

Test-Time Scaling in Reasoning LLMs: Inference Regimes, Evaluation, and Reproducibility

Mohsen Hariri Weicong Chen Nahal Shahini Vikash Singh Kai Ye
 Amirhossein Samandar Debargha Ganguly Sreehari Sankar Yanyan Zhang
 Shouren Wang Jerry Peng Biyao Zhang Michael Hinczewski Vipin Chaudhary
{mohsen.hariri, weicong, nxs814, vikash, kxy406, axs2935, debargha, sxs2284, yxz3106, sxw992, jxp1146, bxz297, mxh605, vipin}@case.edu
Case Western Reserve University

Abstract

Large language models can solve substantially harder reasoning problems when they are allowed to spend more compute at inference time. The term *test-time scaling*, however, now refers to a diverse set of inference algorithms, including methods that extend deliberation along a single trajectory, sample completed candidates and aggregate them through voting or verification, and search over unfinished partial states. These algorithms differ in their statistical structure, compute accounting, and failure modes. Treating these procedures as interchangeable under a single scalar “budget,” or reporting accuracy without the inference protocol that produced it, makes results difficult to compare across studies. We develop a systematic account of test-time scaling along three axes. First, we formalize test-time scaling as budgeted inference over the implicit prefix tree of an autoregressive model and distinguish three structural regimes: single-trajectory sequential scaling, leaf-level scaling with terminal reduction, and prefix-level scaling. Second, we treat the evaluated object as the entire inference system and develop evaluation principles that separate end-to-end system performance from candidate-bank diagnostics. We introduce an evaluation profile whose coordinates and simple functionals recover or bound common repeated-sampling metrics, and prescribe protocol-matched reporting of compute and uncertainty. Third, we specify reproducibility requirements for inference protocols, distinguishing exact replay from distributional reproducibility and identifying the artifacts needed to support each. We also organize the open-weight reasoning ecosystem by model-side and interface mechanisms, apply these principles to broad-knowledge, symbolic-reasoning, and competition-mathematics benchmarks, and assemble over 2 billion full reasoning traces for release with progressively richer verifier and token-level signals.¹

 <https://mohsenhariri.github.io/scorio/tts>
 <https://huggingface.co/datasets/harimo/scorio>

1 Introduction

Large language models (LLMs) often become substantially more capable on difficult reasoning tasks when they are allowed to externalize intermediate reasoning steps and consume additional computation at inference time. Early work on chain-of-thought prompting and self-consistency showed that arithmetic, symbolic, and commonsense reasoning can improve markedly when models generate multi-step solutions and aggregate multiple samples (Wei et al., 2022; Wang et al., 2022). More recent systems treat deliberate reasoning as an explicit design target rather than a prompting artifact: DeepSeek-R1 uses reinforcement learning to elicit long-form reasoning (Guo et al., 2025); Phi-4-reasoning combines curated supervised traces with outcome-based reinforcement learning (Abdin et al., 2025); Qwen3 exposes explicit “thinking” modes and reasoning-budget controls (Team, 2025); and the gpt-oss release provides open-weight reasoning models trained with

¹For the API list, see Appendix F; for the release data, see Appendix G.

large-scale distillation and reinforcement learning (OpenAI, 2025). As a result, reasoning performance is now shaped not only by pretraining and post-training, but also by how computation is allocated at test time.

Despite this progress, the phrase *test-time scaling* is used to describe several materially different inference algorithms. Some methods allocate extra compute along a single evolving trajectory, for instance, by forcing longer deliberation or adaptively intervening on one response (Muennighoff et al., 2025). Others sample multiple completed candidates and reduce them using voting, reranking, verifier-based selection, or minimum-Bayes-risk decoding (Wang et al., 2022; Cobbe et al., 2021; Freitag et al., 2022). Still others search over unfinished partial states, as in Tree-of-Thoughts, RAP, and value-guided decoding (Yao et al., 2023; Hao et al., 2023; Yu et al., 2024). These regimes differ in their statistical structure, induced proposal distributions over candidates, compute accounting, and failure modes. Treating them as interchangeable under a single scalar “budget” obscures the actual procedure being evaluated and makes cross-paper comparisons difficult to interpret (Welleck et al., 2024; Snell et al., 2025).

Evaluation is correspondingly more subtle than benchmarking a base model in isolation. In practice, measured performance depends on the full inference protocol: the prompt template, decoder, search controller, reducer, verifier or judge, stopping rule, and numerical settings. Best-of- N and verifier-guided reranking can improve accuracy, but imperfect proxy scores may themselves be overoptimized as the candidate set grows (Gao et al., 2023; Huang et al., 2025; Khalaf et al., 2025). LLM judges provide flexible evaluation of open-ended outputs, yet prior work documents biases such as position and verbosity effects, and recent work shows that judge quality is highly task- and procedure-dependent in test-time scaling settings (Zheng et al., 2023; Zhou et al., 2025). Even when prompts and methods are fixed, benchmark estimates can vary due to stochastic decoding and implementation details, motivating interval estimates and explicit reproducibility controls (Miller, 2024; Blackwell et al., 2024; Ye et al., 2024; Liu et al., 2025b). A rigorous study of reasoning under test-time scaling thus requires a precise algorithmic vocabulary, evaluation protocols matched to deployed inference algorithms, and model comparisons that are explicit about both model-side training choices and inference-time controls.

This need for explicit system-level evaluation is amplified by the pace and heterogeneity of the open-weight reasoning ecosystem. Recent releases span reinforcement-learning-first reasoning models (Guo et al., 2025), data-centric and verification-focused SFT pipelines (Ye et al., 2025; Guha et al., 2025), open reproductions built around released trace corpora and training code (Hugging Face, 2025a; Open-R1 Team, 2025b), and hybrid systems that combine reasoning and non-reasoning behavior either within a single checkpoint or in parameter space (Team, 2025; Yang et al., 2025). This diversity is scientifically valuable, but it also makes comparisons prone to confounding: two checkpoints may differ simultaneously in their supervision signals, reasoning styles, compute budgets, and aggregation protocols. The field, therefore, benefits from studies that separate these axes and evaluate reasoning as a property of the full inference system rather than of model weights alone. Figure 1 summarizes this shift as a selective chronology: inference-time algorithms, model-side training, and interface controls have co-evolved toward systems whose performance depends on the checkpoint, interface, controller or reducer, evaluator, stopping rule, and budget.

This paper addresses that gap. We formalize test-time scaling as a family of budgeted inference algorithms over the implicit prefix tree of an autoregressive model. Our treatment of sampled inference separates the evidence computed for each trace from decisions on a completed pool; causal stopping is modeled separately because it changes generation cost. Shared-bank comparisons can then isolate aggregation, whereas end-to-end evaluation measures the complete inference system. Candidate-bank quality is reported through structured diagnostics, including a unified family of repeated-sampling metrics for the trade-off between discovery and stability. We instantiate these ideas on a broad set of contemporary open-weight reasoning models and assemble over 2 million reasoning traces to support trace analysis, verifier training, and reproducible study across science, reasoning, and competition mathematics.

Our main contributions are as follows:

- **Formalizing test-time scaling.** We present a unified formalization of test-time scaling as budgeted inference over the implicit prefix tree and distinguish three structural regimes: single-trajectory sequential scaling, leaf-level scaling, and prefix-level scaling.



Figure 1: A selective chronicle of reasoning systems, from elicited chains to budgeted inference. Test-time inference expanded from single-trajectory elicitation and leaf-level sampling with terminal reduction to include prefix-level search and budget-aware control (Section 2.1), while model and interface mechanisms evolved through verified distillation, reward optimization, explicit reasoning controls, and parameter-space composition (Appendices D and E). The lower bands summarize the evaluation and reproducibility requirements developed in Section 3 and the evaluated object used throughout this paper: checkpoint, prompt, decoder, controller or reducer, verifier or judge, budget, and stopping rule, with utility, candidate-bank profile, cost, and uncertainty reported together. Green dots mark families represented in the experimental roster; dashed boxes denote contextual milestones. Era labels indicate shifts in emphasis rather than mutually exclusive periods.

- **Establishing evaluation principles for test-time scaling.** We evaluate the entire inference system: base model, prompting, decoder, inference-time evidence, search or aggregation rule, stopping controller, evaluator, and budget. The resulting principles separate shared-bank aggregation comparisons from end-to-end performance and post-hoc selection from causal stopping. They also require compute reporting on the relevant axes and uncertainty estimates for the actual protocol.
- **Providing a model-centric view of the open reasoning ecosystem.** We organize and compare representative open-weight reasoning models by model-side and interface mechanisms—including verified SFT and distillation, reinforcement learning or preference optimization, inference-time control, and parameter-space fusion—so that comparisons are grounded in the mechanisms that most directly affect reasoning quality, controllability, and cost.
- **Releasing a large-scale reasoning-trace resource.** We release a corpus of over 2 million reasoning-trace sequences spanning science, reasoning, and competition mathematics, intended as a reusable resource for analyzing reasoning behavior, training or calibrating evaluators, and improving reproducibility in reasoning-model research.

2 Formalizing Test-Time Scaling

Test-time scaling is most naturally viewed as a family of *budgeted inference algorithms* built atop a fixed autoregressive model. Recent work studies repeated sampling with terminal reduction, adaptive single-

trajectory deliberation, and search over partial reasoning states as distinct ways of spending additional inference-time computation with a fixed base model (Welleck et al., 2024; Snell et al., 2025; Muennighoff et al., 2025; Wu et al., 2024). The common object behind these methods is the implicit prefix tree induced by generation: extra compute may be spent along a single trajectory, across multiple completed leaves, or over internal prefixes before completion. We formalize that view and use it to distinguish three structural regimes.

2.1 Problem setup and three-regime taxonomy

Let $x \in \mathcal{X}$ be an input prompt and let $p_\theta(\cdot | x)$ be an autoregressive language model. The finite prefixes reachable from x define an implicit rooted tree $\mathcal{T}(x)$; we write $z \preceq y$ when prefix z lies on the path to completed generation y , and $\mathcal{L}(x)$ for the set of terminal leaves (Welleck et al., 2024). A local generation policy π (for example, greedy decoding, temperature sampling, or nucleus sampling) induces a proposal distribution $q_\pi(y | x)$ over completed leaves, which may differ from the raw model distribution because decoding can truncate, renormalize, or otherwise transform token probabilities.

A completed generation y may contain both intermediate reasoning and a task answer. We write $\text{Parse}(y) = (r, a)$, where r is the reasoning trace and a is the extracted answer. When only part of the output can be checked programmatically, we further write $\Psi(y) = (d, s)$, where d is the deterministically verifiable component and s is the remainder. Depending on the task, d may be the final answer string, an executable program, an action sequence, or any other artifact that admits task-specific checking.

Our object of interest is a family of budgeted inference algorithms $\{\mathcal{A}_B\}_{B \in \mathcal{B}}$. Algorithm $\mathcal{A}_B$ may adaptively interleave primitive operations o_t such as token generation, prefix expansion, verifier calls, judge calls, or terminal reduction, subject to a cumulative cost constraint $\sum_t c(o_t) \leq B$. It returns either a completed leaf $\hat{y}_B(x) \in \mathcal{L}(x)$ or an extracted answer $\hat{a}_B(x)$. Let $U_x(y)$ denote the task utility of a completed leaf. In reasoning benchmarks $U_x(y)$ is often exact correctness, but the same abstraction also covers graded utilities such as execution score or preference reward.

In practice, U_x is usually observed only through evaluation signals. A programmatic verifier $V_P(x, d) \in \mathcal{V}$ may return a Boolean decision, a scalar score, a diagnostic object, or a canonicalized answer representation. A learned evaluator $J_\phi(x, \omega)$ scores either a completed leaf $\omega = y$ or a partial state $\omega = z$. Unlike programmatic checks, learned evaluators are proxies whose usefulness depends on their supervision and on the protocol used at test time (Cobbe et al., 2021; Lightman et al., 2023; Zheng et al., 2023; Zhou et al., 2025).

For a task distribution $\mathcal{P}$, the associated test-time scaling curve is

$$G(B) = \mathbb{E}_{x \sim \mathcal{P}, \xi} [U_x(\hat{y}_B(x; \xi))],$$

where ξ denotes the algorithm’s internal randomness, or the corresponding answer-level utility when the method outputs $\hat{a}_B(x)$. The term *scaling* refers to how performance varies with budget; it does not imply that performance must be monotone for every method.

We use the following operational taxonomy, illustrated in Figure 2.

Single-trajectory sequential scaling. At every step, there is at most one unfinished active prefix. Additional computation changes how that prefix is extended, revised, or terminated, but never branches into a competing frontier.

Leaf-level scaling. Additional compute produces a bank of completed candidates. Any interaction between candidate trajectories is deferred until after the candidates have completed, via a terminal reducer.

Prefix-level scaling. Additional compute is allocated based on scores for unfinished prefixes. Expansion, pruning, rollout allocation, or termination decisions depend on partial states before completion.

These regimes are not mutually exclusive at the system level: many practical methods are hybrids, most commonly prefix search followed by a leaf-level reducer.

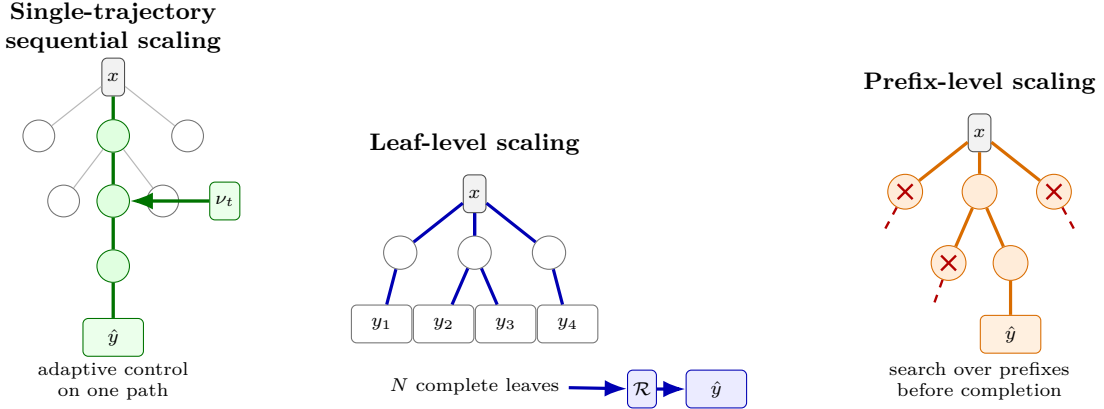


Figure 2: Three regimes of test-time scaling over the implicit prefix tree. Left: single-trajectory sequential scaling allocates compute to one evolving response without maintaining a competing frontier. Middle: leaf-level scaling allocates compute to multiple completed root-to-leaf trajectories and then reduces them to a single output. Right: prefix-level scaling spends compute on internal prefixes, expanding promising continuations and pruning others before committing to a completed leaf.

2.2 Single-trajectory sequential scaling

A single-trajectory method maintains one active state z_t and remaining budget b_t . At step t , a controller chooses a meta-action

$$\nu_t \sim \pi_{\text{seq}}(\cdot \mid x, z_t, b_t),$$

then extends only the active path,

$$z_{t+1} \sim \text{Extend}(z_t; \nu_t, p_\theta), \quad b_{t+1} = b_t - c(\nu_t).$$

Typical actions include continuing generation, suppressing EOS, appending a control string or critique, changing decoding hyperparameters, or stopping. Budget-forcing methods such as S1 are direct instances of this template, and more general inference-time meta-generation procedures also fit it when they keep a single evolving response rather than a branching frontier (Muennighoff et al., 2025; Welleck et al., 2024).

The structural advantage of this regime is low orchestration overhead: all extra compute is devoted to a single evolving candidate. Its structural limitation is that, because no competing unfinished prefix exists, recovery from an early misconception requires explicit self-revision along the same path rather than searching for alternatives.

2.3 Leaf-level scaling: sampling and reduction

In leaf-level scaling, additional compute produces complete leaves and a terminal reducer chooses the output. In the canonical case,

$$\mathcal{Y}_N(x) = \{Y_i\}_{i=1}^N, \quad Y_i \stackrel{\text{i.i.d.}}{\sim} q_\pi(\cdot \mid x),$$

where q_π is a common proposal distribution induced by the base model and decoding policy. Appendix B extends this template to weighted and heterogeneous-proposal banks. For each candidate, let m_i contain metadata available at inference time, and define

$$(r_i, a_i) = \text{Parse}(Y_i), \quad (d_i, s_i) = \Psi(Y_i), \quad v_i = V_P(x, d_i), \quad j_i = J_\phi(x, Y_i).$$

Given the proposal, the terminal stage is specified by a reducer

$$\mathcal{R}_N : (x, \{(Y_i, m_i, v_i, j_i)\}_{i=1}^N) \mapsto \hat{y} \text{ or } \hat{a}.$$

The fixed-bank leaf template separates generation from reduction. Under this template, one unfinished candidate cannot depend on scores assigned to other unfinished candidates; cross-candidate interaction begins inside $\mathcal{R}_N$ after all N leaves are complete. Adaptive stopping can be added without changing this separation. A controller may use completed leaves to decide whether to launch another independent rollout, or it may truncate a rollout using that rollout’s prefix-local evidence, but it does not reallocate compute among competing unfinished prefixes (Section 3.3). Such controllers change the realized sample count or trace length and are not fixed-bank reducers.

2.3.1 Canonical reduction rules

Verifier-constrained selection. Let $\mathcal{I}_{\text{pass}} = \{i : V_P(x, d_i) \in \text{PASS}\}$. A verifier-constrained reducer selects

$$i^* \in \arg \max_{i \in \mathcal{I}_{\text{pass}}} T(x, Y_i, v_i, j_i),$$

where T is a tie-breaking score over the passing candidates (for example, the learned score j_i or sequence likelihood), with a fallback rule if $\mathcal{I}_{\text{pass}}$ is empty. This is the generate-test-select pattern used in program-synthesis and verifier-based reasoning systems (Li et al., 2022; Cobbe et al., 2021).

Answer marginalization and self-consistency. When answers or verifier outputs can be canonicalized, define the empirical answer distribution

$$\hat{q}_N(a \mid x) = \frac{1}{N} \sum_{i=1}^N \mathbf{1}[a_i = a].$$

Self-consistency selects

$$\hat{a} \in \arg \max_a \hat{q}_N(a \mid x).$$

Appendix B shows that this is exactly empirical MBR with agreement utility $u(a, a') = \mathbf{1}[a = a']$ (Wang et al., 2022; Kumar & Byrne, 2004; Bertsch et al., 2023).

Learned reranking (Best-of- N). When programmatic verification is unavailable or incomplete, a standard reducer chooses

$$i^* \in \arg \max_{i \in [N]} j_i.$$

This covers learned verifiers, reward models, and LLM judges used as inference-time rerankers (Cobbe et al., 2021; Snell et al., 2025; Huang et al., 2025).

Empirical expected-utility selection. More generally, with a projection ψ and utility u , one may select

$$i^* \in \arg \max_{i \in [N]} \frac{1}{N} \sum_{k=1}^N u(\psi(Y_i), \psi(Y_k)).$$

This MBR template unifies majority vote, semantic-consensus rules, and neural-metric reranking under a common expected-utility objective (Kumar & Byrne, 2004; Freitag et al., 2022; Bertsch et al., 2023).

These rules cover the main structural forms of leaf-level reduction. Additional variants—including weighted aggregation, semantic kernels, pairwise aggregation, regularized selection, and the separation of answer selection from rationale presentation—are detailed in Appendix B.

2.3.2 Failure modes and budget allocation

The main advantage of leaf-level scaling is that increasing compute increases candidate coverage without risking early pruning. Its failure modes come from the reducer. An incomplete verifier can admit false positives as N grows, while misspecified learned scores can be overoptimized by hard argmax selection, producing the non-monotone Best-of- N behavior documented in recent work on reward-model overoptimization and

inference-time reward hacking (Gao et al., 2023; Huang et al., 2025; Khalaf et al., 2025). Judge-based reducers also inherit protocol sensitivities such as position bias and verbosity bias, so judge design and prompting are part of the method under evaluation (Zheng et al., 2023; Zhou et al., 2025). Finally, leaf count and evaluator count are distinct budgets: MBR-style reducers may require $O(N^2)$ utility calls, motivating approximate implementations such as confidence-based pruning and faster MBR approximations (Cheng & Vlachos, 2023; Jinnai & Ariu, 2024).

A fixed sample count is therefore only one leaf-level design choice. One may instead choose $N = N(x)$ by solving

$$N(x) \in \arg \max_{n: C_{\text{gen}}(n) + C_{\text{eval}}(n) \leq B} \mathbb{E}[U_x(\hat{y}_n(x))],$$

where C_{gen} denotes generation cost and C_{eval} covers inference-time scoring, control, and reduction. Recent inference-scaling studies explicitly optimize this budget allocation and show that difficulty-adaptive compute can dominate fixed- N baselines (Wu et al., 2024; Snell et al., 2025).

2.4 Prefix-level scaling: search over partial states

Prefix-level scaling allocates compute before trajectories are complete. To cover both token-level search and step-level methods such as Tree-of-Thoughts, RAP, and AlphaZero-like decoding, let z denote a search state representing either a literal token prefix or a macro-prefix composed of one or more reasoning steps, and write $\text{Succ}(z)$ for its allowed expansions (Yao et al., 2023; Hao et al., 2023; Wan et al., 2024b).

2.4.1 Frontier, terminal bank, and budget

A prefix-level method maintains an active frontier $\mathcal{F}_t$, a bank of completed leaves $\mathcal{Y}_t$, and remaining budget b_t . A generic iteration is

$$z_t \in \text{Select}(\mathcal{F}_t) \rightarrow \mathcal{C}_t \subseteq \text{Succ}(z_t) \rightarrow \text{score } \mathcal{C}_t \rightarrow (\mathcal{F}_{t+1}, \mathcal{Y}_{t+1}, b_{t+1}).$$

The budget may count generated tokens, node expansions, rollout calls, verifier or judge calls, or wall-clock cost. Unlike in leaf-level scaling, future compute allocation depends explicitly on evaluations of unfinished states.

2.4.2 Prefix evaluation as continuation-value estimation

When a search commits to a single prefix and returns one completion drawn from it, the Bayes-optimal prefix score is the continuation value

$$Q_\rho^*(z) = \mathbb{E}_{Y \sim q_\rho(\cdot | x, z)} [U_x(Y)],$$

where $q_\rho(\cdot | x, z)$ denotes the distribution over completed leaves obtained by continuing from prefix z with rollout policy ρ . If search instead accumulates a bank that is later passed to a reducer $\mathcal{R}$, the exact marginal value of expanding z becomes history-dependent because it depends on how future leaves from z will interact with the leaves already collected. Most practical methods ignore this coupling and use one-leaf surrogates such as correctness probability, process reward, or rollout return.

2.4.3 Canonical prefix evaluators

Likelihood-based scoring. A canonical heuristic is the cumulative, optionally length-normalized, log-probability under the local generation policy,

$$S_{\text{LL}}(z) = \sum_{j=1}^{|z|} \log \pi(z_j | x, z_{<j}).$$

With $\pi = p_\theta$ this is the standard beam-search score. It ranks prefixes by proposal probability rather than by downstream utility.

Outcome-supervised value models. Outcome-supervised value models approximate the probability that a prefix reaches a correct completion and use that estimate for pruning, either directly or in combination with likelihood:

$$\widehat{Q}(z) \approx \Pr_{Y \sim q_\rho(\cdot | x, z)}(U_x(Y) = 1), \quad S(z) = \widehat{Q}(z) \text{ or } S(z) = \alpha S_{\text{LL}}(z) + \beta \widehat{Q}(z).$$

Such value-guided decoding is explored for mathematical reasoning in OVM-style systems (Yu et al., 2024).

Process reward models (PRM). When supervision is available on intermediate reasoning steps, a process reward model can score a partial path directly. If $z = (s_1, \dots, s_t)$ is a sequence of steps, one may use

$$S_{\text{PRM}}(z) = \sum_{k=1}^t R_\phi(x, s_{\leq k}) \quad \text{or} \quad S_{\text{PRM}}(z) = R_\phi(x, z),$$

where R_ϕ scores each step in the context of the problem and the preceding steps and is learned from step-level supervision (Lightman et al., 2023).

Rollout-based evaluation. When learned value estimates are unreliable, continuation value can be estimated by Monte Carlo completion:

$$\widehat{Q}_{\text{MC}}(z) = \frac{1}{M_{\text{roll}}} \sum_{m=1}^{M_{\text{roll}}} U_x(Y^{(m)}), \quad Y^{(m)} \sim q_\rho(\cdot | x, z).$$

MCTS-style reasoning methods can be interpreted as structured rollout allocation together with backup of these estimates (Hao et al., 2023; Wan et al., 2024b).

Sound partial checks. A partial check $H(x, z) \in \{0, 1\}$ is *sound for rejection* if

$$H(x, z) = 0 \implies U_x(y) = 0 \quad \text{for all } y \succeq z.$$

Only sound prefix tests justify safe pruning. Learned value or reward models generally do not provide this guarantee.

2.4.4 Score-induced search procedures

Beam search. Beam search keeps a frontier of size K and updates it by

$$\mathcal{F}_{t+1} = \text{TopK}\left(\bigcup_{z \in \mathcal{F}_t} \text{Succ}(z); S(\cdot)\right).$$

Standard beam search uses $S = S_{\text{LL}}$, while verifier-guided and value-guided variants replace or augment likelihood with learned prefix scores (Yu et al., 2024; Lightman et al., 2023).

Best-first search. Best-first search repeatedly expands the highest-scoring active prefix,

$$z^* \in \arg \max_{z \in \mathcal{F}_t} S(z), \quad \mathcal{F}_{t+1} = (\mathcal{F}_t \setminus \{z^*\}) \cup \text{Succ}(z^*).$$

In classical search, admissible heuristics can yield optimality guarantees; in LLM reasoning, learned prefix scores are generally heuristic rather than admissible (Koh et al., 2025).

MCTS-style search. Monte Carlo tree search alternates selection, expansion, evaluation, and backup. A standard UCT rule selects

$$a_t \in \arg \max_a \left(Q(z, a) + c_{\text{uct}} \sqrt{\frac{\log n(z)}{n(z, a)}} \right),$$

where $n(z)$ and $n(z, a)$ are visit counts and unvisited actions are selected first. Policy-prior variants such as PUCT replace the pure UCT bonus with one modulated by an action prior. In LLM reasoning, RAP, AlphaZero-like search, and related methods instantiate this template with rollout-based or learned value estimates (Kocsis & Szepesvári, 2006; Browne et al., 2012; Hao et al., 2023; Wan et al., 2024b). Tree-of-Thoughts can be viewed as a related search controller operating over thought-level rather than token-level expansions (Yao et al., 2023).

Additional controller variants, including learned search policies and reward-guided objectives, are discussed in Appendix C.

2.4.5 Scaling behavior and comparison to leaf-level methods

Prefix-level search can be markedly more computationally efficient when prefix scores concentrate budget on high-value branches. Its additional difficulty is that pruning decisions driven by prefix scores change the support of reachable completions. If every ancestor of a correct leaf is pruned, no later computation can recover that leaf. Consequently, the relevant statistical property is often not global calibration but the within-instance ordering induced over competing frontier states.

Under a fixed budget B , prefix search also induces its own effective proposal distribution over completed leaves,

$$q_{\text{search}, B}(y \mid x) = \Pr(\hat{y}_B(x) = y),$$

which depends jointly on the scorer, controller, and stopping rule; when the search returns a bank rather than a single leaf, the corresponding object is the per-leaf inclusion probability of the returned bank. Candidate banks returned by prefix search are therefore not i.i.d. draws from $q_{\pi}(\cdot \mid x)$, so direct comparisons to leaf-level Best-of- N must normalize both generation cost and evaluator cost.

Finally, prefix search creates large branching structures with shared prefixes. Efficient tree-structured inference kernels exploit this sharing through key-value cache reuse, which can materially change the practical compute frontier; DEFT is a representative example (Yao et al., 2025). Many real systems are therefore hybrid: prefix search first constructs a search-induced bank of completed leaves, and a leaf-level reducer then selects the final output. The repeated-sampling metrics used later to summarize such candidate banks are introduced in Section 3.4.

3 Evaluating Test-Time Scaling

Under test-time scaling, reported performance depends on the *full inference protocol* defined over the prefix tree in Section 2.1. The protocol determines both the *candidate distribution* and the final output rule. Its specification includes budget allocation, decoding or search, inference-time evidence, aggregation and stopping rules, and any verifier or judge calls. Evaluation should therefore state the benchmark structure, the induced *proposal distribution*, the aggregation and stopping protocol, the reported *estimand*, and the uncertainty procedure.

3.1 Benchmark design as evaluation structure

Benchmark design determines which capability is measured and whether additional inference-time compute can meaningfully change outcomes. Under test-time scaling, benchmark suites should therefore be organized by *evaluation structure*, not only by task domain or historical popularity.

A primary distinction is between *verifiable* and *open-ended* tasks. In verifiable settings such as mathematics and code generation, correctness can be checked exactly or by execution, so repeated sampling and terminal reduction target a comparatively sharp correctness criterion (Chen et al., 2021; Li et al., 2022). In open-ended settings, evaluation relies on rubrics, pairwise preferences, or judge models; consequently, the evaluation protocol itself—including prompts, comparison order, side randomization, and aggregation—is part of the method being evaluated (Liang et al., 2023; Zheng et al., 2023; Chiang et al., 2024). In socially consequential tasks, the evaluation target extends beyond task utility to fairness, with data-level, training-time, and post-processing mitigation strategies evaluated using both predictive-performance and

group-fairness metrics (Kiashemshaki et al., 2025). These regimes interact differently with additional compute: verifier-based selection, agreement-based aggregation, and judge-based re-ranking need not improve at the same rate as budget increases (Cobbe et al., 2021; Lightman et al., 2023; Zheng et al., 2023).

Because open-weight reasoning checkpoints differ materially in verified supervision, preference objectives, and built-in reasoning-budget controls, Section D.2 summarizes these system-level levers separately.

The benchmark structure should also match the task’s interaction model. Single-turn exact-answer tasks probe discovery of a correct terminal leaf, whereas interactive agent benchmarks evaluate full trajectories, including recovery from intermediate errors (Yao et al., 2024; Welleck et al., 2024). For interactive chatbot tasks involving sensitive disclosures, ecological validity also requires modeling users’ contextual privacy decisions and the protective actions exposed by the interface (Nezhad et al., 2026). The same inference algorithm can therefore appear strong under “any correct sample” reporting yet weak under trajectory-level reliability metrics (Section 3.4).

Freshness and difficulty calibration matter especially under test-time scaling. Benchmark-label integrity is also essential, as vision-language-model-based sanitation of noisy and semantically inconsistent labels in manufacturing datasets can materially affect downstream classification accuracy (Mahjourian & Nguyen, 2026). Contamination-limited benchmarks and time-segmented datasets help distinguish genuine search or verification gains from memorization (White et al., 2025; Jain et al., 2025; Oren et al., 2024). Benchmarks should also avoid saturation: if tasks are too easy, additional compute has little effect; if they are uniformly difficult, scaling curves flatten because correct solutions are rarely reachable. Since many reasoning benchmarks are small, dataset size directly limits the statistical resolution of scaling comparisons (Snell et al., 2025; Miller, 2024; Ye et al., 2024) (see Section 3.5).

3.2 Decoding protocols and induced proposal distributions

Under test-time scaling, the object sampled by the inference system is not the raw model distribution p_θ alone, but the proposal distribution induced by a complete decoding protocol. Let Σ denote the token vocabulary. At a partial generation $h_t = (x, y_{<t})$, the model produces logits $\ell_t \in \mathbb{R}^{|\Sigma|}$ and token probabilities

$$p_t(i) = \frac{\exp \ell_t(i)}{\sum_{j \in \Sigma} \exp \ell_t(j)}, \quad i \in \Sigma.$$

A one-token decoder with configuration λ and finite state σ_t returns

$$(\tilde{q}_t, a_t^{\text{tok}}, \sigma_{t+1}) = D_\lambda(h_t, \ell_t, p_t, \chi_t, \sigma_t),$$

where χ_t denotes side information available to the decoder, such as hidden states, token embeddings, auxiliary scores, rollout estimates, or controller state. If $a_t^{\text{tok}} \neq \perp$, the next token is set deterministically to $y_t = a_t^{\text{tok}}$; otherwise $y_t \sim \text{Cat}(\tilde{q}_t)$.

A broad class of stochastic decoders admits an energy-gate form (Figure 3). For a nonnegative token measure $u_t : \Sigma \rightarrow \mathbb{R}_{\geq 0}$ and nonempty support $S_t \subseteq \Sigma$,

$$\tilde{q}_t(i) = \text{Norm}_{S_t}(u_t)_i = \frac{u_t(i) \mathbf{1}\{i \in S_t\}}{\sum_{j \in S_t} u_t(j)}. \quad (1)$$

The measure u_t determines relative weights, while the support S_t determines which tokens remain eligible. Ordinary temperature sampling uses $u_t(i) = \exp(\ell_t(i)/T)$ and $S_t = \Sigma$; ordinary truncation rules restrict S_t and renormalize the same measure; uniform or coverage-based samplers change the weights inside the retained set. Thus support equivalence and distribution equivalence are distinct: two decoders can retain the same candidate set while inducing different token probabilities.

Composing the local kernels until termination gives the leaf-level proposal distribution used in Section 2.3:

$$q_\lambda(y \mid x) = \prod_{t=1}^{\tau(y)} K_{\lambda,t}(y_t \mid x, y_{<t}, \sigma_t), \quad (2)$$

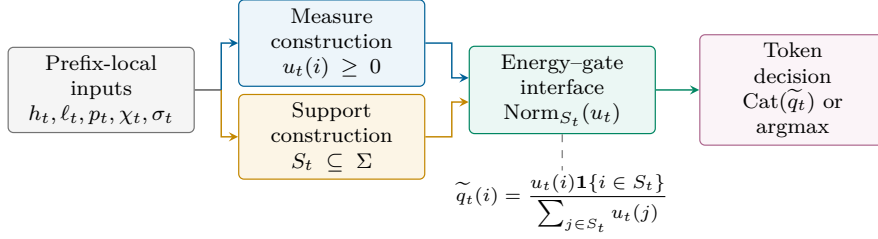


Figure 3: Elementwise view of a one-token sampling protocol. The sequence-level proposal distribution $q_\pi(\cdot | x)$ is induced by composing these local kernels until termination.

where $\tau(y)$ denotes the termination step of y (its length in tokens) and $K_{\lambda,t}$ is a point mass when the decoder takes a deterministic token action and otherwise equals $\tilde{q}_t$. When the protocol π fixes a single configuration λ , we write $q_\pi = q_\lambda$; if the protocol randomizes over configurations, q_π averages q_λ over that outer randomness; if a bank uses heterogeneous protocols, then candidate i is drawn from its own proposal $q_{\pi_i}(\cdot | x)$ rather than from a common i.i.d. proposal.

The reusable part of most truncation rules is the construction of S_t . Each gate below takes the live candidate set A and a reference token score r , typically p_t or a monotone transform of it; under sequential composition, the renormalized $r^{(j)}$ of Eq. (8) plays this role. Three gate families cover the main cases. A *level gate* keeps tokens above a data-dependent threshold,

$$G_{\text{lev}}(A; r, g, c) = \{i \in A : g(i; r, \ell_t, \chi_t) \geq c(r, \ell_t, \chi_t)\}, \quad (3)$$

capturing absolute-probability, mode-relative, entropy-scaled, and logit-band thresholds. A *head-budget gate* sorts the live set A in increasing order of a cost key κ (for probability-ranked truncation, $\kappa(i) = -r(i)$) and retains the first m_B tokens permitted by the budget parameter B (a fixed rank for top- k ; the smallest count whose retained mass reaches the target for cumulative-mass rules),

$$G_{\text{head}}(A; r, \kappa, B) = \{i_{(1)}, \dots, i_{(m_B)}\}, \quad \kappa(i_{(1)}) \leq \dots \leq \kappa(i_{(|A|)}), \quad (4)$$

capturing fixed-rank, cumulative-mass, curvature, entropy-budget, and typicality-based truncation. An *objective gate* chooses a support by solving, or approximating, a set objective,

$$G_{\text{obj}}(A; r, J) \in \arg \min_{\emptyset \neq A' \subseteq A} J(A'; r, \chi_t). \quad (5)$$

This separates the support decision from the weighting rule in Eq. (1).

A useful special case is the family of mode-preserving head gates. Fix a reference distribution r_t whose rank order is the model rank order, for example p_t or a positive-temperature rescaling, and write

$$r_t(i_1) \geq r_t(i_2) \geq \dots \geq r_t(i_{|\Sigma|}), \quad K_m = \{i_1, \dots, i_m\}.$$

If each gate g in a parallel composition $\mathcal{H}$ returns a model-ranked prefix K_{m_g} , then the composed support collapses to the strictest retained prefix:

$$\bigcap_{g \in \mathcal{H}} K_{m_g} = K_{m_\cap}, \quad m_\cap = \min_{g \in \mathcal{H}} m_g. \quad (6)$$

Consequently, any parallel combination of top- k (Fan et al., 2018), nucleus (Holtzman et al., 2020), probability-threshold (Hewitt et al., 2022), mode-relative, or logit-band head gates has the common form

$$\tilde{q}_t^{\mathcal{H}}(i) = \frac{u_t(i) \mathbf{1}\{i \in K_{m_\cap}\}}{\sum_{j=1}^{m_\cap} u_t(i_j)}. \quad (7)$$

The different named rules specify different ways of computing m_g ; once their head supports are evaluated in parallel, only the smallest retained prefix remains active.

This collapse does not apply to all decoders. Locally typical sampling can skip high-probability tokens because it orders tokens by the distance of their surprisal $-\log p_t(i)$ from the conditional entropy of p_t rather than by model probability (Meister et al., 2023); geometry- or objective-based crops can select non-prefix supports; and measure-changing rules can reorder candidates even when the support is fixed. Ordered composition also differs from parallel masking. For probability-only gates, a sequential protocol has the form

$$A_0 = \Sigma, \quad r^{(0)} = p_t, \quad A_j = G_j(A_{j-1}; r^{(j-1)}, \ell_t, \chi_t), \quad r^{(j)} = \text{Norm}_{A_j}(w^{(j)}), \quad (8)$$

where $w^{(j)}$ is the measure exposed by the j -th operator. Thus top- p followed by top- k need not match top- k followed by top- p , because cumulative masses, entropies, thresholds, and ranks may be recomputed after renormalization. Temperature before a mass gate can change the support, whereas temperature after a fixed support only changes within-support weights.

For evaluation, these distinctions determine the estimand. In leaf-level scaling with a fixed stochastic protocol, the usual candidate bank is

$$Y_1, \dots, Y_N \stackrel{\text{i.i.d.}}{\sim} q_\pi(\cdot \mid x).$$

A deterministic configuration sweep instead returns a reproducible set of design points, not Monte Carlo samples from a single proposal. Prefix-level search induces its own budget-dependent proposal $q_{\text{search}, B}(\cdot \mid x)$ through the scorer, controller, branching rule, rollout policy, and stopping criterion, and its returned leaves are generally dependent. Accordingly, the repeated-sampling diagnostics of Section 3.4 have an i.i.d. interpretation only when the bank is generated by independent draws from a declared fixed proposal; otherwise they are descriptive summaries of the returned bank.

A sampling report should therefore specify the token measure, support gates, reference score space, gate ordering, deterministic actions or stochastic draws, state updates, stopping and length rules, number of samples, seeds or randomized configuration policy, and any approximation or fallback behavior. These details are not implementation trivia: they define the proposal distribution over reasoning traces and therefore the test-time scaling system being evaluated.

3.3 Aggregation and stopping under sampled inference

A sampled candidate bank does not by itself define the submitted prediction. The inference protocol must say how evidence is extracted from each trace and how a fixed, completed bank is converted into an output. If either the number or length of rollouts is adaptive, the protocol must also specify when generation stops. This separation matters because stronger evidence or a different decision rule can improve accuracy without reducing generation cost.

Evidence and fixed-bank decisions. For completed candidate i , let m_i contain metadata available at inference time; the benchmark utility U_x is excluded. Examples include chosen-token log-probabilities, next-token distributions, process-reward scores, and verifier or judge outputs. An evidence map produces an evidence object

$$e_i = E_\eta(x, Y_i, m_i) \in \mathcal{E},$$

where $\mathcal{E}$ may be scalar or structured and E_η includes any reduction over tokens or reasoning steps required by the downstream rule. We use scalar $e_i \in \mathbb{R}$ for the score-based rules below. Sequence and mean log-likelihood (Wang et al., 2022), statistics of next-token distributions (Kang et al., 2025; Fu et al., 2026), reductions of per-step process scores (Lightman et al., 2023), and external verifier or reward-model scores (Cobbe et al., 2021) are distinct evidence maps even when passed to the same selector. Pairwise methods may instead construct

$$[\mathbf{W}_n]_{ij} = u(\psi(Y_i), \psi(Y_j)).$$

A fixed-bank decision rule then returns a leaf, an answer, or a set of outputs,

$$\hat{o}_n = \mathcal{G}_\delta(x, ((Y_i, a_i, e_i))_{i=1}^n, \mathbf{W}_n).$$

Together, E_η and $\mathcal{G}_\delta$ refine the generic leaf-level reducer $\mathcal{R}_N$ of Section 2.3 into evidence and decision stages. The evidence source and decision rule are separate choices. Scores from one source can be used for reranking,

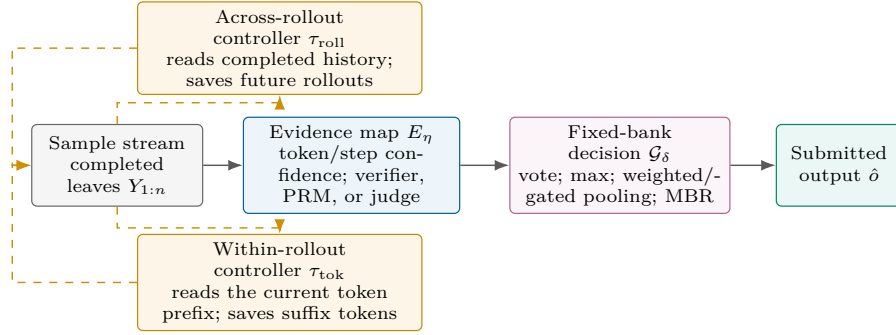


Figure 4: Aggregation and stopping under sampled inference. Evidence extraction and fixed-bank selection operate on completed traces, so they cannot save generation already spent. Causal controllers inspect only revealed history: across-rollout stopping avoids later traces, whereas token-prefix stopping can also truncate the current trace. The evidence map, controller, and final decision are all part of the evaluated protocol.

filtering, or vote weighting. Conversely, a voting rule may take model likelihoods as evidence or replace them with self-evaluations or an external verifier.

For answer-valued tasks, write $a_i = \perp$ when parsing or canonicalization declares a candidate invalid, and let $I_a = \{i : a_i = a\}$ for $a \neq \perp$. Plurality self-consistency² scores a group by $|I_a|$. Hard Best-of- N instead chooses the candidate with largest e_i , which amounts to scoring each answer group by $\max_{i \in I_a} e_i$ (Wang et al., 2022; Cobbe et al., 2021). Score-aware consensus pools evidence within a group using raw sums or means, softmax-normalized weights, rank weights, or calibrated log-odds (Taubenfeld et al., 2025; Kang et al., 2025; Kuang et al., 2026).

Other completed-bank reducers combine filtering, resampling, and gated selection. Offline DeepConf removes low-confidence traces; Majority-of-the-Bests bootstraps Best-of- N decisions and returns their mode; and Best-of-Majority frequency-gates answer groups before reward-ranking the retained responses for Pass@ k (Fu et al., 2026; Rakhsha et al., 2025; Di et al., 2026). MBR instead selects the candidate with the largest empirical row utility, $n^{-1} \sum_j [\mathbf{W}_n]_{ij}$ (Kumar & Byrne, 2004; Freitag et al., 2022; Bertsch et al., 2023). These are decision structures, not evidence sources. If the output contains both a canonical answer and a rationale, the protocol must also specify which trace accompanies the winning answer.

Score semantics are part of the algorithm. Naming the scorer does not fully specify a score-aware reducer. The protocol must state where scores come from, whether larger values are better, and how token- or step-level scores are reduced. Answer canonicalization and invalid-output handling determine which candidates enter the rule. Score transformations, temperatures, and filtering thresholds determine how evidence enters the decision. The protocol must also define ties and fallbacks, and identify any evaluator prompts or models. Calibration parameters and other tunable values must be fixed on held-out data, not test labels.

Order-only rules such as hard Best-of- N , rank-based filtering, rank voting, and Majority-of-the-Bests are invariant to strictly increasing score transformations that preserve ties. Rules based on raw sums or means, softmax weights, or log-odds are not. Sequence and mean log-probability impose different length preferences. A raw verifier score need not be a calibrated probability, and calibrated log-odds may assign negative evidence to low-scoring candidates (Taubenfeld et al., 2025; Kuang et al., 2026). With hard maximization, exposure to proxy-score errors increases as the bank grows (Gao et al., 2023; Huang et al., 2025).

Causal stopping. Generation cost can fall only when a causal controller stops future work; evidence extraction and fixed-bank selection act after their required traces are complete. Let

$$\mathcal{H}_n = \sigma(x, (Y_i, a_i, m_i)_{i \leq n})$$

²We use *plurality voting* because this reducer selects the most frequent extracted answer even when it receives at most half of the votes. A strict majority requires more than half, so *majority voting* can be ambiguous for this rule.

be the information available after n completed rollouts. An across-rollout rule is a bounded stopping time $\tau_{\text{roll}} \leq N_{\text{max}}$ satisfying $\{\tau_{\text{roll}} \leq n\} \in \mathcal{H}_n$, and its final decision may use only the bank observed through τ_{roll} . In plain terms, whether the procedure has stopped by rollout n may depend on the first n completed rollouts, but not on any later candidate or metadata. Adaptive-Consistency stops when the posterior probability that the current count leader has greater latent mass than its runner-up crosses a threshold, whereas Early-Stopping Self-Consistency stops after a valid unanimous answer window (Aggarwal et al., 2023; Li et al., 2024). A stopping policy combines repeated application of the selector with a continuation criterion and a maximum budget.

For rollout i , let $\mathcal{F}_{i,t}$ contain the permitted prior history and the current rollout’s information through token t . A within-rollout stopping time τ_{tok} satisfies $\{\tau_{\text{tok}} \leq t\} \in \mathcal{F}_{i,t}$: it cannot use the hidden suffix or eventual answer to decide whether to truncate the current trace. DeepConf, for example, uses completed warm-up traces to calibrate a confidence threshold and may terminate later traces when sliding-window token confidence falls below it (Fu et al., 2026). Across-rollout rules can avoid only later traces, whereas within-rollout rules can also avoid a suffix of the current trace. Offline filtering of completed traces may change the returned answer but saves no generation tokens.

A recorded fixed bank can support an online claim only through causal retrospective replay. Candidate order must be fixed independently of outcomes, and each simulated action can inspect only the history revealed at that point. A simulated cutoff hides all later tokens and the candidate’s eventual answer. Calibration uses only previously designated traces, with tuning data kept separate from the test set. Cost accounting includes warm-up samples, discarded samples, and runs in which the stopping condition never fires. Any other replay uses information or compute unavailable to the deployed controller.

Comparison and compute accounting. A *shared-bank* comparison measures aggregation conditional on one completed bank. Each offline protocol receives the same serialized candidates and only the inference-time metadata it is permitted to use. Holding E_η , parsing, canonicalization, candidate eligibility, and score transformations fixed isolates $\mathcal{G}_\delta$; otherwise, those differences belong to the compared post-generation protocols. If both evidence and decision components vary, the comparison covers the full post-generation stage $(E_\eta, \mathcal{G}_\delta)$ under a fixed proposal and realized bank. Neither variant can estimate savings from adaptive stopping or changes caused by a different generation policy. Generalization beyond the realized bank still requires uncertainty estimates over prompts and candidate draws. An *end-to-end* comparison instead allows each complete protocol to generate its own stream, acquire its own evidence, execute its stopping rule, and be evaluated by M_B .

The matched budget should decompose as

$$C_{\text{total}} = C_{\text{gen}} + C_{\text{eval}}, \quad C_{\text{eval}} = C_{\text{signal}} + C_{\text{control}} + C_{\text{decision}}.$$

This accounting includes warm-up and discarded tokens, verifier or judge calls, repeated controller evaluations, and aggregation itself. Equal sample counts need not imply equal compute: external scoring adds candidate-wise cost, and naive pairwise MBR requires $O(n^2)$ utility evaluations (Cheng & Vlachos, 2023; Jinnai & Ariu, 2024). Costs that cannot be expressed in commensurate units should be reported separately alongside any scalar budget. Checkpoint-side compression can likewise move the wall-clock frontier at a fixed token budget: structured layer pruning of Qwen3-VL produced measurable time-to-first-token gains while retaining most, but not all, benchmark performance (Madinei et al., 2025). End-to-end utility should be plotted against total cost. Shared-bank results instead identify differences attributable to the post-generation aggregation stage, and holding evidence fixed narrows attribution to the decision rule. The discovery–stability profile in Section 3.4 remains complementary because it describes candidate availability under the proposal, not the success probability of a particular aggregation protocol.

3.4 Evaluation targets and metrics

Evaluation under test-time scaling should separate the performance of the deployed inference system from the shape of the candidate bank it induces.

End-to-end system performance. Let $\mathcal{P}$ denote the task distribution and $\mathcal{D}_{\text{eval}} = \{x_q\}_{q=1}^Q$ the evaluation set. For task utility U_x , budgeted algorithm $\mathcal{A}_B$, and algorithmic randomness ξ ,

$$M_B = \mathbb{E}_{x \sim \mathcal{P}, \xi} [U_x(\hat{y}_B(x; \xi))], \quad \widehat{M}_B = \frac{1}{Q} \sum_{q=1}^Q U_{x_q}(\hat{y}_B(x_q; \xi_q)).$$

This is the estimand for capability claims: it is the scaling curve $G(B)$ of Section 2.1, with the algorithmic randomness ξ written explicitly. Pairwise win rates, judge-aggregated rankings, and other relative comparisons define separate estimands; when they are used, the judge prompt, response order, reference material, and aggregation rule are part of the evaluation protocol.

Discovery–stability profile. For repeated-sampling diagnostics, fix the proposal or inference protocol that produces completed candidates. Let $Z_{q,i} \in \{0, 1\}$ indicate whether candidate i for prompt q is correct. Under the independent-attempt abstraction, let $p_q = \Pr(Z_{q,i} = 1 \mid x_q)$ be the latent single-attempt success probability for prompt q . For a fresh budget of k attempts,

$$X_{q,k} \mid p_q \sim \text{Binomial}(k, p_q).$$

For threshold $t \in \{1, \dots, k\}$, define the binomial tail kernel

$$\kappa_{k,t}(p) = \Pr\{\text{Binomial}(k, p) \geq t\} = \sum_{j=t}^k \binom{k}{j} p^j (1-p)^{k-j}.$$

The dataset-level discovery–stability profile at budget k is

$$S_k = (S_{k,1}, \dots, S_{k,k}), \quad S_{k,t} = \frac{1}{Q} \sum_{q=1}^Q \kappa_{k,t}(p_q).$$

Thus $S_{k,t}$ is the expected fraction of prompts on which a fresh k -attempt evaluation produces at least t correct candidates. Low thresholds measure discovery, high thresholds measure repeatability, and the full profile records how quickly occasional success decays into stable success.

When an observed bank contains N candidates per prompt and $k \leq N$, let $\nu_q = \sum_{i=1}^N Z_{q,i}$. The corresponding finite-bank diagnostic is the without-replacement tail

$$\widehat{S}_{k,t}^{\text{bank}} = \frac{1}{Q} \sum_{q=1}^Q \sum_{j=t}^k \frac{\binom{\nu_q}{j} \binom{N-\nu_q}{k-j}}{\binom{N}{k}},$$

with invalid binomial coefficients interpreted as zero. This statistic is a finite-bank estimate of the same threshold event, whereas S_k is the prospective latent profile. For banks produced by adaptive search or other dependent procedures, the finite-bank profile remains a descriptive summary of the returned bank; the latent i.i.d. interpretation should be invoked only when it matches the sampling protocol.

A Bayesian report separates finite-bank evidence from the latent profile. With $p_q \sim \text{Beta}(\alpha_q^0, \beta_q^0)$, defaulting to $\text{Beta}(1, 1)$ unless an auxiliary prior bank is declared, conjugacy gives

$$p_q \mid Z_{q,1:N} \sim \text{Beta}(a_q, b_q), \quad a_q = \alpha_q^0 + \nu_q, \quad b_q = \beta_q^0 + N - \nu_q.$$

The posterior mean of each profile coordinate is the beta-binomial predictive tail

$$\mu_{k,t} = \frac{1}{Q} \sum_{q=1}^Q \sum_{j=t}^k \binom{k}{j} \frac{B(a_q + j, b_q + k - j)}{B(a_q, b_q)}, \quad t = 1, \dots, k,$$

where $B(\cdot, \cdot)$ is the beta function. Shared posterior draws of $p_{1:Q}$ should be used to evaluate all thresholds and all scalar summaries, preserving their posterior dependence.

For rubric-valued outcomes, the same construction replaces the Bernoulli model with a Dirichlet–categorical model, fixes a category score vector in advance, and computes tail probabilities of the normalized k -sample rubric score. The binary exact-match profile above is the two-category case with score vector $(0, 1)$.

Scalar views of the profile. The profile S_k is the primary bank-level object. A scalar metric is a pre-specified functional $f(S_k)$, not a replacement for the profile. For a linear threshold utility with weights $\omega_{t,k} \geq 0$ and $\sum_{t=1}^k \omega_{t,k} = 1$,

$$U_\omega(k) = \sum_{t=1}^k \omega_{t,k} S_{k,t}.$$

Common repeated-sampling metrics are coordinates or simple functionals of the same profile, or are bounded by them (Chen et al., 2021; Wang et al., 2022; Yao et al., 2024; Liu et al., 2025a):

$$\begin{aligned} \text{Pass}@k &= S_{k,1}, & \text{pass}^k &= S_{k,k}, & \text{Maj}@k &\geq S_{k,\lfloor k/2 \rfloor + 1}, \\ \text{G-Pass}@k_\tau &= S_{k,\lceil \tau k \rceil}, & 0 < \tau &\leq 1, \\ \text{mG-Pass}@k &= \frac{1}{k - \lfloor k/2 \rfloor} \sum_{t=\lfloor k/2 \rfloor + 1}^k S_{k,t}, & \text{Geom}@k &= (S_{k,1} S_{k,k})^{1/2}, \end{aligned}$$

where the averaged form of mG-Pass@ k coincides with the definition of Liu et al. (2025a) for even k . The finite-bank versions are obtained by replacing $S_{k,t}$ with $\widehat{S}_{k,t}^{\text{bank}}$; in particular, $\widehat{\text{Pass}}@k = \widehat{S}_{k,1}^{\text{bank}}$. The strict-majority coordinate $S_{k,\lfloor k/2 \rfloor + 1}$ lower-bounds plurality voting: a strict majority of correct candidates guarantees that the correct answer wins the vote, whereas plurality can also succeed with fewer correct candidates when incorrect answers disagree. Ordinary single-sample pass rate is $S_{1,1}$, and for any k the uniform-threshold area recovers the same quantity:

$$\frac{1}{k} \sum_{t=1}^k S_{k,t} = \frac{1}{Q} \sum_{q=1}^Q p_q.$$

These quantities characterize candidate availability under the proposal; they do not give the success probability of an aggregation rule. Selection rules such as self-consistency, verifier reranking, MBR, and judge-based selection must be evaluated end-to-end through M_B . The same applies to adaptive stopping, which changes the realized bank. Under the complete protocol in Section 3.3, S_k remains a complementary diagnostic of the candidate bank and its discovery–stability shape.

3.5 Reproducibility

Under test-time scaling the object that must reproduce is a randomized inference protocol, not a model checkpoint. Reproducibility norms in machine learning and natural language processing ask whether independent researchers can obtain compatible results from the stated code, data, and procedure, and whether reported scores carry the development traces, budgets, and variability needed to interpret them (Pineau et al., 2021; Dodge et al., 2019; Bouthillier et al., 2021). A leaf-level run draws, or causally stops while constructing, a candidate bank from the proposal $q_\pi(\cdot | x)$ induced by the prompt, decoder, and budget (Section 3.2). It computes inference-time evidence and applies a fixed-bank decision (Section 3.3) before scoring the submitted output or candidate bank (Section 3.4). The bank is therefore one realization of a Monte Carlo experiment. Replaying that realization and reproducing the quantity it estimates are different goals. *Exact replay* reconstructs the same candidate texts, inference-time signals, stopping and aggregation decisions, and reported scores from the same items, weights, tokenizer, chat template, decoding configuration, random-number streams, software stack, and hardware; it underwrites auditing, debugging, and regression testing. *Distributional reproducibility* instead asks that an independent rerun of the declared protocol return statistically compatible estimates of the same estimand, such as M_B , the profile S_k , or a judge-defined win rate. Such a claim presupposes a model of the variation an honest rerun would exhibit. Fixing a seed makes one transcript auditable but characterizes neither the variability over prompts, candidate draws, and judge calls nor that of numerical execution; treating a single seeded run as evidence of replicability, and varying only the seed as a complete uncertainty analysis, are recurring errors (Bethard, 2022).

This variation has both statistical and procedural sources (Miller, 2024; Blackwell et al., 2024), and each enters at a distinct stage of the prefix-tree pipeline. The proposal is fixed by every prefix-local decoding

choice, including temperature, the ordering of truncation gates, deterministic token actions, stopping and length rules, and any adaptive budget controller. These settings constitute q_π rather than merely implement it (Section 3.2). The candidate bank need not be i.i.d.: samples that share prefix-search state, adaptive stopping decisions, verifier feedback, tool state, or a scheduler-dependent random-number stream are dependent, which voids the independent-attempt reading that the repeated-sampling diagnostics of Section 3.4 rely on. Generation does not fix the interpretation or evaluation map. Answer extractors, canonicalizers, unit-test harnesses, judge prompts, and pairwise orderings can treat the same completion differently. The position, verbosity, and protocol sensitivities of LLM judges then affect either the aggregation rule or the benchmark metric, depending on where the judge is used (Zheng et al., 2023; Zhou et al., 2025). Numerical execution is likewise not inert: when logits are close, differences in precision (Hariri et al., 2026b), kernels, or batching can flip the emitted token and propagate it along a long reasoning trace, so even greedy decoding need not be deterministic (Hochlehnert et al., 2025; Yuan et al., 2025), and the resulting run-to-run variability in reasoning outcomes should be measured directly rather than masked by a high average score (Liu et al., 2025a).

The checkable artifact is the bank together with its provenance: rendered prompts, raw completions, parsed answers, token- or step-level metadata, verifier and judge transcripts, transformed scores, stopping and aggregation decisions, and the per-candidate random stream. Releasing these records, as in transparent evaluation frameworks and recent reproducibility audits, lets others separate raw candidate quality from the behavior of the aggregation and stopping protocol (Liang et al., 2023; Hochlehnert et al., 2025). The protocol also determines the estimand. The sample budget N , temperature, evidence map, aggregation and stopping controls, extraction rules, and seed schedule must therefore be declared before outcomes are inspected. Changing them after seeing results turns evaluation into selection on the test set, and any validation-set tuning belongs in the report rather than in the headline number. Sample identities should be fixed before execution through a deterministic map from run, prompt, sample index, and stream name to a counter-based random state. This keeps asynchronous batching from changing which variate is attached to each candidate.

Quantifying variability. Distributional reproducibility is checkable only through an uncertainty estimate that matches the reported quantity. For a binary outcome $z_q \in \{0,1\}$ with dataset success rate $\hat{p} = \frac{1}{Q} \sum_{q=1}^Q z_q$, Wilson intervals remain stable when Q is small or $\hat{p}$ is close to 0 or 1 (Wilson, 1927). For repeated-sampling functionals such as $\text{Pass}@k = S_{k,1}$, the threshold profile $S_{k,t}$, or judge-based aggregates, closed-form variance expressions are rarely available because they depend on the aggregation protocol. A nonparametric bootstrap can instead resample prompts and recompute the full statistic (Efron, 1979); when items share context or document state, resampling should occur at that cluster level. The procedure must be *protocol-consistent*: if the reported output or score is produced by aggregation over N samples, its variability must be propagated through that aggregation rule rather than approximated by a simpler surrogate. Comparisons of two systems should be paired on the shared prompt set and, where the protocols allow, on aligned sample indices or common random streams, which cancels item difficulty without implying that the systems are deterministic. When these estimates are omitted, the small sample budgets common in test-time scaling reports can reorder systems from one rerun to the next; without a protocol-matched uncertainty estimate, the reported ranking is not a distributionally reproducible claim (Miller, 2024; Ye et al., 2024; Hariri et al., 2026c;a).

Numerical and serving settings. Numerical execution must therefore be specified as precisely as decoding: the inference library, the numerical formats for weights, activations, attention, and the key-value cache, and any quantization scheme belong in the report alongside the decoding, aggregation, and stopping settings, since low-bit quantization in particular can degrade reasoning on the hardest items (Liu et al., 2025b). An FP32 or higher-precision audit on a subset of items then separates algorithmic from numerical variation (Yuan et al., 2025). Model-state integrity is an additional serving-side concern: bit flips can corrupt language-model parameters and outputs, and lightweight tensor-auditing methods have been proposed to detect and recover such faults at inference time (Tahmasivand et al., 2025).

Which reproducibility target is claimed dictates what must be controlled: exact replay calls for deterministic kernels, fixed batch schedules, and serialized banks, whereas a capability claim is better served by the protocol-consistent uncertainty over independent executions described above. Exact replay secures au-

Release block	Questions	Banks	N	Traces	Signals beyond full text and outcome
MMLU-Pro + BBH	18,543	27	1	500,661	Parsed answer and document/prompt/target hashes
AIME'24/'25, HMMT'25, and BrUMO'25	120	20	80	192,000	Chosen-token log probability and rank; outcome-verifier score
AIME'26, HMMT Nov.'25/Feb.'26, CMIMC'25, and SMT'25	186	7	80	104,160	Width-20 token alternatives; reference-based and reference-free verifiers
SuperGPQA (subset)	3,600	4	80	1,152,000	Width-20 token alternatives and both verifier families
Total	22,449	–	–	1,948,821	

Table 1: **Reasoning-trace release.** A bank is one model configuration evaluated over a complete question set; N is the number of responses per question in that bank. For BBH, rows produced by two answer extractors are counted once per generated response. The seven signal-rich mathematics banks include three independently generated medium-effort repeats that are retained in the corpus but excluded from effort-comparison curves.

ditability; distributional reproducibility, certified by an uncertainty estimate matched to the protocol, is what makes comparisons between test-time scaling methods scientifically meaningful.

4 Empirical Study and Trace Corpus

Our empirical study uses fixed response banks to measure how quickly additional sampling reveals a correct candidate and how often practical reducers select one. The release retains the responses, outcomes, and recorded signals, so new reducers can be evaluated without regenerating responses. Its four blocks span different tasks, model rosters, sampling protocols, and levels of signal detail (Table 1). We therefore report each block separately.

4.1 Broad evaluation across knowledge and symbolic reasoning

This block covers all 14 MMLU-Pro domains (Wang et al., 2024) and the 23 BIG-Bench Hard tasks (Suzgun et al., 2023). We evaluate 27 open-weight reasoning models with one zero-shot chain-of-thought response per question, yielding 500,661 responses. MMLU-Pro exact match ranges from 3.13% to 70.47% across the roster, while BBH flexible-extraction exact match ranges from 25.97% to 82.20%. We report the two suites separately because their tasks, answer spaces, and parsers differ. The complete roster, prompts, and file census appear in Section G.3.

4.2 Candidate discovery outpaces answer selection

The second block contains 80 seeded responses for every combination of 20 model configurations and 120 questions from AIME'24, AIME'25, HMMT'25, and BrUMO'25 (Jia, 2024; math-ai, 2025; MathArena, 2025c;a), for 192,000 traces. We compute the exact, without-replacement candidate-bank statistics at $k \in \{1, 2, 4, 8, 16, 32, 64, 80\}$, and replay reducers on nested random subsets of the same banks. Because every reducer sees subsets of the same response banks, differences among reducers cannot be attributed to newly generated candidates.

Across the roster, the median Pass@ k rises from 56.49% at $k = 1$ to 82.08% at $k = 80$, whereas the median all-correct coordinate falls to 15.00%. For this roster, mean Spearman agreement with the $N = 80$ accuracy ranking is 0.967 at $k = 1$ and 0.992 at $k = 8$. These are descriptive properties of the 20 configurations, not a general small-sample guarantee.

Qwen3-30B-A3B-Thinking-2507 has the highest single-response accuracy in the roster, at 75.56%. Pass@80 reaches 91.67%, but the probability that all drawn responses are correct falls to 43.33% (Figure 5, left). Ad-

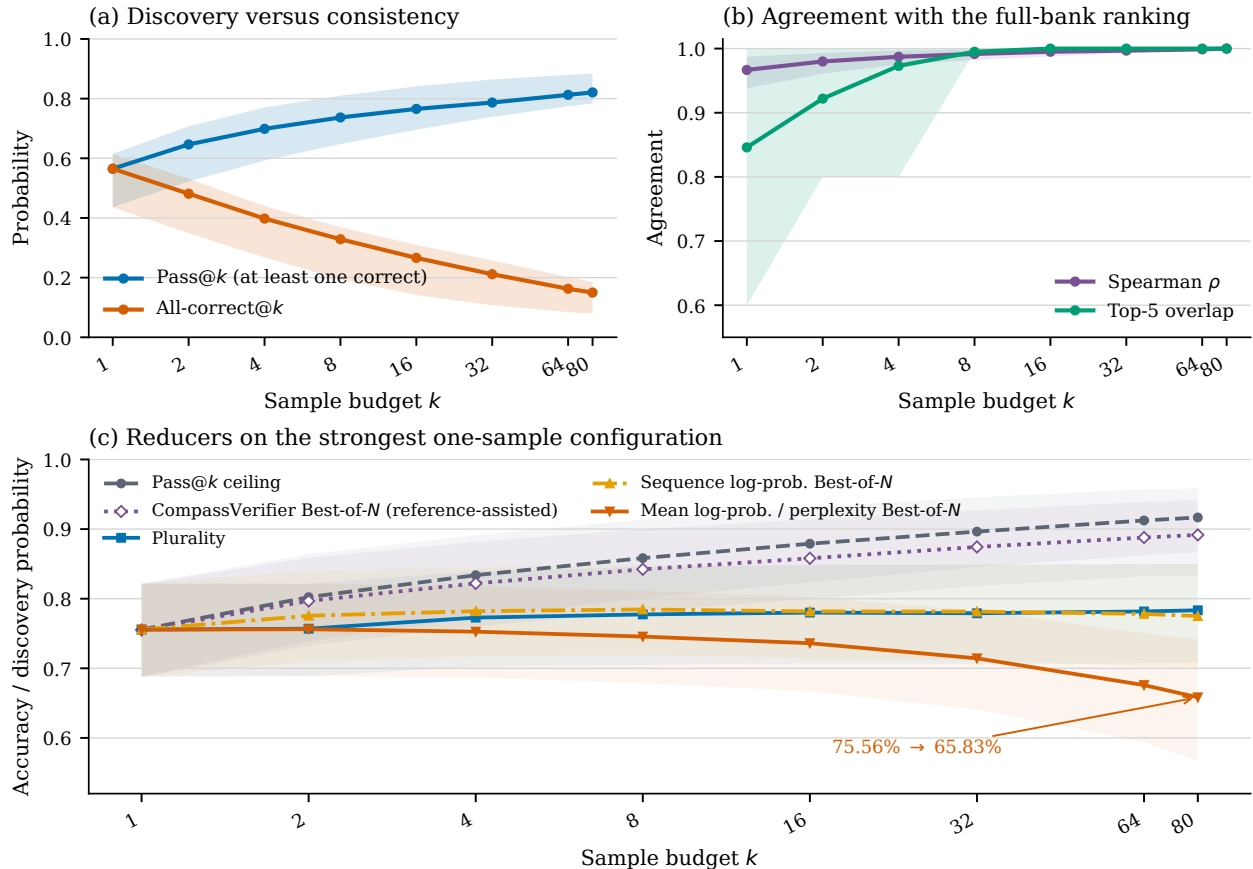


Figure 5: **Candidate availability and reducer accuracy under repeated sampling.** (a) Median exact Pass@ k and pass ^{k} across the 20 configurations; bands span the interquartile range of configurations. (b) Mean agreement of subset rankings with the $N = 80$ ranking; bands are central 95% ranges over 200 paired subset replays. (c) Reducers for Qwen3-30B-A3B-Thinking-2507, the strongest single-response configuration in this roster; bands are 95% prompt-bootstrap intervals. CompassVerifier sees the reference answer and is shown only as a reference-assisted diagnostic. Mean log probability and negative perplexity induce the same ordering, so one curve represents both.

ditional samples increase candidate coverage and expose within-question instability. Literal answer plurality reaches 78.33% at $k = 80$, and sequence-log-probability selection reaches 77.50%. Selecting the response with the highest mean token log probability instead falls from 75.56% to 65.83% as the bank grows (Figure 5, right). More generation compute can thus lower submitted-answer accuracy when the selection score is misaligned with correctness. The reference-assisted CompassVerifier diagnostic reaches 89.17%, close to the candidate ceiling, but it has access to the gold answer and is not an inference-time result.

4.3 Signal-rich competition mathematics (2025–2026)

The third block comprises 186 problems drawn from five recent competition datasets distributed by MathArena: AIME’26, HMMT Feb.’26, HMMT Nov.’25, CMIMC’25, and SMT’25 (MathArena, 2026a;b; 2025d;b;e; Dekoninck et al., 2026).

The effort comparison comprises one Qwen3.6-35B-A3B bank and three gpt-oss-20b banks at low, medium, and high reasoning effort. Every trace records the full response, rule-based outcome, chosen token, and up to 20 token alternatives at every generation position in order to calculate features proposed by DeepConf Fu et al. (2026) and Self-Certainty Kang et al. (2025).

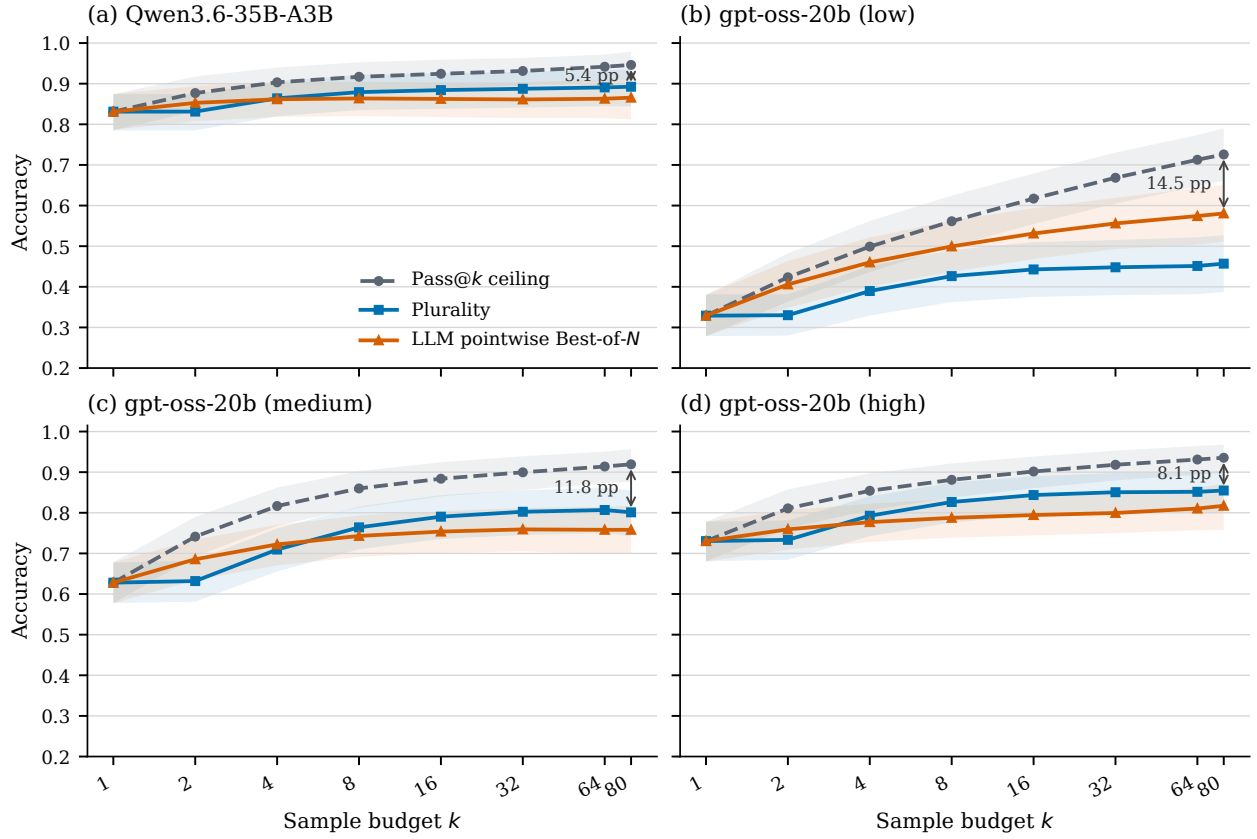


Figure 6: **Finite-bank scaling on the five 2025–2026 competition sets.** Each panel compares the exact $\text{Pass}@k$ ceiling with literal answer plurality and pointwise Best-of- N using the reference-free verifier described below. Arrows mark the $k = 80$ gap between $\text{Pass}@80$ and the better of the two observed reducers. Shading gives 95% prompt-bootstrap intervals conditional on the observed 80-response banks. Intermediate reducer points average 2,000 nested subset replays; $k = 1$ and $k = 80$ use exact endpoints. The horizontal axis counts generated candidates and excludes verifier computation.

$\text{Pass}@k$ increases for all four banks (Figure 6). At $k = 80$, $\text{Pass}@80$ is 94.62% for Qwen3.6 and 72.58%, 91.94%, and 93.55% for gpt-oss low, medium, and high. The corresponding accuracies after reference-free pointwise selection are 86.56%, 58.06%, 75.81%, and 81.72%. The remaining 8.06–16.13 percentage-point gaps measure the share of questions for which a correct candidate is available but the reducer does not select it. Overlapping prompt-bootstrap intervals and differences in response length preclude a total ordering of the generators. Among high-effort responses, 2,399 of 14,880 (16.12%) reach the common 81,920-token cap, compared with none at low effort. Taking the better of pointwise selection and plurality separately for each bank still leaves a 5.38–14.52 point gap to $\text{Pass}@80$.

4.4 Reference-assisted and reference-free verifier signals

The signal-rich blocks record one reference-assisted score and one reference-free score. CompassVerifier-3B (OpenCompass) receives the question, reference answer, and candidate response and predicts A/B/C for correct, incorrect, or invalid responses (Liu et al., 2025c). We retain both its direct label distribution and a null-baseline-adjusted contextual distribution. The reference-dependent score is an outcome diagnostic and cannot be used for inference-time selection when gold answers are unavailable.

Our reference-free score is a pointwise adaptation of the pairwise LLM-as-a-Verifier method (Kwok et al., 2026). Qwen3.6 sees only one problem and one candidate response. It receives no reference answer or ground truth, Compass result, or other candidate. It separately critiques problem understanding, reasoning validity,

and whether the conclusion is supported, then assigns each criterion an ordinal A–T score. We map A through T to 20 through 1, take the expected ordinal score under the returned, visible-mass-renormalized scoring-token probabilities, normalize it to $[0, 1]$, and average the three criteria. This signal can rank candidates without a gold answer and it is an ordinal evaluator score. The exact prompts, decoding passes, score equations, context handling, and audit counts are given in Section G.6.

For SuperGPQA, each bank likewise contains 80 responses per question across broad graduate-level knowledge (M-A-P Team et al., 2025). We construct a field-balanced set by selecting 50 questions from each of 72 EvalScope fields, for 3,600 questions total (ModelScope Team, 2024). The four banks consist of Qwen3.6 and three gpt-oss conditions at low, medium, and high effort. The release includes the full trace, width-20 token alternatives, rule-based multiple-choice outcome, and both verifier records.

5 Conclusion

Test-time scaling is a family of budgeted inference algorithms. Viewing these algorithms over the implicit prefix tree separates single-trajectory deliberation, leaf-level sampling and reduction, and prefix-level search, while exposing the proposal, evidence map, decision rule, and causal stopping controller as distinct components of the inference process. The scientifically meaningful object of comparison is therefore the complete inference system.

This perspective naturally determines how reasoning systems should be evaluated. End-to-end utility and candidate-bank diagnostics answer different questions; shared-bank and end-to-end comparisons support different claims; compute accounting must include generation, evaluation, control, and decision; and uncertainty must propagate through the deployed inference protocol. Our framework also unifies repeated-sampling evaluation through the discovery–stability profile and distinguishes exact replay from distributional reproducibility. The accompanying corpus comprises 1,948,821 full reasoning traces spanning broad knowledge, symbolic reasoning, and competition mathematics, including token-level candidate log probabilities and two verifier families in the signal-rich blocks.

Our empirical study demonstrates the practical consequences of this system-level view. Rankings change with the rubric, sampling budget, benchmark, and commitment standard; runtime signals that predict correctness within trajectories need not reliably rank models; and strict extraction can conflate abstention with error. Our results argue that progress in test-time scaling should be measured using reproducible inference systems whose inference protocols, compute budgets, and uncertainties are explicitly reported, rather than by checkpoints evaluated under incompletely specified procedures.

Acknowledgments

This research was supported in part by NSF awards 2117439, 2112606, and 2320952.

References

- Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, et al. Phi-4-reasoning technical report. *arXiv preprint arXiv:2504.21318*, 2025. doi: 10.48550/arXiv.2504.21318. URL <https://arxiv.org/abs/2504.21318>.
- Marah Abdin et al. Phi-4 technical report, 2024. URL <https://arxiv.org/abs/2412.08905>.
- Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let’s sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.761. URL <https://aclanthology.org/2023.emnlp-main.761/>.

-
- Wasi Uddin Ahmad, Somshubra Majumdar, Aleksander Ficek, Sean Narenthiran, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Vahid Noroozi, and Boris Ginsburg. OpenCodeReasoning-II: A Simple Test Time Scaling Approach via Self-Critique, 2025a. URL <https://arxiv.org/abs/2507.09075>.
- Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. OpenCodeReasoning: Advancing Data Distillation for Competitive Coding. *arXiv preprint arXiv:2504.01943*, 2025b. URL <https://arxiv.org/abs/2504.01943>.
- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL <https://arxiv.org/abs/2402.14740>.
- Arcee AI. mergekit: Tools for merging pretrained large language models. <https://github.com/arcee-ai/mergekit>, 2024. Accessed: 2026-02-26.
- Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating llms on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1d27c01ebd3e3aeb226b44fc970d803-Abstract-Datasets_and_Benchmarks_Track.html.
- Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, et al. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025.
- Amanda Bertsch, Alex Xie, Graham Neubig, and Matthew R. Gormley. It’s MBR all the way down: Modern generation techniques through the lens of minimum bayes risk, 2023. URL <https://arxiv.org/abs/2310.01387>.
- Bespoke Labs. Bespoke curator: Synthetic data curation for post-training. <https://github.com/bespokelabsai/curator>, 2025a. Accessed: 2025-01-31.
- Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. <https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation>, 2025b. Accessed: 2025-01-22.
- Bespoke Labs. Bespoke-stratos-17k dataset card. <https://huggingface.co/datasets/bespokelabs/Bespoke-Stratos-17k>, 2025c. Accessed: 2025-01-31.
- Bespoke Labs. Bespoke-stratos-32b model card. <https://huggingface.co/bespokelabs/Bespoke-Stratos-32B>, 2025d. Accessed: 2025-01-31.
- Bespoke Labs. Bespoke-stratos-7b model card. <https://huggingface.co/bespokelabs/Bespoke-Stratos-7B>, 2025e. Accessed: 2025-01-31.
- Steven Bethard. We need to talk about random seeds. *arXiv preprint arXiv:2210.13393*, 2022. doi: 10.48550/arXiv.2210.13393. URL <https://arxiv.org/abs/2210.13393>.
- Robert E Blackwell, Jon Barry, and Anthony G Cohn. Towards reproducible llm evaluation: Quantifying uncertainty in llm benchmark scores. *arXiv preprint arXiv:2410.03492*, 2024.
- Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Dmitriy Serdyuk, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems*, volume 3, pp. 747–769, 2021. URL https://proceedings.mlsys.org/paper_files/paper/2021/hash/0184b0cd3cfb185989f858a1d9f5c1eb-Abstract.html.

-
- Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfschagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1): 1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation, 2023. URL <https://arxiv.org/abs/2311.17311>.
- Julius Cheng and Andreas Vlachos. Faster minimum bayes risk decoding with confidence-based pruning, 2023. URL <https://arxiv.org/abs/2311.14919>.
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.
- Karl Cobbe et al. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Yuan, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL <https://arxiv.org/abs/2502.01456>.
- DeepSeek-AI. deepseek-ai/deepseek-r1-distill-qwen-32b (model card). <https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B>, 2025a. Accessed: 2026-02-26.
- DeepSeek-AI. Deepseek-r1 (github repository). <https://github.com/deepseek-ai/DeepSeek-R1>, 2025b. Accessed: 2026-02-26.
- Jasper Dekoninck, Nikola Jovanović, Tim Gehringer, Kári Rognvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: MathArena as an evaluation platform for mathematics with llms, 2026. URL <https://arxiv.org/abs/2605.00674>.
- Qiwei Di, Kaixuan Ji, Xuheng Li, Heyang Zhao, and Quanquan Gu. Best-of-majority: Minimax-optimal strategy for pass@k inference scaling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=a6CVQpjbXq>.
- Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2185–2194, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1224. URL <https://aclanthology.org/D19-1224/>.
- Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.
- Bryan Eikema and Wilker Aziz. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, 2020. URL <https://aclanthology.org/2020.coling-main.398/>.
- Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, 2018.

-
- Markus Freitag, David Grangier, Qijun Tan, and Bowen Liang. High quality rather than high model probability: Minimum bayes risk decoding with neural metrics. *Transactions of the Association for Computational Linguistics*, 10:811–825, 2022. doi: 10.1162/tac1_a_00491. URL <https://aclanthology.org/2022.tac1-1.47/>.
- Yichao Fu, Xuwei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=8LqHs0KIM7>.
- FuseAI. Fuseo1-deepseekr1-qwq-skyt1-32b-preview (model card). <https://huggingface.co/FuseAI/FuseO1-DeepSeekR1-QwQ-SkyT1-32B-Preview>, 2025a. Accessed: 2026-02-26.
- FuseAI. Fuseo1-deepseekr1-qwq-skyt1-flash-32b-preview (model card). <https://huggingface.co/FuseAI/FuseO1-DeepSeekR1-QwQ-SkyT1-Flash-32B-Preview>, 2025b. Accessed: 2026-02-26.
- FuseAI. Fuseai: Model fusion for large language models (code and resources). <https://github.com/fanqiwan/FuseAI>, 2025c. Accessed: 2026-02-26.
- GAIR. Gair/limo-v2 (dataset card). <https://huggingface.co/datasets/GAIR/LIMO-v2>, 2025a. Accessed: 2026-02-26.
- GAIR. Gair/limo-v2 (model card). <https://huggingface.co/GAIR/LIMO-v2>, 2025b. Accessed: 2026-02-26.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866, 2023. URL <https://proceedings.mlr.press/v202/gao23h.html>.
- Sajjad Ghiasvand, Mahnoosh Alizadeh, and Ramtin Pedarsani. Decentralized low-rank fine-tuning of large language models. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*. Association for Computational Linguistics, July 2025. doi: 10.18653/v1/2025.realm-1.24. URL <https://aclanthology.org/2025.realm-1.24/>.
- Sajjad Ghiasvand, Mark Beliaev, Mahnoosh Alizadeh, and Ramtin Pedarsani. Realm: Reliable expertise-aware language model fine-tuning from noisy annotations, 2026. URL <https://arxiv.org/abs/2604.17289>.
- Etash Guha, Negin Raoof, Jean Mercat, Eric Frankel, Sedrick Keh, Sachin Grover, George Smyrnis, Trung Vu, Ryan Marten, Jon Saad-Falcon, Caroline Choi, Kushal Arora, Mike Merrill, Yichuan Deng, Ashima Suvana, Hritik Bansal, Marianna Nezhurina, Yejin Choi, Reinhard Heckel, Seewong Oh, Tatsunori Hashimoto, Jenia Jitsev, Vaishaal Shankar, Ludwig Schmidt, Alex Dimakis, and Mahesh Sathiamoorthy. Evalchemy, November 2024.
- Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models, 2025. URL <https://arxiv.org/abs/2506.04178>.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL <https://aclanthology.org/2023.emnlp-main.507/>.

-
- Mohsen Hariri, Michael Hinczewski, Jing Ma, and Vipin Chaudhary. Ranking reasoning LLMs under test-time scaling. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33437–33478, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1544. URL <https://aclanthology.org/2026.acl-long.1544/>.
- Mohsen Hariri, Alan Luo, Weicong Chen, Tianyi Zhang, Qifan Wang, Xiaotian Han, and Vipin Chaudhary. Quantize what counts: More for keys, less for values. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26393–26420, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1314. URL <https://aclanthology.org/2026.findings-acl.1314/>.
- Mohsen Hariri, Amirhossein Samandar, Michael Hinczewski, and Vipin Chaudhary. Don’t pass@k: A bayesian framework for large language model evaluation. In *Proceedings of the 14th International Conference on Learning Representations (ICLR 2026)*, 2026c. URL <https://openreview.net/forum?id=PTXi3Ef4sT>.
- John Hewitt, Christopher D Manning, and Percy Liang. Truncation sampling as language model desmoothing. *arXiv preprint arXiv:2210.15191*, 2022.
- Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandara, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=90UrTTxp50>.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.
- Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan Foster. Is best-of- n the best of them? coverage, scaling, and optimality in inference-time alignment. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 25075–25126, 2025. doi: 10.48550/arXiv.2503.21878. URL <https://arxiv.org/abs/2503.21878>.
- Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025a. URL <https://github.com/huggingface/open-r1>.
- Hugging Face. Open-r1: A fully open reproduction of deepseek-r1. <https://github.com/huggingface/open-r1>, 2025b. Accessed: 2026-02-26.
- Hugging Face. Open r1: Update #2. <https://huggingface.co/blog/open-r1/update-2>, 2025c. Accessed: 2026-02-26.
- Hugging Face. Open r1: Update #3. <https://huggingface.co/blog/open-r1/update-3>, 2025d. Accessed: 2026-02-26.
- Chia-Yu Hung, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Reward-guided tree search for inference time alignment of large language models. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12575–12593, 2025. URL <https://aclanthology.org/2025.naacl-long.625/>.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=chfJJYC3iL>.
- Maxwell Jia. AIME 2024 dataset. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

-
- Yuu Jinnai and Kaito Ariu. Hyperparameter-free approach for faster minimum bayes risk decoding, 2024. URL <https://arxiv.org/abs/2401.02749>.
- Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of- n sampling with minimum Bayes risk objective for language model alignment. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9321–9347, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.472. URL <https://aclanthology.org/2025.naacl-long.472/>.
- Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of- n selection for large language models via self-certainty. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1c7eff166a8e345f664f0faa8f4e4d2e-Abstract-Conference.html.
- Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio du Pin Calmon. Inference-time reward hacking in large language models, 2025. URL <https://arxiv.org/abs/2506.19248>. Accepted to NeurIPS 2025 (Spotlight).
- Kiana Kiashemshaki, Mohammad Jalili Torkamani, Negin Mahmoudi, and Meysam Shirdel Bilehsavar. Simulating a bias mitigation scenario in large language models, 2025. URL <https://arxiv.org/abs/2509.14438>.
- Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pp. 282–293. Springer, 2006.
- Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *Transactions on Machine Learning Research*, 2025. URL <https://openreview.net/forum?id=QF0N3x2XVm>.
- Peng Kuang, Yanli Wang, Xiaoyu Han, Yaowenqi Liu, Kaidi Xu, and Haohan Wang. Optimal aggregation of LLM and PRM signals for efficient test-time scaling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=x85kiYqL4y>.
- Shankar Kumar and William Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004. URL <https://aclanthology.org/N04-1022/>.
- Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. LLM-as-a-Verifier: A general-purpose verification framework, 2026. URL <https://arxiv.org/abs/2607.05391>.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL <https://doi.org/10.1145/3600006.3613165>.
- LG AI Research. Exaone 4.0: Unified large language models integrating non-reasoning and reasoning modes. *arXiv preprint arXiv:2507.11407*, 2025.
- Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hahamaneishi, Shishir G. Patil, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Llms can easily learn to reason from demonstrations: Structure, not content, is what matters!, 2025a. URL <https://arxiv.org/abs/2502.07374>.
- Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025b.

-
- Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=ndR8Ytrzhh>.
- Yujia Li, David Choi, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Ré, Diana Acosta-Navas, Drew A Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J Orr, Lucia Zheng, Mert Yüksesgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Yang Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. URL <https://openreview.net/forum?id=i04LZibEqW>.
- Hunter Lightman et al. Let’s verify step by step, 2023. URL <https://arxiv.org/abs/2305.20050>.
- Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. Are your LLMs capable of stable reasoning? In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17594–17632. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-acl.905. URL <https://aclanthology.org/2025.findings-acl.905/>.
- Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng Yu, Chun Yuan, and Lu Hou. Quantization hurts reasoning? an empirical study on quantized reasoning models. *arXiv preprint arXiv:2504.04823*, 2025b.
- Shudong Liu, Hongwei Liu, Junnan Liu, Linchen Xiao, Songyang Gao, Chengqi Lyu, Yuzhe Gu, Wenwei Zhang, Derek F. Wong, Songyang Zhang, and Kai Chen. CompassVerifier: A unified and robust verifier for LLMs evaluation and outcome reward. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33466–33494, 2025c. doi: 10.18653/v1/2025.emnlp-main.1698. URL <https://aclanthology.org/2025.emnlp-main.1698/>.
- Zihan Liu, Zhuolin Yang, Yang Chen, Chankyu Lee, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron 1.1: Advancing math and code reasoning through sft and rl synergy. *arXiv preprint arXiv:2506.13284*, 2025d.
- M-A-P Team. SuperGPQA dataset card. Hugging Face dataset, 2025. URL <https://huggingface.co/datasets/m-a-p/SuperGPQA>.
- M-A-P Team et al. SuperGPQA: Scaling llm evaluation across 285 graduate disciplines. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a3c5af1f56fc73eef1ba0f442739f5ca-Abstract-Datasets_and_Benchmarks_Track.html.
- Parsa Madinei, Ryan Solgi, Ziqi Wen, Jonathan Skaza, Miguel Eckstein, and Ramtin Pedarsani. Interlace: Interleaved layer pruning and efficient adaptation in large vision-language models, 2025. URL <https://arxiv.org/abs/2511.19676>.
- Nazanin Mahjourian and Vinh Nguyen. Sanitizing manufacturing dataset labels using vision-language models. *Machine Learning with Applications*, 2026. doi: <https://doi.org/10.1016/j.mlwa.2026.100893>. URL <https://www.sciencedirect.com/science/article/pii/S2666827026000587>.
- math-ai. AIME 2025 dataset. Hugging Face dataset, 2025. URL <https://huggingface.co/datasets/math-ai/aime25>.

-
- MathArena. BrUMO 2025 dataset card. Hugging Face dataset, 2025a. URL https://huggingface.co/datasets/MathArena/brumo_2025.
- MathArena. CMIMC 2025 dataset card. Hugging Face dataset, 2025b. URL https://huggingface.co/datasets/MathArena/cmimc_2025.
- MathArena. HMMT February 2025 dataset card. Hugging Face dataset, 2025c. URL https://huggingface.co/datasets/MathArena/hmmt_feb_2025.
- MathArena. HMMT November 2025 dataset card. Hugging Face dataset, 2025d. URL https://huggingface.co/datasets/MathArena/hmmt_nov_2025.
- MathArena. SMT 2025 dataset card. Hugging Face dataset, 2025e. URL https://huggingface.co/datasets/MathArena/smt_2025.
- MathArena. AIME 2026 dataset card. Hugging Face dataset, 2026a. URL https://huggingface.co/datasets/MathArena/aime_2026.
- MathArena. HMMT February 2026 dataset card. Hugging Face dataset, 2026b. URL https://huggingface.co/datasets/MathArena/hmmt_feb_2026.
- Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2023.
- Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward, 2024. URL <https://arxiv.org/abs/2405.14734>.
- Microsoft. microsoft/phi-4-mini-flash-reasoning (model card). <https://huggingface.co/microsoft/Phi-4-mini-flash-reasoning>, 2025. Accessed: 2026-02-26.
- Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024.
- ModelScope Team. EvalScope: Evaluation framework for large models, 2024. URL <https://github.com/modelscope/evalscope>.
- Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 Winning Solution: Building State-of-the-Art Mathematical Reasoning Models with OpenMathReasoning dataset, 2025. URL <https://arxiv.org/abs/2504.16891>.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321. Association for Computational Linguistics, 2025. URL <https://aclanthology.org/2025.emnlp-main.1025.pdf>.
- Mohammad Hadi Nezhad, Francisco Enrique Vicente Castro, and Ivon Arroyo. Understanding users’ privacy reasoning and behaviors during chatbot use to support meaningful agency in privacy, 2026. URL <https://arxiv.org/abs/2601.18125>.
- NovaSky Team. Think less, achieve more: Cut reasoning costs by 50% without sacrificing accuracy. <https://novasky-ai.github.io/posts/reduce-overthinking>, 2025a. Accessed: 2025-01-23.
- NovaSky Team. Sky-t1: Train your own o1 preview model within \$450. <https://novasky-ai.github.io/posts/sky-t1>, 2025b. Accessed: 2025-01-09.
- NovaSky Team. Unlocking the potential of reinforcement learning in improving reasoning models. <https://novasky-ai.github.io/posts/sky-t1-7B/>, 2025c. Accessed: 2025-02-13.
- Open-R1 Team. Codeforces-cots dataset card. <https://huggingface.co/datasets/open-r1/codeforces-cots>, 2025a. Accessed: 2026-02-26.

Open-R1 Team. Mixture-of-thoughts dataset card. <https://huggingface.co/datasets/open-r1/Mixture-of-Thoughts>, 2025b. Accessed: 2026-02-26.

Open-R1 Team. Olympiccoder-32b model card. <https://huggingface.co/open-r1/OlympicCoder-32B>, 2025c. Accessed: 2026-02-26.

Open-R1 Team. Olympiccoder-7b model card. <https://huggingface.co/open-r1/OlympicCoder-7B>, 2025d. Accessed: 2026-02-26.

Open-R1 Team. Openr1-distill-7b model card. <https://huggingface.co/open-r1/OpenR1-Distill-7B>, 2025e. Accessed: 2026-02-26.

Open-R1 Team. Openr1-math-220k dataset card. <https://huggingface.co/datasets/open-r1/OpenR1-Math-220k>, 2025f. Accessed: 2026-02-26.

OpenAI. gpt-oss-20b model (openai api documentation). <https://developers.openai.com/api/docs/models/gpt-oss-20b>, 2025a. Accessed: 2026-02-26.

OpenAI. Introducing gpt-oss. <https://openai.com/index/introducing-gpt-oss/>, 2025b. Accessed: 2026-02-26.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL <https://arxiv.org/abs/2508.10925>.

OpenAI. OpenAI Harmony: Response format for open-weight models. Software, 2025a. URL <https://github.com/openai/harmony>.

OpenAI. openai/gpt-oss-20b (model card). <https://huggingface.co/openai/gpt-oss-20b>, 2025b. Accessed: 2026-02-26.

OpenAI. openai/gpt-oss-safeguard-20b (model card). <https://huggingface.co/openai/gpt-oss-safeguard-20b>, 2025c. Accessed: 2026-02-26.

OpenThoughts Team. Openthinker2-32b model card. <https://huggingface.co/open-thoughts/OpenThinker2-32B>, 2025a. Accessed: 2026-02-26.

OpenThoughts Team. Openthinker3-7b model card. <https://huggingface.co/open-thoughts/OpenThinker3-7B>, 2025b. Accessed: 2026-02-26.

OpenThoughts Team. Open Thoughts. <https://open-thoughts.ai>, February 2025c.

OpenThoughts Team. Openthoughts-114k dataset card. <https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k>, 2025d. Accessed: 2026-02-26.

OpenThoughts Team. Openthoughts2-1m dataset card. <https://huggingface.co/datasets/open-thoughts/OpenThoughts2-1M>, 2025e. Accessed: 2026-02-26.

OpenThoughts Team. Openthoughts3-1.2m dataset card. <https://huggingface.co/datasets/open-thoughts/OpenThoughts3-1.2M>, 2025f. Accessed: 2026-02-26.

Yonatan Oren, Nicole Meister, Niladri S Chatterji, Faisal Ladhak, and Tatsunori B Hashimoto. Proving test set contamination in black-box language models. In *International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=KS8mIvetg2>.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d’Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164): 1–20, 2021. URL <https://jmlr.org/papers/v22/20-303.html>.

-
- Qihoo360. qihoo360/light-r1-14b-ds (model card). <https://huggingface.co/qihoo360/Light-R1-14B-DS>, 2025a. Accessed: 2026-02-26.
- Qihoo360. qihoo360/light-r1-32b (model card). <https://huggingface.co/qihoo360/Light-R1-32B>, 2025b. Accessed: 2026-02-26.
- Qihoo360. qihoo360/light-r1-32b-ds (model card). <https://huggingface.co/qihoo360/Light-R1-32B-DS>, 2025c. Accessed: 2026-02-26.
- Qihoo360. qihoo360/light-r1-7b-ds (model card). <https://huggingface.co/qihoo360/Light-R1-7B-DS>, 2025d. Accessed: 2026-02-26.
- Qwen Team. Qwen/qwq-32b (model card). <https://huggingface.co/Qwen/QwQ-32B>, 2025a. Accessed: 2026-02-26.
- Qwen Team. Qwen/qwen3-30b-a3b-thinking-2507 (model card). <https://huggingface.co/Qwen/Qwen3-30B-A3B-Thinking-2507>, 2025b. Accessed: 2026-02-26.
- Qwen Team. Qwen/qwen3-4b-thinking-2507 (model card). <https://huggingface.co/Qwen/Qwen3-4B-Thinking-2507>, 2025c. Accessed: 2026-02-26.
- Qwen Team. Qwen/qwen3-next-80b-a3b-thinking (model card). <https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Thinking>, 2025d. Accessed: 2026-02-26.
- Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025e. URL <https://qwenlm.github.io/blog/qwq-32b/>.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023. URL <https://arxiv.org/abs/2305.18290>.
- Amin Rakhsha, Kanika Madan, Tianyu Zhang, Amir-massoud Farahmand, and Amir Khasahmadi. Majority of the bests: Improving best-of- n via bootstrapping. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/36556567e8437f137da23047309155dd-Abstract-Conference.html.
- Liliang Ren et al. Decoder-hybrid-decoder architecture for efficient reasoning with long generation, 2025. URL <https://arxiv.org/abs/2507.06607>.
- SaylorTwift. BIG-Bench Hard dataset card. Hugging Face dataset mirror, 2025. URL <https://huggingface.co/datasets/SaylorTwift/bbh>.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, 2025.
- Lin Sun, Guangxiang Zhao, Xiaoqi Jian, Yuhao Wu, Weihong Lin, Yongfu Zhu, Linglin Zhang, Jinzhu Wu, Junfeng Ran, Sai-er Hu, et al. Tinyr1-32b-preview: Boosting accuracy with branch-merge distillation. *arXiv preprint arXiv:2503.04872*, 2025.
- Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, et al. EleutherAI/lm-evaluation-harness: v0.4.8. Software, version v0.4.8, March 2025. URL <https://doi.org/10.5281/zenodo.14970487>.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL <https://aclanthology.org/2023.findings-acl.824/>.

-
- Ahmad Tahmasivand, Noureldin Zahran, Saba Al-Sayouri, Mohammed Fouda, and Khaled N. Khasawneh. Lm-fix: Lightweight bit-flip detection and rapid recovery framework for language models. In *2025 IEEE 43rd International Conference on Computer Design (ICCD)*, 2025. doi: 10.1109/ICCD65941.2025.00068.
- Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111. Association for Computational Linguistics, July 2025. URL <https://aclanthology.org/2025.findings-acl.1030/>.
- Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
- TIGER-Lab. MMLU-Pro dataset card. Hugging Face dataset, 2024. URL <https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro>.
- Shubham Toshniwal, Ivan Sorokin, Aleksander Ficek, Ivan Moshkov, and Igor Gitman. GenSelect: A Generative Approach to Best-of-N. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL <https://openreview.net/forum?id=8LhnmNmUDb>.
- Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio P. Calmon. Soft best-of- n sampling for model alignment, 2025. URL <https://arxiv.org/abs/2505.03156>. Accepted for presentation at ISIT 2025.
- vLLM Project. vLLM v0.23.0. Software release, June 2026. URL <https://github.com/vllm-project/vllm/releases/tag/v0.23.0>.
- Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*, 2024a.
- Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. Fusechat: Knowledge fusion of chat models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21618–21642, 2025.
- Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. AlphaZero-like tree-search can guide large language model decoding and training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 49890–49920. PMLR, July 2024b. URL <https://proceedings.mlr.press/v235/wan24c.html>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2022. URL <https://arxiv.org/abs/2203.11171>.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290, 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL <https://arxiv.org/abs/2201.11903>.
- Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilya Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research*, 2024. URL <https://openreview.net/forum?id=eskQMCIbMS>.

-
- Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, et al. Light-rl: Curriculum sft, dpo and rl for long cot from scratch and beyond. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 318–327, 2025.
- Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh Agrawal, Sandeep Singh Sandha, Siddhartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark. In *International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=sKYHBTaxVa>.
- Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.
- Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.
- Haoran Xu et al. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math, 2025. URL <https://arxiv.org/abs/2504.21233>.
- Ziyi Yang, Fanqi Wan, Longguang Zhong, Canbin Huang, Guosheng Liang, and Xiaojun Quan. Fusechat-3.0: Preference optimization meets heterogeneous model fusion. *arXiv preprint arXiv:2503.04222*, 2025.
- Jinwei Yao, Kaiqi Chen, Kexun Zhang, Jiaxuan You, Binhang Yuan, Zeke Wang, and Tao Lin. Deft: Decoding with flash tree-attention for efficient tree-structured LLM inference. In *International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=2c7pf0qu9k>. ICLR 2025.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL <https://arxiv.org/abs/2305.10601>. NeurIPS 2023 (OpenReview oral).
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. doi: 10.48550/arXiv.2406.12045. Introduces the pass^k metric.
- Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. Benchmarking llms via uncertainty quantification. *Advances in Neural Information Processing Systems*, 37:15356–15385, 2024.
- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.
- Fei Yu, Anningzhe Gao, and Benyou Wang. OVM, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 858–875, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.55. URL <https://aclanthology.org/2024.findings-naacl.55/>.
- Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Give me FP32 or give me death? challenges and solutions for reproducible reasoning. *arXiv preprint arXiv:2506.09501*, 2025. URL <https://arxiv.org/abs/2506.09501>.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL <https://arxiv.org/abs/2507.18071>.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena, 2023. URL <https://arxiv.org/abs/2306.05685>. NeurIPS 2023 Datasets and Benchmarks Track.

Yilun Zhou, Austin Xu, Peifeng Wang, Caiming Xiong, and Shafiq Joty. Evaluating judges as evaluators: The JETTS benchmark of LLM-as-judges as test-time scaling evaluators, 2025. URL <https://arxiv.org/abs/2504.15253>. Accepted to ICML 2025.

A Notation

This appendix consolidates the notation used in the formalization (Section 2), the evaluation framework (Section 3), and Appendices B and C.

Core objects and budgeted inference.

Symbol	Meaning
$x \in \mathcal{X}$	Input prompt or problem instance.
$p_\theta(\cdot x)$	Base autoregressive language model.
$\mathcal{T}(x)$	Implicit rooted prefix tree induced by generation from x .
$\mathcal{L}(x)$	Terminal leaves of $\mathcal{T}(x)$, i.e., completed generations.
$z \preceq y$	Prefix relation: search state or prefix z lies on the path to completed leaf y .
π	Local generation policy or decoder used to extend a trajectory.
$q_\pi(y x)$	Proposal distribution over completed leaves induced by p_θ together with policy π .
$\mathcal{A}_B$	Budgeted test-time algorithm with total cost at most B .
$o_t, c(o_t)$	Primitive operation at step t and its associated cost.
B, b_t	Total test-time budget and remaining budget at step t .
$U_x(y)$	Task utility of completed leaf y on instance x .
$\hat{y}_B(x), \hat{a}_B(x)$	Output leaf or extracted answer returned by algorithm $\mathcal{A}_B$.
ξ	Internal (algorithmic) randomness of $\mathcal{A}_B$; ξ_q is its realization on prompt x_q .
$G(B)$	Test-time scaling curve $\mathbb{E}_{x \sim \mathcal{P}, \xi} [U_x(\hat{y}_B(x; \xi))]$.
π_{seq}, ν_t	Single-trajectory controller and its meta-action at step t (distinct from the correct-candidate count ν_q).

Output interpretation and evaluation signals.

Symbol	Meaning
$\text{Parse}(y) = (r, a)$	Deterministic parser that extracts reasoning trace r and answer a from completed generation y .
$\Psi(y) = (d, s)$	Decomposition of y into a deterministically verifiable component d and a non-verifiable remainder s .
$V_P(x, d)$	Programmatic verifier applied to instance x and verifiable artifact d .
$\mathcal{V}$	Codomain of verifier outputs, which may be Boolean, graded, diagnostic, or canonicalized.
PASS	Subset of verifier outputs treated as acceptable by a reducer.
$J_\phi(x, \omega)$	Learned evaluator on object ω , where ω may be a completed leaf y or a partial state z .

Decoding protocols and induced proposals.

Symbol	Meaning
Σ	Token vocabulary.
h_t, ℓ_t, p_t	Partial generation $(x, y_{<t})$, its logits, and the softmax token probabilities.
$D_\lambda, \sigma_t, \chi_t$	One-token decoder with configuration λ , its finite state, and side information.
u_t, S_t	Nonnegative token measure and retained support of the energy-gate decoder (distinct from the MBR utility $u(\cdot, \cdot)$ and the prefix score $S(z)$).
$\tilde{q}_t$	One-token proposal $\text{Norm}_{S_t}(u_t)$.
$q_\lambda(y x)$	Sequence-level proposal induced by composing the local kernels; written q_π when the protocol is denoted π .

Evaluation under test-time scaling.

Symbol	Meaning
$\mathcal{P}$	Task distribution over problem instances.
$\mathcal{D}_{\text{eval}} = \{x_q\}_{q=1}^Q$	Evaluation set with Q prompts or problem instances.
$M_B, \hat{M}_B$	Population and empirical end-to-end performance of algorithm $\mathcal{A}_B$ under budget B .
$Z_{q,i}$	Binary correctness indicator for candidate i on prompt q inside a sampled bank.
ν_q	Number of correct candidates in the bank for prompt q , i.e., $\sum_{i=1}^N Z_{q,i}$.
p_q	Latent single-attempt success probability for prompt q under the fixed proposal.
k, t	Attempt budget of a repeated-sampling diagnostic and success-count threshold, $1 \leq t \leq k$; finite-bank versions require $k \leq N$.
$\kappa_{k,t}(p)$	Binomial tail kernel $\Pr\{\text{Binomial}(k, p) \geq t\}$.
$S_{k,t}$	Discovery–stability profile coordinate: expected fraction of prompts with at least t correct candidates among k fresh attempts.
$\hat{S}_{k,t}^{\text{bank}}$	Finite-bank (without-replacement) estimate of the same threshold event computed from an N -candidate bank.
$\mathcal{H}_n, \mathcal{F}_{i,t}$	Information available after n completed rollouts and through token t of rollout i , respectively.
$\tau_{\text{roll}}, \tau_{\text{tok}}$	Across-rollout and within-rollout stopping times.

Leaf-level scaling.

Symbol	Meaning
N	Number of completed candidates in a leaf bank.
$\mathcal{Y}_N(x) = \{Y_i\}_{i=1}^N$	Multiset of completed candidates generated for input x .
Y_i	i th completed candidate (leaf).
a_i, m_i	Canonical answer parsed from Y_i and its inference-time metadata; $a_i = \perp$ denotes an invalid candidate when applicable.
v_i, j_i	Verifier output $V_P(x, d_i)$ and learned score $J_\phi(x, Y_i)$ for candidate Y_i .
$E_\eta, \mathcal{E}, e_i$	Evidence map, evidence space, and candidate evidence object $e_i = E_\eta(x, Y_i, m_i)$; e_i is scalar for score-based rules.
$\mathbf{W}_n$	Pairwise utility matrix over the first n completed candidates.
$\mathcal{G}_\delta, \hat{o}_n$	Fixed-bank decision rule and its returned leaf, answer, or output set.
$\mathcal{R}_N$	Leaf-level reducer that maps a candidate bank to a final leaf or answer.
$T(x, Y_i, v_i, j_i)$	Tie-breaking score used by verifier-constrained selection.
i^*	Index of the selected candidate when the reducer returns one sampled leaf.
$\hat{q}_N(a \mid x)$	Empirical answer distribution induced by the leaf bank.
$\psi(\cdot)$	Deterministic projection of a leaf used inside an MBR objective.
$u(\cdot, \cdot)$	Utility or similarity function used by an MBR-style reducer.
$C_{\text{gen}}(n), C_{\text{eval}}(n)$	Generation cost and inference-time scoring, control, and reduction cost for a bank of size n .
$C_{\text{signal}}, C_{\text{control}}, C_{\text{decision}}, C_{\text{total}}$	Evidence-acquisition, controller, decision, and total protocol cost.

Prefix-level scaling.

Symbol	Meaning
z	Search state: a literal token prefix or a macro-prefix consisting of one or more reasoning steps.
$\text{Succ}(z)$	Allowed successor states obtained by expanding prefix z .
$\mathcal{F}_t$	Active frontier of unfinished search states at step t .
$\mathcal{Y}_t$	Bank of completed leaves accumulated by a prefix search up to step t .
ρ	Rollout policy used to continue a prefix to completion.
$q_\rho(\cdot \mid x, z)$	Distribution over completed leaves obtained by continuing from prefix z with rollout policy ρ .
$Q_\rho^*(z)$	Continuation value of prefix z under rollout policy ρ .
$\hat{Q}(z)$	Approximate continuation-value estimate used for pruning or prioritization.
$S(z)$	Generic prefix score used by the search controller.
$S_{\text{LL}}(z)$	Likelihood-based prefix score.
$R_\phi, S_{\text{PRM}}(z)$	Process reward model (step-level scorer) and the prefix score it induces.
$H(x, z)$	Partial check; sound for rejection when $H(x, z) = 0$ implies $U_x(y) = 0$ for all $y \succeq z$.
π_{ctrl}, a_t	Search controller and its meta-action (expand, generate successors, allocate rollouts, prune, terminate).
$\hat{Q}_{\text{MC}}(z)$	Monte Carlo estimate of continuation value using M_{roll} rollouts from prefix z .
K	Beam width in beam search.
M_{roll}	Number of rollouts used when estimating $\hat{Q}_{\text{MC}}(z)$.
$n(z), n(z, a)$	Visit counts for state z and action a in an MCTS-style search tree.
$q_{\text{search}, B}(y \mid x)$	Effective leaf distribution induced by a prefix-level search algorithm under budget B .

B Leaf-level reduction: additional derivations and variants

This appendix records a few consequences of the leaf-level formalism from Section 2.3: self-consistency is an empirical MBR rule, weighted candidate banks fit the same template, and answer selection should be kept separate from rationale presentation.

B.1 Self-consistency as empirical MBR

Let $\psi(y)$ be the extracted answer, i.e., $\psi(y) = a$ with $(r, a) = \text{Parse}(y)$, and let $u(a, a') = \mathbf{1}[a = a']$. The empirical MBR objective over a leaf bank $\mathcal{Y}_N(x) = \{Y_i\}_{i=1}^N$ is

$$\hat{R}_N(i) = \frac{1}{N} \sum_{k=1}^N u(a_i, a_k) = \frac{1}{N} \sum_{k=1}^N \mathbf{1}[a_i = a_k] = \hat{q}_N(a_i \mid x).$$

Therefore,

$$i^* \in \arg \max_{i \in [N]} \hat{R}_N(i) \iff \hat{a} \in \arg \max_a \hat{q}_N(a \mid x).$$

Majority-vote self-consistency is thus exactly empirical MBR with a zero-one answer-agreement utility (Wang et al., 2022; Kumar & Byrne, 2004).

B.2 Weighted banks and heterogeneous proposals

A weighted variant replaces the uniform empirical distribution by

$$\hat{q}_w(a \mid x) = \frac{\sum_{i=1}^N w_i \mathbf{1}[a_i = a]}{\sum_{i=1}^N w_i}, \quad w_i \geq 0.$$

The corresponding weighted MBR objective is

$$\hat{R}_w(i) = \sum_{k=1}^N \bar{w}_k u(\psi(Y_i), \psi(Y_k)), \quad \bar{w}_k = \frac{w_k}{\sum_{j=1}^N w_j}.$$

This covers confidence-weighted consensus as well as banks built from heterogeneous proposal distributions. If candidate Y_i is drawn from proposal $q_i(\cdot | x)$ but the desired risk is defined under target proposal $q_\star(\cdot | x)$, then importance weights

$$w_i \propto \frac{q_\star(Y_i | x)}{q_i(Y_i | x)}$$

recover the usual reweighted estimator whenever these ratios are available. In practice, exact correction is often unavailable under heavily truncated decoding or adaptive search, so many systems rely on heuristic weights instead.

B.3 Beyond exact-match consensus

Exact answer matching is often too brittle for open-ended generation. A softer alternative replaces equality with a utility or similarity kernel over projected candidates,

$$s_i = \sum_{\ell=1}^N u(\psi(Y_i), \psi(Y_\ell)), \quad i^\star \in \arg \max_i s_i.$$

Neural MBR instantiates this template with learned reference-based metrics (Freitag et al., 2022), while Universal Self-Consistency uses an LLM-mediated notion of consistency to extend self-consistency beyond exact answer extraction (Chen et al., 2023).

B.4 Proxy misspecification and regularized selection

When a reducer scores candidates using an imperfect proxy j_i , hard argmax selection can overoptimize proxy error as the bank grows (Gao et al., 2023). A simple hedge is a stochastic or regularized selection,

$$\Pr(\hat{y} = Y_i | x, \mathcal{Y}_N) \propto \exp(\alpha j_i),$$

which interpolates between near-uniform selection and hard Best-of- N . More generally, one can combine proxy scores with agreement or Bayes-risk terms so that the selected candidate is not determined by the proxy alone (Jinnai et al., 2025; Verdun et al., 2025). Because the reducer is part of the inference system being evaluated, changing the strength or regularization of selection changes the object under comparison.

B.5 Answer selection versus rationale presentation

For many reasoning tasks, benchmark utility depends only on the final answer rather than on the exact rationale string. It is therefore useful to separate answer selection from rationale presentation:

$$\hat{a} = \arg \max_a \hat{q}_N(a | x), \quad i^\star \in \arg \max_{i: a_i = \hat{a}} \tilde{T}(x, Y_i).$$

The first decision chooses the answer that should be returned; the second chooses, according to a rationale-quality score $\tilde{T}$ (for example, the learned score j_i), which sampled rationale should be shown as a representative explanation for that answer. Keeping these decisions separate prevents silent reversion to full-string MAP selection, which need not optimize output quality even when the answer is correct (Eikema & Aziz, 2020).

C Prefix-level search: controller and evaluator variants

Prefix search allocates computation using scores on unfinished states, with three consequences for evaluation: local evaluation and control factorize into separate design choices, search shifts the proposal distribution over leaves, and prefix methods consume compute on axes that are invisible if one reports only the number of completed candidates.

C.1 Evaluator–controller factorization

A prefix-level search procedure can be written as a pair (S, π_{ctrl}) , where S maps a partial state to a score or summary statistic and π_{ctrl} uses those summaries to choose meta-actions,

$$a_t \sim \pi_{\text{ctrl}}(\cdot \mid x, \mathcal{F}_t, \mathcal{Y}_t, b_t).$$

The action a_t may specify which node to expand, how many successors to generate, how many rollouts to allocate, which states to prune, or when to terminate. Beam search, best-first search, Tree-of-Thoughts-style search, RAP, and MCTS-style procedures all instantiate this evaluator–controller pattern with different scorers and control rules (Kocsis & Szepesvári, 2006; Browne et al., 2012; Yao et al., 2023; Hao et al., 2023). This factorization is useful in ablations: improvements may come from better local evaluation, better compute allocation, or both.

C.2 Search-induced proposal shift

Any controller together with its scorer and stopping rule induces an effective leaf distribution

$$q_{\text{search}, B}(y \mid x) = \Pr(\hat{y}_B(x) = y),$$

which generally differs from the decoder-induced proposal $q_\pi(y \mid x)$; when the search returns a bank rather than a single leaf, the corresponding object is the per-leaf inclusion probability of the returned bank (Section 2.4). Prefix search is therefore not only a compute allocator but also a proposal shaper. This matters for evaluation because banks returned by search are correlated and biased toward branches that the controller explored. Threshold metrics such as Pass@ k and the profile coordinates $\hat{S}_{k,t}^{\text{bank}}$ (Section 3.4) can still be computed on such banks, but they should be interpreted as descriptive summaries of the search-induced bank unless the experimental protocol explicitly averages over repeated executions of the full search algorithm.

When S is a learned reward or value model, prefix search inherits the familiar problem of proxy misspecification and adds a branch-elimination failure mode: a promising branch may be pruned before any high-utility leaf from that branch is ever generated. This failure mode has no direct analog in pure leaf-level reranking, where the candidate already exists before selection (Gao et al., 2023; Hung et al., 2025).

C.3 Compute accounting

Prefix search incurs compute costs for partial-state scoring, rollouts, and tree maintenance, in addition to final generations. Reporting only the number of completed leaves can therefore be misleading: two methods may return the same number of leaves while using very different amounts of model computation and evaluator computation. At minimum, empirical comparisons should separate generated-token cost from evaluator cost, and note whether specialized systems support shared-prefix execution. This distinction matters because tree-structured kernels can exploit prefix sharing through key–value cache reuse, substantially changing the wall-clock frontier for search-based methods (Yao et al., 2025).

D Reasoning LLMs and Evaluated Models

Recent open-weight LLMs have produced a distinct class of *reasoning models* trained or configured to emit intermediate reasoning steps for multi-hop problems in mathematics, science, and code. In practice, these gains rarely come from additional pretraining alone; they arise primarily from model-side adaptation and inference interfaces: adapting a pretrained backbone to structured problem solving using explicit reasoning traces, verifiable rewards, or preference signals, and configuring how reasoning is elicited or budgeted at inference time. Figure 1 gives the main-text chronology connecting these developments to inference-time scaling; this appendix provides the detailed taxonomy and representative open-weight model descriptions used in our evaluation, emphasizing the design choices that most directly affect reasoning accuracy, robustness, and inference cost.

D.1 Common Post-Training Pipeline Archetypes

Recent open-weight reasoning pipelines largely fall into three recurring post-training archetypes. Parameter-efficient adaptation has also been extended to decentralized settings, where clients exchange low-rank updates without centralizing their private data (Ghiasi et al., 2025).

(1) SFT→RL pipelines. A common recipe first applies supervised fine-tuning (SFT) on curated chain-of-thought (CoT) (Wei et al., 2022) demonstrations to elicit explicit multi-step reasoning, then uses reinforcement learning (RL) to optimize behavior under a reward signal. The reward may reflect verifiable correctness (e.g., exact-match math or unit-test code), human or model preferences (e.g., helpfulness or format adherence), or efficiency objectives such as conciseness to mitigate overthinking. This paradigm underlies systems such as SKY-T1/FLASH (NovaSky Team, 2025b;a;c), LIGHT-R1/TINYR1 (Wen et al., 2025; Sun et al., 2025), and Phi-4 reasoning variants (Abdin et al., 2025), where post-training explicitly trades off correctness, reasoning length, and style.

(2) Distillation-based pipelines. Distillation treats reasoning acquisition primarily as a data-generation and filtering problem: a strong teacher (e.g., DEEPSEEK-R1 (Guo et al., 2025) or QWQ (Qwen Team, 2025a)) produces reasoning traces, which are then transferred to a smaller student by standard cross-entropy SFT. Compared with RL, distillation is typically easier to reproduce and much cheaper to train, while still transferring the teacher’s reasoning style and decomposition heuristics. Open workflows such as BESPOKE-STRATOS (Bespoke Labs, 2025b), OPENTHOUGHTS/OPENTHINKER (OpenThoughts Team, 2025c), and OPEN-R1 (Hugging Face, 2025a; Open-R1 Team, 2025e) exemplify this line: they scale either the amount of *verified* reasoning data (from ~10K to > 1M examples) or the rigor of filtering (e.g., exact-match checks and unit tests), often achieving large gains without specialized RL.

(3) Parameter-space fusion and merging. A third line avoids gradient-based updates and instead fuses multiple pretrained or post-trained experts directly in parameter space. Modern merging rules, such as *Select-Calculate-Erase* (SCE), compute structured weighted combinations of aligned weights so that a single merged model inherits complementary behaviors from its parents, such as long-CoT mathematical reasoning and shorter, more direct responses. This approach underlies the FUSEO1 family and related FuseAI releases (Wan et al., 2024a; Yang et al., 2025), and also appears in branch-merge schemes such as TINYR1 (Sun et al., 2025). Fusion is attractive when training budgets are tight because it can be performed quickly and without requiring new data, although its effectiveness depends heavily on checkpoint compatibility.

D.2 Cross-Cutting Mechanisms and Inference Controls

While Section D.1 describes recurring pipeline archetypes, we compare reasoning systems along four cross-cutting mechanism classes used in Figure 1 and Table 2: verified SFT and trace distillation, RL and preference optimization, inference-time control and aggregation, and parameter-space composition.

CoT distillation and verified SFT. At the low-compute end, CoT distillation transfers solution decompositions from a strong teacher to a student by standard SFT. The dominant factor is not raw data volume but the *informativeness and correctness* of supervision, including the reliability of the annotators who provide labels (Ghiasi et al., 2026); reasoning pipelines also use rejection sampling and verification, such as exact-match math checks or unit tests for code, before training. This yields compact but high-yield corpora that can reliably improve reasoning over strong instruction-tuned bases (NovaSky Team, 2025b; Bespoke Labs, 2025b; OpenThoughts Team, 2025c).

RL and preference optimization. When additional generalization or behavioral control is needed, RL-style post-training can optimize policies directly for correctness or preference-aligned behavior. In open-weight work, this often appears as outcome-based RL in verifiable domains such as math and code, or as preference optimization to reduce overthinking and stabilize response length. Direct Preference Optimization (DPO) (Rafailov et al., 2023) is widely used as a practical alternative to Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) because it avoids an explicit reward model and value function while still optimizing a preference objective. In reasoning models, such objectives often prefer shorter correct chains over longer ones, or penalize redundant verification, as in SKY-T1-FLASH (NovaSky Team, 2025a).

Inference-time scaling and aggregation. Reasoning accuracy is also sensitive to the allocation of inference-time compute. Test-time scaling methods sample multiple candidate solutions and aggregate them, for instance, by majority vote in self-consistency, to improve reliability without changing weights (Wang et al., 2022). Other controls allocate a “thinking budget” by encouraging longer deliberation or constraining it to reduce latency, thereby trading off cost and robustness. Because these procedures modify the effective decision rule beyond the base conditional distribution, they should be treated as part of the evaluated system and reported explicitly.

Model merging and fusion. Finally, parameter-space fusion provides a non-gradient mechanism for combining complementary reasoning behaviors, such as long-CoT competence and concise answering. FuseAI-style systems (FuseAI, 2025a;b) use heterogeneous model fusion to produce merged checkpoints that often behave like a single hybrid expert, without multi-stage training runs (Wan et al., 2024a; Yang et al., 2025). In this setting, fusion is especially relevant because it can substantially alter reasoning length and style while keeping the backbone family and parameter count fixed.

D.3 Comparison of Representative Open-Weight Reasoning Models

We instantiate this taxonomy on families represented in, or directly contextualizing, the evaluated roster. Table 2 summarizes the representative open-weight reasoning models examined in this work, organized by mechanism class, training-data scale, base model, and the technical choices that most directly affect both accuracy and cost.

The models evaluated span the principal open paradigms used in modern reasoning LLMs, including verified SFT, SFT+RL/preference optimization, inference-time scaling controls, and parameter-space fusion. This coverage matters for evaluation because different paradigms induce different failure modes and cost profiles: distillation and verified SFT tend to be stable and reproducible; RL and preference optimization can improve correctness or conciseness but are sensitive to reward construction and verification; inference-time scaling improves reliability but increases test-time compute; and fusion can alter reasoning style substantially without additional data. Consequently, any meaningful benchmarking must report both the model checkpoint and the inference protocol (sampling and aggregation), since the latter can change effective reasoning performance even when weights are fixed.

Table 2: Family-level comparison of representative open-weight reasoning models.

Model / Family	Paradigm	Training data	Base model	Technical remarks
DeepSeek-R1 (teacher)	SFT + RL (verifiable)	Multi-stage (cold-start + RL)	DeepSeek-V3-Base	RL-first reasoning (R1-Zero) + multi-stage refinement (R1) to improve usability; teacher for many open distillations.
DeepSeek-R1-Distill [†]	Distillation (verified SFT)	~800k teacher CoTs	Qwen2.5 (1.5B–32B), Llama-3.1-8B	Teacher-generated CoT SFT into standard backbones; strong open baselines for small/mid models.
QwQ-32B	RL reasoning	RL post-training (blog-reported)	Qwen2.5-32B	Dense 32B RL reasoner; reference “thinking” model used as teacher in multiple open pipelines.
Qwen3 Thinking [†] (4B / 30B-A3B / Next-80B-A3B)	SFT + RL (thinking)	Long-CoT cold start + reasoning RL + fusion + distill (TR)	Qwen3; sparse MoE variants	Thinking-specialized checkpoints; large native context; Next-80B uses high-sparsity MoE + hybrid attention with GSPO for RL stability.
s1 / s1.1 (SimpleScaling)	SFT-only (data-minimized) + inference-time control	1k curated CoTs (s1K; math-dominant)	Qwen2.5 (1.5B–32B variants)	Minimal SFT plus budget forcing (“Wait”/termination) to control test-time compute; compatible with sampling/aggregation protocols.

Table 2: Family-level comparison of representative open-weight reasoning models (continued).

Model / Family	Paradigm	Training data	Base model	Technical remarks
Sky-T1 [†] (NovaSky)	SFT + preference/RL	~17k curated prompts + preference pairs (Flash)	Qwen2.5-7B/32B	Rejection-sampled and verified traces; Flash uses length-sensitive preference optimization (SimPO-style) + optional rewriting (FCS+1) to reduce overthinking.
Bespoke-Stratos [†]	Distillation (verified SFT)	17k verified CoTs (math/code/sci)	Students: Qwen2.5-7B/32B / Teacher: DeepSeek-R1	Correctness-filtered distillation (exact-match math, unit tests for code); open Curator pipeline.
OpenThinker [†] (OpenThoughts)	SFT-only (verified)	114k–1.2M verified CoTs (math/-code/sci/puzzles)	Qwen2.5 (1.5B–32B)	Verification-centric SFT; systematic recipe search and controlled verified vs. unverified comparisons; Evalchemy evaluation.
Open-R1 [†] (Distill)	SFT-only (verified)	350k verified traces (math/code/sci)	Qwen2.5-Math-7B	Mixture-of-Thoughts: verified DeepSeek-R1 traces; open reproduction baseline for scaling and verifiable-RL follow-ons.
Open-R1 (Math) / OlympicCoder	SFT-only (verified/partial)	220k math (verified); CodeForces-CoTs (~100k traces; partial)	Qwen2.5-Math-7B; Qwen2.5-Coder-7B/32B	Math Verify + judge fallback for a subset; code traces not exhaustively verified (public tests); emphasizes contamination control and benchmark transparency.
LIMO / LIMO-v2 / LIMR [†] (GAIR)	SFT-only (data-minimized) + RL (verifiable)	800–817 curated math; 1.4k RL-selected items	Qwen2.5-32B-Instruct (LIMO); Qwen2.5-Math-7B (LIMR)	Extreme data efficiency (SFT) + learning-impact selection for RL; v2 updates curation/trace sources while keeping minimal-SFT philosophy.
Light-R1 / TinyR1 [†] (Qihoo360)	SFT + preference/RL; merge (branch–merge)	76k + 3k CoTs; ~3k preference pairs	Qwen2.5; DeepSeek-R1-Distill backbones	Curriculum/step-wise SFT + DPO; GRPO on distilled variants; TinyR1 uses branch specialization then merge for cross-domain gains.
FuseO1 [†] (FuseAI)	Merge (parameter fusion)	N/A	DeepSeek-R1-Distill-32B, QwQ-32B-Preview, Sky-T1-32B (incl. Flash)	SCE-style parameter-space merging (via MergeKit) to fuse long-CoT and concise experts; data-free at merge time.
gpt-oss-20b [†] (OpenAI)	Distillation + RL (MoE)	Large-scale distill + RL (model-card)	OpenAI MoE (21B total / 3.6B active)	MXFP4 quantization; reasoning-effort knob (low/medium/high) for cost/quality trade-off.
Phi-4 Reasoning [†] (Microsoft)	SFT + RL (verifiable)	Distilled traces + ~6k RL math	Phi-4 (14B)	Teachable-prompt selection + long-CoT SFT; outcome-based RL in “plus”; designed to benefit from inference-time compute allocation.
Nemotron Reasoning [†] (NVIDIA)	SFT + RLHF / verifiable RL	Large-scale synthetic post-training	Qwen2.5 (OpenReasoning, AceReason); Llama-3.1/3.3; hybrid Mamba–Transformer (Nano v2)	Family spans distilled small baselines and verifiable-RL “AceReason”; includes reasoning toggle/budget controls and hybrid efficiency variants (e.g., Nano v2).
EXAONE-4.0 [†] (LG AI)	SFT + preference/RL	Multi-domain math/code/science mixtures	EXAONE-4.0-32B / 1.2B	Unified modes (chat vs. reasoning); long-context support; preference/RL stages to balance correctness and verbosity.

† At least one checkpoint or configuration from this family appears in the experiment rosters in Appendix G. Unmarked rows provide field context.

The detailed model-by-model discussion below expands the families summarized in Table 2.

D.3.1 DeepSeek R1 and distilled students: RL-first reasoning and high-yield distillation

DeepSeek-R1 demonstrates that multi-step reasoning can be induced by large-scale RL on verifiable tasks. Guo et al. introduce **DeepSeek-R1-Zero**, trained with RL directly from a base model (no SFT warm-start), which exhibits emergent self-verification and reflection but produces less readable outputs and may mix languages. **DeepSeek-R1** addresses these issues with a multi-stage recipe that introduces cold-start data prior to RL and combines SFT and RL stages (including rejection sampling within the overall pipeline) to preserve reasoning while improving instruction-following and output quality (Guo et al., 2025).

The **DeepSeek-R1-Distill** family transfers R1 behavior into standard dense backbones rather than running RL directly on small policies. The released students include Qwen2.5-based checkpoints at 1.5B/7B/14B/32B, and a Llama-3.1-based 8B checkpoint (Guo et al., 2025). The Hugging Face model cards report that the Qwen2.5 distilled students are fine-tuned on $\sim 800\text{K}$ DeepSeek-R1-curated samples, positioning distillation as a compute-stable alternative to small-model RL (DeepSeek-AI, 2025a). The accompanying project documentation further argues that distilling reasoning patterns from a strong teacher is more effective than rediscovering them via RL at small scales, supporting distillation as a standard small-model alternative in open-weight settings (DeepSeek-AI, 2025b).

D.3.2 Qwen reasoning models: RL-scaled thinking, long-context, and sparse MoE

Qwen’s reasoning line treats “thinking” as an explicit post-training target and emphasizes architectural efficiency (long context and sparse MoE) for models intended to emit intermediate reasoning by default. **QwQ-32B** is a dense 32B reasoning model trained using RL; the Qwen team reports competitive performance against contemporary reasoning systems such as DeepSeek-R1 and o1-mini (Qwen Team, 2025e;a). This provides an RL-first reference point at moderate scale, separate from large distillation pipelines.

Qwen3 extends this direction by separating *thinking* and *non-thinking* behaviors and describing a staged post-training pipeline (long-CoT cold start, reasoning RL, “thinking mode fusion,” general RL, and strong-to-weak distillation) (Team, 2025). In our evaluated set, **Qwen3-4B-Thinking-2507** (Qwen Team, 2025c) is a thinking-specialized checkpoint (no toggle) with a reported native 262K context window. **Qwen3-30B-A3B-Thinking-2507** (Qwen Team, 2025b) reduces active compute via sparse MoE (30.5B total parameters, 3.3B active; 128 experts with 8 active) while retaining the same reported 262K context length. Beyond the evaluated roster, **Qwen3-Next-80B-A3B-Thinking** (Qwen Team, 2025d) extends this design with hybrid attention and high-sparsity MoE; the model card attributes RL stability/efficiency in this regime to Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025). The model card also notes that “thinking” is enforced by the default prompting template (automatic `<think>`) and that generations may contain longer thinking traces than earlier Qwen3 variants.

D.3.3 NovaSky: high-quality distillation on a budget

NovaSky (UC Berkeley Sky Computing Lab) releases data recipes, verification code, training scripts, and evaluation tooling alongside weights. The Sky-T1 line focuses on two objectives: inducing reliable long-CoT behavior from instruction-tuned backbones with small verified corpora, and reducing overthinking-driven token inflation.

Sky-T1-32B-Preview (Li et al., 2025a) serves as the anchor model for the series. Li et al. show that QWEN2.5-32B-INSTRUCT can acquire strong long-CoT behavior from a $\sim 17\text{K}$ set of demonstrations distilled from QWQ-32B-PREVIEW via rejection sampling, with large gains on math/code benchmarks and reported competitiveness with o1-preview (NovaSky Team, 2025b). The paper reports that trace *structure* matters more than trace *content*: perturbing surface tokens is less harmful than disrupting step order or logical coherence (e.g., shuffling or deleting steps).

Sky-T1-32B-Flash (NovaSky Team, 2025a) targets deployment cost by explicitly penalizing overthinking. It forms preference pairs from multiple sampled candidates per prompt (shortest-correct vs. longest-correct) and adds hard-negative pairs (short incorrect vs. long correct) on difficult prompts; it then applies length-sensitive preference optimization and optionally rewrites math solutions to retain only the first correct solution plus one additional solution (“FCS+1”), while leaving code outputs unchanged. The preference stage uses SimPO-style objectives (Meng et al., 2024), making reasoning-length control a direct training target rather than a post-hoc decoding heuristic.

Sky-T1-7B (NovaSky Team, 2025c) explores smaller-budget post-training with an iterative SFT/RL schedule. Starting from QWEN2.5-MATH-7B, it performs SFT on a small verified set of QwQ-distilled traces, applies RL with PRIME-style prompt filtering and rollouts on Eurys-2-RL-Data (Cui et al., 2025), re-distills from the intermediate model for another SFT stage, and finishes with an RLOO-based RL stage (Ahmadian et al., 2024).

D.3.4 Bespoke-Stratos: reasoning distillation as data curation

Bespoke-Stratos (Bespoke Labs, 2025b) frames reasoning post-training as a curation and verification problem: given a strong teacher that produces long-form traces, a comparatively small but *verified* corpus can transfer substantial reasoning behavior via cross-entropy SFT. The pipeline distills DEEPSEEK-R1 traces into BESPOKE-STRATOS-17K and fine-tunes QWEN2.5-INSTRUCT backbones at multiple scales.

The core artifact is **Bespoke-Stratos-17k** (16.7K examples) (Bespoke Labs, 2025c), implemented in BESPOKE CURATOR (Bespoke Labs, 2025a) and derived from a Sky-T1-style distillation recipe. Candidate traces are sampled from DEEPSEEK-R1 and filtered by rejection sampling with task-specific verifiers (strict answer checks for mathematics; unit-test execution for code), yielding supervision dominated by correct trajectories rather than noise. This verifier-first design trades dataset scale for correctness, improving the signal-to-noise ratio of SFT updates.

Two checkpoints instantiate the recipe. **Bespoke-Stratos-32B** (Bespoke Labs, 2025d) fine-tunes QWEN2.5-32B-INSTRUCT on BESPOKE-STRATOS-17K; **Bespoke-Stratos-7B** (Bespoke Labs, 2025e) applies the same procedure to QWEN2.5-7B-INSTRUCT. Reported results via EVALCHEMY (Guha et al., 2024) indicate sizeable gains over the base backbones on standard math and code benchmarks, making these models a sample-efficient distillation baseline.

D.3.5 OpenThoughts/OpenThinker: scaling verified supervised reasoning data

OpenThoughts (Guha et al., 2025; OpenThoughts Team, 2025c) aims to make reasoning post-training reproducible at the level of data recipes by releasing pipelines for synthetic trace generation, verification, ablations, and evaluation alongside model weights. The central claim is that a large portion of open-weight reasoning gains can be achieved with *SFT-only* training when the supervision data are verified and systematically composed.

OpenThoughts-114K (OpenThoughts Team, 2025d) establishes the baseline. The pipeline generates long-form traces over curated math/science/code/puzzle prompts using a strong teacher and applies aggressive filtering to retain traces that are both correct and consistently formatted. The corresponding **OpenThinker** models fine-tune QWEN2.5 instruction-tuned backbones with standard cross-entropy SFT and evaluate via EVALCHEMY, enabling controlled comparisons of verified vs. unverified traces under matched training and evaluation code. The paper identifies verification as a primary driver: it keeps supervision aligned so dataset scaling remains productive rather than saturating on noisy traces.

The project then scales both volume and recipe quality. **OpenThoughts2-1M** (OpenThoughts Team, 2025e) expands to one million verified examples and trains **OpenThinker2-32B** (OpenThoughts Team, 2025a); the authors report matching DEEPSEEK-R1-DISTILL-32B on key reasoning benchmarks (e.g., AIME, LiveCodeBench) while retaining a fully public pipeline. **OpenThoughts3-1.2M** (OpenThoughts Team, 2025f) centers on systematic recipe search (1,000+ controlled experiments) and switches the trace teacher to QwQ-32B for improved small-model transfer. **OpenThinker3-7B** (OpenThoughts Team, 2025b), trained SFT-only on the 1.2M mixture (reported as 850K math, 250K code, 100K science) (OpenThoughts Team,

2025f), is positioned as state-of-the-art open-data 7B reasoning on AIME’25, LiveCodeBench, and GPQA Diamond without RL. In our evaluation, OpenThoughts is a controlled baseline because it isolates the effect of verified supervised data from RL, reward modeling, and preference collection.

In the evaluated roster for this paper, the OpenThoughts family is represented by **OpenThinker2-32B** and **OpenThinker3-1.5B**, giving a matched pair of large- and small-scale SFT-only baselines under the OpenThoughts2 and OpenThoughts3 recipes, respectively.

D.3.6 OpenAI gpt-oss-20b: MoE reasoning with distillation+RL and deployment-oriented packaging

gpt-oss-20b (OpenAI, 2025b) is OpenAI’s medium-size open-weight reasoning model. It is a mixture-of-experts transformer trained with large-scale distillation and reinforcement learning, targeting a favorable accuracy–cost point for local or latency-sensitive deployment (OpenAI, 2025). The OpenAI API documentation reports ~ 21 B total parameters with ~ 3.6 B active parameters per token, reflecting a design that trades routing overhead for lower active compute than dense baselines (OpenAI, 2025a). Consistent with this cost-aware framing, the inference interface exposes a *reasoning-effort* control with three levels (**low**, **medium**, **high**) that trades off latency/token usage against accuracy without changing weights.

Operationally, OpenAI distributes weights on Hugging Face and documents native MXFP4 quantization to reduce memory requirements (e.g., ~ 16 GB) (OpenAI, 2025b). The release also includes companion “safeguard” models (OpenAI, 2025c) for policy-grounded safety reasoning use cases.

D.3.7 Microsoft Phi-4 reasoning models

Microsoft’s Phi-4 reasoning line specializes in a 14B foundation model via curated supervision and, for one variant, a short outcome-based RL stage. Starting from PHI-4 (Abdin et al., 2024), **Phi-4-reasoning** (Abdin et al., 2025) applies SFT on “teachable” prompts—diverse and challenging but within the student’s learnable regime—paired with long-form reasoning demonstrations from a stronger teacher (reported as O3-MINI). The report highlights teachable-prompt selection as a primary lever for converting long-CoT supervision into measurable gains and emphasizes inference-time compute allocation (longer reasoning budgets, repeated sampling) as part of the intended operating regime.

Phi-4-reasoning-plus (Abdin et al., 2025) adds a short outcome-based RL stage that rewards final-answer correctness on a verified math set, improving difficult reasoning while increasing average reasoning length and token usage. Microsoft also releases compact variants to probe the small-model regime. **Phi-4-mini-reasoning** (3.8B) (Xu et al., 2025) uses a staged recipe (mid-training on distilled long-CoT data, high-quality long-CoT SFT, rollout-DPO preference optimization, and verifiable-reward RL for math). **Phi-4-mini-flash-reasoning** (Microsoft, 2025) targets high-throughput long-context reasoning (64K) with a hybrid SAMBAY design and Differential Attention; the accompanying work reports substantial decoding-throughput gains (e.g., under vLLM) while maintaining strong reasoning performance, and the “flash” recipe achieves its reported results without RL in that specific variant (Ren et al., 2025).

D.3.8 NVIDIA Nemotron reasoning models

NVIDIA’s open reasoning releases cover three distinct operating points relevant to this study. **OpenReasoning-Nemotron-1.5B** is a compact reasoning model used as a small-scale open baseline in math/code settings, while **AceReason-Nemotron-1.1-7B** focuses more explicitly on verifiable reasoning through the OpenMathReasoning and OpenCodeReasoning pipelines (Toshniwal et al., 2025; Moshkov et al., 2025; Ahmad et al., 2025a;b; Liu et al., 2025d). Together they mark the “small dense reasoner” and “verifiable-reasoning specialist” ends of NVIDIA’s lineup.

NVIDIA-Nemotron-Nano-9B-v2 (Basant et al., 2025) adds a different design point: a hybrid efficiency-oriented model with controllable reasoning behavior. Relative to the smaller OpenReasoning and AceReason checkpoints, Nano v2 is useful in evaluation because it tests whether hybrid architectures and reasoning toggles can preserve reasoning quality while reducing active compute. In our model roster, the three Nemotron variants therefore provide complementary evidence on small-model reasoning, verifiable post-training, and efficiency-oriented reasoning control within one vendor family.

D.3.9 Open-R1 and OlympicCoder: open reproductions for math and competitive programming

Open-R1 (Hugging Face, 2025a) is an open, staged reproduction of DEEPSEEK-R1-style reasoning post-training that emphasizes auditable artifacts (datasets, scripts, and intermediate checkpoints) to enable controlled studies of data mixture, verification, and inference protocol choices. Methodologically, it serves as a reference pipeline in which the relevant object of study is the full system (data + post-training + inference), especially for reasoning benchmarks where test-time compute can materially change outcomes.

A central component is **Mixture-of-Thoughts** (Open-R1 Team, 2025b), a $\sim 350\text{K}$ dataset of verified traces distilled from DEEPSEEK-R1 across math, code, and science. **OpenR1-Distill-7B** (Open-R1 Team, 2025e) is trained by SFT on this mixture and provides a transparent baseline for analyzing inference-time scaling (e.g., repeated sampling and aggregation) and subsequent verifiable-reward RL on a fixed checkpoint. Open-R1 also provides a math-only corpus, **OpenR1-Math-220k** (Open-R1 Team, 2025f), derived from NuminaMath 1.5 with multiple traces per problem and automatic verification via MATH VERIFY; LLAMA-3.3-70B-INSTRUCT is used as a judge for the 12% of cases where rule-based checks are insufficient (Hugging Face, 2025c).

OlympicCoder extends this methodology to competitive programming, where correctness is partially verifiable but the quality of supervision depends on test coverage. **CodeForces-CoTs** (Open-R1 Team, 2025a) contains 10K Codeforces problems with up to five DEEPSEEK-R1 traces per problem ($\sim 100\text{K}$ total); the dataset documentation notes that traces are not exhaustively filtered for correctness, with about 84% of Python solutions passing public tests (Hugging Face, 2025b). **OlympicCoder-7B** (Open-R1 Team, 2025d) and **OlympicCoder-32B** (Open-R1 Team, 2025c) fine-tune QWEN2.5-CODER instruction models on decontaminated Codeforces-style CoT planning data and evaluate on standardized competitive-programming benchmarks (including an IOI’2024 subset) and LiveCodeBench (Hugging Face, 2025d). Together, Open-R1/OlympicCoder provides a transparent testbed for studying how verification strength, contamination control, and inference-time computation interact in code reasoning.

D.3.10 GAIR “Less is More”: data-minimized reasoning and impact-aware RL

GAIR’s “Less is More” line argues that post-training efficiency is driven more by data selection and supervision structure than by raw volume. **LIMO** (Ye et al., 2025) demonstrates this with SFT of QWEN2.5-32B-INSTRUCT on 817 curated math problems paired with long, high-quality CoT solutions. The study formalizes this as the Less-Is-More Reasoning Hypothesis and presents structured demonstrations and inference-time compute allocation as complementary levers for extracting latent reasoning capability. GAIR subsequently released **LIMO-v2** (GAIR, 2025b), updating the training set to 800 examples (GAIR, 2025a) and releasing a corresponding QWEN2.5-32B-INSTRUCT fine-tune checkpoint. The updated paper describes the revised set as selecting, for each problem, the highest-quality solution sampled from DEEPSEEK-R1, DEEPSEEK-R1-DISTILL-QWEN-32B, and QWQ-32B, preserving the minimal-SFT philosophy while upgrading the trace sources (Ye et al., 2025).

LIMR (Li et al., 2025b) transfers the same efficiency goal to RL by focusing on which prompts drive learning. It introduces *Learning Impact Measurement* (LIM) to select high-impact training samples by aligning reward trajectories with the model’s learning dynamics. LIMR reports that RL on a selected subset of 1,389 prompts can match or exceed RL on the full 8,523-prompt pool, challenging the assumption that RL gains require scaling prompt count. The paper also highlights a regime difference: extreme SFT minimization is most effective at larger scales, whereas impact-aware RL remains effective at smaller scales when selection maximizes learning signal (Li et al., 2025b).

D.3.11 FuseAI/FuseO1: parameter-space aggregation for behavior composition

Fusion and checkpoint merging provide a non-gradient route to composing reasoning behaviors by combining trained “experts” directly in parameter space. Many reasoning checkpoints differ primarily in reasoning style (long-CoT vs. concise “flash”) and domain emphasis; if interference is controlled, merging can approximate multi-expert behavior in a single set of weights with negligible additional compute and no new supervision data.

FuseAI (Wan et al., 2024a; 2025; Yang et al., 2025) studies heterogeneous model fusion via distribution-level and chat-oriented pipelines and introduces **SCE** (Select–Calculate–Erase), a parameter-matrix merge rule built around fusion vectors (weight differences from a pivot) and a structured per-matrix merge procedure. **FuseO1** (FuseAI, 2025c;a;b) applies SCE-style merging to open reasoning checkpoints, combining DEEPSEEK-R1-DISTILL-QWEN-32B, QWQ-32B-PREVIEW, and SKY-T1-32B variants to retain long-CoT competence while reducing verbosity when a Flash parent is included. The model cards report gains under pass@1 and consistency-style evaluation and note that merging is data-free at merge time. Operationally, the merges are typically implemented using MERGEKIT (Arcee AI, 2024), which enables rapid iteration over merge coefficients and parent selection. Branch-merge variants such as TinyR1 are discussed with the Qihoo360 models below.

D.3.12 Light-R1 and TinyR1: curriculum post-training and branch-merge distillation

Light-R1 and TinyR1 represent compute-conscious routes to long-CoT reasoning: curriculum-style post-training to induce long-CoT behavior in backbones that lack it, and branch-merge distillation to compose domain expertise.

Light-R1 (Wen et al., 2025) targets long-CoT training with an explicit staged pipeline. The report describes a two-stage curriculum SFT followed by semi-on-policy DPO to stabilize long-form reasoning and shape response style. In the “from scratch” setting, **Light-R1-32B** (Qihoo360, 2025b) starts from QWEN2.5-32B-INSTRUCT and is trained on math-focused data, with reported generalization beyond math to broader benchmarks. The paper also emphasizes data leverage via a 3K “stage-2” hard long-CoT set: applying this set as additional SFT substantially improves DeepSeek-distilled students, yielding **Light-R1-7B-DS** (Qihoo360, 2025d) and **Light-R1-32B-DS** (Qihoo360, 2025c) from DEEPSEEK-R1-DISTILL-QWEN checkpoints. Finally, Light-R1 studies RL once long-CoT competence is present: using GRPO, it further improves **Light-R1-14B-DS** (Qihoo360, 2025a) and reports the expected coupling between deeper reasoning and higher reward (often with increased length).

TinyR1-32B-Preview (Sun et al., 2025) addresses a different lever: accuracy and cross-domain breadth at fixed scale, via Branch-Merge Distillation. It distills DEEPSEEK-R1 into domain-specialized branches using domain-specific SFT, then merges them to transfer complementary knowledge and improve generalization beyond any single branch. TinyR1 reports consistent gains over DEEPSEEK-R1-DISTILL-QWEN-32B across math/code/science and approaches teacher performance on AIME’24.

D.3.13 SimpleScaling s1.1 family: minimalist SFT and budget forcing for test-time scaling

SimpleScaling’s s1.1 family (Muennighoff et al., 2025) preserves the original s1 recipe of lightweight SFT on a small, curated reasoning set together with budget forcing at inference time, while replacing the earlier Gemini-generated traces with DeepSeek-R1 traces. Budget forcing can either truncate reasoning or extend it by suppressing the end-of-thinking signal and appending “Wait,” making test-time compute directly controllable. The resulting recipe relies on stronger supervision and explicit inference control rather than additional architectural or pipeline complexity.

Across the 1.5B, 3B, 7B, 14B, and 32B variants, s1.1 applies the same post-training recipe to Qwen2.5-Instruct backbones without scale-specific methodological changes. The smaller checkpoints mainly illustrate transfer across model sizes, whereas s1.1-32B serves as the primary reference model. The paper situates budget forcing within a broader test-time scaling framework by combining sequential scaling with parallel aggregation methods such as majority voting and REBASE, so the s1.1 recipe remains compatible with sampling-and-aggregation protocols rather than a single decoding setup.

D.3.14 LG EXAONE-4.0: unified chat-reasoning modes and long-context post-training

EXAONE-4.0 treats chat-style assistance and explicit reasoning as two modes of a single model family rather than as separate systems. LG implements this with a unified post-training recipe that jointly optimizes non-reasoning and reasoning behavior, followed by preference-learning stages that balance correctness, brevity,

and language consistency. The emphasis is on aligning usability, deliberate reasoning, and agentic behavior within one family across both the 32B and 1.2B models (LG AI Research, 2025).

The 32B model also incorporates several targeted engineering choices. Instead of relying on fully global attention, it combines sliding-window local attention with global attention, adjusts positional handling on the global-attention path to reduce long-context degradation, and adopts a revised normalization scheme to improve training stability. Its long-context capability is framed as a training strategy rather than only a larger context window, and its RL stage introduces a customized optimization scheme aimed at improving reasoning quality without losing capabilities acquired during SFT. The tool-use setup follows the same design principle, emphasizing multi-turn, long-horizon interaction with execution feedback rather than isolated tool calls.

E Prompt Templates and Inference Interfaces

Table 3 records the evaluation-time prompting protocol for each of the 20 model configurations in our benchmark, including the prompt interface, any explicit reasoning-control setting, and the answer-format convention used for scoring. Whereas Table 2 summarizes model families and their post-training strategies, the table below documents the inference-time interface choices that define the evaluated systems.

Prompt identifiers. We use three prompt/interface identifiers in the table. *Boxed-answer* denotes the standard instruction, “Please reason step by step, and put your final answer within `\boxed{}`.” *Nemotron math* denotes the OpenReasoning-Nemotron instruction, “Solve the following math problem. Make sure to put the answer (and only the answer) inside `\boxed{}`.” *Harmony* denotes the OpenAI Harmony chat serialization used for `gpt-oss-20b`, with the reasoning-effort setting shown separately.

Table 3: Prompt interfaces, reasoning controls, and answer-format notes for the evaluated model configurations.

Model / configuration	Prompt/interface used	Reasoning control	Output / parsing note
DeepSeek-R1-Distill-Qwen-1.5B	Boxed-answer	Default	Reasoning trace may include <code><think></code> ; final answer extracted from <code>\boxed{}</code>
LIMO-v2	Boxed-answer	Default	Final answer extracted from <code>\boxed{}</code>
OpenThinker2-32B	Boxed-answer	Default	OpenThoughts formatting handled during answer extraction
OpenThinker3-1.5B	Boxed-answer	Default	OpenThoughts formatting handled during answer extraction
Qwen3-30B-A3B-Thinking-2507	Boxed-answer	Thinking mode enabled by checkpoint/template	Thinking traces may be emitted before the boxed answer
Sky-T1-32B-Flash	Boxed-answer	Default	Final answer extracted from <code>\boxed{}</code>
gpt-oss-20b-low	Harmony	<code>effort=low</code>	Harmony-formatted reasoning; final answer canonicalized after generation
gpt-oss-20b-medium	Harmony	<code>effort=medium</code>	Harmony-formatted reasoning; final answer canonicalized after generation
gpt-oss-20b-high	Harmony	<code>effort=high</code>	Harmony-formatted reasoning; final answer canonicalized after generation
EXAONE-4.0-1.2B	Boxed-answer	Reasoning mode enabled	Final answer extracted from <code>\boxed{}</code>
OpenReasoning-Nemotron-1.5B	Nemotron math	Default	Nemotron-specific math prompt; boxed answer required
Phi-4-reasoning	Boxed-answer	Default	Final answer extracted from <code>\boxed{}</code>
Phi-4-reasoning-plus	Boxed-answer	Default	Final answer extracted from <code>\boxed{}</code>
OpenR1-Distill-7B	Boxed-answer	Default	Reasoning trace may include <code><think></code> ; boxed answer used for scoring
FuseO1-DeepSeekR1-QwQ-SkyT1-Flash-32B-Preview	Boxed-answer	Default	Merged checkpoint; reasoning style varies by prompt
Light-R1-14B-DS	Boxed-answer	Default	Reasoning trace may include <code><think></code>
AceReason-Nemotron-1.1-7B	Boxed-answer	Default	Final answer extracted from <code>\boxed{}</code>

Table 3: Prompt interfaces, reasoning controls, and answer-format notes for the evaluated model configurations (continued).

Model / configuration	Prompt/interface used	Reasoning control	Output / parsing note
NVIDIA-Nemotron-Nano-9B-v2	Boxed-answer	Reasoning mode enabled	Final answer extracted from <code>\boxed{}</code>
Qwen3-4B-Thinking-2507	Boxed-answer	Thinking mode enabled by checkpoint/template	Thinking traces may be emitted before the boxed answer
Bespoke-Stratos-7B	Boxed-answer	Default	Final answer extracted from <code>\boxed{}</code>

F Software

Scorio exposes the aggregation protocol of Section 3.3 as three composable stages: confidence or reward signals reduce trace-level metadata to candidate scores; fixed-bank rules select an answer from aligned answer and score arrays; and online predicates determine whether sampling or generation should stop given only the history observed so far. Keeping these stages separate permits the same candidate bank to be evaluated under several decision rules, or the same rule to be paired with different evidence sources, without conflating candidate generation with post-generation selection.

The interface accepts a one-dimensional candidate bank for a single problem or a two-dimensional array for a batch. Score-free consensus, score-based selection, and across-rollout stopping share the same extracted-answer representation. When a rationale must accompany the selected answer, the fixed-bank routines can also return the index of a representative candidate. Listing 1 illustrates the core workflow: plurality and Best-of- N can disagree on one bank, a score-aware vote interpolates between these extremes, and a causal stopping predicate consumes only the answers revealed so far.

```

1  from scorio import agg
2
3  answers = ["A", "A", "A", "B"]
4  scores = [0.30, 0.30, 0.30, 0.85]
5
6  # Fixed-bank decisions over the same candidates
7  plurality = agg.majority_vote(answers)           # "A"
8  best = agg.best_of_n(answers, scores)           # "B"
9  weighted = agg.softmax_weighted_vote(
10     answers, scores, temperature=0.1
11 )                                                 # "B"
12
13 # A stopping check reads only the prefix observed at decision time
14 observed = ["A"] * 8 + ["B"] * 2
15 stop, probability = agg.adaptive_consistency_stop(
16     observed, threshold=0.95, return_prob=True
17 )                                                 # (True, 0.9673...)

```

Listing 1: Examples of fixed-bank aggregation and causal stopping. Higher candidate scores indicate stronger evidence.

G Data, Experiments, and Reproducibility

This appendix documents the response banks used in Section 4 and the artifacts that will accompany them. We use *response* for one model completion and *bank* for the responses produced by one model configuration over a fixed question set. A *capture set* is one concrete execution of such a bank. These distinctions matter for two reasons. First, BBH serializes one response under two extraction filters. Second, the signal-rich mathematics collection contains several independently executed capture sets with the same effective reasoning effort. All trace totals count generated responses, not duplicated evaluator rows.

We separate two reproducibility goals. The released banks support exact replay of every analysis in this paper: the response identities, outcomes, scores, and subset seeds are fixed. Regenerating the same response distribution is a stronger goal and also requires the model revision, rendered prompt, decoder, serving stack, and numerical settings. Where an older archive omitted one of those fields, we state the omission rather than infer it.

G.1 Data sources and frozen inputs

Table 4 gives the realized question sets. The two older blocks retain the source documents inside their records but do not record the dataset-hub revision.

Block	Source	Split	Questions	Snapshot status
Broad	MMLU-Pro (TIGER-Lab, 2024; Wang et al., 2024)	test	12,032	Rows and hashes stored; revision unavailable
Broad	BBH mirror (SaylorTwift, 2025; Suzgun et al., 2023)	test	6,511	Rows and hashes stored; revision unavailable
Repeated math	AIME’24 (Jia, 2024)	train	30	Rows stored; revision unavailable
Repeated math	AIME’25 (math-ai, 2025)	train	30	Rows stored; revision unavailable
Repeated math	HMMT’25 (MathArena, 2025c)	train	30	Rows stored; revision unavailable
Repeated math	BrUMO’25 (MathArena, 2025a)	train	30	Rows stored; revision unavailable
Signal-rich math	Five MathArena-hosted sets	train	186	Exact revisions in Table 5
Broad extension	SuperGPQA (M-A-P Team, 2025; M-A-P Team et al., 2025)	train	3,600	Deterministic, field-balanced sample

Table 4: Question sources. In the Hugging Face releases used here, **train** is the public benchmark split; it is not training data for the generators. The repeated-mathematics block combines two independently hosted AIME sets with two sets distributed through MathArena.

The signal-rich mathematics study uses every row in the default configuration and sole split of the five datasets in Table 5. A read-only audit compares the exact (**problem**, **str(answer)**) pairs against the distinct (**question**, **ground_truth**) pairs in the frozen response index: all 186 pairs match, with neither missing nor extra pairs. We obtained these rows from five MathArena-hosted repositories, but the combined question collection, generation settings, repeated-sampling design, rule grader, and two verifier pipelines are our protocol. Consequently, none of the reported accuracies is an official MathArena leaderboard result (Balunovic et al., 2025; Dekoninck et al., 2026).

Dataset	(Q)	Repository	Frozen revision
AIME’26 (MathArena, 2026a)	30	MathArena/aime_2026	d2de22f3c656b4f56cf8981212186377d1e23bc3
HMMT Feb.’26 (MathArena, 2026b)	33	MathArena/hmmt_feb_2026	02fba4f74d8e68e73e66a02d540fd979c05c274c
HMMT Nov.’25 (MathArena, 2025d)	30	MathArena/hmmt_nov_2025	118dbfb45c4c9467c672268ed55166642897aa46
CMIMC’25 (MathArena, 2025b)	40	MathArena/cmimc_2025	8f27517ce4ad1d05e023c0e7eca4344754ba2a04
SMT’25 (MathArena, 2025e)	53	MathArena/smt_2025	d8e99d83f1583842f46d4953cd960eea5d887e45

Table 5: Source revisions for the five MathArena-hosted competition datasets. Full revisions are tabulated as reproducibility metadata rather than embedded in the narrative.

For SuperGPQA, we obtained the authors’ public release and selected 50 items from each of the 72 fields exposed by EvalScope (ModelScope Team, 2024). Within each field, items are ordered by the SHA-256 digest of a fixed seed, the field name, and the source UUID; the first 50 are retained. The relevant provenance is collected in Table 6.

Artifact	Identifier
Source repository revision	e31e8bf3e7089aef808be5ddecafd4538dee6e41
Source JSONL SHA-256	28b998e70205ee95e540317b5adc06a06552a3961fb50b153df126b833f7a910
Selection seed	supergpqa-50-per-evalscope-subset-v1
Selected JSONL SHA-256	2c8841eead2f672305995918b8e55dcc6f95f066108ea928551e44e3aefea129

Table 6: SuperGPQA source and deterministic field-balanced materialization. The source has 26,529 questions; the materialization contains 3,600.

G.2 Metrics, reducers, and uncertainty

For a question with N stored responses, let c be the number marked correct by the benchmark adapter. The exact without-replacement coordinates used throughout are

$$\text{Pass}@k = 1 - \frac{\binom{N-c}{k}}{\binom{N}{k}}, \quad \text{pass}^k = \frac{\binom{c}{k}}{\binom{N}{k}}. \quad (1)$$

The first asks whether a size- k subset contains at least one correct candidate; the second asks whether every candidate in that subset is correct. They describe a bank and do not specify which response a system submits.

We evaluate submitted-answer accuracy by replaying reducers on subsets of the same bank. Budgets are $k \in \{1, 2, 4, 8, 16, 32, 64, 80\}$. A counter-derived seed gives each question a reproducible random permutation, and prefixes form nested uniform subsets without replacement. In the 20-configuration study we enumerate all 80 singleton choices and use 200 paired subset replays for each $k > 1$. The signal-rich mathematics study uses 2,000 replays for $1 < k < 80$ and exact endpoints. Ties are resolved by the earliest candidate in the randomized subset order (Hariri et al., 2026c). The released analysis tables include subset seeds, selection rates, and candidate-eligibility rates.

The reported intervals resample questions after integrating over subset replays. Experiment 2 uses 2,000 percentile bootstrap replicates; the signal-rich mathematics analysis uses 10,000. These intervals are conditional on the observed 80-response banks. They measure sensitivity to the question composition, not the additional variability that would arise from generating a new bank. Pooled “all” rows are item-weighted micro-averages. For SuperGPQA, field-balanced summaries give each of the 72 constructed fields equal weight because every field contributes 50 questions.

The reducers in Table 7 are run only when the bank contains their required inputs. In particular, the 2024–2025 block does not contain a full next-token alternative distribution, and neither mathematics block contains process-reward scores or a validated answer-start boundary. We do not synthesize those missing signals.

Reducer or statistic	Required record fields	Interpretation
Pass@ k , pass ^{k}	Binary outcome	Exact bank diagnostic
String plurality	Extracted answer	Deployable terminal reducer; literal strings
Sequence/mean log probability	Chosen-token log probabilities	Deployable confidence heuristic
Normalized-likelihood vote	Answer string and response likelihood	Deployable weighted terminal vote
Pointwise Best-of- N	Reference-free Qwen score	Deployable after paying verifier cost
Compass Best-of- N	Question, reference, and candidate	Reference-assisted diagnostic only

Table 7: Inputs and roles of the reported analyses. “Deployable” here means that the reducer does not require a benchmark reference answer; its compute is still additional to candidate generation.

G.3 Broad knowledge and symbolic reasoning

Protocol. MMLU-Pro contributes 12,032 questions across 14 domains with ten choices per question (Wang et al., 2024). The BBH invocation expands the canonical 23-task suite into 27 harness configurations because several tasks are split by object count or sequence length (Suzgun et al., 2023). It should not be interpreted as a new 27-task version of BBH. MMLU-Pro extracts a final letter A–J; BBH records both a flexible and a strict answer filter, and we use the flexible filter for its reported score.

The manifests identify `lm-evaluation-harness 0.4.8 (e4a7b69)` (Sutawika et al., 2025). Generation uses vLLM (Kwon et al., 2023) with bfloat16 weights, zero demonstrations, temperature 0.6, top- $p = 0.95$, frequency penalty 0.1, a 32,768-token generation cap, and seed 777 for the harness, NumPy, PyTorch, and few-shot sampler. The stop strings are `</s>`, `Q:`, and `<|im_end|>`. No system message or chat template is recorded.

Census and record contents. The curated snapshot contains 1,213 files and 14,542,539,395 logical bytes. All 1,107 sample JSONL files and 54 result manifests parse, and all recomputed group scores agree with the manifests. The 54 runs are 27 checkpoint labels crossed with two suites. MMLU-Pro contributes 324,864 responses and BBH175,797, for 500,661 generated responses. BBH writes each response once for each of its two filters, so the serialized-row count is 676,458.

Each row retains the source document, target, rendered generation arguments, raw response, filtered response, exact-match outcome, and hashes of the document, prompt, and target. Token IDs, token-level probabilities, structured finish reasons, and dataset revisions are absent. We therefore report character, UTF-8-byte, and whitespace-word lengths for this block and do not infer length termination.

Configuration	MMLU-Pro	BBH
Bespoke-Stratos-32B	67.37	75.41
Bespoke-Stratos-7B	46.68	51.50
DeepSeek-R1-Distill-Llama-8B	29.42	64.57
DeepSeek-R1-Distill-Qwen-1.5B	3.13	25.97
DeepSeek-R1-Distill-Qwen-14B	48.85	77.88
DeepSeek-R1-Distill-Qwen-32B	46.02	81.68
DeepSeek-R1-Distill-Qwen-7B	6.68	53.11
FuseO1-DeepSeekR1-QwQ-32B-Preview	45.05	81.22
FuseO1-DeepSeekR1-QwQ-SkyT1-32B-Preview	44.96	82.20
FuseO1-DeepSeekR1-QwQ-SkyT1-Flash-32B-Preview	44.52	81.45
FuseO1-DeepSeekR1-Qwen2.5-Instruct-32B-Preview	46.62	82.14
LIMO	65.61	74.41
Light-R1-14B-DS	38.23	74.09
Light-R1-32B	55.60	74.44
Light-R1-7B-DS	7.68	54.57
OlympicCoder-32B	50.78	57.79
OlympicCoder-7B	40.88	56.89
OpenR1-Qwen-7B	15.97	31.45
OpenThinker-32B	66.62	75.26
OpenThinker-7B	47.45	52.22
QwQ-32B	70.47	58.45
Qwen2.5-32B-Instruct	67.10	56.52
Sky-T1-32B-Flash	60.29	79.14
Sky-T1-32B-Preview	59.48	79.67
Sky-T1-7B	15.08	39.35
TinyR1-32B-Preview	58.32	77.39
s1.1-32B	64.23	41.48

Table 8: Broad-block exact match in percent. BBH uses its flexible extractor. The columns are separate descriptive results and are not averaged.

There are 1,929 empty primary BBH extractions and 126 genuinely empty BBHresponses; all remain in the denominator and are scored incorrect. MMLU-Pro has neither case. The two Sky-T1-7B directories lack `config.json`; their result manifests identify NovaSky-AI/Sky-T1-7B, but no resolved checkpoint revision is recoverable. The file-level audit and corrected source description are in `experiments/experiments/experiment1.md`.

G.4 Twenty configurations on 2024–2025 mathematics

Grid and models. The four sets are AIME’24, AIME’25, HMMT’25, and BrUMO’25, with 30 questions each. HMMT’25 and BrUMO’25 are distributed through MathArena, but the four-set combination is not itself a MathArena benchmark. Every one of the 20 configurations in Table 9 is evaluated with seeds 1234–1313, giving

$$20 \text{ configurations} \times 120 \text{ questions} \times 80 \text{ seeds} = 192,000 \text{ responses.}$$

Configuration	Stored model identifier
AceReason-Nemotron-1.1-7B	nvidia/AceReason-Nemotron-1.1-7B
Bespoke-Stratos-7B	bespokelabs/Bespoke-Stratos-7B
DeepSeek-R1-Distill-Qwen-1.5B	deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B
EXAONE-4.0-1.2B	LGAI-EXAONE/EXAONE-4.0-1.2B
FuseO1-DeepSeekR1-QwQ-SkyT1-Flash-32B-Preview	FuseAI/FuseO1-DeepSeekR1-QwQ-SkyT1-Flash-32B-Preview
LIMO-v2	GAIR/LIMO-v2
Light-R1-14B-DS	qihoo360/Light-R1-14B-DS
NVIDIA-Nemotron-Nano-9B-v2	nvidia/NVIDIA-Nemotron-Nano-9B-v2
OpenR1-Distill-7B	open-r1/OpenR1-Distill-7B
OpenReasoning-Nemotron-1.5B	nvidia/OpenReasoning-Nemotron-1.5B
OpenThinker2-32B	open-thoughts/OpenThinker2-32B
OpenThinker3-1.5B	open-thoughts/OpenThinker3-1.5B
Phi-4-reasoning / plus	microsoft/Phi-4-reasoning / microsoft/Phi-4-reasoning-plus
Qwen3-30B-A3B / Qwen3-4B Thinking	Qwen/Qwen3-30B-A3B-Thinking-2507 / Qwen/Qwen3-4B-Thinking-2507
Sky-T1-32B-Flash	NovaSky-AI/Sky-T1-32B-Flash
gpt-oss-20b low / medium / high	openai/gpt-oss-20b, three effort values

Table 9: The 20 repeated-mathematics configurations. Rows with slashes expand to multiple configurations; the released inventory has one row per configuration.

Generation records. Every record reports top- p sampling with temperature 0.6, top- $p = 0.95$, one completion, a 32,768-token cap, and retained special tokens. It stores the prompt, full response, prompt/completion token counts, finish reason, extracted answer, binary correctness, human solution when available, and the realized token’s log probability and rank at every generation position. It does not store the other next-token alternatives. Sequence likelihood, mean log probability, and realized-rank summaries are therefore supported; entropy and other full-distribution methods are not.

CompassVerifier-7B field. This older block contains a separately generated CompassVerifier-7B outcome score. Its A/B/C labels mean correct, incorrect, and invalid, and the score uses `contextual_probs.A` (Liu et al., 2025c). All 192,000 rows contain the verifier key; 178,142 contain finite numerical probabilities and 13,858 contain the sentinel `raw_score="length"`. Twenty-six numerical rows lack a parsed candidate answer, leaving 178,116 eligible candidates. Selection excludes ineligible candidates without imputation. The archived script is not the exact execution manifest for these stored 7B scores, so we treat the field as fixed released evidence rather than claim that its generation can be exactly reproduced. It sees the gold answer and is used only as a reference-assisted diagnostic.

The leading configuration’s Acc.@1 has a 95% prompt-bootstrap interval of [68.81, 82.18]. Across the 20 configurations, the mean Spearman correlation between subset and full-bank accuracy rankings is 0.967 at $k = 1$ and 0.992 at $k = 8$. This stability describes the present roster; it is not a general sample-size guarantee. The archive contains 160,521 `stop` and 31,479 `length` finishes. A length finish does not imply a missing verifier score: 17,621 length-finished responses still have numerical Compass values.

Configuration	Acc.@1	Pass@8	Pass@80
Qwen3-30B-A3B-Thinking-2507	75.56	85.83	91.67
Qwen3-4B-Thinking-2507	66.55	80.40	87.50
Phi-4-reasoning-plus	65.56	84.19	91.67
gpt-oss-20b high	63.99	83.94	90.83
gpt-oss-20b medium	63.65	86.49	93.33
AceReason-Nemotron-1.1-7B	60.74	74.70	80.83
Phi-4-reasoning	59.97	82.81	90.83
OpenThinker2-32B	59.75	76.77	82.50
FuseO1-DeepSeekR1-QwQ-SkyT1-Flash-32B-Preview	58.52	74.05	81.67
Light-R1-14B-DS	58.27	74.09	81.67
NVIDIA-Nemotron-Nano-9B-v2	54.72	72.12	79.17
LIMO-v2	53.45	73.33	82.50
gpt-oss-20b low	47.39	72.39	85.00
EXAONE-4.0-1.2B	44.29	67.78	83.33
OpenR1-Distill-7B	43.68	65.66	78.33
OpenThinker3-1.5B	43.38	62.13	75.00
OpenReasoning-Nemotron-1.5B	40.73	62.07	78.33
DeepSeek-R1-Distill-Qwen-1.5B	25.30	47.47	64.17
Sky-T1-32B-Flash	25.27	43.95	55.83
Bespoke-Stratos-7B	18.39	36.22	54.17

Table 10: Complete fixed-roster results for the 2024–2025 mathematics block, in percent. Acc.@1 averages the 80 response outcomes per question; Pass@ k is the exact without-replacement discovery statistic.

Qwen3-30B-A3B-Thinking-2507	$k = 1$	$k = 8$	$k = 80$
Pass@ k	75.56	85.83	91.67
pass ^{k}	75.56	61.50	43.33
String plurality	75.56	77.74	78.33
Sequence-log-probability Best-of- N	75.56	78.45	77.50
Mean-log-probability Best-of- N	75.56	74.56	65.83
Compass Best-of- N (reference-assisted)	75.51	84.23	89.17

Table 11: Selected repeated-mathematics results in percent, pooled over 120 questions. The $k = 1$ reducer values enumerate all singletons; $k = 8$ uses 200 paired subset replays; $k = 80$ is the full bank. Negative perplexity is identical to mean-log-probability selection.

G.5 Signal-rich competition-math capture

Generators and prompts. The generators are Qwen3.6-35B-A3B and gpt-oss-20b. The realized model snapshots and the evaluator snapshots are listed in Table 12. Generation launches resolved the two generator snapshots from cached `main` rather than passing an explicit revision; a reproduction should pin the listed revisions.

Role	Repository	Realized revision
Generator and pointwise verifier	Qwen/Qwen3.6-35B-A3B	995ad96eacd98c81ed38be0c5b274b04031597b0
Generator	openai/gpt-oss-20b	6cee5e81ee83917806bbde320786a8fb61efebec
Reference-based verifier	opencompass/CompassVerifier-3B	27d6d660bdf6c2b8a9f45906d4b76c508af65ab0

Table 12: Model provenance for the signal-rich mathematics capture.

The mathematics prompt is zero-shot:

```
1 Solve the following math problem step by step. Put your answer inside \boxed{}.
2
3 {question}
4
5 Remember to put your answer inside \boxed{}
```

No example, reference, correctness label, or evaluator score is shown to the generator. The text is supplied as the user message before applying each model’s chat template. EvalScope 1.8.1 extracts the final boxed answer and checks numeric or symbolic equivalence, including its SymPy rules; no LLM judge overwrites this primary binary outcome (ModelScope Team, 2024).

Qwen uses temperature 1, top- $p = 0.95$, top- $k = 20$, and presence penalty 1.5. GPT-OSS uses temperature 1, top- $p = 1$, no presence penalty, and requested top- $k = -1$. Both use one completion, seeds 0–79, retained special tokens, and an 81,920-token generation cap. The servers run the stock `vllm/vllm-openai:v0.23.0` image (vLLM Project, 2026). Eight H200 GPUs host eight independent replicas (TP=PP=DP=1), with at most 12 outstanding requests per replica.

Effective effort and sampling. The corpus contains seven complete capture sets of $186 \times 80 = 14,880$ responses each. One uses Qwen; one each uses correctly rendered gpt-oss low, medium, and high effort. In three earlier captures, effort was supplied inside `chat_template_kwargs`, whereas Harmony reads a top-level request field (OpenAI, 2025a). All three therefore rendered medium. They are valid independent medium-effort repeats and remain in the release, but they are never treated as low or high or pooled into an $N = 320$ bank. Their Acc.@1 values range from 63.08% to 63.47%; the corrected-medium bank is 62.80%.

GPT-OSS request metadata says top- $k = -1$, which vLLM 0.23.0 documents as disabling top- k . Nevertheless, none of approximately 40.4 million audited chosen completion tokens falls outside the contemporaneously logged top 20. We therefore describe top- $k = 20$ as the observed effective behavior without attributing an unverified mechanism. Qwen explicitly requested top- $k = 20$. This case illustrates why rendered prompts and realized token traces are more reliable provenance than request metadata alone.

Token-level fields. At every prompt and completion position, the full gzip archive stores token text, ID, bytes, the chosen-token log probability and rank, and up to 20 candidate tokens with their IDs, bytes, log probabilities, and ranks. The chosen token is one of those entries, not an additional twenty-first token. Candidate arrays must be sorted numerically before being interpreted as ranked because vLLM may serialize the sampled token first. Completion ranks are right-censored at 20, while prompt-token ranks may exceed 20.

The values are raw model log probabilities before presence penalties and sampling truncation. Renormalizing GPT-OSS’s observed width-20 values recovers its effective categorical distribution in this capture because temperature is one, full nucleus mass is retained, and no penalty is applied. The same is not generally true for Qwen: its presence penalty can change membership before top- k and top- p filtering.

The compact Parquet view retains chosen-token and width-20 log-probability arrays but omits raw text and candidate token identities; analyses that need token identity must read the full gzip archive. The validated compact snapshot contains 1,832,922,842 completion positions and 36,658,456,840 candidate log-probability values. Neither view contains a process reward or a validated answer-start boundary, so we do not run process-reward or split-span methods.

G.6 Verifier implementations and diagnostics

The two verifier fields are complementary by construction (Table 13). CompassVerifier provides a strong reference-assisted outcome view. The pointwise Qwen evaluator provides a reference-free signal that can be used for Best-of- N , but it judges a different target and has different failure modes.

	CompassVerifier-3B (OpenCompass)	Pointwise Qwen adaptation
Input	Problem, reference answer, candidate response	Problem and one candidate response; no reference or other candidate
Judgment	A/B/C final-answer outcome; reasoning explicitly ignored	Problem understanding, reasoning validity, and conclusion support
Passes per unique input	Direct and contextual one-token passes, plus a shared null baseline	Three greedy critiques and three one-token score-prefill passes
Score	Direct A/B/C softmax; null-adjusted contextual A/B/C softmax	Expected A–T ordinal score per criterion, then mean
Role	Reference-assisted outcome diagnostic	Reference-free Best-of- N evidence
Context handling	32,768 tokens; deterministic head–tail shortening for 8,750 inputs	Candidate retained intact in 262,144-token context; critique capped at 4,096 tokens
Main caveat	Gold access and neither distribution is calibrated	Ordinal, not calibrated; Qwen evaluates its own Qwen traces

Table 13: The two verifier pipelines. The pointwise pipeline is inspired by LLM-as-a-Verifier (Kwok et al., 2026) but is not its pairwise tournament. CompassVerifier follows the released outcome-verifier prompt and checkpoint (Liu et al., 2025c).

CompassVerifier-3B (OpenCompass). The official prompt supplies the problem, standard answer, and candidate. It asks for A (correct), B (incorrect), or C (invalid), instructs the verifier to compare final answers, and explicitly says to ignore errors in the reasoning when the final answer is correct (Liu et al., 2025c). In the direct pass, generation is one token with temperature zero, top- $p = 1$, top- $k = 0$, and repetition penalty 1.05. We leave generation unconstrained but request the unmasked log probabilities (ℓ_A, ℓ_B, ℓ_C) of the one-token labels A/B/C (token IDs 32/33/34), choose the stable A/B/C argmax, and store

$$p_{\text{direct}}(z) = \text{softmax}_{z \in \{A, B, C\}}(\ell_z).$$

The legacy `raw_score.probs` field contains only the exponentiated log probability of the selected label; it must not be read as the three-class distribution. The explicit `abc_probs` field is the distribution.

The contextual pass appends **Final Judgment:** (. It uses one-token decoding with temperature 1, top- $p = 1$, top- $k = 0$, repetition penalty 1, and seed 0. Let ℓ_z^{slot} denote its label log probability. The null prompt fills the problem, reference, and candidate slots of the official template with the literal string [N/A]; let its label log probability be ℓ_z^{null} . The stored distribution is

$$p_{\text{ctx}}(z) = \text{softmax}_{z \in \{A, B, C\}} \left(\frac{\ell_z^{\text{slot}} - \ell_z^{\text{null}}}{1.5} \right).$$

This adjustment is a diagnostic used by our pipeline; it is not a calibrated correctness probability or the production label of CompassVerifier.

Exact length-framed deduplication maps 104,160 archive locations to 97,786 unique question–reference–candidate triples. If a prompt exceeds the 32,768-token context, only the candidate is shortened. The question, reference, instructions, first 1,024 candidate tokens, a marker, and a candidate suffix sized under a conservative 64-token guard are retained. Both rendered prompts are remeasured and the suffix is reduced until they fit. This affects 8,750 unique inputs (8.95%). The final audit covers all 2,800 files and 104,160 locations. Among the 97,786 unique inputs, the direct labels are 68,725 A, 25,709 B, and 3,352 C.

Reference-free pointwise Qwen evaluator. The corpus-wide score adapts the rubric and expected-score idea of LLM-as-a-Verifier to one candidate at a time (Kwok et al., 2026). It replaces the cited method’s jointly prompted candidate pair and Gemini judge with one trajectory and Qwen, uses one critique and one 20-level score per criterion, and ranks candidates directly. It does not run pairwise orientations, repeated comparisons, a tournament, or Bradley–Terry aggregation. For each of three criteria, Qwen first produces a critique with thinking disabled, temperature zero, top- $p = 1$, top- $k = 0$, a 4,096-token cap, and `<score>` as a stop string. A second request appends `<score>` as an assistant prefill and uses temperature 1, top- $p = 1$, top- $k = 0$, a one-token cap, and a deterministic seed derived from the content hash and criterion. It requests 20 log-probability entries over the constrained bare and leading-space spellings of A–T; the sampled token is ignored. Duplicate spellings are collapsed by their maximum probability, visible mass is renormalized, and unobserved letters receive zero mass.

Let $q_j(L)$ be the resulting distribution for criterion j , with $v(A) = 20, \dots, v(T) = 1$. We compute

$$r_j = \frac{\sum_{L=A}^T q_j(L)v(L) - 1}{19}, \quad s_{\text{point}} = \frac{1}{3} \sum_{j=1}^3 r_j.$$

The score is in $[0, 1]$, but the construction is ordinal rather than probabilistic calibration. The score pass requests only 20 log-probability entries for 40 allowed token spellings, so low-mass letters may be censored.

Deduplication again maps the 104,160 locations to 97,786 unique problem–candidate pairs. A preflight over all 293,358 criterion prompts found zero context violations and a maximum rendered prompt of 124,721 tokens; after reserving 4,096 critique tokens and 32 scoring-prefill tokens, the minimum remaining margin is 133,295 tokens. Candidate responses are therefore retained intact inside the 262,144-token context. Worker logs indicate that 57,433 critiques (19.58%) reached the 4,096-token cap; their partial critiques were still passed to scoring. This count combines standard worker manifests with a pre-restart log segment, because the archive payload does not retain a per-criterion cap flag. Qwen judging Qwen-generated responses is self-evaluation; judging gpt-oss responses is cross-model evaluation.

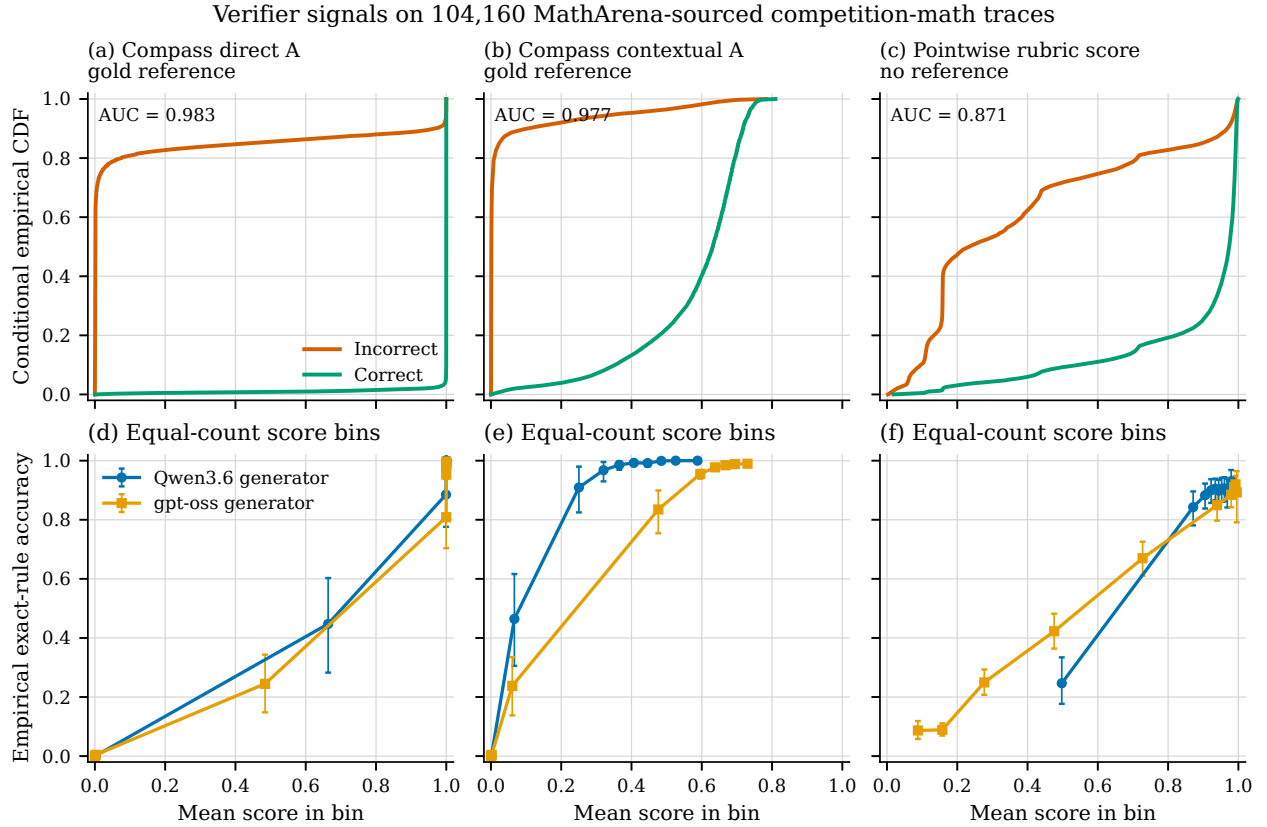


Figure 7: Verifier signals against the independent rule-based outcome on 104,160 competition-math traces. The figure compares direct and contextual Compass A scores with the reference-free pointwise Qwen score through conditional distributions and equal-count score bins. The lower panels separate Qwen-generated self-evaluations from gpt-oss cross-model evaluations. Compass sees the reference answer; pointwise Qwen does not. The binned empirical correctness curves describe ranking behavior on this fixed archive and should not be interpreted as calibration guarantees.

Against the rule-based outcome, the descriptive trace-level ROC AUC is 0.983 for direct Compass A, 0.977 for contextual Compass A, and 0.871 for the pointwise score. The pointwise AUC is 0.744 on Qwen-generated traces and 0.876 on gpt-oss traces. This split does not identify a causal self-evaluation effect because the

generator families also differ in prompts, output distributions, and base accuracy. It warns against pooling evaluator behavior across generators.

Scaling results. Table 14 reports the four canonical banks. String plurality groups literal extracted strings rather than mathematically equivalent forms. Pointwise Best-of- N uses only the reference-free score; Compass fields are released but excluded from these deployed-selection results.

Canonical bank	Acc.@1	Pass@8	Pass@80	Str.-Plur.@80	Pnt.-BoN@80	Caps
Qwen3.6	83.12	91.70	94.62	89.25	86.56	675
gpt-oss low	32.90	56.14	72.58	45.70	58.06	0
gpt-oss medium	62.80	85.99	91.94	80.11	75.81	32
gpt-oss high	73.03	88.12	93.55	85.48	81.72	2,399

Table 14: Finite-bank and shared-bank results on the 186 MathArena-sourced competition questions, in percent. Caps count responses that reach 81,920 tokens out of 14,880. Pnt.-BoN is pointwise Best-of- N .

At $k = 80$, the 95% prompt-bootstrap intervals for Pass@80 are [90.86, 97.85], [66.13, 79.03], [87.63, 95.70], and [89.78, 96.77] in table order. Pointwise Best-of- N intervals are [81.72, 91.40], [51.08, 65.05], [69.35, 81.72], and [75.81, 87.10]. Several intervals overlap. Mean completion counts per response are 34,382 tokens for Qwen and 1,781, 12,341, and 37,264 for gpt-oss low, medium, and high. These costs exclude prompt tokens and verifier computation. The high-effort cap rate is 16.12%, so the effort comparison is inseparable from the shared length limit.

Across all seven capture sets, the source audit finds 104,160 responses, 103,867 parsed answers, 65,706 rule-correct outcomes, and 3,192 generation-cap events. The four canonical banks account for 3,106 of those events. The released `source_conversion_manifest.json` checks 35 dataset-capture rows, 1,302 capture-question shards, 80 seeds per shard, schemas, and token-array widths, and records its source digest.

G.7 Field-balanced SuperGPQA extension

SuperGPQA contains 26,529 multiple-choice questions across 285 graduate disciplines (M-A-P Team et al., 2025). Our 3,600-question materialization has 50 items in each of 72 EvalScope fields. It contains 1,447 easy, 1,607 middle, and 546 hard questions; 940 are marked as calculation questions. The deterministic construction in Table 6 makes the sample rebuildable and ensures that large source fields do not dominate the result.

The exact zero-shot user template is:

```

1 Answer the following multiple choice question. The last line of your response
2 should be of the following format: 'ANSWER: [LETTER]' (without quotes) where
3 [LETTER] is one of {letters}. Think step by step before answering.
4
5 {question}
6
7 {choices}
```

The adapter extracts the final option letter and records binary accuracy. The four generation banks are Qwen3.6 and correctly rendered gpt-oss low, medium, and high, each with seeds 0–79. Sampling, serving, and trace capture follow the signal-rich mathematics protocol. The verifier fields use the same model snapshots and score constructions: Compass traverses response-record locations, whereas pointwise Qwen deduplicates within generator groups and reconciles cross-group duplicates. Its rubric is generalized to factual or technical accuracy, logical coherence, and completeness. The complete design contains $3,600 \times 4 \times 80 = 1,152,000$ responses.

The finalized gpt-oss high rule-outcome matrix provides one view of the heterogeneity that a single aggregate hides. Its 288,000 responses have mean accuracy 45.03%. Exact Pass@80 is 81.94%, while pass⁸⁰ is 9.47%. Of the 3,600 questions, 650 are never answered correctly and 341 are answered correctly in all 80 responses.

Field-level response accuracy ranges from 21.33% in Aquaculture to 76.78% in Mathematics. Figure 8 reports finite-bank scaling and field variation from the 3,600 by 80 binary matrix. These values are descriptive of the equal-field construction, not estimates under SuperGPQA’s original field frequencies.

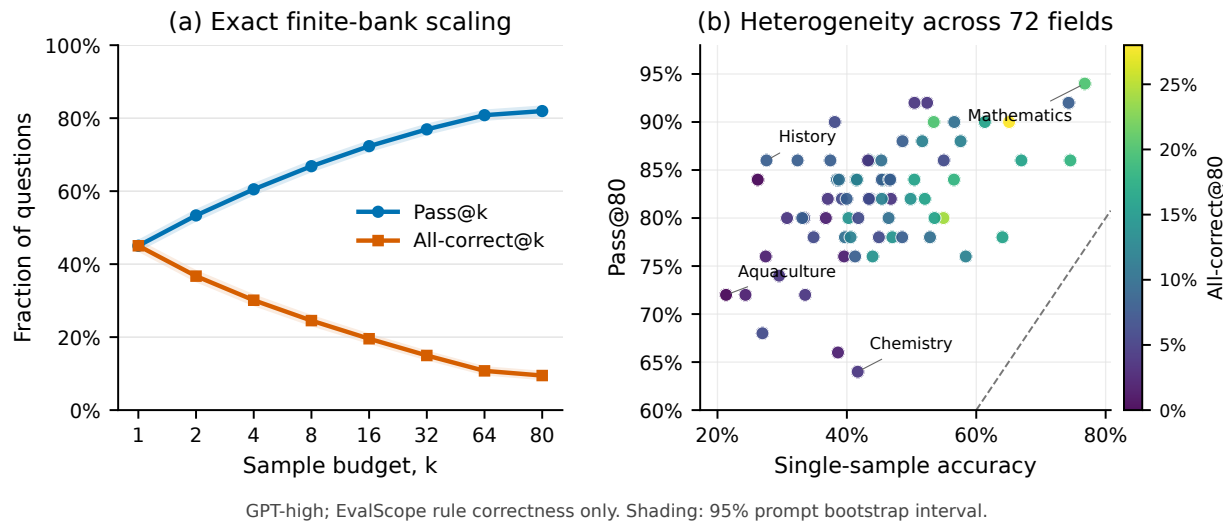


Figure 8: **SuperGPQA scaling and field heterogeneity for the finalized gpt-oss high bank.** The left panel gives exact finite-bank discovery and stability over 3,600 questions. In the right panel, each point is one of 72 equally sized fields: the axes compare single-response accuracy with Pass@80, and color shows the all-correct@80 coordinate. This figure uses only the audited rule-based correctness matrices; it does not substitute an LLM verifier for the benchmark outcome.

The response records include full text, width-20 token alternatives, the rule-based multiple-choice result, CompassVerifier-3B, and the pointwise Qwen record. Compass remains reference-assisted; pointwise Qwen remains reference-free. Because the sampling unit is a fixed 50-item field slice, any overall number should be reported as field-balanced rather than as a natural SuperGPQA micro-average.

G.8 Release schema and environments

Record layers. Table 15 summarizes the common fields and the intentional differences between blocks. The release preserves raw response text even when an answer extractor fails. Exact duplicate candidates in the signal-rich math block share verifier computations, but every original archive location keeps its own copy of the resulting payload.

Field family	Broad	Repeated math	Rich math	SuperGPQA
Prompt, full response, identity	Yes	Yes	Yes	Yes
Parsed answer and rule outcome	Yes	Yes	Yes	Yes
Finish reason and token counts	–	Yes	Yes	Yes
Chosen-token log probability/rank	–	Yes	Yes	Yes
Width-20 token alternatives	–	–	Yes	Yes
Reference-based verifier	–	Compass-7B	Compass-3B	Compass-3B
Reference-free pointwise verifier	–	–	Yes	Yes

Table 15: Principal release fields. A dash means that the source archive does not support the field; it is not reconstructed during conversion.

Software and hardware. The rich-capture environment is summarized in Table 16. The broad and repeated-mathematics archives do not contain a hardware manifest, so we do not guess their accelerator type or GPU count.

Stage	Software	Hardware and parallelism
Generation	vLLM 0.23.0; PyTorch 2.11.0+cu130; EvalScope 1.8.1; <code>datasets</code> 4.8.4; SymPy 1.14.0	Eight H200 GPUs; one replica/GPU; TP=PP=DP=1; 12 requests/replica
CompassVerifier-3B	vLLM 0.23.0; Transformers 5.12.0; BF16	One A100-80GB per worker; up to eight workers; batch 32
Pointwise Qwen	vLLM 0.23.0; Transformers 5.12.0; BF16	One H200 per worker; eight logical shards; batch 4
Analysis	Python analysis environments	Up to 128 CPU workers for source audits

Table 16: Runtime environment for the signal-rich blocks. Generator and verifier computation use separate fleets.

The analyses reject missing records, duplicate identities, nonfinite required values, out-of-range scores, and unexpected grids. They read source archives without modifying them. Release-level manifests validate full-file schemas, token-array widths, compression integrity, and file hashes.

Licensing and use. At the audited snapshots, the dataset cards declare MIT for MMLU-Pro and AIME’24, Apache-2.0 for AIME’25, CC BY-NC-SA 4.0 for the cited MathArena-hosted sets, and ODC-BY for SuperGPQA (TIGER-Lab, 2024; Jia, 2024; math-ai, 2025; MathArena, 2025c;a; 2026a;b; 2025d;b;e; M-A-P Team, 2025). The BBH mirror used by the harness does not declare a license on its dataset card (SaylorTwift, 2025). Dataset-card metadata does not by itself determine the terms of generated model outputs or underlying component questions. The release therefore preserves source attribution and model identifiers, and users remain responsible for the terms applicable to each component.