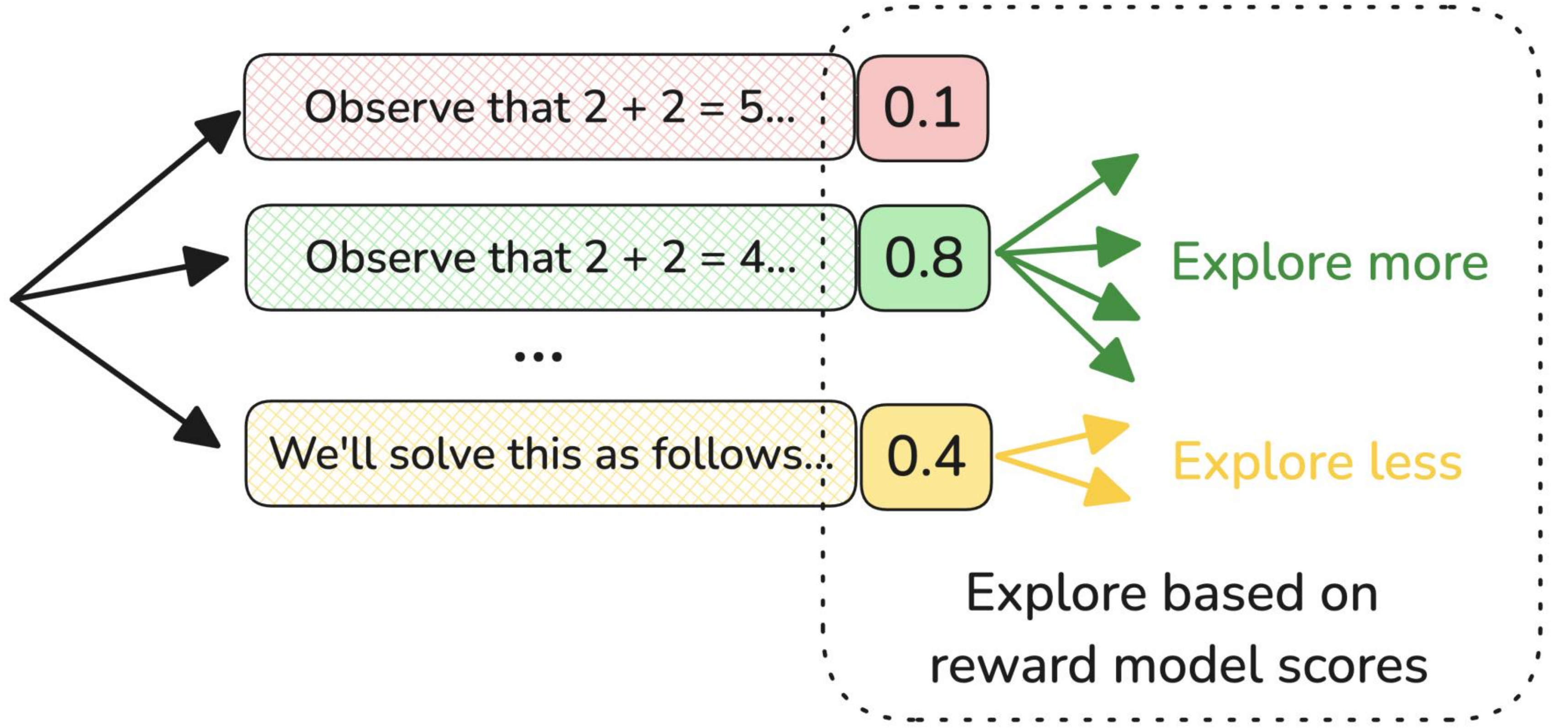
- REBASE builds answers token-by-token, as a tree
- Each node = a partial solution

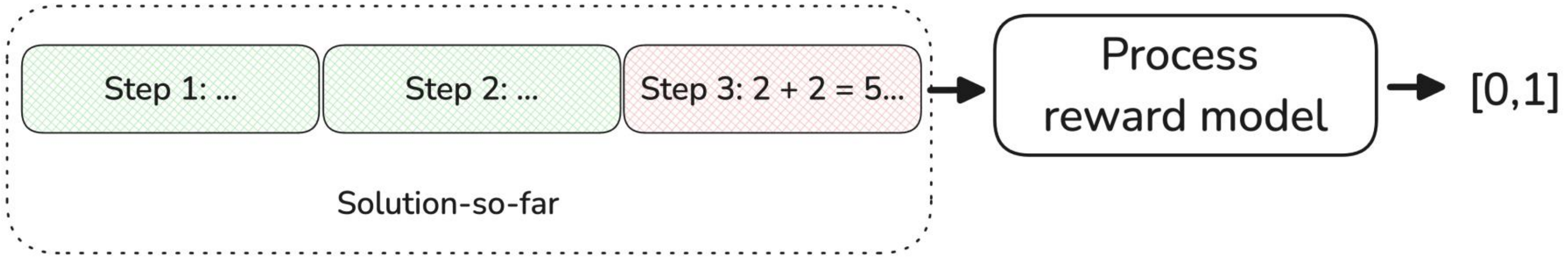
Reward-guided expansion:

- A learned reward model scores each partial solution
- Nodes are expanded based on softmax-normalized scores
- Higher scores → more children explored

Compute-efficient:

- Avoids full rollouts (like MCTS)
- Prioritizes only the most promising paths


Observe that 2 + 2 = 5...
0.1
Observe that 2 + 2 = 4...
0.8
Explore more
...
We'll solve this as follows...
0.4
Explore less
Explore based on reward model scores


Step 1: ...
Step 2: ...
Step 3: 2 + 2 = 5...
Solution-so-far
Process reward model
[0,1]

Experiment Setup

Benchmarks:

- GSM8K – Grade-school math word problems (easy, short reasoning)
- MATH – High school competition problems (long, multi-step reasoning)

Models tested:

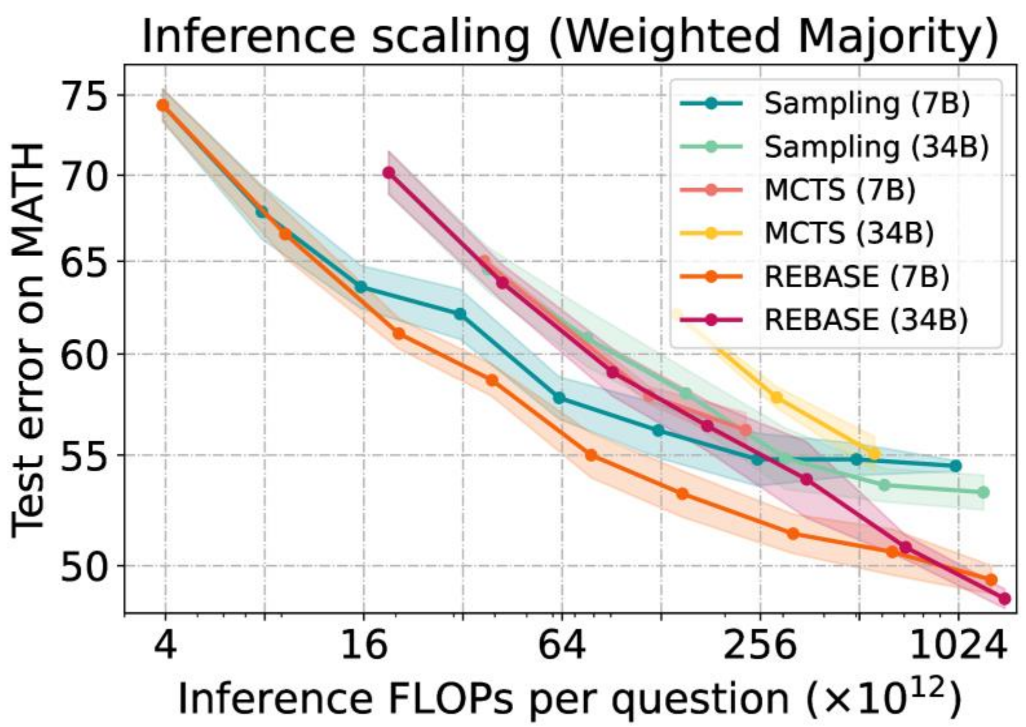
- Llemma-7B and Llemma-34B (fine-tuned for math)
- Pythia and Mistral-7B (open LLM baselines)

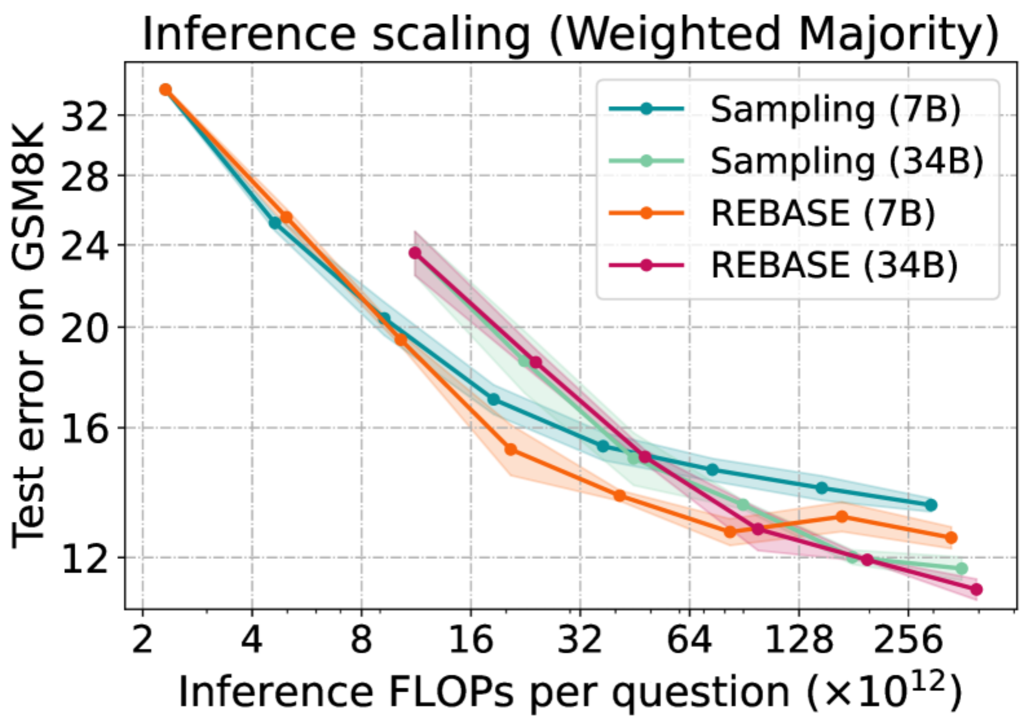
Inference methods evaluated:

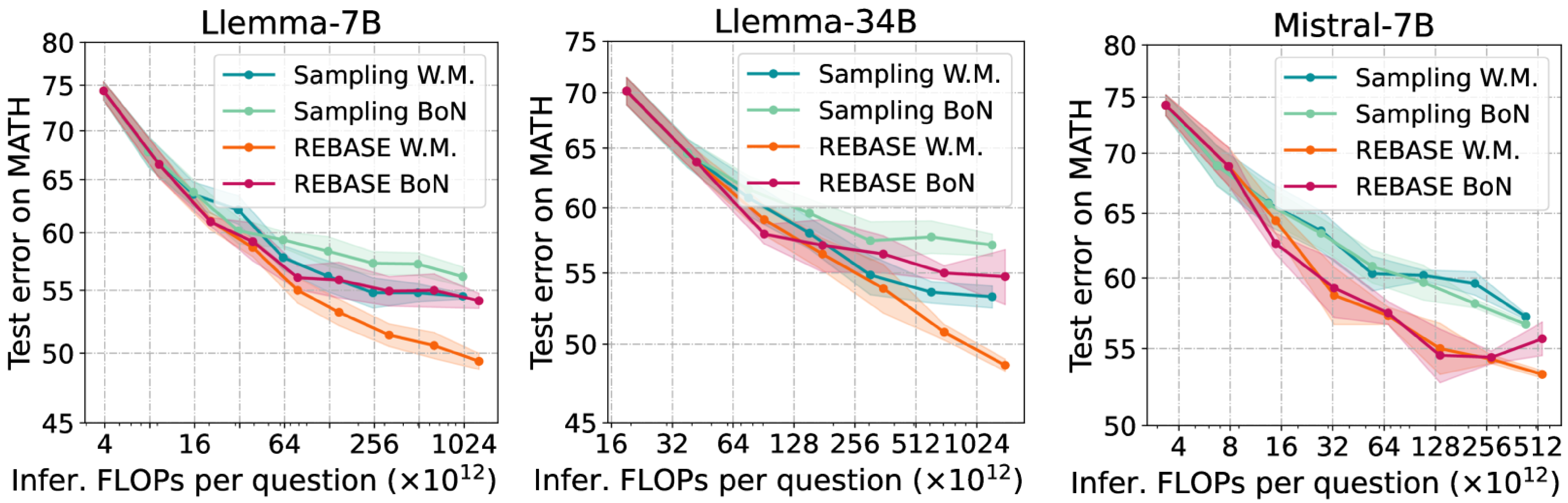
- Greedy decoding, Sampling, Majority & Weighted voting
- MCTS (baseline tree search)
- REBASE (proposed method)

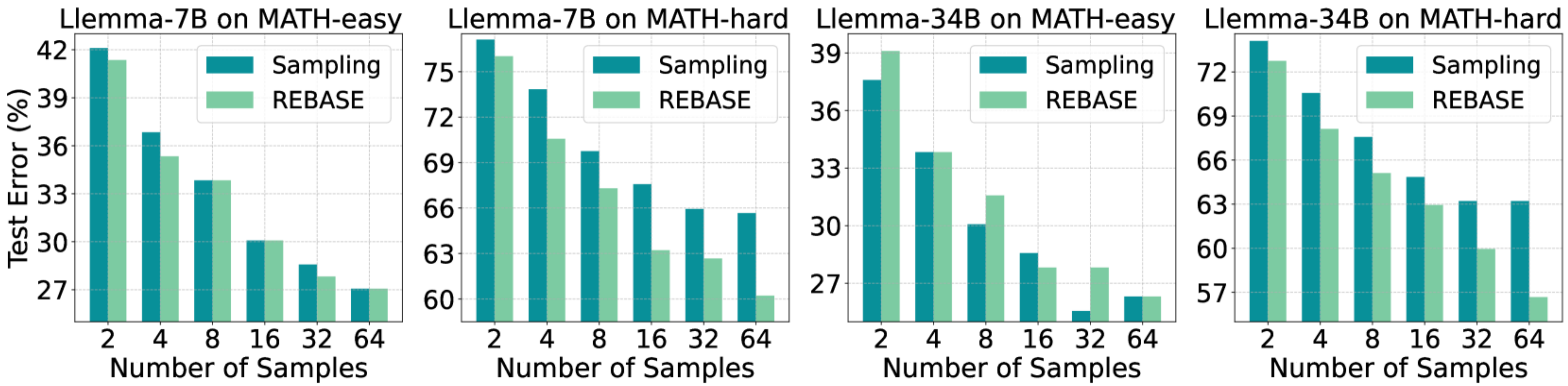
Evaluation metric:

- Test error (lower is better)
- Inference FLOPs per question – total compute used to generate a final answer


Inference scaling (Weighted Majority)
Test error on MATH
Inference FLOPs per question ($\times10^{12}$)
Sampling (7B)
Sampling (34B)
MCTS (7B)
MCTS (34B)
REBASE (7B)
REBASE (34B)
75
70
65
60
55
50
4
16
64
256
1024


Inference scaling (Weighted Majority)
Test error on GSM8K
Inference FLOPs per question ($\times10^{12}$)
Sampling (7B)
Sampling (34B)
REBASE (7B)
REBASE (34B)
32
28
24
20
16
12
2
4
8
16
32
64
128
256


Llemma-7B
Llemma-34B
Mistral-7B
Sampling W.M.
Sampling BoN
REBASE W.M.
REBASE BoN
Test error on MATH
Infer. FLOPs per question (×10^12)
80
75
70
65
60
55
50
45
4
8
16
32
64
128
256
512
1024


Llemma-7B on MATH-easy
Llemma-7B on MATH-hard
Llemma-34B on MATH-easy
Llemma-34B on MATH-hard
Test Error (%)
Number of Samples
2
4
8
16
32
64
Sampling
REBASE

Conclusion

1. Small models + smart inference > big models
- Better performance at the same compute budget

2. Sampling saturates
- More samples ≠ better results after a point

3. REBASE is compute-optimal
- Best accuracy-cost trade-off across all budgets