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

Page: https://testtimescaling.github.io/

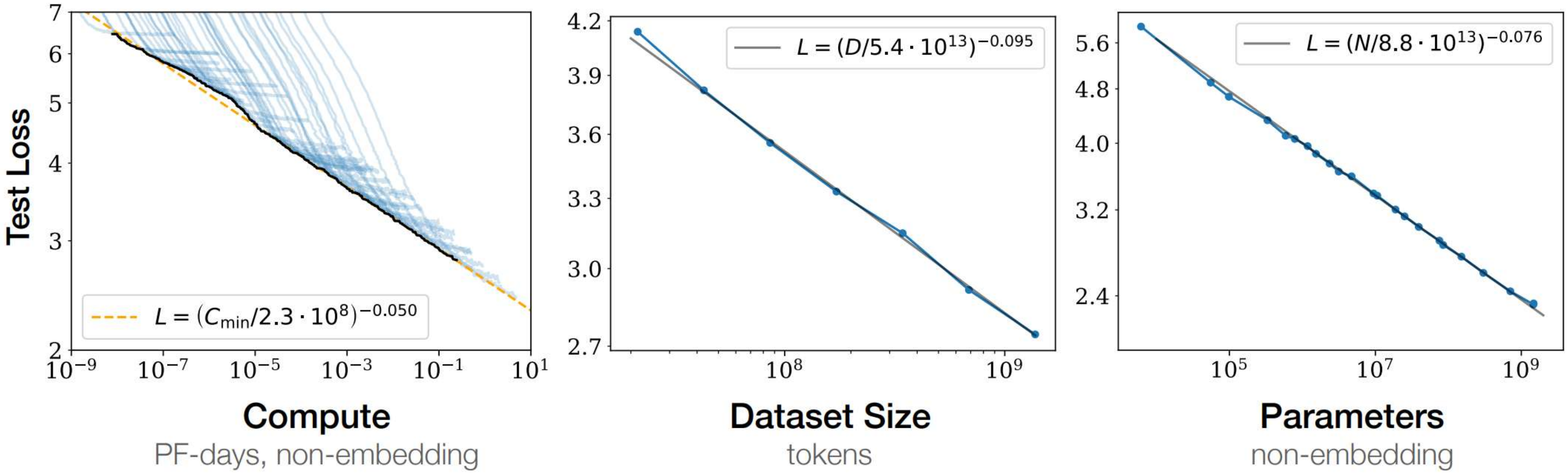
Presenter: Fuyuan Lyu & Qiyuan Zhang

Mila

McGill UNIVERSITY

CityU
香港城市大學
City University of Hong Kong

The Age of Pretraining (2020-2024)



[1] Scaling Laws for Neural Language Models

Pretraining scaling has gradually slowed

2 - The limit of text scaling data

is probability already here. As we observe that GPT-4 Turbo / Gemini Ultra / Claude 3 Opus / Llama 3 400B are all about the same range (MMLU around 85). To continue scale up text we need more data, the problem is whether it is possible to substantially increase the amount of tex data beyond Llama 3's 15T tokens.

There are the following directions, ranked by the potential scale of new data:

- CC is only part of the whole internet.
- We have not yet finished digging and crawling from CC.
- Relaxing the filtering and deduplication threshold.
- Use existing models to produce synthetic data.
- Scanning more books from libraries

We discuss them one by one.

[2] Fu's Blog (Apr 2024)

Pre-training as we know it will end

Compute is growing:

- Better hardware
- Better algorithms
- Larger clusters

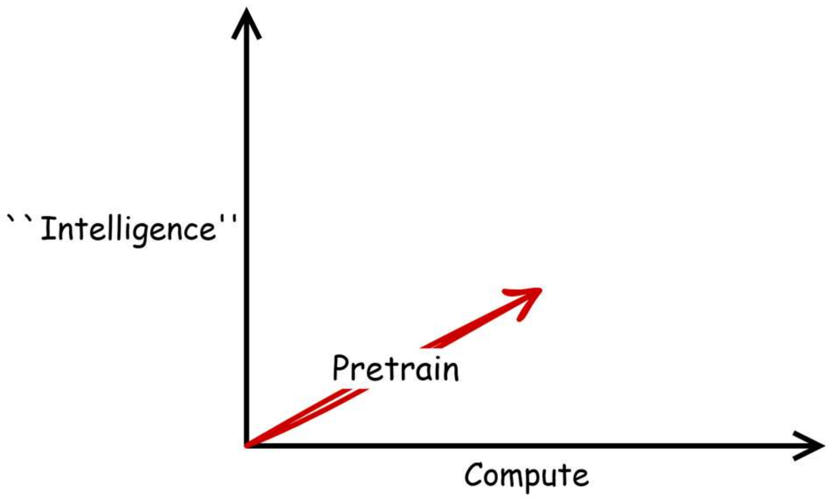
Data is not growing:

- We have but one internet
- **The fossil fuel of AI**

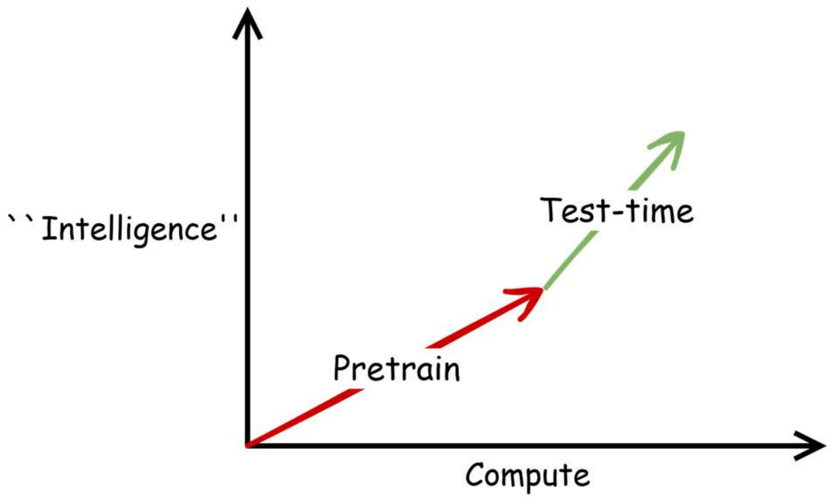
Ilya's Speech (Dec 2024)

[2] Llama 3 Opens the Second Chapter of the Game of Scale

Pretraining ⟶ Test-time



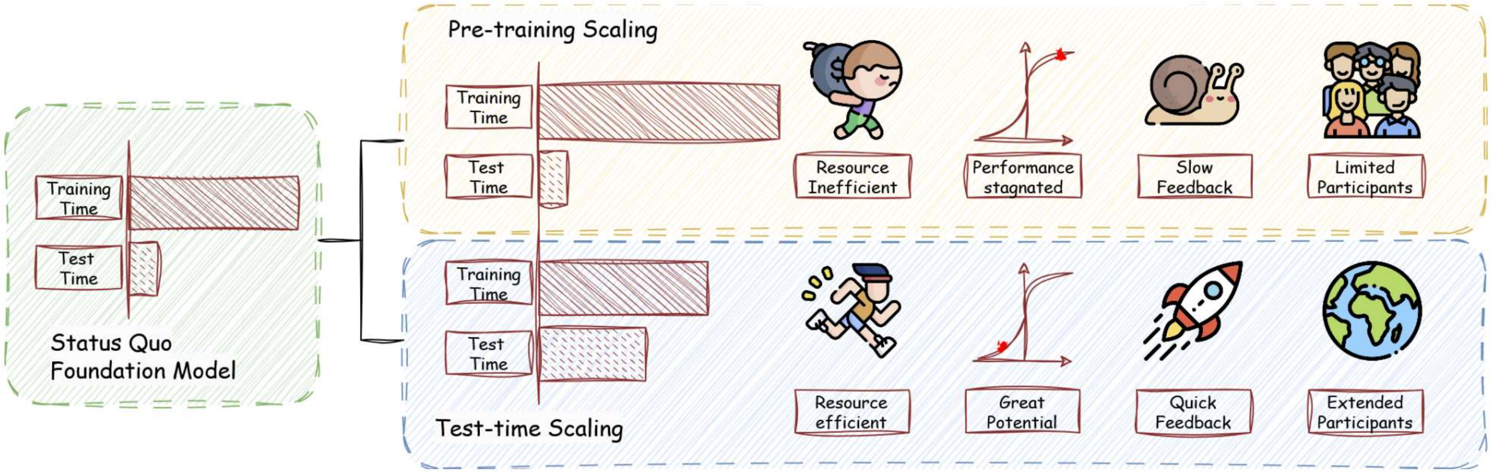
Increasing pretraining-time compute yields consistent performance improvements



Increasing test-time compute yields consistent performance improvements

Test-time Scaling

Test-time scaling (also referred to as *inference scaling*, *test-time compute*) progressively elicits the model's intelligence in the test-time phase.



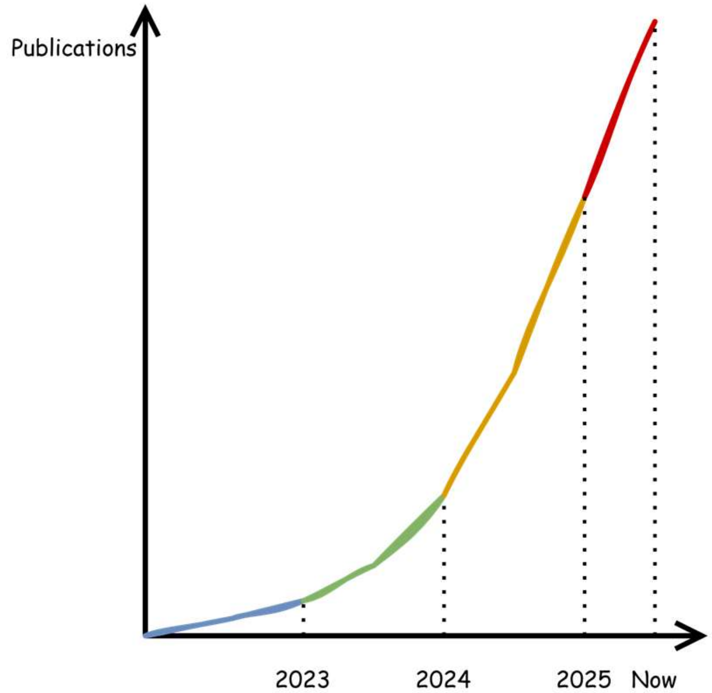
Comparison of Scaling Paradigms in Pre-training and Test-time Phases

Situation

- **Researchers:** When faced with overwhelming and complex literature, how should we cope?

- **Practitioners:** where should we focus our efforts for innovation?

- **Both:** How should we discuss them?



Survey Overview

- **A Unified, Multi-Dimensional Taxonomy**. **(For Researchers and Practitioners)**
  We propose a four-axis taxonomy—***what to scale***, ***how to scale***, ***where to scale***, **and** ***how well to scale***—that supports structured classification, comparison, and extensibility for TTS methods.

- **Systematical Literature Organization and Pragmatic Analysis. (For Practitioners)**
  Using our taxonomy, we survey the TTS landscape, analyze representative methods, and present guidelines for research application and deployment.

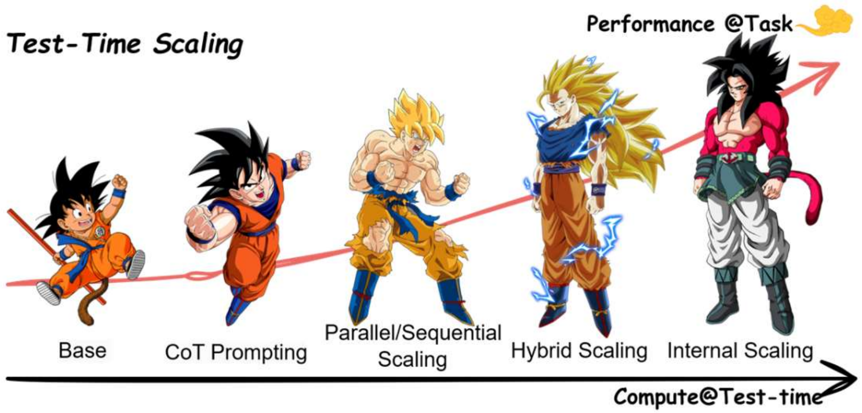
- **Challenges, Insights, and Forward Directions. (For Researchers)**
  Building on our organized perspective, we uncover critical challenges—ranging from advancing scaling to clarifying essence—and outline promising research directions that could shape future progress. Our unified framework facilitates the mapping of these open questions to concrete dimensions of TTS, enabling more targeted and impactful advancements.

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[*], Wenyue Hua[☆], Haolun Wu[★], Zhihan Guo[§], Yufei Wang[||], Niklas Muennighoff[★], Irwin King[§] Xue Liu[*] Chen Ma[+]

City University of Hong Kong[+], McGill University & MILA[*], Gaoling School of Artificial Intelligence, Renmin University of China[‡], Chinese University of Hong Kong[§] Salesforce AI Research[¶] Macquarie University[||] Stanford University[★] University of California, Santa Barbara[☆]

Paper Code arXiv



Dissecting TTS into 4 Key Orthogonal Dimensions

TTS

- **What to scale**

  refers to the specific form of TTS that is scaled to enhance an LLM's performance during inference.

- **How to scale**

  depicts how different TTSs are implemented.

- **Where to scale**

  covers the tasks and datasets where these TTSs are applied.

- **How well to scale**

  refers to both the metrics of evaluating TTS and the optimization directions.

Taxonomy Credit to Yufei Wang

What to Scale

- When applying TTS, researchers typically choose a specific empirical hypothesis.

  e.g., longer CoT, multiple sampling, advanced search ...

What to Scale

- **Parallel Scaling**

  Generating multiple outputs in parallel.

- **Sequential Scaling**

  Updating intermediate states iteratively.

- **Hybrid Scaling**

  Balancing exploration and exploitation.

- **Internal Scaling**

  Determining autonomously how much computation to allocate.

Chapter Credit to Qiyuan Zhang

Parallel Scaling

Normal: LLMs generate a single response per query.

Parallel scaling

- generates multiple outputs in parallel and then aggregates them into a final answer.
- e.g. Best-of-N, Majority-Voting

Sequential Scaling

Normal: LLMs generate a single response per query.

Parallel scaling

- generates multiple outputs in parallel and then aggregates them into a final answer.
- e.g. Best-of-N, Majority-Voting

Sequential scaling

- involves explicitly directing later computations based on intermediate steps.
- e.g. Self-Refine, CoT

Hybrid Scaling

Normal: LLMs generate a single response per query.

Parallel scaling

- generates multiple outputs in parallel and then aggregates them into a final answer.
- e.g. Best-of-N, Majority-Voting

Sequential scaling

- involves explicitly directing later computations based on intermediate steps.
- e.g. Self-Refine, CoT

Hybrid scaling

- exploits the complementary benefits of parallel and sequential scaling.
- e.g. ToT, FoT, AoT, MCTS

Internal Scaling

Normal: LLMs generate a single response per query.

Parallel scaling

- generates multiple outputs in parallel and then aggregates them into a final answer.
- e.g. Best-of-N, Majority-Voting

Sequential scaling

- involves explicitly directing later computations based on intermediate steps.
- e.g. Self-Refine, CoT

Hybrid scaling

- exploits the complementary benefits of parallel and sequential scaling.
- e.g. ToT, FoT, AoT, MCTS

Internal scaling

- Internalize the scaling process into a model and autonomously determine how much computation to allocate
- e.g. R1, o1/o3

What to Scale - Taxonomy

- **What to Scale** (§2)
  - **Parallel Scaling** (§2.1)
    - Self-Consistency (Brown et al., 2024; 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., 2024f; 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., 2024h), SoS (Gandhi et al., 2024), REBASE (Wu et al., 2024d), 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., 2025),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

- **How to Scale**
  - **Inference Strategies**
    - Stimulation
    - Verification
    - Search
    - Aggregation
  - **Tuning Strategies**
    - Supervised Finetuning
    - Reinforcement Learning

Inference-based Approaches

Inference-based approaches dynamically adjust computation during deployment.

This paradigm includes 4 essential components:

(i) **Stimulation** encourages the model to generate longer or multiple candidate outputs;
(ii) **Verification** filters or scores outputs based on correctness or other criteria;
(iii) **Search** systematically explores the sample space;
(iv) **Aggregation** consolidates multiple outputs into the final output.

Chapter Credit to Fuyuan Lyu

Inference-based Approaches: Stimulation

Stimulation encourages the model to allocate more computation to thinking.

Prompt Strategy

This behavior requires the backbone LLM's ability to follow instructions.

Decode Strategy

This approach modifies the decoding process to encourage LLM to generate longer, more detailed samples adaptively.

Latent Strategy

It encourages deeper or recurrent thinking within the hidden representations themselves through continuous internal states.

Self-Repetition Strategy

It generates multiple samples instead of individual ones.

Mixture-of-Model Strategy

It requires gathering the "wisdom of the crowd".

Summary of Certain Stimulation Techniques

Category	Approach	Approach Description
Prompt	CoT (Wei et al., 2022)	Contains a series of intermediate reasoning steps in prompts
	Step-by-step (Lightman et al., 2023)	Stimulate step-by-step thinking via prompt
	QuaSAR (Ranaldi et al., 2025)	Decompose CoT into Quasi-Symbolic Language
	CoD (Xu et al., 2025b)	Generate concrete representations and distill into concise equation
	Hint-infer (Li et al., 2025b)	Inserting artificially designed hints in the prompt
	Think (Li et al., 2025b)	Prompt LLM with "Think before response"
	Think About World (Jin et al., 2024)	Prompt LLM with "Think About the World" to enforce larger inference
Decode	Filler-token (Pfau et al., 2024)	uses arbitrary, irrelevant filler tokens before answering
	Budget-forcing (Muennighoff et al., 2025)	suppress the generation of the end-of-thinking token
	AFT (Li et al., 2025f)	iteratively aggregating proposals and aggregate for future proposals
	Predictive-Decoding (Ma et al., 2025a)	re-weight decoding distribution given evaluation of foresight
	Adaptive Injection (Jin et al., 2025)	Injecting a predefined injection phrase under certain condition
Latent	Coconut (Hao et al., 2024)	Perform chain-of-thought in hidden space without explicit token generation
	CoDI (Shen et al., 2025c)	Compress chain-of-thought into continuous vectors via self-distillation
	Looped (Recurrent) Transformers (Saunshi et al., 2025)	Unroll model depth at inference by repeatedly refining hidden states
	Heima (Shen et al., 2025b)	Encode each reasoning step into a single latent token to reduce output length
	LTV (Kong et al., 2025)	Introduce a latent thought variable to guide text generation
Self-Repetition	Self-Repetition (Wang et al., 2023)	prompt LM in parallel
	Self-Refine (Madaan et al., 2023)	Naively prompt LM to iteratively refine answer
	DeCRIM (Ferraz et al., 2024)	Self-correlation for multi-constrained instruction following
Mixture-of-Model	MoA (Wang et al., 2025a)	Prompt different models in parallel and iteratively improve
	RR-MP (He et al., 2025)	Propose Reactive and Reflection agents to collaborate
	BRAIN (Chen et al., 2024g)	Propose frontal & parietal lobe model to inspire brain
	Collab (Chakraborty et al., 2025)	Propose decoding strategies to leverage multiple off-the-shelf aligned LLM policies

Inference-based Approaches: Verification

The ***verification*** process plays an important role in the test-time scaling

- directly selects the output sample among various ones (Parallel Scaling);
- guides the stimulation process and determines when to stop (Sequential Scaling);
- serves as the criteria in the search process (Hybrid Scaling);
- determines what sample to aggregate and how to aggregate them, *e.g.*, weights.

Outcome Verification.

plays a crucial role in ensuring the correctness and consistency of generated outputs.

Process Verification.

verifies the sample outcomes and the process of obtaining such an outcome.

Summary of Certain Verification Techniques

Category	Approach	Approach Description
Outcome	Naive ORM (Cobbe et al., 2021)	Naively process to train solution-level and token-level verifiers on labeled-dataset
	OVM (Yu et al., 2024b)	Train a value model under outcome supervision for guided decoding
	Heuristic (DeepSeek-AI, 2025)	Heuristic check for domain-specific problems
	Functional (Lee et al., 2025)	Functional scoring for task-specific problems
	Bandit (Sui et al., 2025)	Train a bandit algorithm to learn how to verify
	Generative Verifier (Zhang et al., 2025d)	Exploit the generative ability of LLM-based verifiers via reformulating the verification
	Self-Reflection Feedback (Li et al., 2025g)	formulate the feedback utilization as an optimization problem and solve during test-time
	Discriminator (Chen et al., 2024h)	SFT a domain-specific LM as a discriminator
	Unit Test (Saad-Falcon et al., 2024)	Verify each sample as unit tests
	XoT (Liu et al., 2023b)	Passive verification from external tools and Activate verification via re-thinking
	WoT (Zhang et al., 2024c)	Multi-Perspective Verification on three aspects: Assertion, Process, and Result
	Multi-Agent Verifiers (Lifshitz et al., 2025)	Multi-Perspective Verification without explicit semantic meanings
Process	Naive PRM (Lightman et al., 2023)	SFT an LM as a PRM on each reasoning step over mathematical tasks
	State Verifier (Yao et al., 2023b)	SFT an LM as a state verifier and evaluate states either independently or jointly
	Deductive PRM (Ling et al., 2023)	Deductively verify a few statements in the process
	Self-Evaluation (Xie et al., 2023)	Prompting the same LM to evaluate the current step given previous ones
	PoT (Chen et al., 2023a)	delegate computation steps to an external language interpreter
	Tool (Li et al., 2025b)	Relies on external toolbox for verification
	V-STaR (Hosseini et al., 2024)	Verifier trained on both accurate and inaccurate self-generated data

Inference-based Approaches: Search

Employing search algorithms during inference provides a structured way to explore the solution space, significantly enhancing performance in complex tasks.

- **Beam Search and Variants**

  Beam search-based methods enhance traditional beam search by incorporating additional dimensions such as stochasticity, self-evaluation, and diversity.

- **Graph-Structured Search**

  They extend search strategies beyond simple tree structures, modeling outputs explicitly as graphs to exploit relational and complex structural reasoning.

- **Tree-Structured Search**

  These approaches leverage classical tree search algorithms to organize potential outputs into structured trees, explicitly exploring reasoning or planning steps.

  Naive Tree Search Methods (e.g., DFS, BFS) **Monte-Carlo Tree Search (MCTS)**

- **Systematic and Optimized Search Approaches**

  These works provide systematic analyses, optimizations, and enhancements to traditional search techniques, e.g., reward-balanced search.

Inference-based Approaches: Aggregation

Aggregation techniques consolidate multiple solutions into a final decision to enhance the reliability and robustness of model predictions at test time.

Selection

selects the best-performed sample among all candidates, where the selection criteria may vary across different approaches.

Fusion

fuses multiple samples into one through tricks like weighting or generation.

Category	Approach	External Verifier	Approach Description	Also Utilized in
Selection	Majority Voting (Wang et al., 2023)	✗	Select the most common sample	(Chen et al., 2024d)
	Best-of-N (Irvine et al., 2023)	✓	Select the highest scored sample	(Song et al., 2024)
	Few-shot BoN (Munkhbat et al., 2025)	✓	BoN with few-shot conditioning	
	Agentic (Parmar et al., 2025)	✗	agent considering both current and previous status	
Fusion	Weighted BoN (Li et al., 2023a)	✓	Weight each sample by its score	(Brown et al., 2024)
	Synthesize (Jiang et al., 2023)	✗	Fuse the selected samples via GenAI	(Wang et al., 2025a; Li et al., 2025c)
	Ensemble Fusion (Saad-Falcon et al., 2024)	✗	Conduct ensemble before fusion	

Tuning-based Approaches

To activate a model's ability to devote cost at test time, directly **tuning its parameters** is an effective strategy.

1) Supervised Finetuning (SFT)

Training an LLM via next-token prediction.

Key Factor: Data (synthetic or distilled long CoTs)

2) Reinforcement Learning (RL)

By leveraging feedback from a reward model on inference tasks, the policy model is automatically updated.

Tuning-based Approaches: SFT

Imitation

generate long CoT demonstrations using test-time “planner” algorithms and then fine-tune the model to imitate those demonstrations.

Distillation

aim to transfer the capabilities of a stronger model (or ensemble of models) into a target model via supervised learning.

Warmup

refer to an initial SFT phase applied to an LLM after its unsupervised pretraining but before other post-training steps like RL.

Chapter Credit to Haolun Wu

Tuning-based Approaches: RL

Reward Model-Free Approaches

These methods do not rely on explicitly learned reward models but instead use intrinsic or implicit signals to guide model optimization.

Representative works: rule-based reward, preference optimization, Value Function, Dynamic Sampling…

Open-Source Training Frameworks

SimpleRL, DeepScaler, SimpleRL-Zoo, X-R1, TinyZero, Open-Reasoner-Zero. OpenR, OpenRLHF, OpenR1, Logic-RL, AReaL.

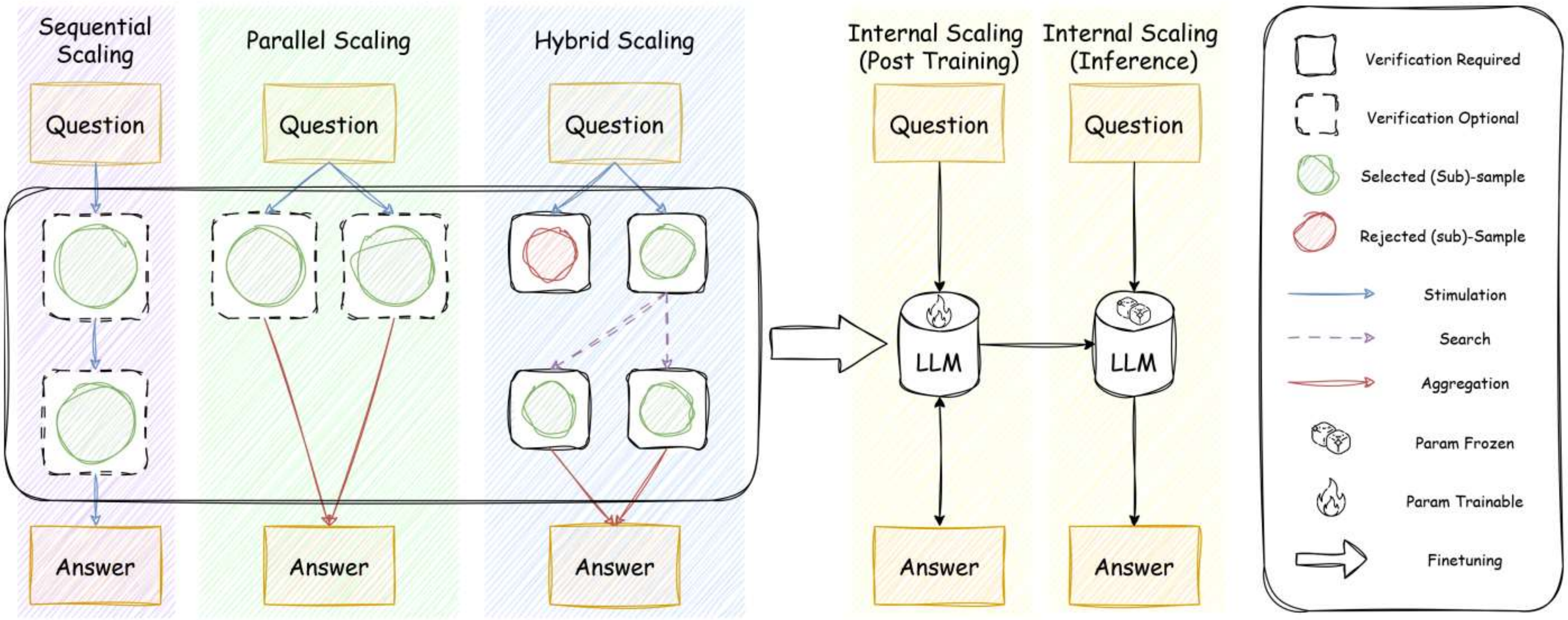
Reward Model-Based Approaches

These methods explicitly utilize trained reward models, typically guided by human preferences or learned value models.

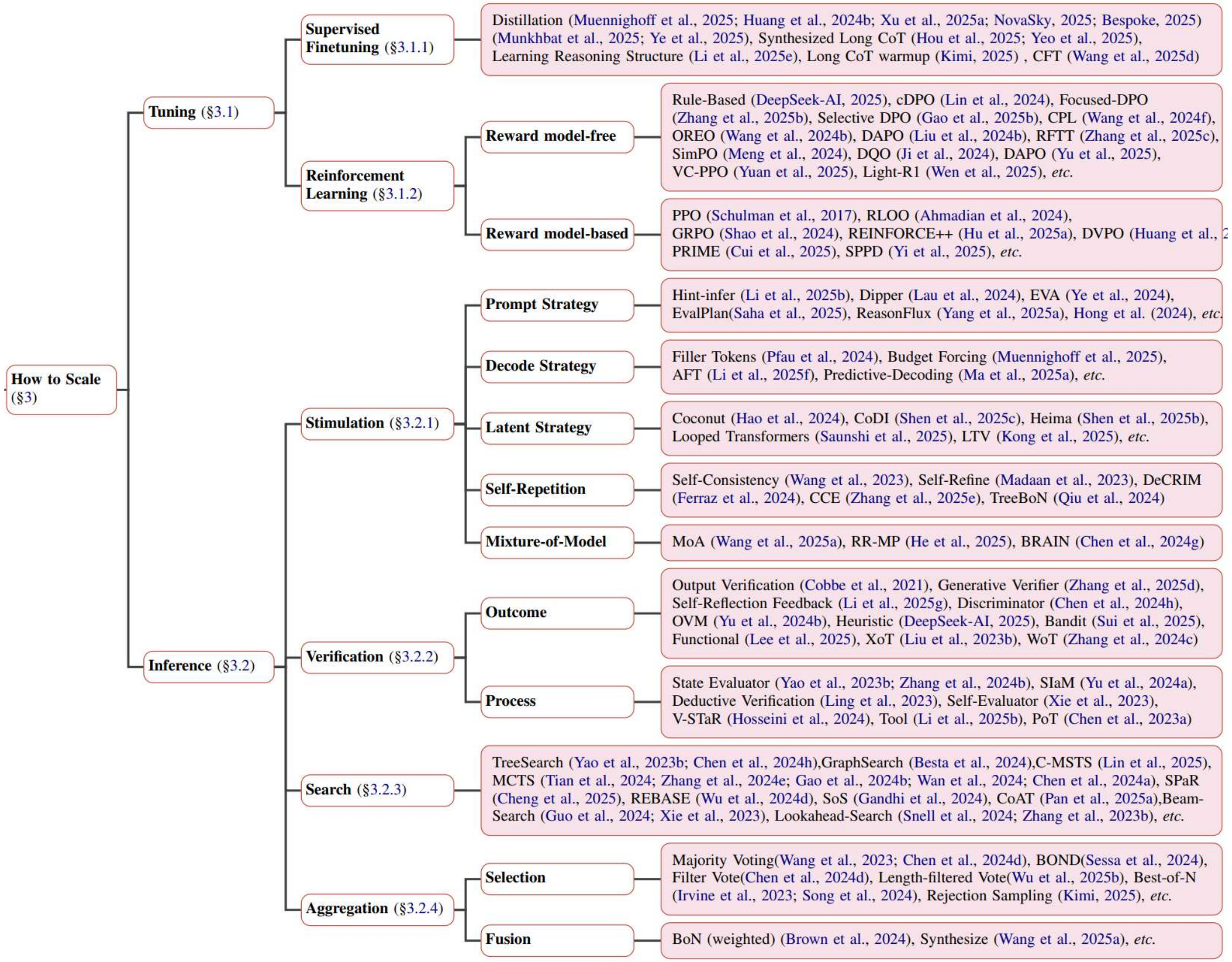
Representative works: Human-Preference Optimized Reward Models, Process-Based Reward Model, Enhanced Reward Models…

Chapter Credit to Zexu Sun

A Visual Map and Comparison:
From What to Scale to How to Scale.



How to Scale - Taxonomy



Where to Scale - Taxonomy

Chapter Credit to Weixu Zhang, Wenyue Hua & Zhihan Guo

Summary of Benchmarks

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
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	

How Well to Scale

How Well to Scale

- Performance

  assess the correctness of generated solutions.

- Controllability

  evaluate whether test-time methods can consistently adhere to pre-defined resource constraints (compute budgets or output length targets).

- Scalability

  measure how effectively test-time scaling methods can leverage increased compute to improve performance.

- Efficiency

  assess the computational and resource cost.

Chapter Credit to Lei Wang

How Well to Scale - Performance

Pass@1 evaluates the correctness of a model's first output attempt, which is frequently used in tasks such as mathematical reasoning and coding benchmarks.

Pass@K extends Pass@1 by measuring whether at least one of the model's k sampled outputs is correct, which is widely adopted in program synthesis and formal theorem-proving tasks.

Cons@k (Consensus@K) measures the majority vote correctness from k independently sampled outputs.

Pairwise Win Rate is based on comparing against baselines using human or LLM-based judges.

Task-Specific Metrics For instance, Codeforces Percentile and Elo Rating.

How Well to Scale - Controllability

Control Metric

measures the fraction of test-time compute values that stay within given upper and lower bounds.

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

where $\mathcal{A}$ is the set of observed compute values such as thinking tokens, and $\mathbb{I}(\cdot)$ is the indicator function.

Length Deviation Metric

Mean Deviation from Target Length quantifies the average relative difference between the generated output length and the target length

$$\text{Mean Deviation} = \mathbb{E}_{x \sim D} \left[\frac{|n_{\text{generated}} - n_{\text{gold}}|}{n_{\text{gold}}} \right]$$

Root Mean Squared Error (RMSE) of Length Deviation captures the variance in length control

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\frac{n_{\text{generated},i} - n_{\text{gold},i}}{n_{\text{gold},i}} \right)^2}.$$

k–ϵ Controllability

quantifies whether a model can be guided to produce a target output within a bounded prompt length and allowable deviation.

How Well to Scale - Scalability

Scalability metrics measure how effectively test-time scaling methods can leverage increased compute (e.g., token budgets, samples, inference steps) to improve performance.

Scaling Metric

captures the average slope of performance gains as compute increases

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

Scaling Curves (Accuracy vs. Compute)

visualizes how metrics such as accuracy, pass rate, or EM improve as token budgets, iteration depth, or the number of samples increase.

How Well to Scale - Efficiency

Token Cost

measures the total number of tokens generated during inference, including intermediate reasoning steps and final outputs.

FLOPs-based Efficiency Analysis

Underthinking score

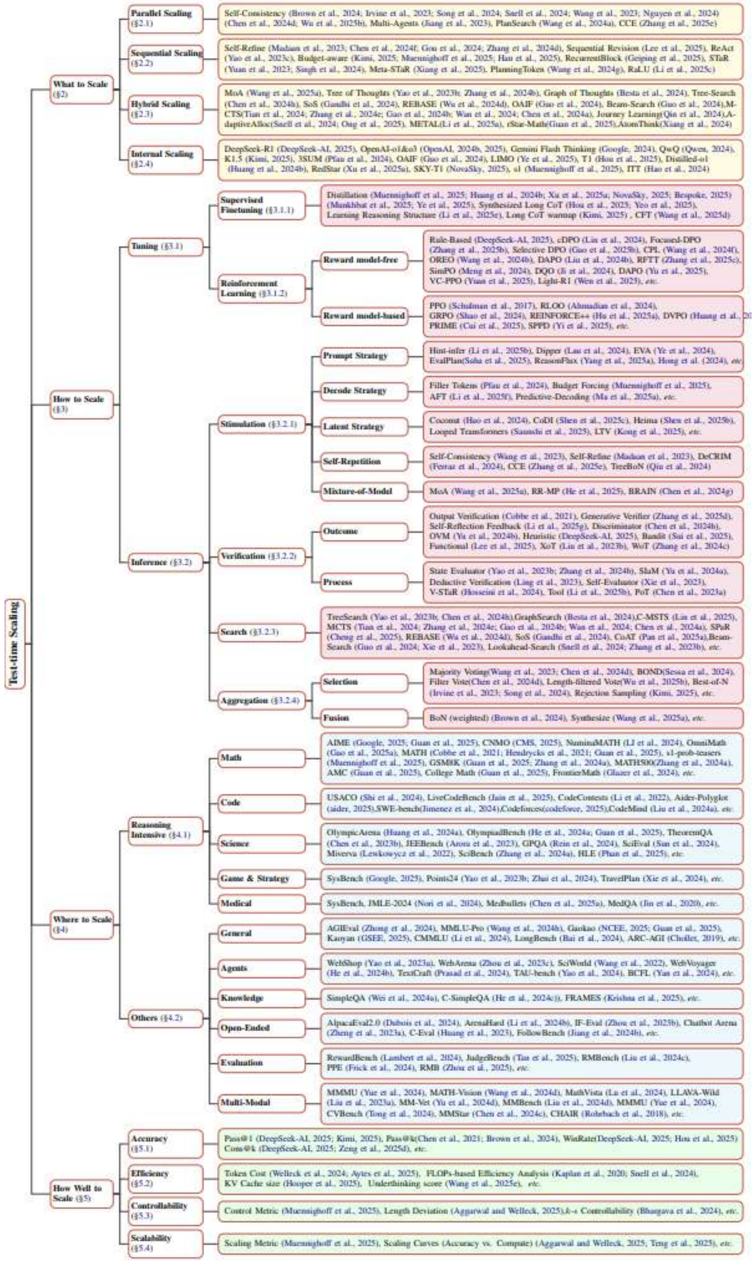
measures how early in the response the first correct thought appears, relative to the total length of the response, in cases where the final answer is incorrect.

Formally, the underthinking score ξ_{UT} is defined as:

$$\xi_{\mathrm{UT}} = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{\hat{T}_i}{T_i} \right)$$

- N: Number of incorrect responses in the test set.
- T_i: Total number of tokens in the i-th incorrect response.
- $\hat{T}_i$: Number of tokens from the beginning of the response up to and including the first correct thought.

Taxonomy

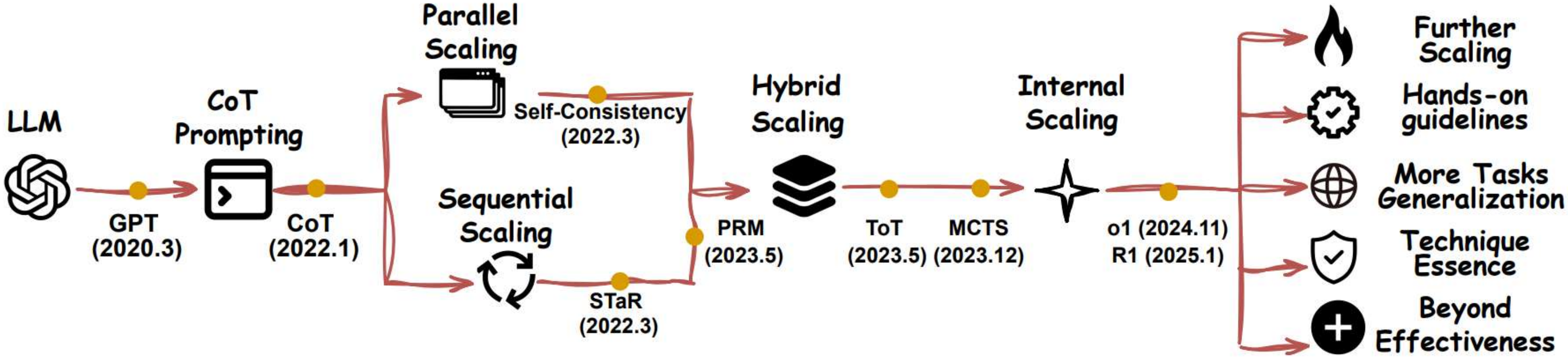

Test-time Scaling
What to Scale (§2)
Parallel Scaling (§2.1)
Sequential Scaling (§2.2)
Hybrid Scaling (§2.3)
Internal Scaling (§2.4)
How to Scale (§3)
Tuning (§3.1)
Supervised Finetuning (§3.1.1)
Reinforcement Learning (§3.1.2)
Reward model-free
Reward model-based
Inference (§3.2)
Stimulation (§3.2.1)
Prompt Strategy
Decode Strategy
Latent Strategy
Self-Repetition
Mixture-of-Model
Verification (§3.2.2)
Outcome
Process
Search (§3.2.3)
Aggregation (§3.2.4)
Selection
Fusion
Where to Scale (§4)
Reasoning Intensive (§4.1)
Math
Code
Science
Game & Strategy
Medical
Others (§4.2)
General
Agents
Knowledge
Open-Ended
Evaluation
Multi-Modal
How Well to Scale (§5)
Accuracy (§5.1)
Efficiency (§5.2)
Controllability (§5.3)
Scalability (§5.4)

Existing Literature Organization using Our Taxonomy

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	HOW: RL	HOW: STIMULATION	HOW: SEARCH	HOW: VERIFICATION	HOW: AGGREGATION	WHERE	HOW WELL
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
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

Organization and Trends in Test-time scaling



- **These techniques are complementary**
- **There is no one simple scaling solution that works for all problems**
- **The boundary between inference-based and tuning-based approaches is blurring.**

Chapter Credit to Qiyuan Zhang

A Hand-on Guideline for Test-time Scaling

Hands-on Guidelines: Common Problems

Q: What kind of task does *TTS* help?
A: Almost any task! While traditional reasoning tasks—such as Olympiad-level mathematics, complex coding, and game-based challenges—have been shown to significantly improve with *TTS* , community observations suggest that *TTS* can also enhance performance in open-ended tasks, such as comment generation or evaluation. However, due to the long-form nature of outputs and the lack of centralized, objective benchmarks, these tasks are inherently more difficult to evaluate quantitatively, making it harder to draw conclusive claims. Beyond that, more realistic, complex, and long-horizon scenarios, like medical reasoning and law, have also shown promising gains through *TTS* strategies.

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.

Q: Are these pipelines mutually exclusive? How should I design a frontier-level *TTS* strategy?
A: These pipelines are by no means mutually exclusive—they can be seamlessly integrated. For instance, R1 inherently necessitates SFT through rejection sampling as a preliminary warmup step. When employing RL, practitioners should continue leveraging synthesized CoT methods and introduce additional structured inference strategies to tackle increasingly complex scenarios effectively.

Q: What are some representative or widely-used *TTS* methods that can serve as baselines?
A: Parallel–Self-Consistency, Best-of-N; Sequential–STaR, Self-Refine, PRM; Hybrid–MCTS, ToT; Internal–Distilled-R1, R1.

Q: Is there an optimal go-to solution so far?
A: No free lunch. Optimal computing is often dependent on the hardness and openness of the question.

Q: How should we evaluate the performance of a *TTS* method? In addition to standard accuracy, what other aspects should we pay attention to?
A: The evaluation is largely task-aware, but metrics like accuracy remain the most critical indicators. In addition, efficiency (the trade-off between performance and cost) is another key concern in practical settings. As *TTS* becomes a more general-purpose strategy, researchers have also begun evaluating a range of secondary attributes, including robustness, safety, bias, and interpretability, to better understand the broader impacts of *TTS* .

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.

Open Hands-on Guidelines / 开放手册

We understand that an individual's strength is limited. I hope our survey provides an open and practical platform where everyone can share their experiences in TTS practice within the community we are building. These experiences are invaluable and will benefit everyone. If the guidelines you provide are valuable, we will include them in the PDF version of the paper.

Submit your Guidelines

- Looking forward to anyone giving problems and summaries of what you've encountered in your practice #3 · by testtimescaling

Open Comments supported by Our Page

Comments & Discussion

0 reactions

2 comments – *powered by giscus*

Oldest | Newest

testtimescaling yesterday Owner

Let's see what happens!

↑ 1

0 replies

Write a reply

testtimescaling yesterday Owner

So, is RL the optimal strategy?

↑ 1

0 replies

Write a reply

Challenges and Opportunities

- Advancing Scalability is the Frontier.
- Clarifying the Essence of Techniques in Scaling is the Foundation.
- Optimizing Scaling is the Key
- Generalization across Domains is the Mainstream

More Scaling is the Frontier

Parallel Scaling

Challenges:
- Diminishing returns at saturation
- Naive best-of-N lacks diversity

Opportunities:
1. Smart Coverage Expansion: Diverse reasoning paths
2. Verifier-Augmented Sampling: Real-time filtering

Sequential Scaling

Challenges:
- Coherence degradation
- Error accumulation

Opportunities:
1. Structured Self-Refinement: Targeted step repair
2. Verification-Enhanced Iteration: Real-time consistency checks

Hybrid Scaling

Generalized Hybrid Scaling Architectures

Multi-Agent & Interactive Scaling

Internal Scaling

Effective Compute Allocation

Stability and Consistency

Interpretability and Controllability

Clarifying the Essence

Their roles and interactions within the pipeline demand a deeper investigation.

1. Theoretical Gaps in Scaling Techniques

   How do core techniques (SFT, RL, reward modeling) contribute to test-time scaling? how should SFT and RL be optimally combined?

2. Re-evaluating Reward Modeling

   whether PRMs actually improve multi-step inference? Does the classic reward model incorporate noise and unnecessary complexity?

3. Mathematical Properties of Test-Time Scaling

   How does performance scale with increased inference steps? Is there an optimal stopping criterion? Are there fundamental constraints on how much test-time scaling can improve reasoning performance?

Clarifying the Essence

4. Chain-of-Thought Reasoning Priorities

   Which aspects of the chain-of-thought are most crucial for effective test-time scaling?

5. Adaptive Test-Time Scaling

   How can we make a model automatically adjust its inference process based on the problem at hand? As empirical observations on certain property models show blindly scaling over test-time may lead to over-thinking.

6. Thoughtology

   How do the reasoning patterns in its language help improve reasoning effectiveness by treating a finetuned reasoning model as an agent?

Optimizing TTS via Metrics

Goal: Holistic evaluation & efficient deployment

Directions to Optimize:

- Accuracy
- Efficiency
- Robustness
- Interpretability
- Bias/Safety

Trend: Multi-metric, task-sensitive optimization strategies emerging

Domain Generalization

Emerging Domains:

- Medicine
- Finance
- Law
- Math Proof & Physics
- AI Evaluation
- open-domain QA
- other high-stakes or knowledge-intensive areas

Challenges:

1. **Balancing Cost and Accuracy**:

   Unlike general NLP tasks, specialized domains often require strict computational efficiency and reliability;

2. **Ensuring Domain-Specific Interpretability**:

   In fields like medicine and law, outputs must be transparent and justifiable;

3. **Integrating External Knowledge & Real-World Constraints**:

   Many domains require retrieval-augmented generation, real-time data analysis, or interactive query refinement;

4. Future research must identify generalizable test-time scaling strategies that are **robust** across diverse reasoning tasks.

Conclusion

In the post-training era, TTS has been one of the dominant directions.

a. Structured Taxonomy

b. Practical Utility

c. Open Community

Our Amazing and Inspiring Team

Qiyuan Zhang
@CityU

Fuyuan Lyu
@McGill & MILA

Zexun Sun
@Gaoling RUC

Lei Wang
@Saleforce AI

Weixu Zhang
@McGill & MILA

Wenyue Hua
@UCSB

Haolun Wu
@Stanford &
McGill & MILA

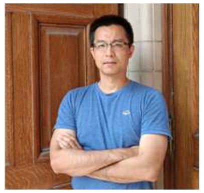
Zhihan Guo
@CUHK

Yufei Wang
@Macquire

Niklas Muennighoff
@Stanford

Irwin King
@CUHK

Steve Liu
@MBZUAI &
McGill & MILA



Chen Ma
@CityU

Thanks for the invitation and your watching!

Page: https://testtimescaling.github.io/

Email: qzhang732-c@my.cityu.edu.hk
fuyuan.lyu@mail.mcgill.ca

paper